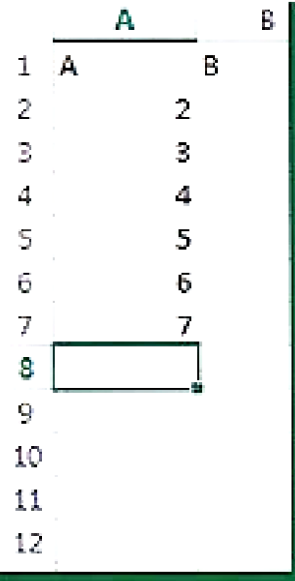


Числовые расчеты в Excel

А. Н. Васильев

А. Н. ВАСИЛЬЕВ

ЧИСЛОВЫЕ РАСЧЕТЫ В EXCEL

УЧЕБНОЕ ПОСОБИЕ

САНКТ-ПЕТЕРБУРГ
МОСКВА
КРАСНОДАР
2014

ББК 22.18я73
В 19

Васильев А. Н.

В 19 Числовые расчеты в Excel: Учебное пособие. — СПб.: Издательство «Лань», 2014. — 608 с.: ил. — (Учебники для вузов. Специальная литература).

ISBN 978-5-8114-1580-9

Книга посвящена методам решения вычислительных задач с помощью приложения Excel. Тематика книги охватывает алгебраические уравнения и системы, интерполирование и аппроксимацию функциональных зависимостей, дифференцирование и интегрирование, решение дифференциальных и интегральных уравнений, а также некоторые другие темы из области вычислительных методов. Помимо этого, в книге описываются основные приемы работы с приложением Excel, обсуждаются способы организации рабочих документов, анализируются методы ввода и редактирования данных в рабочих документах, изучаются возможности применения форматов и стилей, иллюстрируются принципы использования встроенных вычислительных утилит, а также даются основы программирования в VBA.

Книга может использоваться в качестве учебного пособия при изучении курсов вычислительной математики и математического программирования, в качестве самоучителя или справочного пособия при решении вычислительных задач средствами приложения Excel.

ББК 22.18я73

Рецензенты:

В. Ю. РЕШЕТНЯК — доктор физико-математических наук, профессор кафедры теоретической физики физического факультета Киевского национального университета им. Т. Шевченко;

С. И. ВИЛЬЧИНСКИЙ — доктор физико-математических наук, профессор, зав. кафедрой квантовой теории поля физического факультета Киевского национального университета им. Т. Шевченко;

Д. В. АНЧИШКИН — доктор физико-математических наук, ведущий научный сотрудник отдела физики высоких плотностей энергии Института теоретической физики НАН Украины им. Н. Н. Боголюбова.

Обложка
Е. А. ВЛАСОВА

ВСТУПЛЕНИЕ

О КНИГЕ, ПРИЛОЖЕНИИ EXCEL И ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКЕ

Я в отпуске и потому работаю дома.
И вообще, мои опыты абсолютно безопасны.

Из к/ф «Иван Васильевич меняет профессию»

Книг по Excel написано очень много и самых разных. Естественным образом возникает вопрос: зачем нужна еще одна? Вопрос прямой и откровенный, поэтому требует такого же ответа. Вместе с тем, ответ будет не очень кратким, и состоит он из нескольких пунктов.

Во-первых, эта книга не совсем об Excel — точнее, об Excel, но не только: речь будет идти о решении вычислительных задач с помощью Excel. Причем задачи будем рассматривать не просто «вычислительные» — это будут те задачи, что изучаются в курсе вычислительной математики: решение алгебраических уравнений и систем, интерполирование и аппроксимация функциональных зависимостей, интегрирование и дифференцирование, решение дифференциальных уравнений.

Во-вторых, мы попытаемся сломать определенный стереотип о приложении Excel, за которым закрепилась слава «лучшего помощника бухгалтера». Конечно, никто не оспаривает действительно уникальные возможности Excel в области составления всевозможной финансовой документации, но на самом деле это только вершина айсберга — у Excel очень широкие возможности для применения в самых различных областях.

В-третьих, даже если мы задействуем готовые и хорошо проверенные алгоритмы числовых расчетов для решения той или иной задачи и при этом в качестве вычислительного средства используем Excel, адаптация этих самых алгоритмов к вычислительной среде Excel требует некоторых навыков и сноровки. Об этом тоже пойдет речь в книге.

ОСОБЕННОСТИ КНИГИ

Я бы на месте правительства всем,
кто заочно учится, ордена бы выдавала.

Из к/ф «Девчата»

Несложно сообразить, что тематика книги достаточно широка и объемна, особенно если учесть, что в ней описывается как само приложение, так и конкретные математические задачи и алгоритмы. Чтобы выдержать объем книги в разумных пределах, на вооружение был взят *принцип (закон) Парето*.

На заметку

Вильфредо Парето (1848–1923) — экономист и социолог. Карьеру начинал как инженер (закончил Политехническую школу в Турине). Автор публикаций по экономической теории и математической экономике. Является создателем теории элит. В 1893 г. — профессор политической экономики в Лозаннском университете (Швейцария). Диктатор Бенито Муссолини считал Парето своим учителем. Закон Парето лишь назван в честь ученого, а сформулирован его последователями.

Закон Парето утверждает, что *в любого рода деятельности 20% усилий приносят 80% результата, а остальные 80% усилий дают лишь 20% результата.* К написанию книг по Excel это все, на первый взгляд, не имеет никакого отношения. Но это только на первый взгляд.

Главное достоинство Excel состоит в том, что это очень «емкое» приложение. Более-менее полное его описание является задачей крайне проблематичной и малоперспективной. Поэтому самые толстые книги обычно ограничиваются хотя и относительно полным, но во многих отношениях поверхностным описанием утилит Excel. В основном такие «тяжеловесные» издания играют роль справочников. Они, безусловно, нужны, но нужны не только справочники. Хотя в процессе учебы хорошо иметь под рукой объемный справочник, учиться по нему тяжело. Учеба — это процесс. И здесь важно выделить те 20% от возможностей Excel, которые дадут 80% результата. Собственно, поиску заветных 20% и посвящена эта книга.

Книга состоит из четырех частей. В первой части (четыре главы) описываются основные приемы работы с приложением Excel. В главах первой части можно найти полезную информацию относительно того, как выглядит рабочее окно приложения, какие существуют режимы работы, как выполняются основные настройки.

Вторая часть (также четыре главы) посвящена простым вычислениям в Excel. Здесь можно найти примеры несложных прикладных вычислительных задач. Особенность ситуации в том, что задачи в этой части книги рассматриваются в контексте методов работы с приложением.

Третья часть книги состоит из трех глав, в которых описываются методы программирования в среде VBA (Visual Basic for Applications). В этой части (которая, надо сказать, не очень большая) рассматриваются такие задачи: запись макросов, составление программных кодов и создание пользовательских форм. Сведения, полученные в этой и предыдущих частях книги, будут полезными при рассмотрении прикладных вычислительных задач в четвертой части книги.

Четвертая часть состоит из четырех глав. В главах четвертой части рассматриваются задачи по решению алгебраических уравнений и систем, интерполированию и аппроксимации функциональных зависимостей, числовому дифференцированию и интегрированию (в том числе речь пойдет о вычислении несобственных и двойных интегралов), решению дифференциальных и интегральных уравнений. В общей сложности (по объему) эта часть немногим уступает трем предыдущим частям в совокупности. Поэтому в известном смысле можно утверждать, что книга состоит из двух блоков: в пер-

вом описываются методы работы с приложением, а во втором — методы решения в Excel вычислительных задач.

Материал в первых трех частях книги подбирался разнородный. Некоторые вопросы освещены достаточно поверхностно, некоторые не включены вовсе. Есть и в деталях разобранные задачи. Понятно, что во многом такой подход субъективен. Вместе с тем при написании книги разумного объема с неизбежностью приходится чем-то жертвовать.

На заметку

Одной из таких «жертв» стали сводные таблицы. Это исключительно полезная и функциональная опция в Excel. К сожалению, о сводных таблицах нужно писать много и подробно или не писать вовсе. К тому же, книга все-таки ориентирована на реализацию числовых алгоритмов, а красота и элегантность сводных таблиц проявляется скорее при обработке больших массивов данных, что несколько уводит нас от центральной темы.

В некоторых случаях жертвы были неизбежными. В некоторых — умеренно просчитанными. Генеральный план состоял в том, чтобы выделить некий «остов» из знаний и навыков в области работы с приложением Excel, дабы впоследствии читатель смог самостоятельно и без особых проблем разобраться со всеми вопросами, которые не описаны в книге и могут у него возникнуть при работе с электронными таблицами.

Что касается четвертой части книги, то она базируется на классическом курсе числовых методов, который читается для студентов естественнонаучных и инженерных специальностей университетов. Правда, в силу объективных причин некоторая строгость изложения материала нарушена. Тем не менее, минимальные (необходимые для понимания происходящего) теоретические сведения приводятся в каждой главе четвертой части. Поэтому в четвертой части книги представлена своеобразная «проекция» теории числовых расчетов в прикладную плоскость, определяемую возможностями приложения Excel.

НЕМНОГО О ПРИЛОЖЕНИИ EXCEL

Если вы нашли в подвале машину времени,
никогда не включайте ее без разрешения взрослых.

Из к/ф «Гостья из будущего»

Начиная с версии Microsoft Office 2007 все приложения этого пакета (включая и Excel) получили новый «ленточный» интерфейс. С тех пор появилось еще две версии «ленточного» приложения: Excel 2010 и Excel 2013. Большинство элементов у этих трех версий приложения сходны, однако имеются и некоторые различия. В книге за базовую выбрана последняя (на момент написания) версия приложения — Excel 2013. При этом все, что касается вычислительных задач в четвертой части, остается справедливым и для более ранних версий продукта, причем не только до версии Excel 2007, но и для более ранних версий. В общем и целом это же замечание относится ко второй и третьей части книги — с той лишь поправкой, что в «неленточных»

версиях приложения доступ к утилитам получаем не через пиктограммы ленты, а через команды меню и кнопки панелей инструментов. В плане настроек, которые описываются в первой части, конечно же, не все проходит гладко с обратной совместимостью (т. е. совместимостью с предыдущими версиями). В тех местах книги, где между Excel 2010 и Excel 2013 имеются существенные отличия, приводятся врезки с соответствующими комментариями.

О РОЛИ EXCEL ПРИ РЕШЕНИИ ВЫЧИСЛИТЕЛЬНЫХ ЗАДАЧ

Ипохондрия есть жестокое любострастие,
которое содержит дух
в непрерывном печальном положении.
Тут медицина знает разные средства,
лучшее из которых и самое безвредное — беседа.
Слово лечит, разговор мысль отгоняет.

Из к/ф «Формула любви»

Не будет ошибкой сказать, что первые три части играют вспомогательную роль и предназначены в первую очередь для тех читателей, кто не знаком или мало знаком с приложением Excel. Методы прикладного использования Excel описываются в четвертой части. Здесь следует учесть, что возможности для использования приложения Excel для проведения вычислений широчайшие и не ограничиваются чисто математическими расчетами. Приложение идеально подходит для решения общих статистических задач, выполнения регрессионного и дисперсионного анализа, проверки гипотез и многого другого. И это не учитывая «традиционной» финансово-экономической составляющей. Даже больше. Стандартные возможности приложения можно существенно расширить за счет подключаемых надстроек, среди которых особое место занимают надстройки для совместного использования Excel и Maple, а также Excel и MATLAB. Эти надстройки не входят в комплект поставки Excel (они устанавливаются вместе с соответствующими математическими пакетами), но они есть и их можно использовать. В свете сказанного возникает вопрос: к чему тогда использовать Excel при решении задач вычислительной математики, если для этих целей предназначены специальные математические и статистические пакеты? Ответов может быть несколько.

Во-первых, следует учесть, что нередко профессиональному исследователю (или студенту) необходимо создать собственный (или усовершенствовать существующий) алгоритм решения той или иной задачи, с одной стороны. С другой стороны, приложение Excel является достаточно распространенным. Поэтому при наличии соответствующих навыков работы с приложением автоматически отпадает вопрос выбора среды разработки для решения задачи. В пользу выбора Excel говорит и тот факт, что приложение обладает серьезными возможностями в плане визуализации результатов.

Во-вторых, даже «классические» алгоритмы проведения числовых расчетов не всегда просты в практическом применении. Другими словами, большое значение имеет выбор адекватной (для решения поставленной вычисли-

тельной задачи) среды разработки. Приложение Excel во многих случаях как раз является «адекватным» (хотя, конечно, и не всегда). Проверять это утверждение мы будем на задачах стандартного курса вычислительной математики (или числовых методов — названия могут варьироваться). Разумеется, весь курс охватить нам не удастся, но такая цель и не стоит. Мы проиллюстрируем возможности Excel в решении «основных» задач, которые на практике встречаются наиболее часто. В этом отношении книга может рассматриваться как учебное пособие по курсу вычислительной математики. Во всяком случае, хочется верить, что она будет полезной читателю и поспособствует пропаганде приложения Excel как эффективного вычислительного средства.

АВТОРА НА СЦЕНУ

— А мы где-то с Вами встречались!
— Ничего удивительного.
Меня ж весь Киев знает.

Из к/ф «За двумя зайцами»

Обычно начинающим политтехнологам-студентам рассказывают, что самая главная ошибка — думать, будто все остальные думают так же, как ты. Когда книга пишется, всегда есть идея, которую автор пытается реализовать. Идею бывает сложно выразить кратко, но в хорошей книге она есть. Что касается книги, которую читатель держит в руках, то уместным было бы сказать следующее: главная идея книги — это адаптация стандартных (или «классических») алгоритмов решения задач вычислительной математики для реализации в приложении, имеющем «ячеечную» структуру, т. е. структуру электронной таблицы. Разумеется, есть у книги и другие «цели» — но они в основном второстепенные. Насколько удалось реализовать данную концепцию, судить, конечно, читателю. В любом случае автор будет признателен за мнения, замечания, предложения и конструктивную критику. Все это предлагается высказывать по адресу **vasilev@univ.kiev.ua**.

На заметку

Некоторую полезную информацию (по книге и не только) можно будет найти на сайте **www.vasilev.kiev.ua**. Если читатель там для себя не найдет полезной информации, имеет смысл предложить ее там разместить. Адрес электронной почты для обратной связи приведен выше (он же есть и на сайте).

ЧАСТЬ ПЕРВАЯ

ОСНОВЫ EXCEL

ГЛАВА 1

ЗНАКОМСТВО С EXCEL

Воображаю, что сейчас будет!
Только бы не скандал.
Они так утомляют, эти скандалы.

Из к/ф «Иван Васильевич меняет профессию»

Приложение Excel представляет собой нечто среднее между офисным редактором и специальным пакетом для обработки больших массивов данных. Работать с Excel легко, удобно и приятно. Достигается это благодаря высокой степени совместимости Excel с другими офисными пакетами и специальными математическими приложениями, удобному интерфейсу, а также простой и элегантной организации рабочих документов Excel.

Приложение Excel — это электронная таблица. Данные в этой таблице разбиты по ячейкам. Данные могут быть как текстовыми, так и числовыми. Для обработки данных в Excel предлагается широкий набор встроенных функций. Важно то, что значения в ячейках могут рассчитываться на основе значений других ячеек. Все это обеспечивает неизменный успех и популярность Excel на рынке офисных программных продуктов.

На заметку

На приложении Excel, которое входит в состав пакета Microsoft Office, образно выражаясь, свет клином не сошелся. Есть и другие электронные таблицы. Тем не менее, в сегменте электронных таблиц Excel является абсолютным «законодателем моды» и конкурентов, по большому счету, не имеет.

РАБОЧЕЕ ОКНО EXCEL

— Здесь вот и будет происходить действие.
— Хорошо. А вот здесь вот что будет происходить?
— А здесь ничего не будет происходить!

Из к/ф «Старый знакомый»

После запуска приложения Excel открывается достаточно стереотипное для офисных пакетов рабочее окно. Главная особенность окна Excel состоит в том, что центральная рабочая область разграфлена в виде сетки и представляет собой массив ячеек. Именно в эти ячейки вводятся данные. На рисунке 1.1 показано пустое рабочее окно приложения версии Excel 2013.

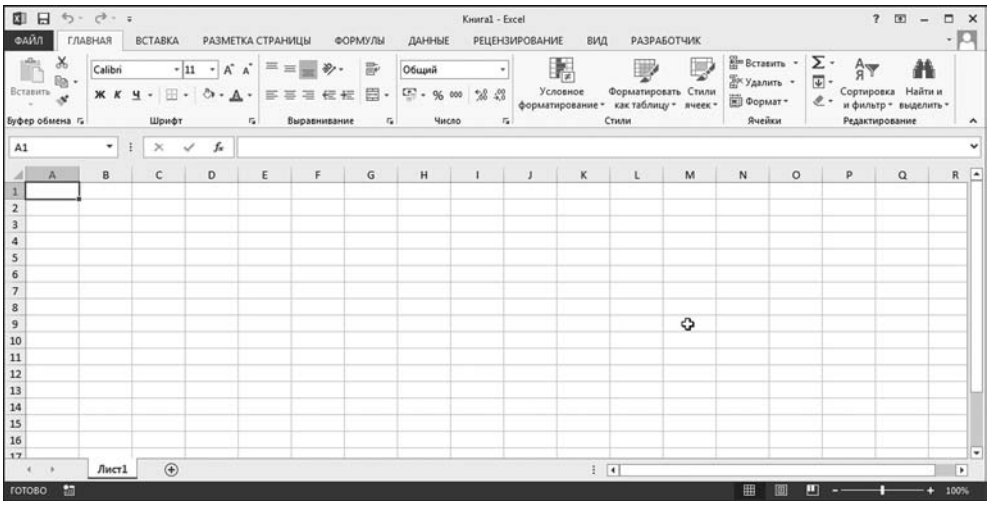


Рис. 1.1
Рабочее окно приложения Excel 2013

Если разбивка рабочей области на ячейки является стандартной для всех версий приложения Excel, то такой атрибут, как лента, появился в версии Excel 2007, был доработан в версии Excel 2010 и усовершенствован в Excel 2013. Лента размещена в верхней части окна приложения и содержит несколько вкладок с пиктограммами всевозможных команд. Фактически лента — это симбиоз панелей инструментов и панелей меню ранних версий Excel (с учетом появившихся новых возможностей). Главная причина перехода от классических панелей меню и инструментов к ленте — повышение эффективности и скорости взаимодействия пользователя и приложения. Пожалуй, можно констатировать, что эта цель достигнута. Во всяком случае, лента во многих случаях более удобна в использовании по сравнению с меню и панелями инструментов. Хотя это достаточно новый интересный элемент интерфейса приложения Excel, не он является главным. Основные действия разворачиваются в рабочей области. На ней и остановим свое внимание.

На заметку

Лента будет описываться позже и в основном по мере необходимости — в контексте того, как выполнять те или иные действия.

Ячеек в рабочей области очень много — практически бесконечное количество. Мало ввести в ту или иную ячейку значение — необходимо как-то различать эти ячейки. Наиболее оптимальный способ добиться этого состоит в адресации. Адресация ячеек подразумевает, что каждая ячейка в рабочем документе имеет свой уникальный адрес. В частности, в Excel традиционно используется принцип адресации, который состоит в том, что для ячейки указывается строка и столбец, на пересечении которых находится ячейка. По умолчанию строки нумеруются последовательностью натуральных чисел, а столбцы — латинскими буквами. Если с нумерацией строк все более-менее

ясно, то относительно нумерации (точнее, маркировки) столбцов возникает естественный вопрос: что делать, если закончатся буквы латинского алфавита? Ответ очень простой: в этом случае используются комбинации букв. Например, после столбца, маркированного буквой Z, следует столбец AA, затем AB. Если будут перебраны все комбинации из двух букв, используются комбинации из трех букв и так далее. Фактически, речь идет о нумерации столбцов, при которой в позиционном представлении «номера» столбца используются не цифры, а латинские литеры.

На заметку

Один рабочий лист (а их в рабочей книге по умолчанию обычно три) содержит огромное количество ячеек — более 17 млрд. Понятно, что задействовать все эти ячейки в вычислениях просто нереально, причем в силу самых разных причин. Для нас важно знать, что ячеек в документе так много, что на фактор ограниченности их количества можно не обращать внимания. Тем не менее, заметим, что последняя строка имеет номер **1048576**, а последний столбец имеет название **XFD** (всего **16384** столбца).

Чтобы однозначно определить ячейку, необходимо и достаточно указать ее адрес, который традиционно формируется объединением «номера» (точнее, имени) столбца и номера строки. Например, ячейка B3 находится на пересечении столбца, помеченного литерой B и строки с номером 3. Для удобства пользователя, опять же, по умолчанию, номера строк отображаются в вертикальной полосе вдоль левой границы рабочей области (полоса заголовков строк), а названия столбцов (полоса заголовков столбцов) — в горизонтальной полосе в верхней части рабочей области (рис. 1.1).

На самом деле рабочее окно приложения Excel содержит не одну, а несколько рабочих областей, которые называются *листами*. В нижней части рабочего окна приложения находятся вкладки (корешки) листов рабочего документа Excel. Сам рабочий документ называется книгой (или рабочей книгой). Чтобы выделить тот или иной рабочий лист, достаточно щелкнуть по корешку этого листа. На рисунке 1.2 представлены корешки рабочих листов книги Excel — их может быть разное количество (обычно при создании новой рабочей книги отображается три корешка, хотя соответствующие настройки могут быть изменены).

Рис. 1.2
Корешки рабочих листов книги Excel

Рабочие листы не являются обособленными компонентами окна приложения. Между ячейками листов можно задавать функциональные зависимости. Но прежде, чем рассматривать методы работы с несколькими листами в одной книге имеет смысл обсудить базовые операции, которые выполняются с ячейками и диапазонами ячеек в области одного листа.

По большому счету, при работе непосредственно с ячейками рабочего документа речь может идти о вводе данных или об их редактировании. Сюда же относится и форматирование уже введенных в документ данных — применение специального способа отображения содержимого в ячейке (отлич-

ного от того, что используется по умолчанию). Принципиальное значение имеет тип данных, которые введены в ячейку документа. Условно данные можно разделить на три группы:

- текст;
- числа;
- формулы.

На заметку

Нередко отдельно выделяют логические значения (ИСТИНА и ЛОЖЬ), которые используются при проверке условий и вычислении логических выражений. Также некоторое обособленное место занимают данные типа дата/время. Хотя формально это числовые данные, к которым применяется специальный тип форматирования, «логика» их обработки достаточно специфична.

Если относительно текста и чисел особых вопросов не возникает, то *формулы* как тип данных требуют пояснений. Именно формулы делают Excel уникальным вычислительным приложением. Функциональное связывание ячеек в документе происходит фактически с помощью формул.

В ячейке рабочей области значение может быть определено пользователем непосредственно при вводе (это числа и текст) или вычислено на основе значений в других ячейках. В последнем случае по определенным правилам в ячейку вводится формула, которая, как правило, содержит ссылки на другие ячейки (адреса этих ячеек). Если значения ячеек, на которые ссылается ячейка с формулой, меняются, автоматически пересчитывается и формула (во всяком случае, такой режим используется по умолчанию). При этом ячейки с формулами могут содержать ссылки на ячейки как с числами и текстом, так и с формулами. Неудивительно поэтому, что рабочие документы Excel могут иметь довольно сложную структуру на уровне функциональных зависимостей ячеек.

ВВОД ДАННЫХ В ЯЧЕЙКИ

Ну, пошли дела кое-как.
Что же это изобретатель
свою машину времени назад не крутит?

Из к/ф «Иван Васильевич меняет профессию»

Существует два способа ввода данных в ячейки электронной таблицы: непосредственно в выделенную ячейку или в строку формул (рис. 1.3).

И в том, и в другом случае предварительно выделяется ячейка, в которую вводится значение. Для выделения на ячейке выполняют щелчок мышью либо перемещают рамку выделения ячейки в нужное место рабочего документа с помощью клавиш со стрелками. Также можно воспользоваться полем названий (белое поле с адресом ячейки в левом верхнем углу рабочей области документа — слева от строки формул).

Рис. 1.3
Строка формул

В поле названия можно вести адрес ячейки и нажать клавишу <Enter>, в результате чего эта ячейка выделится (данный способ выделения ячеек удобен при работе с большими документами). Далее осуществляется ввод значения в ячейку. При этом автоматически выполняется переход в режим ввода значения в ячейку: в области выделенной ячейки мигает курсор ввода (рис. 1.4).

В строке формул при вводе значения в ячейку также отображается это значение. Слева от строки формул в поле названия отображается имя (адрес) активной на данный момент ячейки. Для завершения ввода достаточно нажать клавишу <Enter>, клавишу со стрелкой или щелкнуть мышью на другой ячейке.

Рис. 1.4
Ввод значения в ячейку

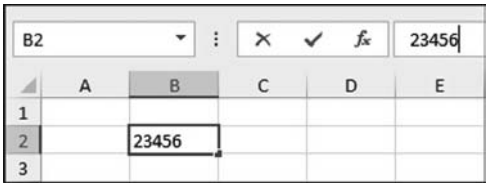


Рис. 1.5
Ввод значения через строку формул

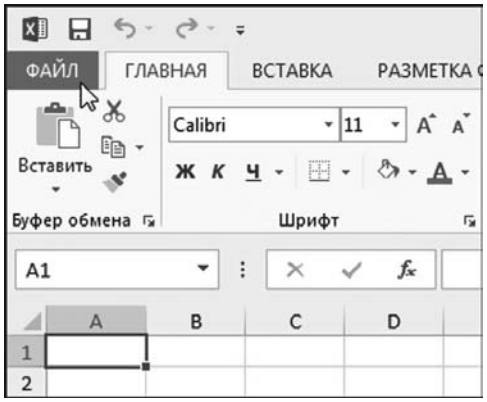


Рис. 1.6
Щелкаем ярлычок вкладки **Файл**

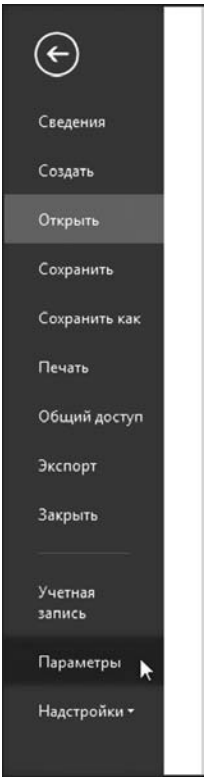


Рис. 1.7
В списке разделов вкладки файл выбираем позицию **Параметры**

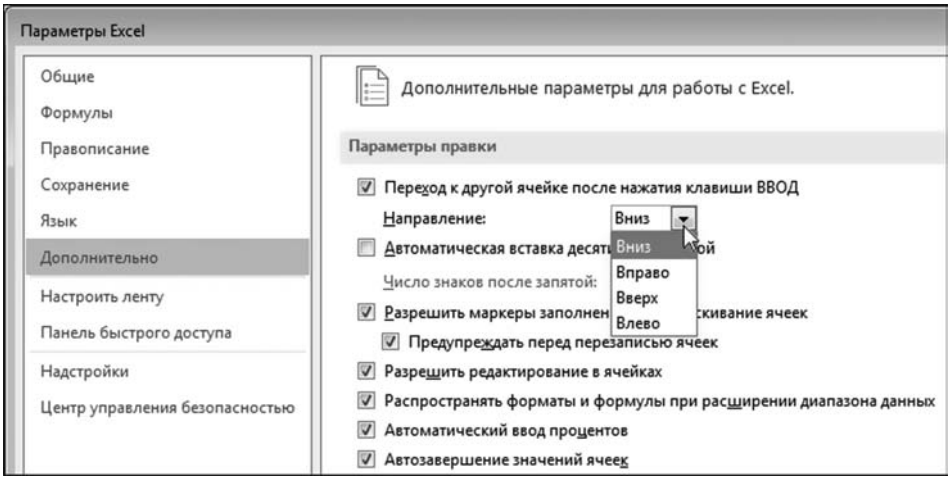


Рис. 1.8
В окне **Параметры Excel** в разделе **Дополнительно** выполняем настройки в группе **Параметры вставки**

Для ввода значения через строку формул сначала выделяют правильную ячейку, после чего выполняют щелчок мышью в строке формул и выполняют ввод значения (рис. 1.5).

Ситуация зеркальная к предыдущему случаю: в строке формул мигает курсор ввода, а в выделенной ячейке синхронно отображается вводимое в строку формул значение. После нажатия клавиши <Enter> ввод значения завершается, а активной становится ячейка под той, куда вводилось значение.

На заметку

В Excel по умолчанию используется режим, при котором нажатие клавиши <Enter> приводит к выделению ячейки снизу. Но такое поведение приложения можно откорректировать. Делается это в окне настроек приложения **Параметры Excel**. Чтобы открыть окно, например, щелкаем ярлычок вкладки **Файл** (рис. 1.6).
На этой вкладке находим команду **Параметры** (рис. 1.7).
Открывается диалоговое окно **Параметры Excel**, которое необходимо открыть в разделе **Дополнительно** (рис. 1.8).
В этом случае в окне появляется группа утилит управления **Параметры правки**. В частности, интерес представляет опция **Переход к другой ячейке после нажатия клавиши ВВОД** и раскрывающийся список **Направление**.

Имеет свои особенности процесс редактирования ячеек, т. е. внесение изменение в ячейки, которые уже содержат значения. Дело в том, что если просто выделить ячейку и начать ввод, то старое значение ячейки автоматически удаляется. Это не всегда удобно. Для редактирования, а не замены значения ячейки, ячейка выделяется, после чего нажимают клавишу <F2>. Можно также выполнить двойной щелчок мышью на редактируемой ячейке. Еще один способ редактирования ячейки — выделить ее и перейти к строке формул.

На заметку

Описанный выше способ редактирования данных в ячейках доступен по умолчанию. Но, откровенно говоря, в зависимости от настроек приложения к ячейке с данными можно и не добраться. За режим редактирования данных в ячейке отвечает опция **Разрешить редактирование в ячейках** в разделе **Дополнительно** окна настроек **Параметры Excel** (рис. 1.8). Стоит убрать флажок у этой опции, и редактировать значения ячеек в самих ячейках уже не получится — только в строке формул.

ДИАПАЗОНЫ ЯЧЕЕК

Как вы знаете, родные мои, у нас по проекту 33 бокса.
Поэтому у нас 32 пайщика. Один гараж,
как вы знаете, предназначен для ремонта.

Из к/ф «Гараж»

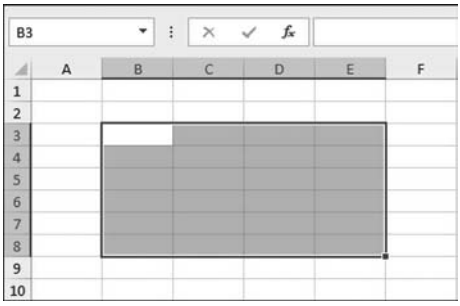


Рис. 1.9
Выделен диапазон ячеек

Операции в рабочем документе можно выполнять не только с отдельными ячейками, но и с диапазонами ячеек, т. е. несколькими ячейками одновременно. Общее правило состоит в том, что для выполнения тех или иных действий с диапазоном ячеек этот диапазон в рабочей области необходимо както обозначить или выделить. Самый простой способ — удерживая нажатой левую кнопку мыши, захватить нужный диапазон ячеек, как, например, это показано на рисунке 1.9.

Адресация (именование) диапазона ячеек выполняется по следующему принципу: сначала указывается адрес левой верхней ячейки диапазона, после чего, через двоеточие, указывается адрес правой нижней ячейки диапазона. На рисунке 1.9 выделен диапазон ячеек B3:E8. Отметим, что при выделении диапазона ячеек среди всех ячеек диапазона имеется одна активная. Она выделена в диапазоне ячеек белым цветом. В строке названий отображается ее адрес. Если нажать клавишу <Enter>, активной станет следующая ячейка, расположенная снизу под текущей. Последовательно нажимая клавишу <Enter>, можно перебрать все ячейки диапазона. Ячейки перебираются по столбикам: после нижней ячейки в столбике активной становится верхняя ячейка следующего столбика. В таком режиме можно заполнять диапазон ячеек значениями.

Можно выделить сразу несколько диапазонов. Для этого при выделении каждого следующего диапазона необходимо держать нажатой клавишу <Ctrl>. На рисунке 1.10 показан документ, в котором выделено одновременно несколько диапазонов ячеек.

Если во все ячейки диапазона необходимо ввести одинаковые значения, достаточно в активную ячейку диапазона ввести это значение и нажать ком-

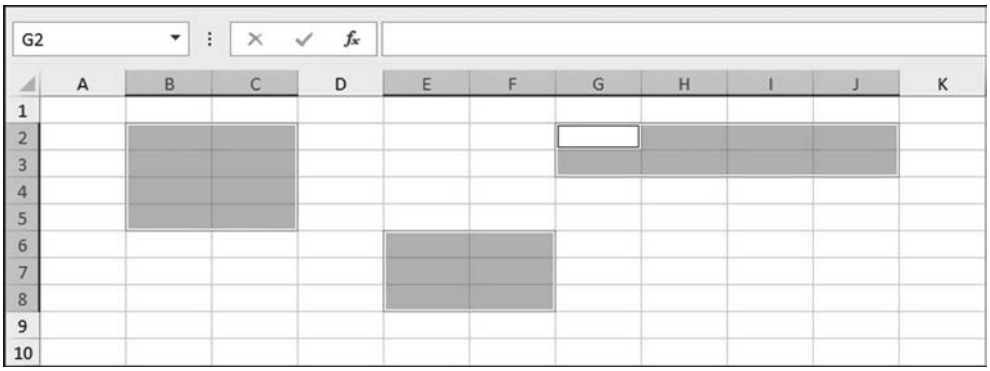


Рис. 1.10
Выделено несколько диапазонов ячеек

бинацию клавиш <Ctrl>+<Enter>. Результат заполнения диапазона ячеек одинаковыми значениями приведен на рисунке 1.11.

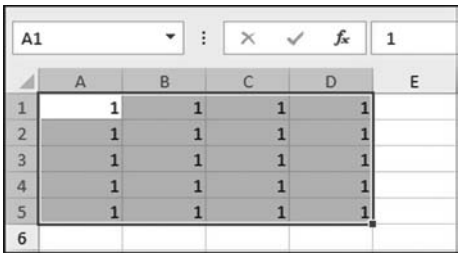


Рис. 1.11
Все ячейки диапазона заполнены одинаковыми значениями

Нередко возникает необходимость выделить всю строку или столбец. Такая операция выполняется очень просто. Для выделения строки наводим курсор мыши на ленту нумерации строк (в соответствующей позиции с номером выделяемой строки). Курсор мыши при этом приобретает вид стрелки, направленной вдоль строки. Щелчок левой кнопкой мыши приводит к выделению строки. Так же выделяются столбцы, с той лишь разницей, что курсор мыши наводится на ленту именования столбцов. Наконец, чтобы выделить весь рабочий лист, наводим курсор в область пересечения лент именования столбцов и индексации строк и нажимаем левую кнопку мыши.

ФОРМУЛЫ

И еще один маленький,
но довольно-таки большой вопрос.

Из к/ф «Гараж»

Формула в Excel — это специального вида выражение, с помощью которого вычисляется значение ячейки, содержащей это выражение, т. е. формулу. Как отмечалось выше, формулы обычно содержат ссылки на другие ячейки. Ссылка на ячейку — это ее адрес. Кроме адресов ячеек, в формулах могут использоваться знаки арифметических операций (таких, как сложение +, вычитание -, умножение *, деление / и возведение в степень ^), а также функции (встроенные функции Excel и функции, созданные пользователем

в VBA). Функции рассмотрим позже, а здесь остановимся на простых формулах, содержащих только арифметические операции.

Любая формула Excel, и простая, и сложная, начинается со знака равенства (т. е. =). Именно знак равенства является индикатором того, что в ячейку введена формула, а не обычный текст. Во всем остальном процесс ввода формулы в ячейку аналогичен вводу в ячейку числа или текста. Одно важное отличие касается отображения значений ячеек с формулами. Дело в том, что по умолчанию если в ячейку введена формула, то отображается значение, которое рассчитано по этой формуле.

На заметку

Опять же, этот режим используется по умолчанию. Однако есть режим, при котором в ячейках с формулами отображаются формулы. Любители острых ощущений могут поискать в окне настроек **Параметры Excel** в разделе **Дополнительно** опцию **Показывать формулы, а не их значения**.

Чтобы увидеть формулу, необходимо выделить ячейку — формула отображается в строке формул. Можно также перейти в режим редактирования ячейки. На рисунке 1.12 приведен фрагмент рабочего документа, в котором есть ячейка, содержащая формулу.

В данном случае в ячейку B2 введена формула =A2*A1-B1. Знак равенства, напомним, является опознавательным индикатором формулы. Само выражение после знака равенства содержит адреса ячеек A1, A2 и B1, а также оператор умножения * и вычитания -, и означает, что значение ячейки A1 необходимо умножить на значение ячейки A2 и из полученного значения вычесть значение ячейки B1. Учитывая, что в ячейки A1, A2 и B1 введены соответственно числовые значения 3, 5 и 4, в результате для значения ячейки B2 получаем значение 11, которое и отображается в ячейке. При этом в строке формул (при выделенной ячейке B2) отображается формула =A2*A1-B1.

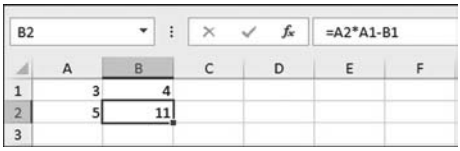


Рис. 1.12
Ячейка с формулой

Красота формул Excel проявляется в динамике. Достаточно изменить значение хотя бы одной из ячеек, на которые содержится ссылка в формуле, как автоматически будет пересчитано и значение ячейки с формулой. На рисунке 1.13 показан тот же документ, что и ранее, но с измененным значением ячейки A1 (теперь в эту ячейку введено значение 10).

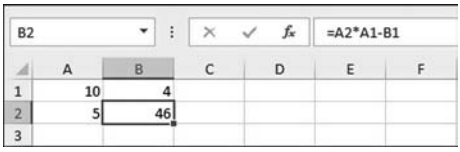


Рис. 1.13
При изменении значения ячеек документа формулы пересчитываются автоматически

При вводе формулы для выполнения ссылки на ту или иную ячейку нет необходимости вводить ее адрес вручную. Достаточно щелкнуть на этой ячейке мышью. Адрес ячейки будет автоматически вставлен в формулу в месте размещения курсора ввода. Важно только помнить, что это должна быть

формула, т. е. выражение, которое начинается со знака равенства. Если знак равенства не указать, вводимое значение воспринимается как текст (или число — в зависимости от контекста вводимого значения) и щелчок на другой ячейке в этом случае приведет к тому, что данная ячейка станет активной.

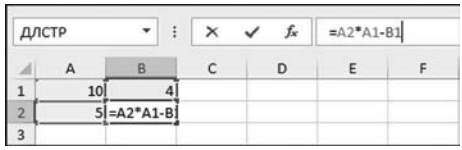


Рис. 1.14
При редактировании ячейки с формулой автоматически выделяются ячейки, на которые есть ссылки в формуле

Если перейти в режим редактирования ячейки, содержащей формулу, автоматически те ячейки, на которые в формуле есть ссылки, будут выделены цветными рамками, как показано на рисунке 1.14.

Это достаточно удобно, особенно при отслеживании ошибок в документе. Перейдя в режим редактирования формулы (клавиша <F2> при выделенной ячейке с формулой или щелчок в строке формул), легче проконтролировать значения, на основе которых вычисляется формула.

На заметку

При этом поле названия (поле имен), которое находится слева от строки формул, содержит имя последней использовавшейся функции Excel. Также отметим, что приложение Excel имеет специальные встроенные утилиты для поиска и устранения ошибок в рабочем документе. Они тоже будут кратко описаны в книге.

Наиболее интересные свойства формул вообще и ссылок в формулах в частности связаны с их копированием. Чтобы скопировать значение ячейки, в том числе и ячейки с формулой, эта ячейка выделяется, и выполняется щелчок на пиктограмме копирования (вкладка **Главная** на ленте приложения, группа **Буфер обмена**), или используется комбинация клавиш <Ctrl>+<C>. На рисунке 1.15 в буфер обмена копируется содержимое ячейки B2.

После того как значение скопировано, в рабочем документе выделяется ячейка для вставки этого значения. На рисунке 1.16 выделена ячейка C3, после чего выполняется щелчок на пиктограмме вставки содержимого буфера

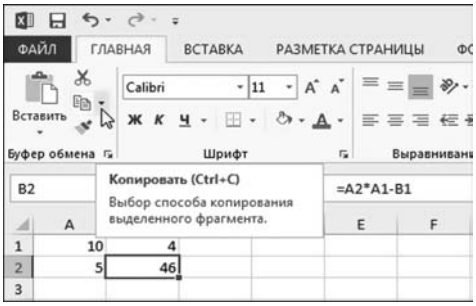


Рис. 1.15
Копирование значения ячейки с формулой в буфер обмена

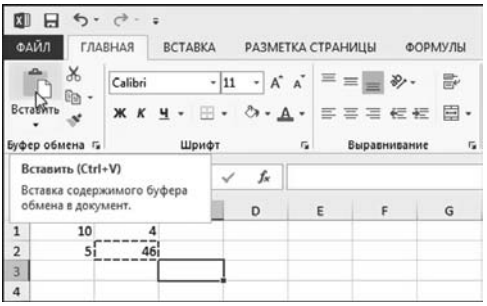


Рис. 1.16
Вставка содержимого буфера обмена в ячейку

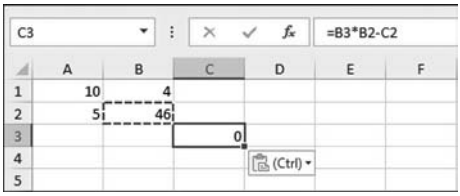


Рис. 1.17
Результат вставки формулы в ячейку — ссылки на ячейки изменились

обмена (группа **Буфер обмена** вкладки **Главная**) или используется комбинация клавиш <Ctrl>+<V>.

Результат копирования и вставки формулы представлен на рисунке 1.17.

Интерес в данном случае представляет не столько непосредственно числовое значение, отображаемое в ячейке C3, а формула, по которой оно было вычислено. В частности, ячейка C3 содержит формулу =B3*B2-C2, хотя ячейка B2, из которой эта формула копировалась, содержит формулу =A2*A1-B1. Другими словами, при копировании формула изменилась. Точнее не вся формула, а только входящие в эту формулу ссылки на ячейки. Объяснение достаточно простое — по умолчанию ссылки в формулах рассматриваются как *относительные*. Относительность ссылок означает, что при копировании они изменяются так, чтобы относительное положение ячейки с формулой и ячейки, на которую в этой формуле есть ссылка, не менялось. Например, в формуле =A2*A1-B1 в ячейке B2 есть, кроме прочего, ссылка на ячейку A1. Эта ячейка расположена на один столбец влево и на одну строку вверх по отношению к ячейке с формулой, т. е. к ячейке B2. Поэтому при копировании формулы из ячейки B2 в ячейку C3 ссылка A1 меняется на ссылку B2. Ячейка B2 находится по отношению к ячейке C3 на один столбец влево и на одну строку вверх. Если бы в копируемой формуле было несколько ссылок на ячейку A1, все бы они изменились на B2. Аналогично, ячейка A2 расположена в той же строке, что и ячейка B2, но на один столбец левее. При копировании формулы в ячейку C3 ссылка A2 меняется на ссылку B3, поскольку взаимное расположение ссылок C3 и B3 такое же, как ссылок B2 и A2. Ссылка B1 меняется на ссылку C2. Обращаем внимание читателя, что способ изменения ссылок зависит от того, в какую ячейку копируется формула. При копировании одной и той же формулы в разные ячейки ссылки меняются по-разному.

Что касается вычисленного по скопированной формуле нулевого значения в ячейке C3, то здесь следует учесть, что по умолчанию пустые ячейки интерпретируются как ячейки, содержащие нулевые значения. Скопированная в ячейку C3 формула содержит ссылки на две пустые ячейки: B3 и C2.

На первый взгляд может показаться, что такой способ копирования ссылок в формулах очень неудобен, хотя на самом деле все как раз наоборот. В Excel существует режим автоматического заполнения ячеек (он будет описан чуть позже), который позволяет быстро и удобно заполнять копированием целые диапазоны ячеек. Использование относительных ссылок в формулах позволяет в этом случае создавать четко структурированные документы. Также кроме относительных ссылок, в формулах могут использоваться и абсолютные ссылки. Таким образом, интрига нарастает. А может, и нет — это как посмотреть.

АБСОЛЮТНЫЕ И ОТНОСИТЕЛЬНЫЕ ССЫЛКИ

Передайте Зинаиде Михайловне,
что Розалия Францевна говорила,
что Анне Ивановне Капитолина Никифоровна
дубленку предлагает.

Из к/ф «Иван Васильевич меняет профессию»

Абсолютная ссылка на ячейку отличается от относительной ссылки на ячейку тем, что при копировании формулы абсолютная ссылка не меняется. Следовательно, если формула содержит абсолютную ссылку на ячейку, находящуюся, например, на пересечении второго столбца и третьей строки (это ячейка B3), то куда бы формула ни была скопирована, в новой ячейке ссылка будет указывать на ту же самую ячейку (в данном случае на ячейку, находящуюся на пересечении второго столбца и третьей строки, т. е. ячейку B3). Выше отмечалось, что по умолчанию ссылки являются относительными. Другими словами, если просто указать адрес ячейки, то это будет относительная ссылка и при копировании формулы она изменится. Для того чтобы сделать ссылку абсолютной, в имени ячейки используют символ доллара $. Если знак доллара указать в имени ячейки и перед именем столбца, и перед номером строки, то это будет абсолютная ссылка (например, $A$1 является абсолютной ссылкой на ячейку A1). Можно выполнять так называемые *смешанные* ссылки, когда ссылка является абсолютной только по номеру строки или по имени столбца. В этом случае знак доллара указывается соответственно перед номером строки или перед именем столбца в названии ячейки. Если ссылка абсолютная по строке, то при копировании формулы с такой ссылкой меняется столбец, но неизменной остается строка. Напротив, если ссылка абсолютная по столбцу, то при копировании формулы с этой ссылкой неизменным остается столбец, а строка меняется. Например, ячейка B3 содержит формулу со ссылкой A$2. Это смешанная ссылка. Она является относительной по столбцу и абсолютной по строке. При копировании формулы, скажем, в ячейку D5, ссылка A$2 изменится на C$2. Поскольку по строке ссылка абсолютная, номер строки не меняется. По столбцу ссылка относительная, поэтому имя столбца в ссылке при копировании меняется. При этом количество столбцов между ячейками B3 и A2 такое же, как между ячейками D5 и C2.

Если из ячейки B3 формула, содержащая ссылку $A2, копируется в ячейку D5, ссылка изменится на $A4. Смешанная ссылка $A2 является абсолютной по столбцу и относительной по строке. Поэтому при копировании столбец не меняется, меняется только номер строки в ссылке. Количество строк между ячейками B3 и $A2 такое же, как между ячейками D5 и $A4.

На заметку

При вводе абсолютных или смешанных ссылок знак доллара может вводиться с клавиатуры. Однако проще воспользоваться клавишей <F4>. В режиме редактирования формулы при наведенном на адрес ячейки курсоре последовательное нажатие этой клавиши приводит к тому, что ссылка становится абсолютной (по строке и столбцу), смешанной (сначала абсолютной по строке, а затем абсолютной по столбцу), и снова относительной.

ЛЕНТА ПРИЛОЖЕНИЯ

— Знаете, я убежден, что Вы девушка со вкусом
и способны чувствовать тонко,
и в то же время масштабно.
— Вы думаете?
— Убежден!

Из к/ф «Старый знакомый»

Начиная с версии Microsoft Office 2007 интерфейс всех приложений, входящих в пакет, в том числе и Excel, претерпел серьезные изменения. Основу этих изменений составляет относительно новый элемент, который получил название лента. Лента представляет собой нечто среднее между традиционной панелью меню и панелью инструментов. Располагается она в верхней части окна документа, содержит несколько вкладок, на каждой из которых представлены пиктограммы команд (рис. 1.18).

В отличие от версии Excel 2007, в версии Excel 2010 и Excel 2013 ленту можно настраивать. Другими словами, в Excel 2013 пользователь может по своему усмотрению (до известных пределов, разумеется) определять, какие вкладки должны быть представлены на ленте. Особенностям интерфейса приложения Excel 2013, в том числе и методам работы с лентой, посвящена следующая глава книги. Здесь лишь остановимся на некоторых особенностях этого достаточно нового и интересного элемента интерфейса.

Для выполнения той или иной команды необходимо сначала определиться с тем, на какой вкладке ленты представлена пиктограмма для этой команды. Хорошо, если пользователь помнит, где на ленте представлена какая пиктограмма. Но даже если это не так, практически всегда можно догадаться — названия вкладок ленты достаточно точно соответствуют назначению команд на вкладках. Например, вкладка **Главная** содержит в основном команды, предназначенные для применения и создания стилей, форматов, работы с буфером обмена, поиском данных в документе — это те основные задачи, с которыми обычно приходится иметь дело пользователю. На вкладке **Формулы** представлены команды для работы с формулами и функциями, а также утилиты отслеживания ошибок в формулах. Вкладка **Разработчик** содержит утилиты, полезные при создании макросов и функций пользователя, для вызова редактора VBA (сокращение от *Visual Basic for Applications*). Чтобы выбрать ту или иную вкладку, достаточно щелкнуть на ее корешке.

Команды на каждой вкладке разбиты по группам. Каждая группа имеет название, которое отображается внизу в области группы. Некоторые группы

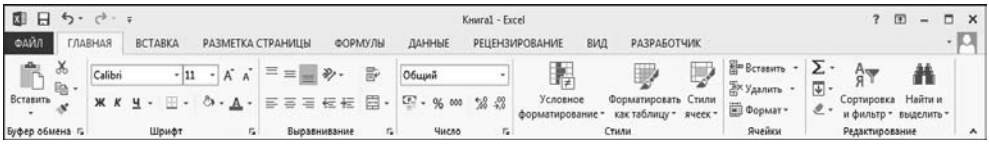


Рис. 1.18
Лента приложения Excel открыта на вкладке **Главная**

состоят всего из одной пиктограммы (как правило, это пиктограмма — раскрывающийся список). Некоторые группы в правом нижнем углу содержат активную метку, щелчок на которой приводит, в зависимости от группы, к раскрытию окна настройки параметров. Назначение большинства из этих окон описывается в следующих главах книги.

ОСНОВНЫЕ ОПЕРАЦИИ С ПРИЛОЖЕНИЕМ

Нептуна окружают русалки.
Это наши советские девушки — только с хвостами.
Они выполняют всяческие физкультурные упражнения.

Из к/ф «Старый знакомый»

Среди всех вкладок ленты одна стоит особняком. Это вкладка **Файл**. После щелчка на корешке этой вкладки открывается окно, представленное на рисунке 1.19.

Вкладка **Файл** отличается от прочих вкладок ленты и внешне, и по назначению. На ней собраны команды и меню для создания, сохранения, открытия и закрытия документов, вывода их на печать, выполнения базовых настроек всего приложения и завершения работы с приложением. Особый интерес представляет пиктограмма **Параметры**, щелчок на которой приводит к открытию окна настройки параметров приложения Excel. Настройки приложения мы будем обсуждать небольшими порциями и только в случае крайней необходимости.

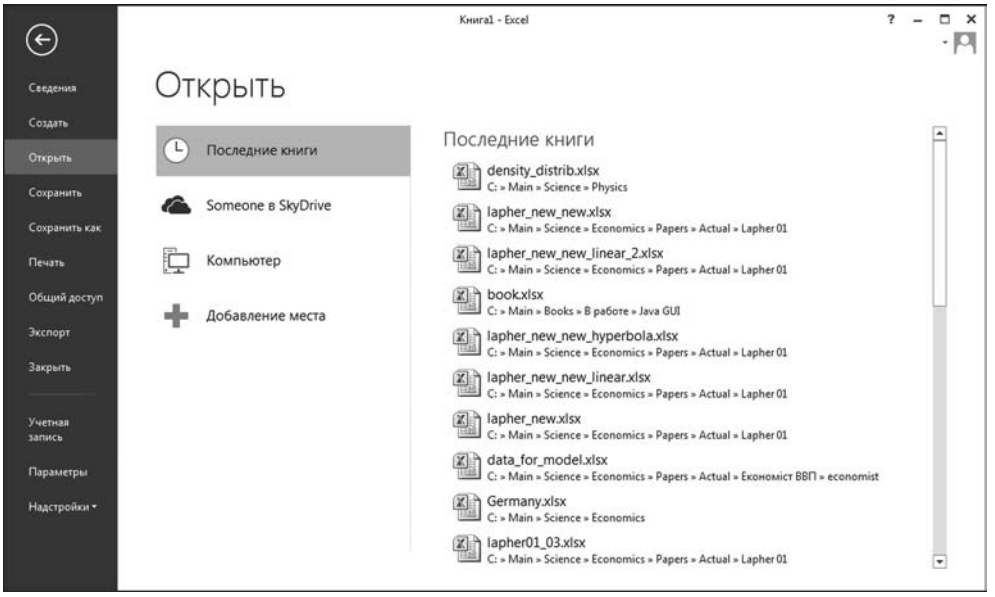


Рис. 1.19
Содержимое вкладки **Файл**

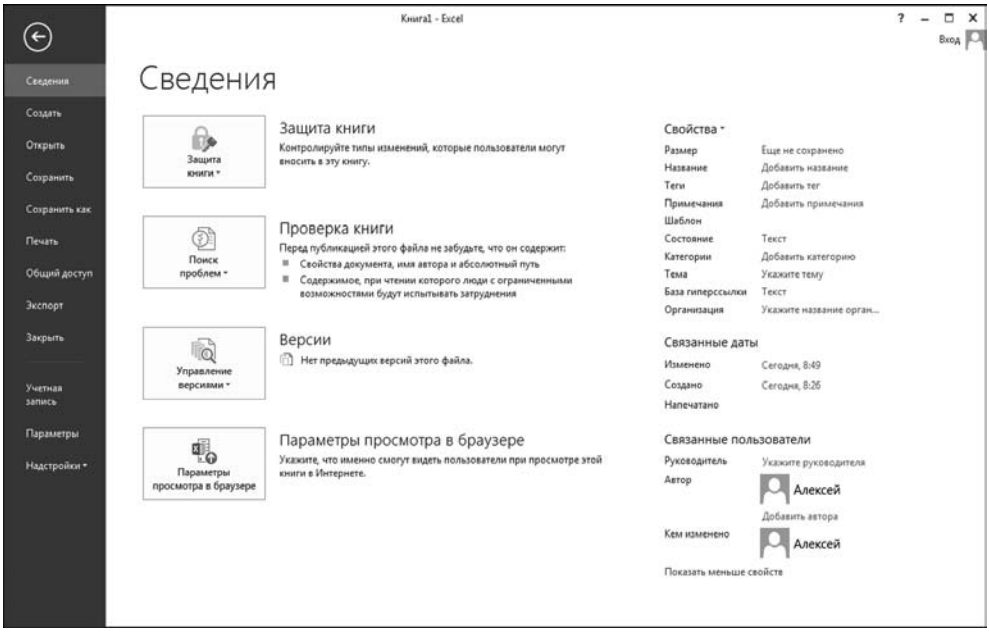


Рис. 1.20
Вкладка **Файл** открыта в разделе **Сведения**

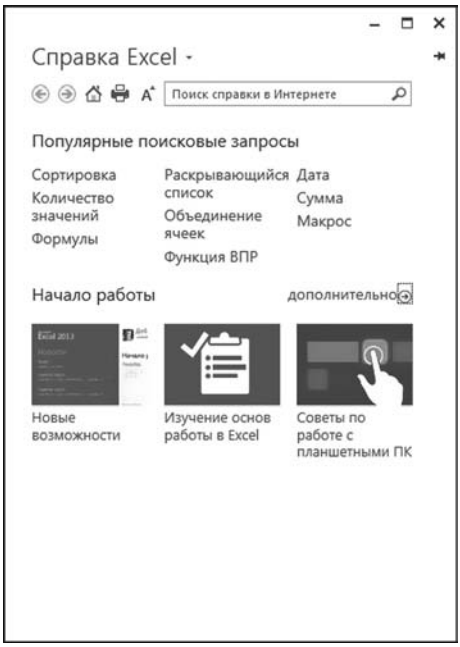


Рис. 1.21
Окно справки Excel

Некоторая полезная информация о документе может быть получена через пиктограмму **Сведения** вкладки **Файл** (рис. 1.20).

Разумеется, в Excel существует достаточно мощная справочная система, которая позволяет искать нужную информацию как на компьютере пользователя, так и в сети. На рисунке 1.21 представлено окно справки Excel.

В поле поиска водится ключевое слово или фраза. Можно также воспользоваться набором пиктограмм и гиперссылок для получения тематической справки по тому или иному вопросу. Открыть окно справочной системы просто — достаточно щелкнуть на специальной пиктограмме с вопросительным знаком в правом верхнем углу рабочего окна документа (рис. 1.22).

Пиктограмма **Создать** на вкладке **Файл** предназначена для создания нового документа. На рисунке 1.23 пока-

зано окно вкладки **Файл**, открытое в разделе **Создать**.

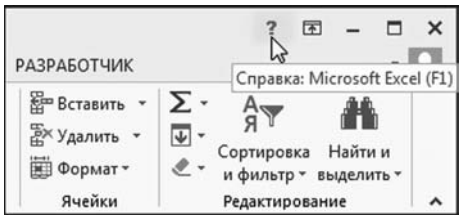


Рис. 1.22
Щелчок на пиктограмме приводит к открытию окна справки приложения

С помощью пиктограмм раздела можно создавать как пустые рабочие книги, так и документы на основе шаблонов и уже существующих книг. В частности, для создания новой пустой книги выбираем пиктограмму **Пустая книга**. В результате создается новая книга, параметры которой определяются глобальными настройками приложения. Наконец, для открытия, закрытия и сохранения документа на вкладке **Файл** предназначены соответственно пиктограммы **Открыть**, **Закрыть** и **Сохранить**. Также для сохранения документа может использоваться пиктограмма **Сохранить как**. В этом случае можно выбрать формат сохраняемого файла.

Вместе с тем, для сохранения документа удобнее использовать пиктограммы *панели быстрого доступа*. Эта панель по умолчанию расположена в

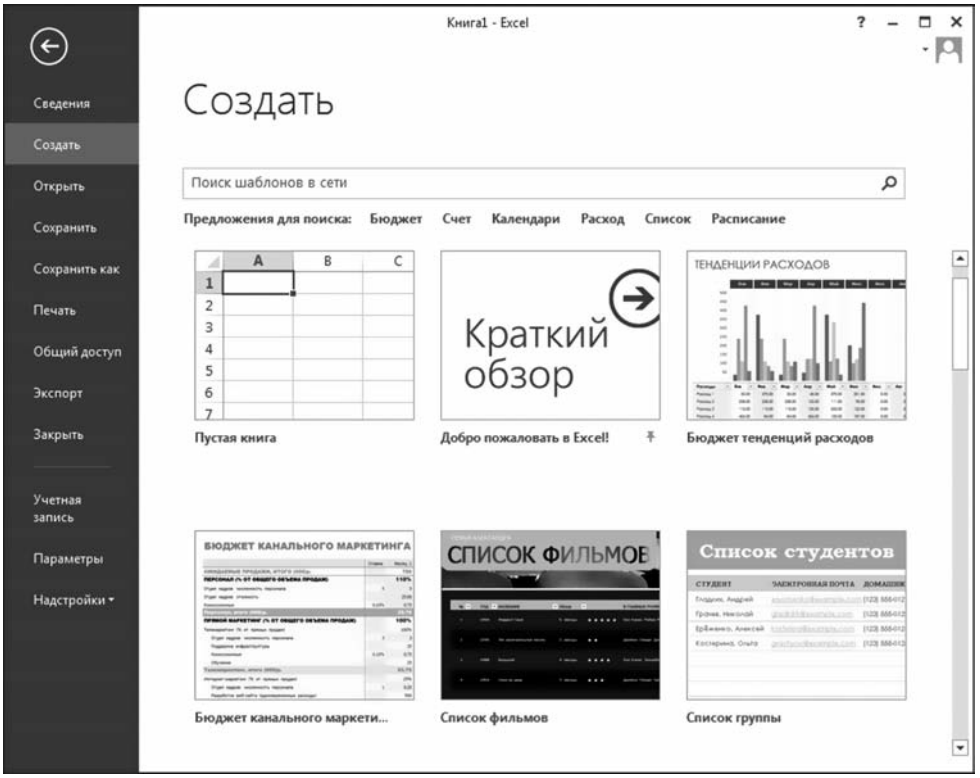


Рис. 1.23
Вкладка **Файл** открыта в разделе **Создать**

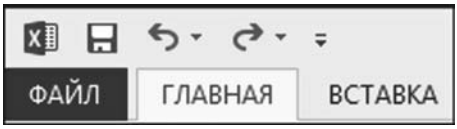


Рис. 1.24
Панель быстрого доступа

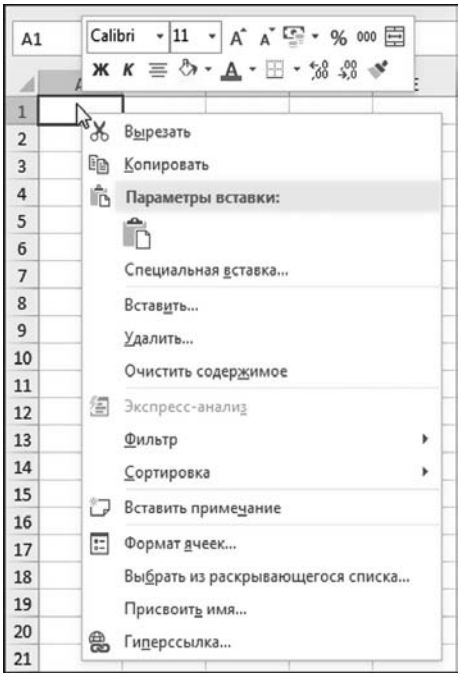


Рис. 1.25
Контекстное меню
для рабочей области
электронной таблицы

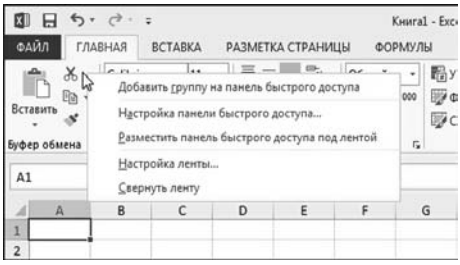


Рис. 1.26
Контекстное меню
для группы кнопок ленты

верхней части окна приложения Excel и содержит, опять же, по умолчанию, три кнопки: для сохранения документа, отмены последнего действия и его повторения (рис. 1.24).

Панель быстрого доступа настраивается, т. е. на нее можно добавлять и прочие кнопки (и даже целые группы кнопок). Панель быстрого доступа можно настраивать как для отдельного документа, так и для приложения в целом. Обычно на панель быстрого доступа выносятся кнопки для базовых команд, таких как создание нового документа, открытие документа, вывод документа на печать. Кроме этого нередко на панель быстрого доступа добавляются пиктограммы для запуска макросов пользователя.

Еще один достаточно быстрый и удобный способ взаимодействия пользователя с приложением — воспользоваться контекстным меню. Контекстное меню раскрывается при щелчке правой кнопкой мыши в рабочей области документа или на элементе управления приложения. На рисунке 1.25 показано окно контекстного меню для рабочей области документа — окно открывается при щелчке правой кнопкой мыши на ячейке электронной таблицы.

В этом контекстном меню содержатся команды и меню для выполнения основных действий по форматированию данных. Вообще же содержимое контекстного меню существенно зависит от элемента, для которого оно вызывается. На рисунке 1.26 представлено контекстное меню для группы кнопок ленты.

Команд в этом случае немного, и они предназначены для перехода в режим настройки ленты, изменения положения панели быстрого доступа и добавления группы кнопок на панель быстрого доступа. Для любого элемента

управления приложения или объекта таблицы контекстное меню содержит основные команды, допустимые при работе с этим элементом или объектом.

ФОРМАТЫ ФАЙЛОВ

Эврика! Царские шмотки.
Одевайся — царем будешь.

Из к/ф «Иван Васильевич меняет профессию»

При работе с приложением Excel желательно иметь хотя бы общее представление о типах и, соответственно, форматах файлов, с которыми придется иметь дело.

Во-первых, стандартным расширением для книг версии Excel 2013 является xlsx. Если в рабочей книге имеются макросы, она будет иметь расширение xlsm. Еще два популярных расширения — это xltx и xltm. Первое расширение имеют файлы шаблонов. Это специального типа документы, которые служат основой для создания рабочих книг Excel. Обычно шаблон содержит базовое форматирование и элементы оформления для будущего документа. Если шаблон содержит макросы, у файла такого шаблона будет расширение xltm.

К сожалению или к счастью, среди пользователей приложения Excel достаточно много консерваторов, которые не спешат переходить на новые версии приложения. Чтобы рабочий документ был совместим с предыдущими версиями Excel (версии до Excel 2007) его сохраняют в специальном режиме совместимости, и в этом случае файл рабочего документа имеет расширение xls. Такое же расширение имеют файлы рабочих книг Excel, которые создаются старыми версиями приложения (до версии 2007 г.). Хотя с одной стороны обратная совместимость — штука удобная, при сохранении документов в формате xls некоторые свойства документа не могут поддерживаться. В режиме обратной совместимости можно сохранять и файлы шаблонов. Для таких файлов используется расширение xlt.

Еще один достаточно популярный тип файлов, которые часто и эффективно используются при работе с приложением Excel, — надстройки. Надстройки используются для подключения дополнительных утилит и ресурсов. В версии Excel 2013 (как и в версиях Excel 2010 и Excel 2007) надстройки имеют расширение xlam. В более ранних версиях приложения надстройки имеют расширение xla. Несложно догадаться, что для надстроек также предусмотрен режим обратной совместимости (надстройка сохраняется с расширением xla).

Помимо перечисленных, существует достаточно много прочих форматов, в которых можно сохранять рабочие документы Excel. Хотя уместнее в данном случае было бы сказать *конвертировать*, поскольку после сохранения в одном из таких форматов документ Excel фактически перестает быть таковым. Однако интерес они представляют больше теоретический, поэтому здесь останавливаться на них не будем.

РАБОТА С МЫШЬЮ

Совершенно внезапно появляется герой оперы Садко.
Он исполняет «Подмосковные вечера».

Из к/ф «Старый знакомый»

Счастливые обладатели мышек с колесиком прокрутки (такие мыши называют *IntelliMouse*) при работе с приложением Excel могут получить весомые бонусы от наличия этого чудесного колеса. Кроме очевидного преимущества, связанного с прокруткой колеса для просмотра документа, есть еще одна военная хитрость, которая позволяет просматривать документ в более удобном режиме. Для этого в рабочей области достаточно щелкнуть на колесе, в результате чего в месте размещения курсора появится элегантный геометрический знак, напоминающий четыре направленных в разные стороны стрелки с увесистой точкой в центре (рис. 1.27).

Если отвести курсор мыши в сторону, он будет иметь вид стрелки, указывающей направление автоматической прокрутки документа. Чем больше расстояние от курсора до первоначального его положения — тем выше скорость прокрутки документа. Чтобы завершить это действо, достаточно, например, нажать любую клавишу или щелкнуть кнопкой мыши.

Важную информацию о режиме работы с документом или элементом графического интерфейса можно получить по виду курсора мыши. В таблице 1.1 приведены основные изображение курсора и кратко описано, когда курсор мыши может принимать тот или иной вид.

Конечно, плохо, что курсор не умеет разговаривать, однако иногда даже одного его вида бывает достаточно чтобы понять, какие действия можно выполнить с тем или иным объектом в рабочем документе. Более детально большинство из этих действий описываются в следующих главах книги.

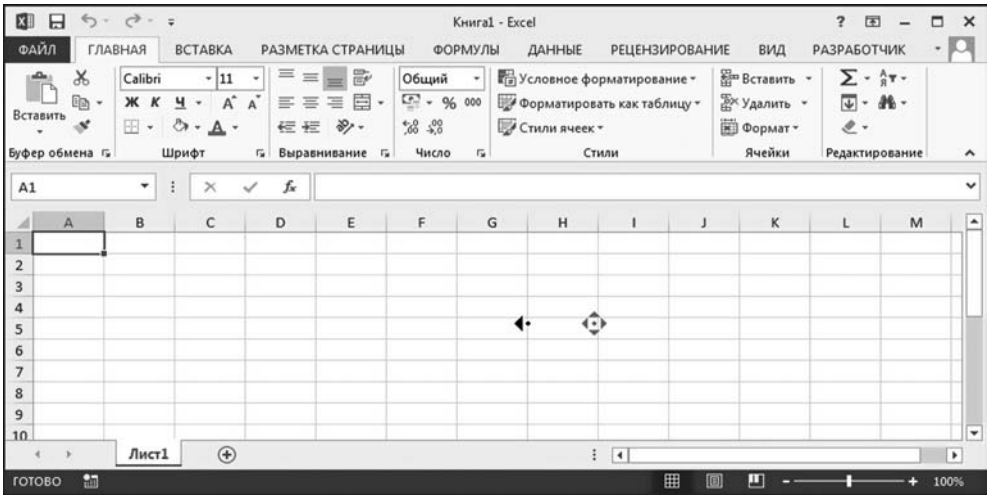


Рис. 1.27
Использование мыши IntelliMouse

Таблица 1.1

Курсор мыши

Изображение курсора	Описание
	Вид курсора мыши в рабочей области (над ячейками электронной таблицы)
	Вид курсора в области строки формул или при наведении на выделенную для ввода данных ячейку
	Обычный вид курсора, например, при наведении на управляющие элементы интерфейса
 и 	Вид курсора мыши при наведении на ленту индексации строк и столбцов соответственно (полосы по краям рабочей области с номерами строк и буквенными названиями столбцов). Нажатие левой кнопки мыши приводит к выделению строки или столбца
 и 	Вид курсора мыши в области ленты индексации строк и столбцов на границе между соседними строками или столбцами. Удерживая левую кнопку мыши, можно изменять высоту строк и ширину столбцов
 или  или  или 	Вид курсора мыши при наведении на границу выделенного объекта в рабочем документе: верхнюю или нижнюю, правую или левую, левый верхний или нижний правый угол, левый нижний или правый верхний угол. Удерживая левую кнопку мыши, можно изменять размеры объекта
	Вид курсора мыши в области выделенного объекта, готового к перемещению в рабочем документе
	Такой вид имеет курсор при наведении на маркер заполнения (темное утолщение в правом нижнем углу рамки выделения ячейки или диапазона ячеек)
	Вид курсора при наведении на гиперссылку
	Вид курсора при наведении на якорь графического объекта в рабочем документе (маленький зеленый кружочек рядом с объектом) для вращения объекта
	Вид курсора при выделении области в рабочем документе для вставки графических фигур
 или 	Вид курсора при перемещении ползунка границ закрепления областей
 или 	Перемещение или копирование рабочего листа (область списка корешков рабочих листов)
	Копирование формата ячеек

ЗНАКОМСТВО С ДИАГРАММАМИ

Нет, Тамбов такого не видел!

Из к/ф «Старый знакомый»

Одно из принципиальных преимуществ приложения Excel состоит в том, что оно позволяет не только выполнять качественный анализ и обработку больших массивов данных, но еще и представлять результаты в наглядном и эффектном виде. Не последнюю роль при этом играют всевозможные графические утилиты и, в первую очередь, диаграммы и другие графики. Здесь мы очень кратко опишем способ быстрого создания диаграмм в Excel. Более детально вопросы создания диаграмм и вставки в рабочий документ других графических объектов описываются в отдельной главе.

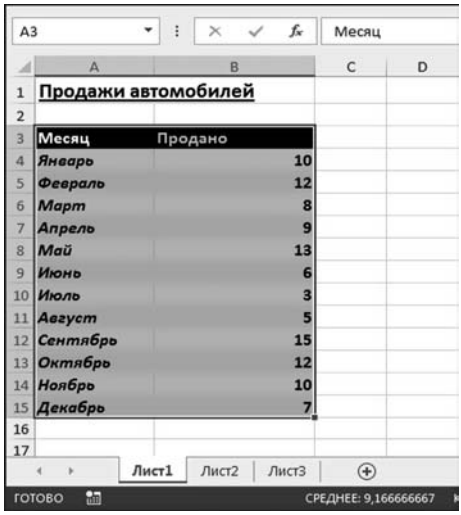


Рис. 1.28
Документ с данными для создания диаграммы

Для создания диаграммы необходимы как минимум данные, на основании которых создается диаграмма. На рисунке 1.28 представлен документ, который содержит данные (условные) об объемах продаж автомобилей в фактических единицах (диапазон ячеек B4:B15) в зависимости от месяца (диапазон ячеек A4:A15). Диапазон ячеек A3:B3 содержит заголовки для столбцов означенной мини-таблицы. Существует несколько простых и надежных способов создания

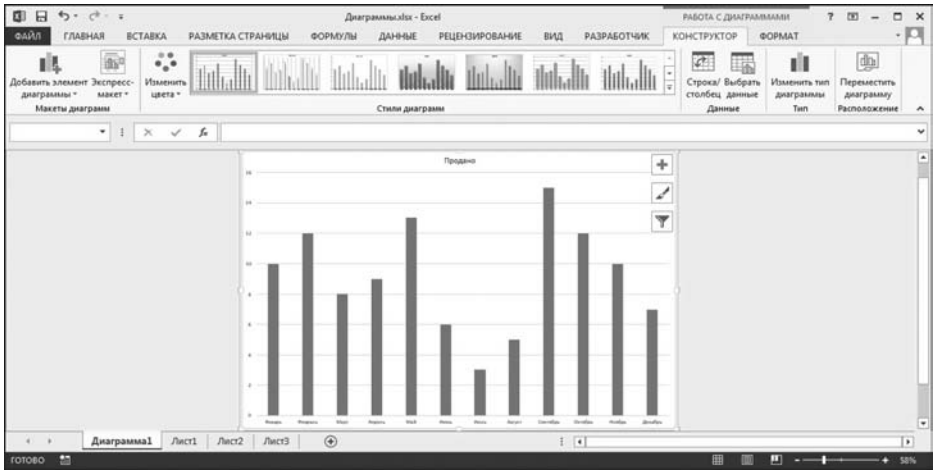


Рис. 1.29
На основе числовых данных в новом окне создана диаграмма

диаграмм. Простой до банальности способ состоит в том, что в рабочем документе выделяется диапазон ячеек, на основе которого строится диаграмма, после чего нажимаем клавишу <F11>. На рисунке 1.28 выделен диапазон ячеек A3:B15. Нажатие на кнопке <F11> автоматически добавляется новый рабочий лист с диаграммой, как показано на рисунке 1.29.

Лист (который называют листом диаграммы) содержит единственную диаграмму и ничего больше. По умолчанию название такого листа **Диаграмма1** (следующий лист с диаграммой будет **Диаграмма2**, затем **Диаграмма3**, и т. д.). Что касается самой диаграммы, то стоит обратить внимание на наличие корректных подписей у горизонтальной оси, и названия **Продано** для ряда данных (это же название используется для заголовка диаграммы, который, кстати, довольно просто поменять). Другими словами, процесс создания диаграммы в автономном режиме прошел достаточно качественно.

Диаграмму можно разместить и непосредственно в рабочем листе с исходными данными. На рисунке 1.30 в рабочем листе с данными выделен диапазон ячеек, после чего в раскрывающемся списке **Гистограмма** в группе **Диаграммы** вкладки **Вставка** выбирается одна из позиций (объемная гистограмма с группировкой).

Результат показан на рисунке 1.31.

Эту диаграмму в рабочем листе можно перемещать (перетаскиванием мышью), менять параметры диаграммы (редактировать диаграмму), и еще делать много-много полезных и интересных вещей.

На заметку

Если вид диаграммы, по мнению пользователя, далек от совершенства — не стоит впадать в отчаяние. Уже созданная диаграмма легко редактируется, вплоть до изменения ее типа и иных основных характеристик.

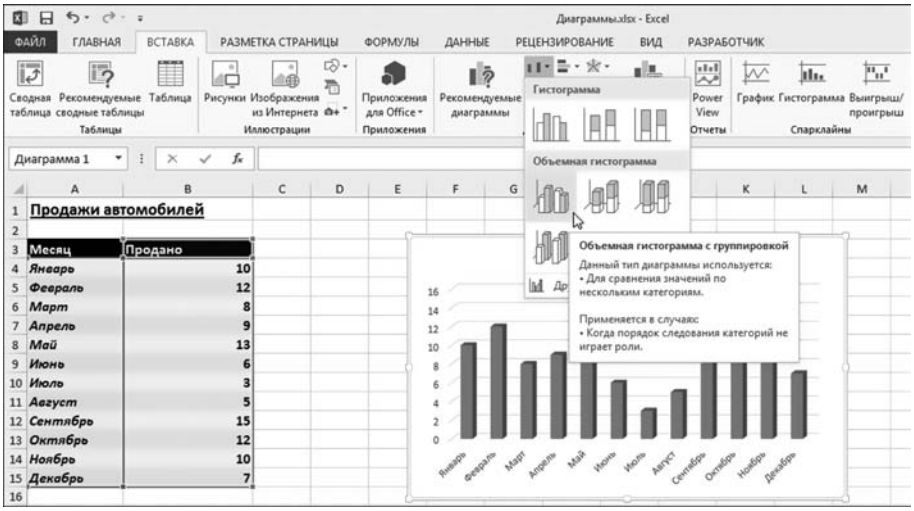


Рис. 1.30
Для создания диаграммы используем утилиты группы **Диаграммы** вкладки **Вставка**

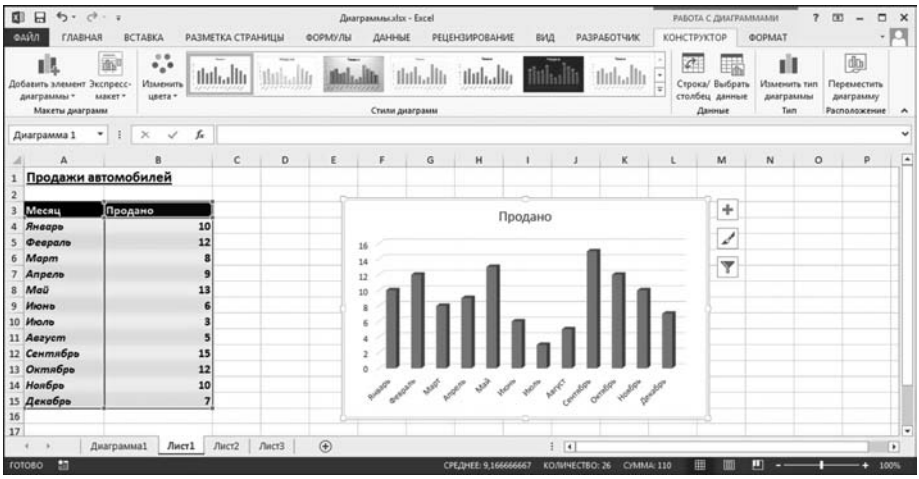


Рис. 1.31
Диаграмма создана в текущем рабочем листе

Сонм утилит для работы с диаграммами собран на контекстной вкладке **Работа с диаграммами**, которая появляется, если в рабочем документе выделена диаграмма. У вкладки имеются подвкладки **Конструктор** и **Формат**, которые содержат пиктограммы для команд, что называется, на все случаи жизни (рис. 1.31). Также при выделении диаграммы автоматически выделяются (рамками разного цвета) диапазоны ячеек, на основе которых создавалась диаграмма.

На заметку

Похожий режим используется при работе с формулами. Если в рабочем документе выделить ячейку с формулой и перейти в режим редактирования содержимого такой ячейки, то те ячейки рабочего документа, на которые есть ссылка в формуле, будут выделяться рамками разного цвета.

О диаграммах речь еще будет идти. Пока же мы ограничимся той информацией, что приведена выше — на первых порах нам ее будет вполне достаточно.

РАБОТА С ИЗОБРАЖЕНИЯМИ

— Слово бухгалтерии.
— Мне нравится, когда музыка и танцы.
И когда все погружается.
Но меня интересует одно: во что это выльется?

Из к/ф «Старый знакомый»

Помимо диаграмм, в Excel есть и другие «художественные» конструкции — не такие полезные, как диаграммы, но не менее красивые. Утилиты для управления элементами художественного оформления рабочего документа собраны в основном на вкладке **Вставка** ленты — например, в группе **Иллюстрации** (рис. 1.32).

Например, чтобы вставить в рабочий документ изображение, щелкаем пиктограмму **Рисунок** (рис. 1.32). В результате открывается диалоговое окно **Вставка рисунка**, в котором, собственно, и выбирается рисунок для вставки (рис. 1.33).

Кнопка **Вставка** в этом окне представляет собой раскрывающийся список с несколькими командами, которые влияют на режим вставки изображения. На рисунке 1.34 показан результат вставки в рабочий документ изображения.

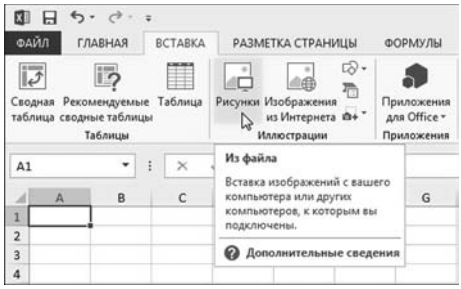


Рис. 1.32
Пиктограммы группы **Иллюстрации** вкладки **Вставка** используются для вставки изображений и графических объектов

На заметку

Кроме обычной вставки, вставляемое в рабочий документ изображение можно связать с исходным файлом. В этом случае изменения в исходном файле будут отражаться на изображении в рабочем документе.

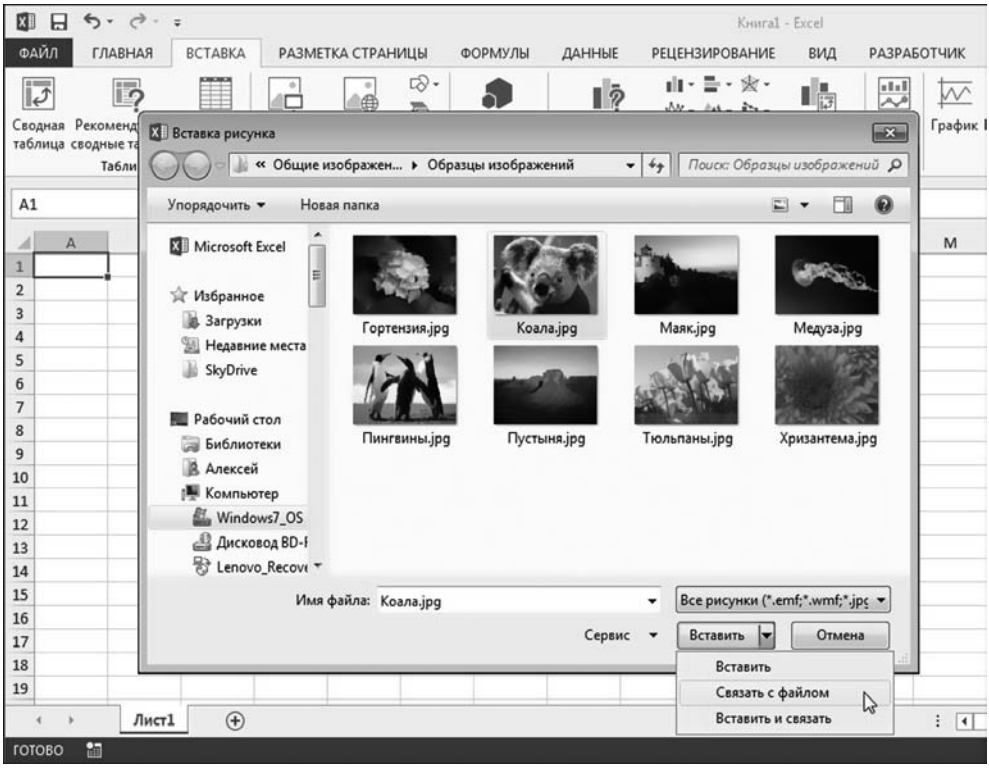


Рис. 1.33
Вставка рисунка в рабочий документ

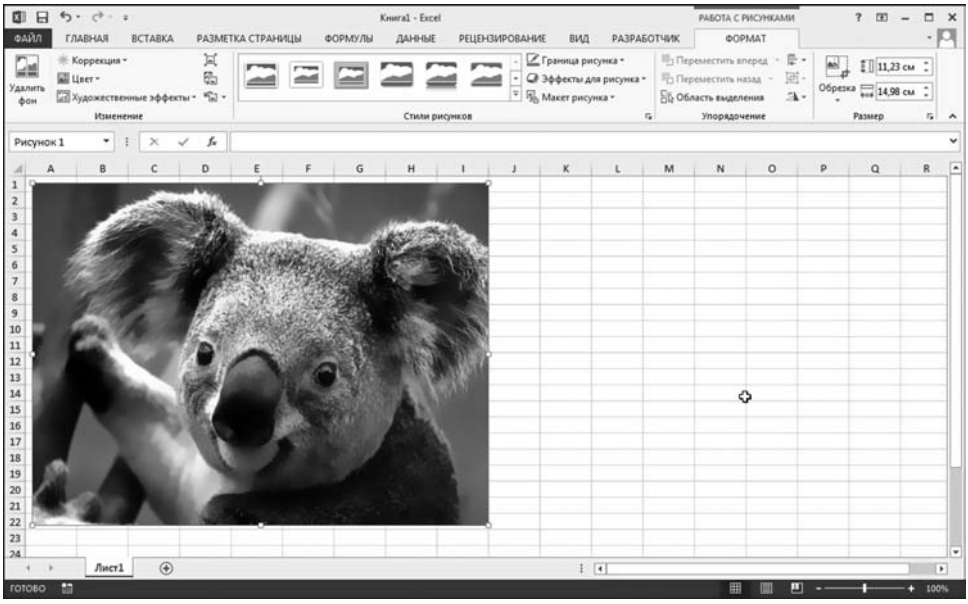


Рис. 1.34
Рисунок вставлен в рабочий документ

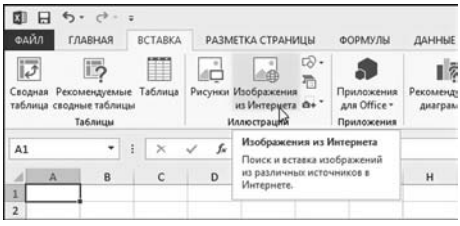


Рис. 1.35
В рабочий документ вставляется картинка

Пиктограмма **Изображения из Интернета** в группе **Иллюстрации** используется для вставки картинок из внешних источников (рис. 1.35).

В результате откроется окно **Вставка картинок**, в котором выбирается нужное изображение для вставки в рабочий документ (рис. 1.36).

Весьма полезной и интригующей является пиктограмма **Фигуры**, которая представляет собой раскрывающийся список со множеством изображений в виде всяческих геометрических фигур и фигурок (рис. 1.37).

Для удобства все это хозяйство разбито на группы. Нужное изображение выбирается в списке с помощью мыши (растягиваем рамку для области вставки изображения с нажатой левой кнопкой мыши). На рисунке 1.38 в рабочий документ вставлено скромное, но очень обаятельное изображение (графическая фигура).

Если графическая фигура выделена в документе, появляется дополнительная вкладка **Средства рисования** с единственной подвкладкой **Формат**, которая содержит множество пиктограмм для настройки внешнего вида выделенной графической фигуры.

При создании иллюстративных материалов нередко используются *структурные диаграммы* — красивые картинки, организованные в виде

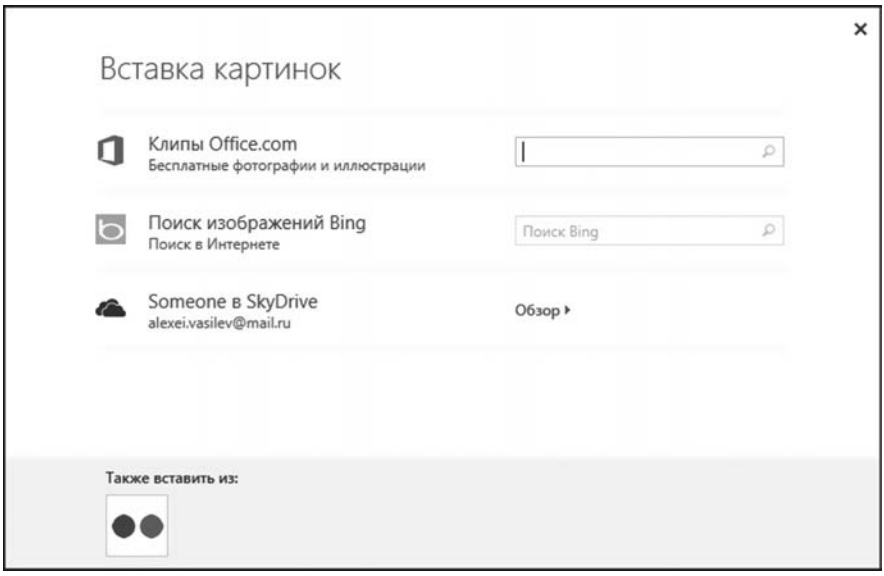


Рис. 1.36
Выбор картинки для вставки

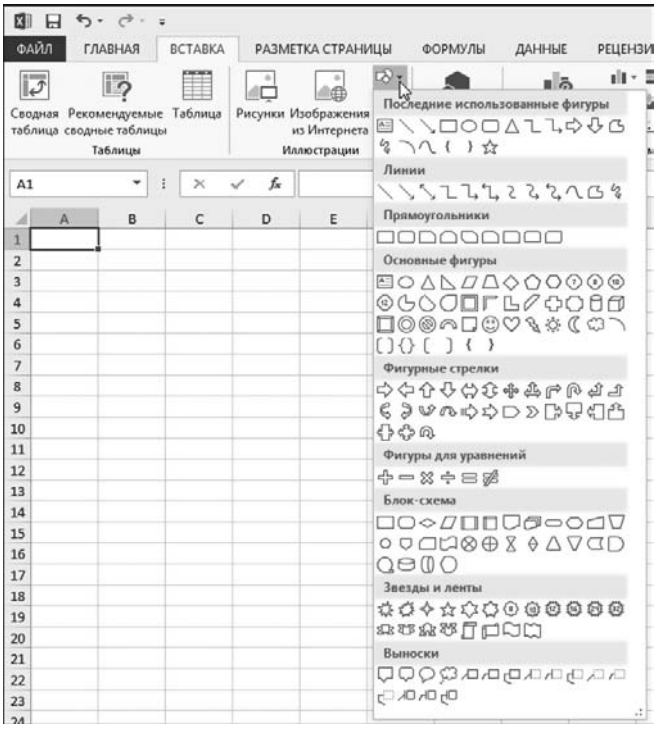


Рис. 1.37
Вставка в объект геометрической фигуры

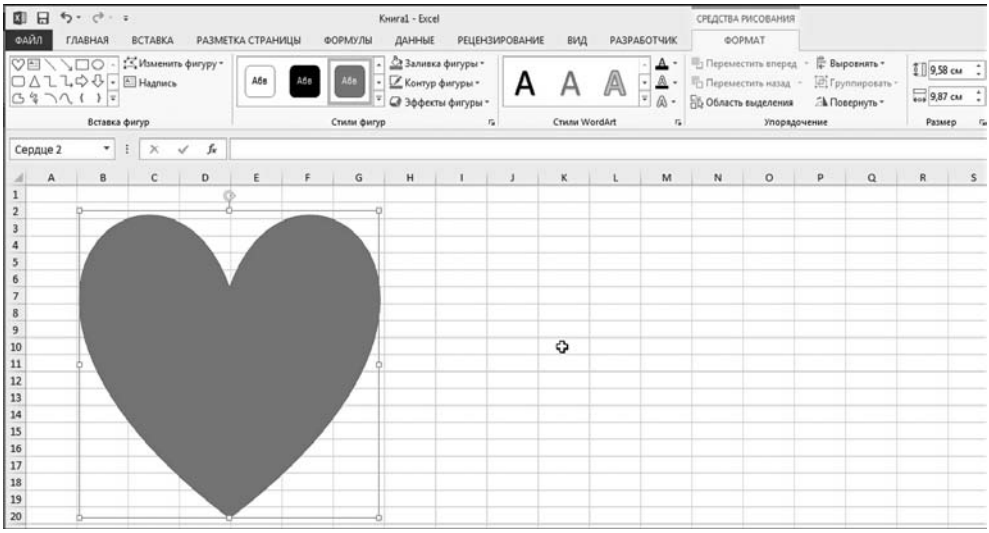


Рис. 1.38
В документ вставлена графическая фигура

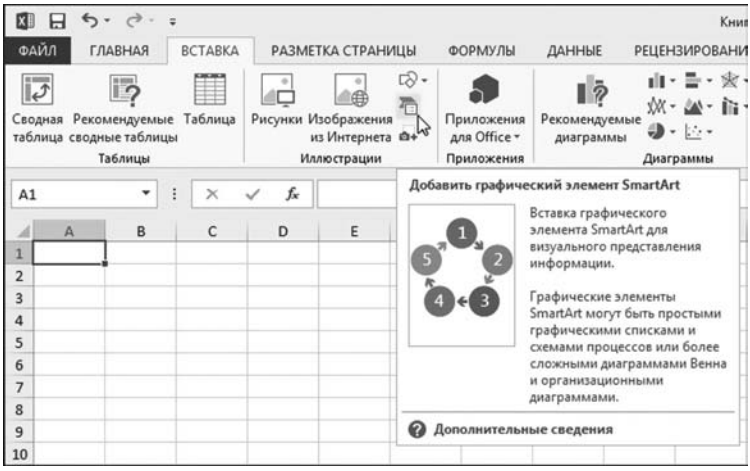


Рис. 1.39
Вставка изображения SmartArt

иерархических структур самых разных типов и видов. Все эти сокровища спрятаны за скромной и непримечательной пиктограммой **SmartArt** (рис. 1.39).

После щелчка на пиктограмме открывается диалоговое окно **Выбор графического элемента SmartArt**, в котором есть картинки на любой вкус и цвет (рис. 1.40). Следуя старой доброй традиции, картинки разбиты на группы, названия которых представлены в левой части диалогового окна **Выбор графического элемента SmartArt**.

Принцип простой: выбираем группу, выбираем в списке понравившуюся пиктограмму и в правой части диалогового окна изучаем (если в этом есть необходимость) справку по выбранной структурной диаграмме.

На рисунке 1.41 показан рабочий документ, в который вставлена структурная диаграмма. Как и в предыдущих случаях, для редактирования уже вставленного объекта в документ этот объект выделяется, в результате чего на ленте появляется дополнительная вкладка — в данном случае это вкладка **Работа с рисунками SmartArt.**

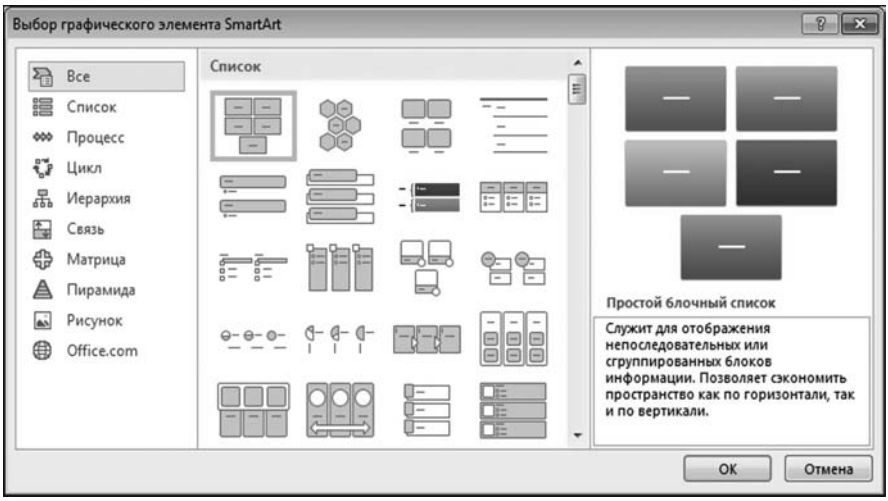


Рис. 1.40
Выбор типа вставляемого элемента в окне
Выбор графического элемента SmartArt

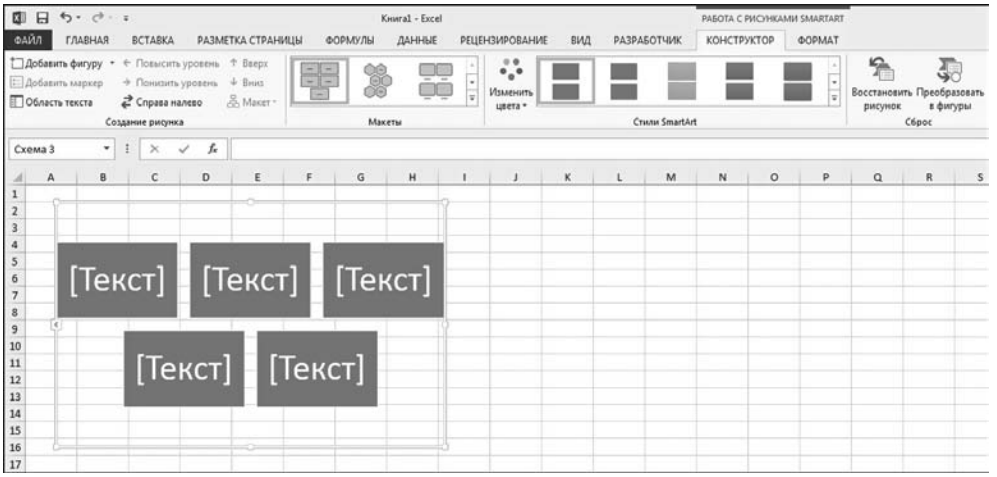


Рис. 1.41
В документ вставлен объект SmartArt

Возможности для редактирования объекта при этом самые широкие — вплоть до изменения типа структурной диаграммы. Желающие могут поупражняться самостоятельно.

Есть несколько полезных утилит в группе **Текст** вкладки **Вставка**. Например, пиктограмма **Надпись** используется в том случае, если мы хотим разместить в рабочем документе текстовую надпись (рис. 1.42).

Такая надпись размещается в специальной области, которая накладывается на подложку рабочего листа (т. е. надпись не связана с какой-то конкретной ячейкой) и может по ней перемещаться (выделением и перетаскиванием мышью). На рисунке 1.43 проиллюстрирован процесс ввода текстового значения в текстовую область, вставленную в рабочий документ.

Как может выглядеть готовая текстовая надпись, показано на рисунке 1.44.

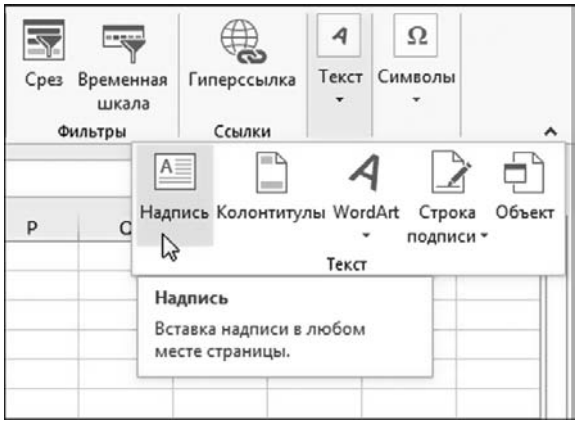


Рис. 1.42
Вставка в документ текстовой надписи

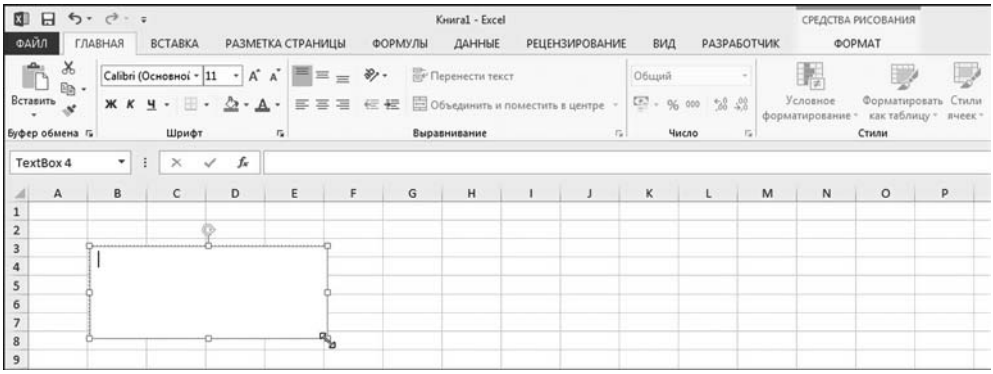


Рис. 1.43
Ввод текста в поле текстовой надписи

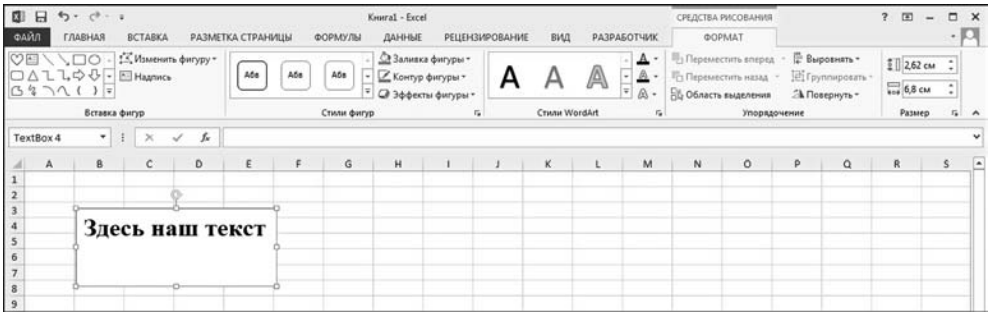


Рис. 1.44
Так может выглядеть выделенная в рабочем документе текстовая надпись

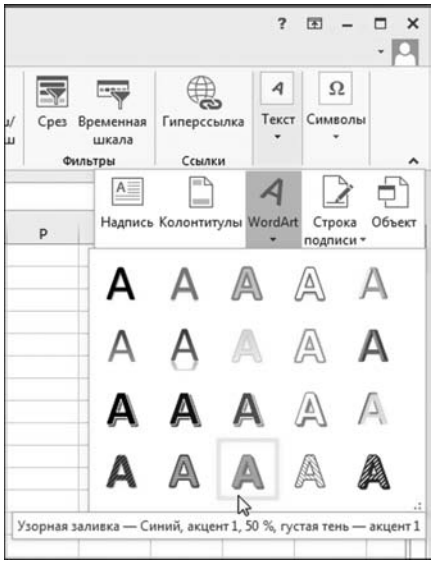


Рис. 1.45
Вставка в документ художественного текста

На заметку

При выделении текстовой области на ленте приложения отображается дополнительная вкладка **Средства рисования** с подвкладкой **Формат** (рис. 1.44). Несложно догадаться, что эта дополнительная вкладка содержит средства для редактирования и настройки текстовой области с надписью. Вместе с тем не стоит пренебрегать и стандартными средствами редактирования текста доступными, например, на вкладке **Главная** ленты.

Весьма интересная штука — художественный текст. Строго говоря, художественный текст — это графический объект, который имеет вид текста. Чтобы вставить в документ художественный текст, раскрываем список пиктограммы **WordArt** в группе **Текст** и выбираем понравившийся стиль для

художественного текста (если не понравился никакой — не страшно, ведь все равно стиль можно будет поменять). Процесс выбора типа художественного текста проиллюстрирован на рисунке 1.45.

На рисунке 1.46 показан «свежевставленный» в рабочий документ художественный текст.

Текст можно редактировать — как само содержимое, так и внешний вид. Для настройки этих параметров используются как нехитрые методы банального редактирования текстов, так и хитроумные приемы интеллектуального переключения кнопок, опций и прочих магических действий. Полезной будет дополнительная вкладка **Средства рисования**, которая появляется при выделении художественного текста (рис. 1.46).

Вообще же перечень объектов, которые можно инкапсулировать в рабочий документ Excel, просто огромен. Общая схема действий для вставки

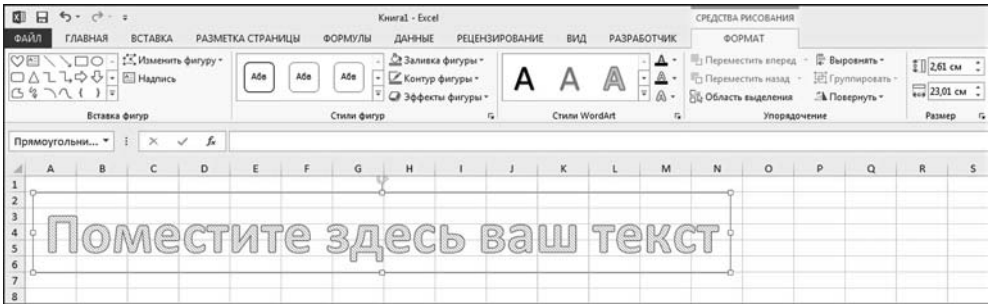


Рис. 1.46
В рабочий документ вставлен готовый к редактированию художественный текст

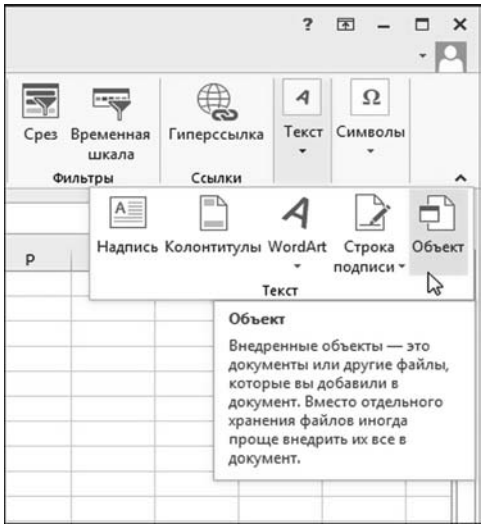


Рис. 1.47
Вставка в документ объекта

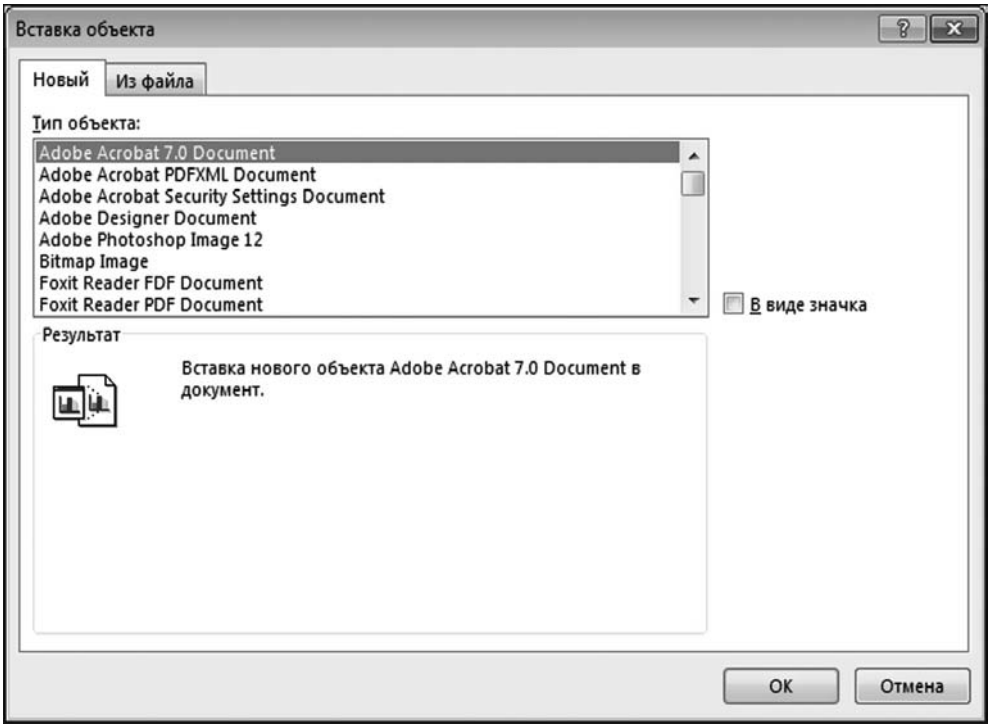


Рис. 1.48
Выбор типа объекта для вставки в рабочий документ

объекта в рабочий документ подразумевает щелчок на пиктограмме **Объект** в группе **Текст** на вкладке **Вставка** (рис. 1.47).

В результате открывается диалоговое окно **Вставка объекта**, которое показано на рисунке 1.48.

У окна две вкладки: **Новый** и **Из файла**, которые используются при вставке соответственно нового объекта и уже существующего. Для успешного выполнения операции по вставке объекта необходимо выбрать тип объекта (определяется в основном типом приложения) или файл с объектом.

ВВОД ГРАФИЧЕСКИХ ФОРМУЛ И СИМВОЛОВ

— Как бы символика?
— Она самая!

Из к/ф «Старый знакомый»

В принципе, излагаемый далее материал смело можно было бы отнести к предыдущему разделу. Тем не менее, то, о чем пойдет речь здесь, имеет особую важность в свете прикладного использования приложения Excel в инженерных и научных расчетах. Поэтому такое «особое» внимание, думается, вполне оправдано. Итак, поговорим о символах и *графических формулах*.

На заметку

Под символами здесь подразумеваются экзотические буквы. Они, кстати, иногда бывают настолько экзотическими, что и буквами-то их назвать сложно. С графическими формулами все наоборот — это не формулы Excel, а самые обычные математические формулы, красиво написанные и не имеющие никакого отношения к вычислениям значений ячеек Excel. Именно чтобы отличить эти «написанные» формулы от формул, вводимых в ячейки рабочих документов, мы и используем уточняющее слово «графические».

Для вставки символов и графических формул на вкладке **Вставка** в группе **Символы** представлено две пиктограммы **Уравнение** и **Символ**, названия которых говорят сами за себя (рис. 1.49). Начнем с символов.

Для вставки символа щелкаем, как несложно догадаться, на пиктограмму **Символ** и наблюдаем диалоговое окно с таким же названием и двумя вкладками. На рисунке 1.50 показано окно **Символ**, открытое на вкладке **Символы**.

Ключевым здесь, пожалуй, является раскрывающийся список **Шрифт**, в котором выбирается тип шрифта для отображения символов. Например, греческие буквы лучше вводить с использованием шрифта **Symbol**. В центральной области вкладки отображается набор символов, которые можно вставить в документ. На вкладке **Специальные знаки** представлен список иероглифических комбинаций, которые тоже подпадают под понятие «символ» (рис. 1.51).

Главное удобство окна **Символ** связано с тем, что сразу видно, с каким символом мы имеем дело. Здесь же кроется и главное неудобство — выбирать символы из палитры — далеко не самое быстрое и интеллектуальное занятие.

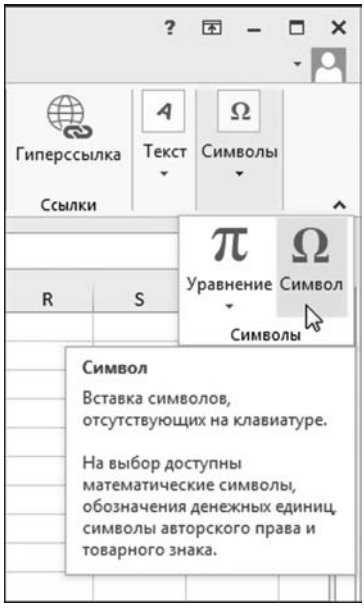


Рис. 1.49
В группе **Символы** пиктограммы **Уравнение** и **Символ** предназначены для вставки соответственно формул (графических) и символов в документ

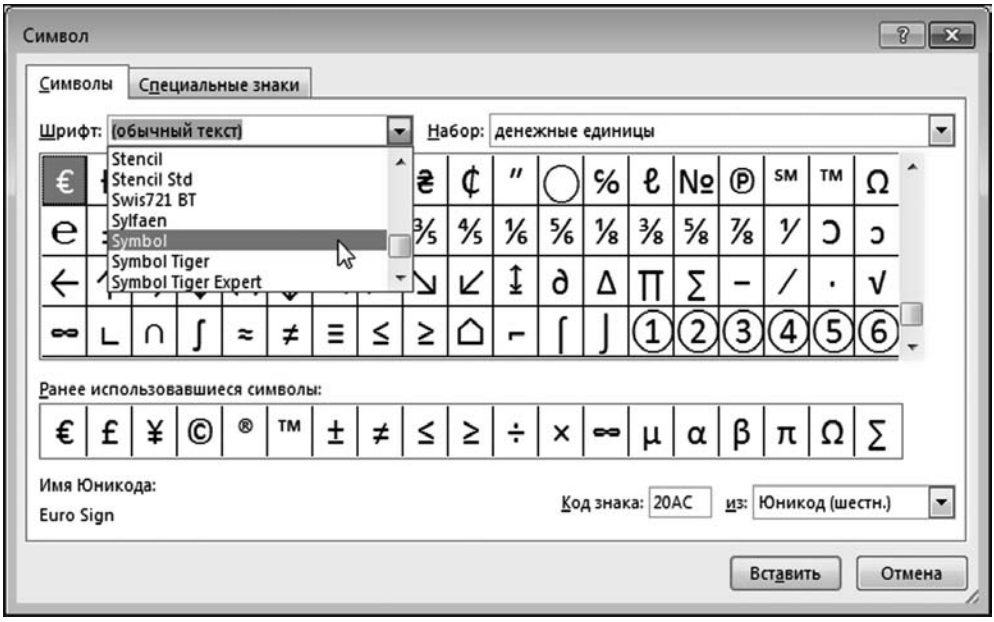


Рис. 1.50
Окно **Символ** открыто на вкладке **Символы**

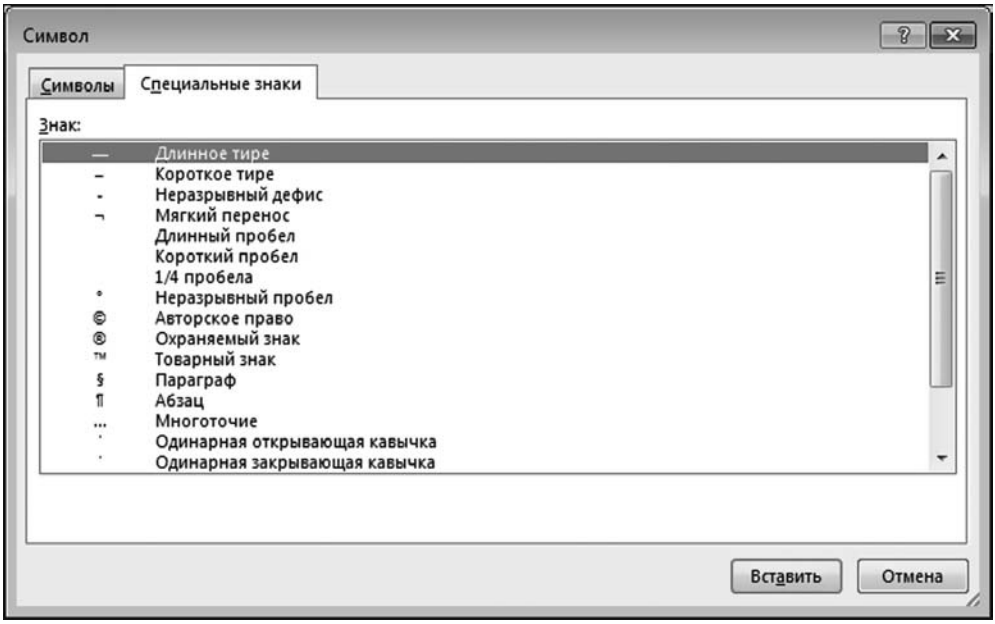


Рис. 1.51
Окно **Символ** открыто на вкладке **Специальные знаки**

На заметку

Те же греческие символы можно вводить с клавиатуры, если предварительно переключиться на нужный шрифт. На вкладке **Главная** в группе **Шрифт** выбираем из списка доступных шрифтов какой-нибудь экзотический шрифт — например, шрифт **Symbol** (рис. 1.52).
Затем с клавиатуры после нажатия обычных, казалось бы, клавиш появляются привычные для греков и математиков буквы.

Теперь остановимся на том, как в рабочем документе вводятся формулы — ведь одних символов в этом случае будет недостаточно.

На заметку

В пакете Microsoft Office 2007 появилась уникальная штука — встроенный редактор формул. Он достаточно удобный, а набранные с его помощью формулы выглядят вполне элегантно. Разумеется, есть такой редактор и в версиях Excel 2010 и Excel 2013. Об этом редакторе дальше и пойдет речь.

Если раскрыть список пиктограммы **Уравнение**, появится область с несколькими шаблонами для формул, которыми можно воспользоваться (рис. 1.53).

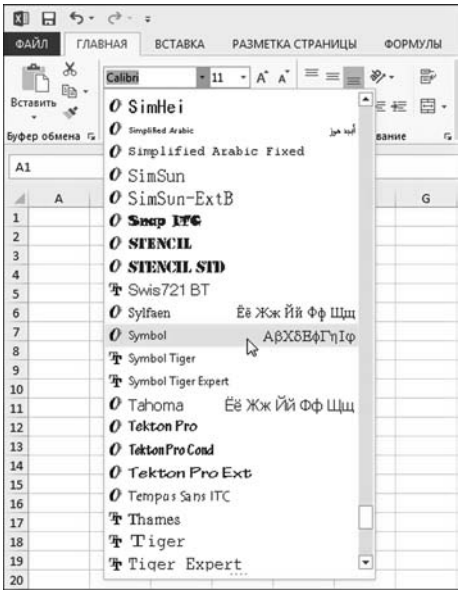


Рис. 1.52
Выбор шрифта
для отображения символов

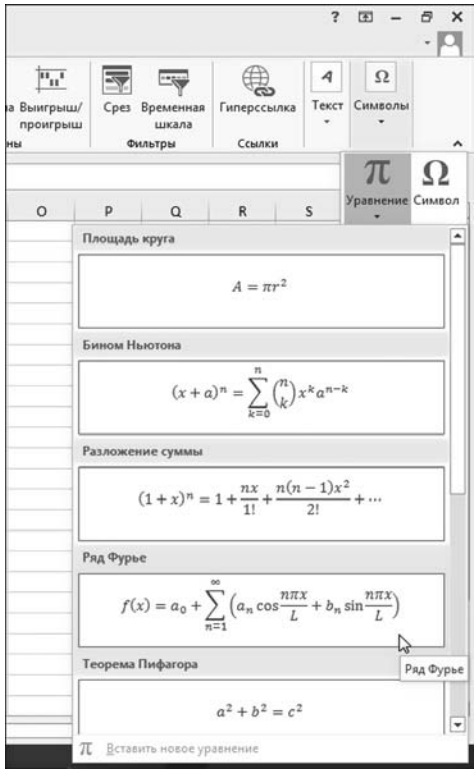


Рис. 1.53
Выбор шаблона вставляемой формулы

Рис. 1.54
Вставка новой (пустой) формулы

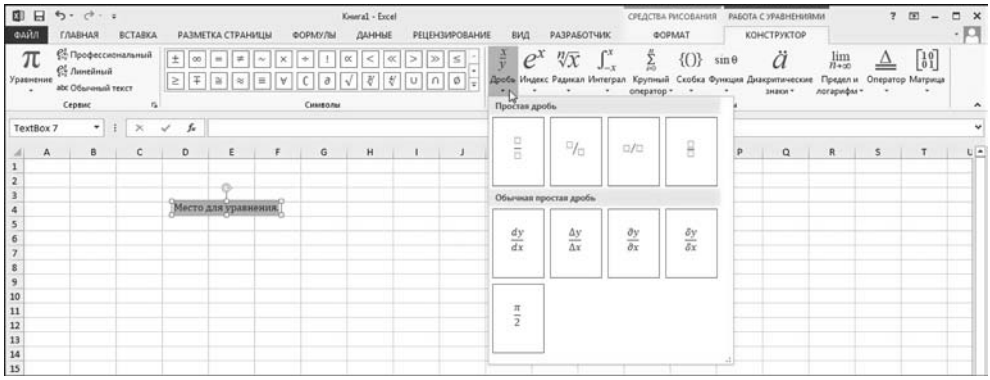


Рис. 1.55
Ввод формулы с помощью утилит дополнительной вкладки **Работа с уравнениями**

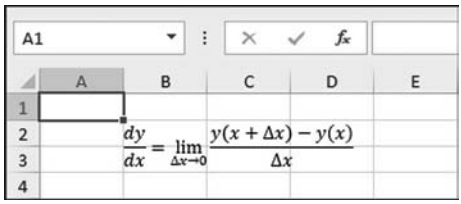


Рис. 1.56
Формула в рабочем документе

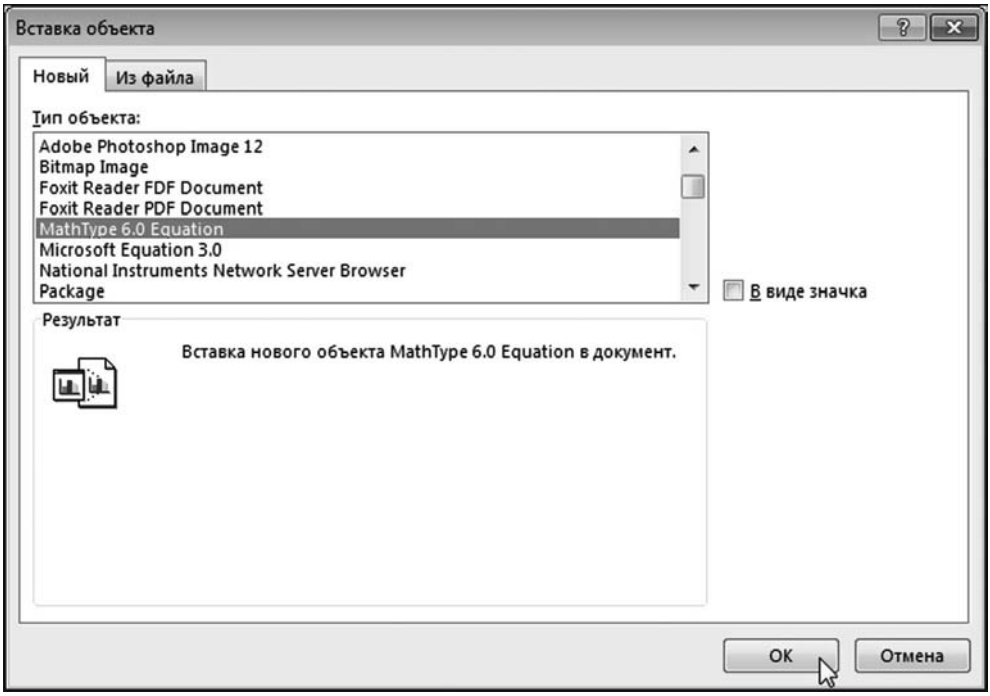


Рис. 1.57
Вставка объекта MathType в рабочий документ

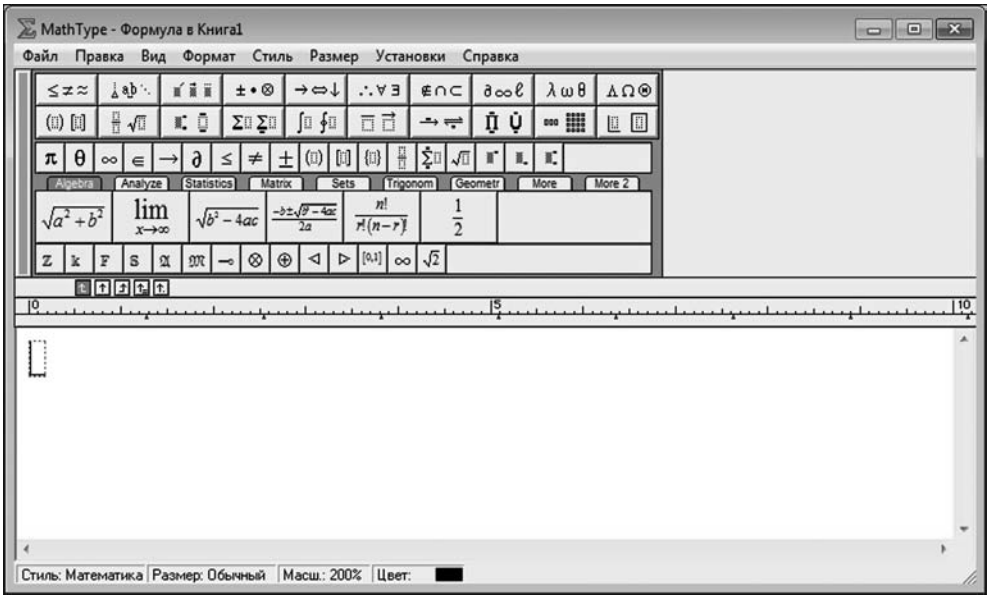


Рис. 1.58
Окно редактора формул MathType

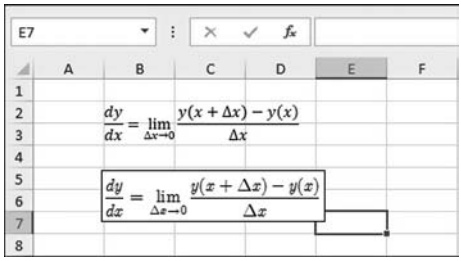


Рис. 1.59
Рабочий документ с формулами,
набранными с помощью встроенного редактора формул Excel
и редактора формул MathType

Даже если шаблон не очень подходит — не страшно. После вставки формулы ее, разумеется, можно будет редактировать. Вместе с тем, можем просто щелкнуть на пиктограмме **Уравнение** (рис. 1.54), что приведет к вставке в рабочий документ новой пустой формулы.

При выделении формулы в документе, помимо привычной для графических объектов вкладки **Средства рисования**, появляется еще и дополнительная вкладка **Работа с уравнениями** (рис. 1.55).

Вкладка содержит множество полезных списков, пиктограмм и прочих элементов, с помощью которых вводится нужная формула. На рисунке 1.56 показана графическая формула (формальное определение производной для функции), набранная с помощью встроенного редактора Excel.

На заметку

Достаточно популярен редактор формул MathType, который нередко является базовым редактором при наборе формул в текстовом редакторе Word. Формульный редактор MathType доступен и в Excel. Если соответствующая программа уже установлена, то при попытке вставить объект в рабочий документ (напомним, для вставки объекта следует щелкнуть на пиктограмме **Объект** в группе **Текст**) в окне **Вставка объекта** среди списка типов объектов будет и объект приложения MathType, который следует выбрать (рис. 1.57).
Формула набирается в специальном редакторе, окно которого показано на рисунке 1.58.
На рисунке 1.59 показан рабочий документ с двумя идентичными графическими формулами — одна (верхняя) набрана с помощью встроенного редактора, а другая (нижняя) набрана с помощью редактора формул MathType.
Формулы практически идентичны, но разница между ними есть. Желающие могут поупражняться в поиске различий.

ГЛАВА 2 ОСОБЕННОСТИ ИНТЕРФЕЙСА

Зачем нашим советским людям скрывать свое лицо?
Зачем, товарищи? Это нетипично!
Но не в этом дело.
Перейдем к существу вопроса.

Из к/ф «Карнавальная ночь»

Прежде чем непосредственно приступить к изучению приемов эффективной работы в Excel, неплохо хотя бы вкратце познакомиться с основными элементами графического интерфейса этого приложения. Именно графическому интерфейсу Excel посвящена данная глава. В предыдущей главе некоторые элементы упоминались вскользь. Здесь им уделяется больше внимания. Кроме основных элементов интерфейса, в главе описывается ряд возможностей и режимов Excel, которые напрямую к интерфейсу не относятся, но влияют на внешний вид окна приложения. Вообще, интерфейс Excel достаточно гибкий в плане настроек. Описание некоторых из них читатель также найдет в этой главе.

На заметку

Нас будет интересовать даже не столько сам интерфейс, а скорее «способ взаимодействия» пользователя с приложением. Поэтому в данную главу, например, добавлен раздел, посвященный выводу документа на печать, — хотя прямого отношения к интерфейсу все это, как несложно догадаться, не имеет.

РАБОТА С ЛЕНТОЙ

Смотрите!
Сейчас мы пойдем через пространство.

Из к/ф «Иван Васильевич меняет профессию»

Лента в новых версиях приложения Excel является главным функциональным элементом интерфейса, удобным, простым и элегантным. Большинство возможностей приложения реализованы посредством именно ленты. В предыдущей главе лента упоминалась лишь кратко. Здесь остановимся на ней подробнее.

Обычно лента содержит девять вкладок (включая специальную вкладку **Файл**). Поскольку вкладка **Файл** стоит особняком в списке вкладок, далее, если явно не указано, под вкладками ленты будут подразумеваться «иные»

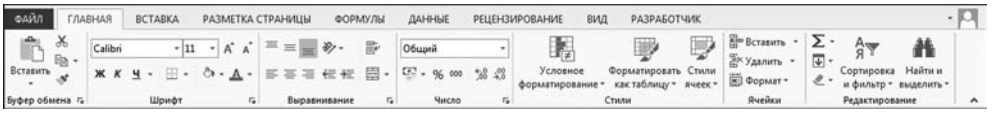


Рис. 2.1
Лента открыта на вкладке **Главная**

Таблица 2.1

Группы и пиктограммы вкладки Главная

Группа или пиктограмма	Описание
Буфер обмена	Пиктограммы для работы с буфером обмена: копирование и удаление данных в буфер, вставка данных из буфера и копирование форматов
Шрифт	Пиктограммы для применения настроек и форматов к данным в ячейках рабочей области: выделение полужирным шрифтом или курсивом, тип шрифта, изменение цвета, заливка фона и др.
Выравнивание	Группа пиктограмм для определения способа выравнивания (по горизонтали и вертикали) данных в ячейках рабочих листов
Число	Группа пиктограмм для применения форматов (в основном числовых, но не только) к данным в ячейках
Стили	Пиктограммы для применения стилей, условных форматов и форматов таблиц к ячейкам рабочих листов
Ячейки	Группа пиктограмм для вставки и удаления ячеек, а также изменения их параметров, например, таких как ширина и высота
Редактирование	Пиктограммы для выполнения основных редакторских операций, поиска и сортировки данных в рабочих документах

вкладки. Каждая вкладка состоит из *групп*, каждая группа, в свою очередь, содержит пиктограммы для выполнения тех или иных команд. Открытая на вкладке **Главная** лента представлена на рисунке 2.1.

Описание групп пиктограмм вкладки **Главная** приведено в таблице 2.1.

Вкладка **Главная** оправдывает в полной мере свое название, и основные операции, связанные с редактированием и форматированием данных, выполняются с помощью ее пиктограмм.

На заметку

Многие пиктограммы на вкладке достаточно стереотипны для приложений пакета Microsoft Office и, в частности, встречаются на вкладках ленты приложения Word. Некоторые из пиктограмм/утилит описываются в книге — в основном в контексте решаемых задач, и только в случае, если методы работы с ними имеют принципиальное значение для понимания излагаемого материала.

Утилиты вкладки **Вставка** полезны при вставке в документ всевозможных объектов, но в основном графических, а также диаграмм, гиперссылок и текстовых элементов (художественного текста и текстовых полей). Вкладка **Вставка** представлена на рисунке 2.2.

Описание групп пиктограмм вкладки **Вставка** приведено в таблице 2.2.

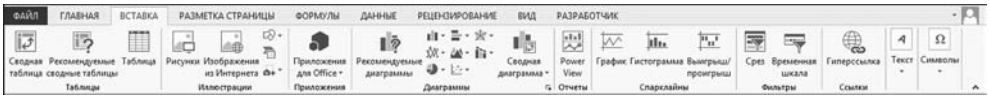


Рис. 2.2
Лента открыта на вкладке **Вставка**

Таблица 2.2

Группы и пиктограммы вкладки Вставка

Группа или пиктограмма	Описание
Таблицы	Пиктограммы для работы с таблицами (структурные элементы рабочих листов)
Иллюстрации	Пиктограммы для вставки графических объектов (рисунки, картинки, графические фигуры, снимки) в документ
Приложения	Группа используется для управления дополнительными программами, предназначенными для работы с приложением
Диаграммы	Пиктограммы для создания диаграмм и работы с ними
Отчеты	Группа для работы с надстройкой составления отчетов **Power View**
Спарклайны	Пиктограмма для вставки в документ и редактирования *спарклайнов*. Это новый элемент, который появился в последних версиях Excel и представляет собой миниатюрную, размером в одну ячейку, диаграмму, отображающую структуру данных для предопределенного диапазона ячеек
Фильтры	Используется для фильтрации данных путем выполнения «среза» данных
Ссылки	Вставка гиперссылок в документ
Текст	Вставка в документ колонтитулов, художественного текста, текстовых областей и пр.
Символы	Вставка в документ символов и формул (формулы специального формульного редактора пакета Microsoft Office)

Фактически, через вкладку **Вставка** в документ добавляется все, что не связано с непосредственным вводом числовых и текстовых данных в ячейки документа или подключением к внешним базам данных. Вкладка **Вставка**, как и вкладка **Главная**, на практике используется достаточно часто.

Пиктограммы, представленные на вкладке **Разметка страницы**, используются в основном при выводе документа на печать. Вкладка имеет вид, показанный на рисунке 2.3.

Описание групп пиктограмм вкладки **Разметка страницы** приведено в таблице 2.3.

На заметку

Нехитрые приемы, с помощью которых документ выводится на печать, описываются в конце этой главы.

Пользы в электронных таблицах было бы мало, если бы в них не использовались формулы. Для работы с формулами предназначена вкладка **Формулы**, представленная на рисунке 2.4.

Описание групп пиктограмм вкладки **Формулы** приведено в таблице 2.4.

Пиктограммы этой вкладки полезны при вставке встроенных функций в рабочие документы, а также при отслеживании структурных связей между ссылками в формулах и настройке режима вычислений документа.

Формулы нужны для обработки данных. Кроме формул, обработка данных реализуется посредством ряда встроенных утилит, основные из которых

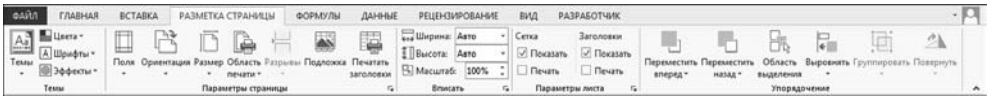


Рис. 2.3
Лента открыта на вкладке **Разметка страницы**

Таблица 2.3

Группы и пиктограммы вкладки Разметка страницы

Группа или пиктограмма	Описание
Темы	Пиктограммы для применения тем: предопределенных наборов форматов и стилей
Параметры страницы	Группа пиктограмм для выполнения настроек параметров печати, определения полей страницы, ее размера, ориентации, добавления подложки (фона документа) и пр.
Вписать	Группа пиктограмм для определения параметров масштабирования, ширины и высоты печатного содержимого при выводе документов на печать
Параметры листа	Утилиты для определения параметров рабочих листов при выводе на печать
Упорядочение	Группа пиктограмм для группировки графических объектов, их выравнивания, выполнения поворотов и определения способа отображения объектов, которые накладываются друг на друга

Рис. 2.4
Лента открыта на вкладке **Формулы**

Таблица 2.4

Группы и пиктограммы вкладки Формулы

Группа или пиктограмма	Описание
Библиотека функций	Группа пиктограмм для вставки встроенных функций в документ
Определенные имена	Группа пиктограмм для работы с именованными ячейками и диапазонами
Зависимости формул	Группа пиктограмм для отслеживания взаимных ссылок в формулах документа
Вычисление	Группа пиктограмм для настройки параметров вычислений

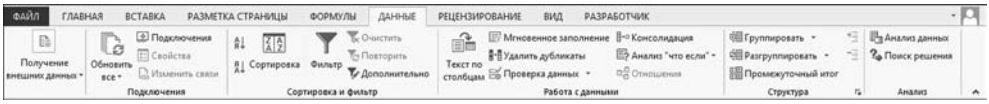


Рис. 2.5
Лента открыта на вкладке **Данные**

Таблица 2.5

Группы и пиктограммы вкладки Данные

Группа или пиктограмма	Описание
Получение внешних данных	Пиктограмма используется при получении доступа к внешним данным
Подключения	Группа пиктограмм для управления и контроля внешних связей документа
Сортировка и фильтр	Пиктограммы для выполнения сортировки и фильтрации данных
Работа с данными	Группа пиктограмм для выполнения некоторых операций с данными в документе: подбора параметров, запуска менеджера сценариев, консолидации данных и т. д.
Структура	Пиктограммы для работы со структурами
Анализ	Пиктограммы запуска надстроек для анализа данных. Группа отображается только если предварительно выполнено подключение надстроек (в данном случае заранее были подключены надстройки **Поиск решения** и **Анализ данных**)

представлены пиктограммами на вкладке **Данные** ленты. Вкладка показана на рисунке 2.5.

Описание групп пиктограмм вкладки **Данные** приведено в таблице 2.5.

На некоторые утилиты, реализованные через пиктограммы вкладки, хочется обратить особое внимание пользователя. Одна из этих утилит называется **Подбор параметра** (на вкладке она представлена в раскрывающемся списке команд пиктограммы **Анализ «что-если»** группы **Работа с данными**). Утилита позволяет, с использованием функциональной связи между ячейками, задавать значение в одной ячейке (зависимая ячейка) путем изменения значения в другой ячейке (исходная ячейка). Зависимая ячейка — это ячейка с формулой, содержащей ссылку на исходную ячейку. Надстройка **Поиск решения** позволяет решать более сложные задачи. Но эту надстройку надо сначала подключить. Если надстройка подключена, на вкладке **Данные** будет отображаться группа **Анализ**, содержащая пиктограмму для запуска надстройки.

Вкладка **Рецензирование** показана на рисунке 2.6.

Описание групп пиктограмм вкладки **Рецензирование** приведено в таблице 2.6.

Пиктограммы вкладки используются для проверки орфографии, вставки и работы с примечаниями и настройки режимов доступа к документу.

За внешний вид рабочего окна приложения ответственны утилиты, представленные на вкладке **Вид**. Сама вкладка отображена на рисунке 2.7.

Описание групп пиктограмм вкладки **Вид** приведено в таблице 2.7.

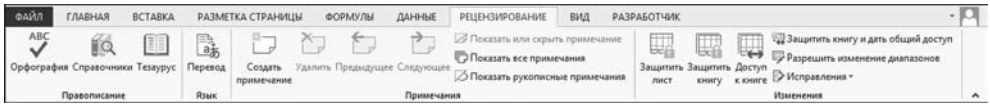


Рис. 2.6
Лента открыта на вкладке **Рецензирование**

Таблица 2.6

Группы и пиктограммы вкладки Рецензирование

Группа или пиктограмма	Описание
Правописание	Утилиты для проверки правописания
Язык	Утилита для выполнения автоматического перевода
Примечания	Группа пиктограмм для добавления примечаний в документ и работы с ними
Изменения	Группа пиктограмм для определения режима доступа к документу и отслеживания изменений в документе

Рис. 2.7
Лента открыта на вкладке **Вид**

Таблица 2.7

Группы и пиктограммы вкладки Вид

Группа или пиктограмма	Описание
Режимы просмотра книги	Группа пиктограмм, используемых для выбора режима просмотра рабочей книги
Показ	Утилита для установки режима отображения сетки, линейки (в режиме отображения страниц документа), полос индексации строк и столбцов и строки формул
Масштаб	Выбор масштаба отображения документа
Окно	Управление окнами и областями
Макросы	Утилита для работы с макросами

Некоторые пиктограммы этой вкладки упоминаются далее в главе при описании настроек внешнего вида рабочего окна.

На вкладке **Разработчик** представлены пиктограммы, полезные при записи макросов и работе с программными кодами. Вкладка показана на рисунке 2.8.

Описание групп пиктограмм вкладки **Разработчик** приведено в таблице 2.8.

Пиктограммы вкладки **Разработчик** выйдут на первый план, когда мы будем создавать всевозможные программные коды — случится это в *третьей части* книги (гл. 9–11).

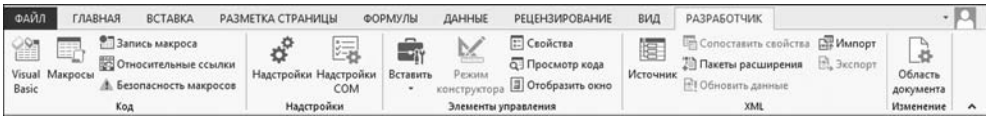


Рис. 2.8
Лента открыта на вкладке **Разработчик**

Таблица 2.8

Группы и пиктограммы вкладки Разработчик

Группа или пиктограмма	Описание
Код	Группа пиктограмм для запуска редактора VBA, записи макросов и выполнения ряда сопутствующих операций по работе с программным кодом
Надстройки	Пиктограммы для подключения надстроек
Элементы управления	Пиктограммы для работы с элементами управления
XML	Группа пиктограмм для работы с данными в формате XML
Изменение	Утилита определения шаблона информационной панели документа

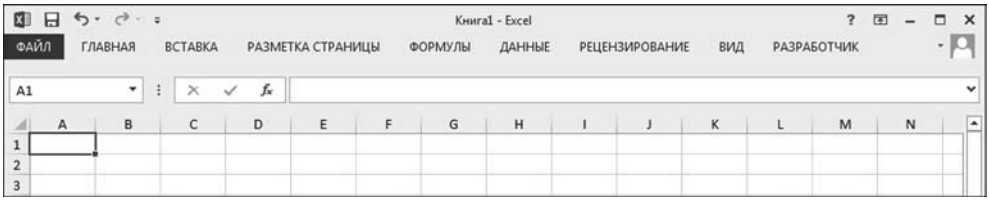


Рис. 2.9
Лента в свернутом виде

Это был обзор вкладок ленты приложения. Несколько слов скажем относительно ленты как таковой. Например, если на любой из вкладок ленты выполнить двойной щелчок мышью, вкладка свернется так, что отображаться будут только корешки ее вкладок, как показано на рисунке 2.9.

Это достаточно удобный режим, особенно если учесть, что после щелчка на корешке одной из вкладок лента выпадает вниз в рабочую область, как показано на рисунке 2.10.

Причем именно выпадает, а не раскрывается. Она при этом перекрывает строку формул и начальные строки рабочей области. После того как на вкладке ленты выбрана нужная пиктограмма, лента снова окажется свернутой. Перейти в режим развернутой ленты можно, либо выполнив двойной щелчок на корешке вкладки, либо воспользовавшись специальной пиктограммой в правом верхнем углу ленты (рис. 2.11).

Пиктограмма представляет собой раскрывающийся список с тремя командами. Выбрав команду **Автоматически скрывать ленту**, переходим в режим, при котором окно приложения отображается со скрытой лентой. В режиме команды **Показать вкладки** отображаются только корешки вкладок

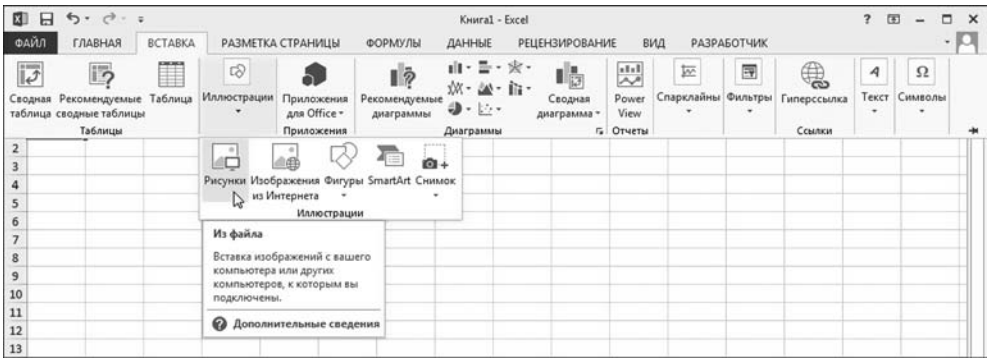


Рис. 2.10
При щелчке на корешке вкладки свернутой ленты она выпадает в рабочую область

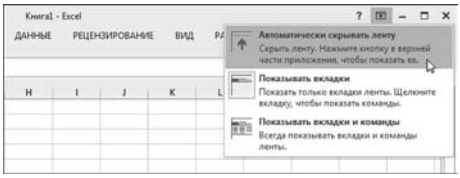


Рис. 2.11
Пиктограмма позволяет развернуть ленту

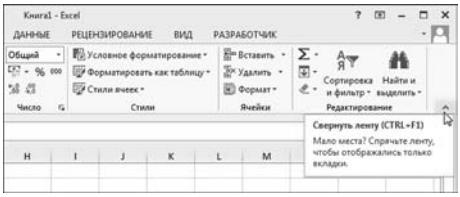


Рис. 2.12
Пиктограмма позволяет свернуть ленту

ленты. Чтобы лента отображалась полностью, выбираем команду **Показать вкладки и команды**.

Для перехода в режим свернутой ленты также может использоваться пиктограмма в правом нижнем углу ленты (рис. 2.12).

Кроме того, свернуть или развернуть ленту поможет комбинация клавиш <Ctrl>+<F1>.

На заметку

Ленту можно настраивать. Как это делается, рассказано в одном из следующих разделов этой главы.

МАСШТАБ ОТОБРАЖЕНИЯ

Такое случалось в моей практике
и приносило плоды.

Из к/ф «Формула любви»

Масштаб отображения рабочей области изменяется достаточно легко. Самый простой способ — воспользоваться ползунком выбора масштаба в строке состояния (рис. 2.13).

Рис. 2.13
Ползунок выбора масштаба в строке состояния

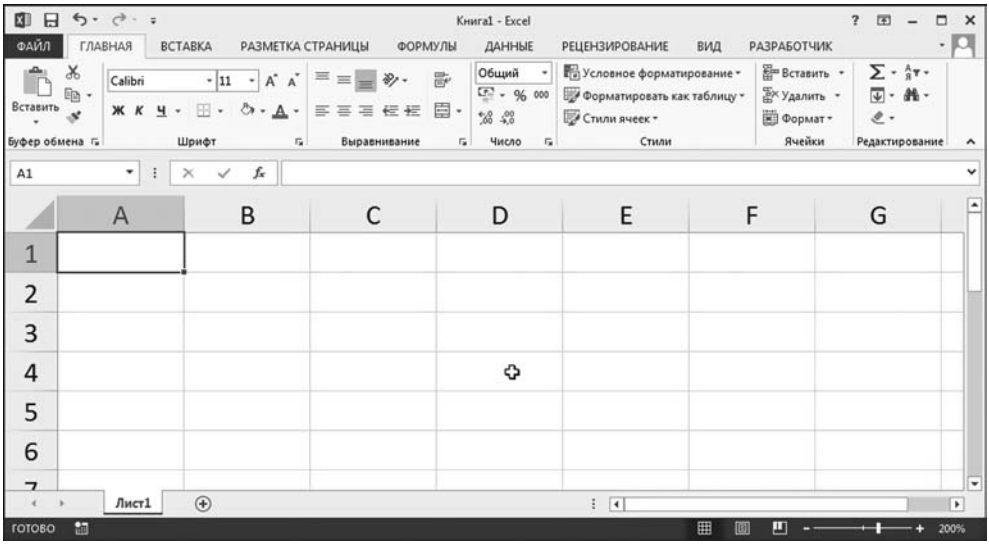


Рис. 2.14
Масштаб отображения **200%**

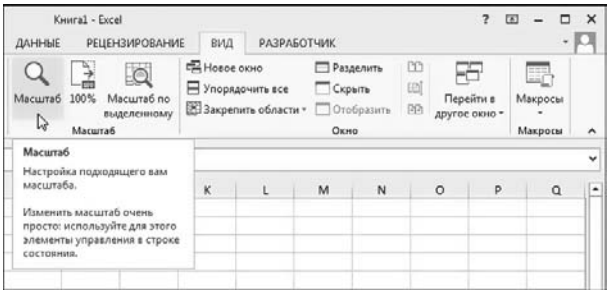


Рис. 2.15
Выбор масштаба отображения
с помощью пиктограммы на ленте

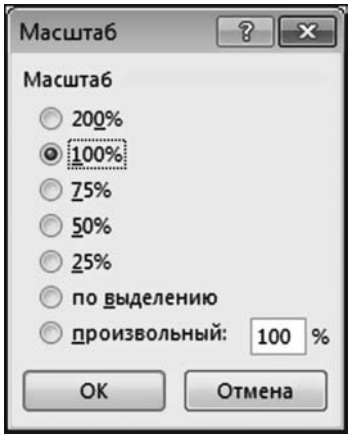


Рис. 2.16
Окно выбора масштаба

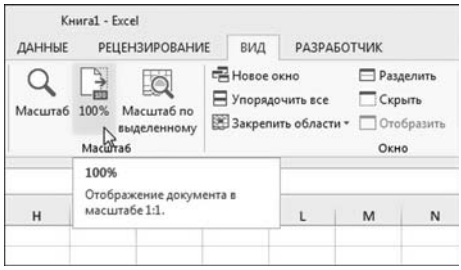


Рис. 2.17
Пиктограмма установки стандартного режима масштабирования

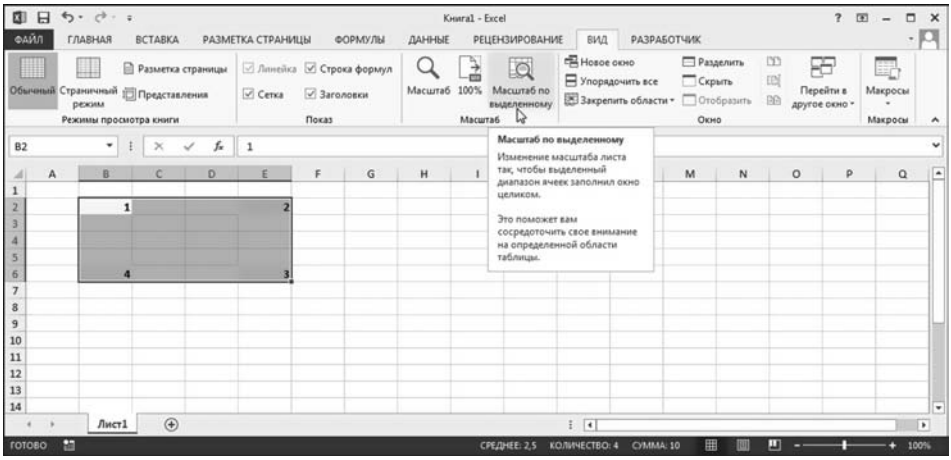


Рис. 2.18
Пиктограмма установки масштаба по выделенной области

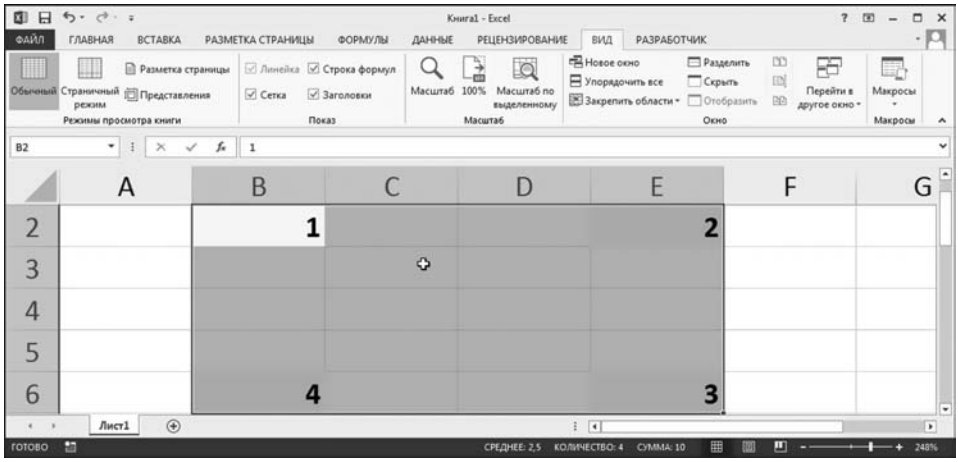


Рис. 2.19
Результат установки масштаба отображения по выделенной области

Перетаскивают или непосредственно ползунок, или щелкают на пиктограммах со знаками «плюс» (увеличение масштаба) и «минус» (уменьшение масштаба). Сам масштаб по умолчанию отображается слева от ползунка (в процентах). Окно приложения с установленным масштабом **200%** показано на рисунке 2.14.

Выбранный масштаб применяется только к текущему рабочему листу. Для остальных листов используется масштаб отображения по умолчанию (т. е. **100%**).

Для изменения масштаба отображения используется также вкладка **Вид**. На этой вкладке в группе **Масштаб** имеется несколько пиктограмм, полезных для настройки масштаба отображения. Так, установить произвольный масштаб можно, щелкнув по одноименной пиктограмме в группе **Масштаб** (рис. 2.15).

В результате откроется диалоговое окно **Масштаб**, представленное на рисунке 2.16.

Масштаб можно выбрать из списка предлагаемых значений или указать собственный в поле **произвольный**. Полезна также пиктограмма **100%** в группе **Масштаб** (см. рис. 2.17).

Щелчок на ней приводит к отображению рабочего листа в стандартном масштабе отображения, т. е. в масштабе **100%** (он же называется масштабом 1:1). Это удобно, поскольку всего одним щелчком (или точнее двумя —

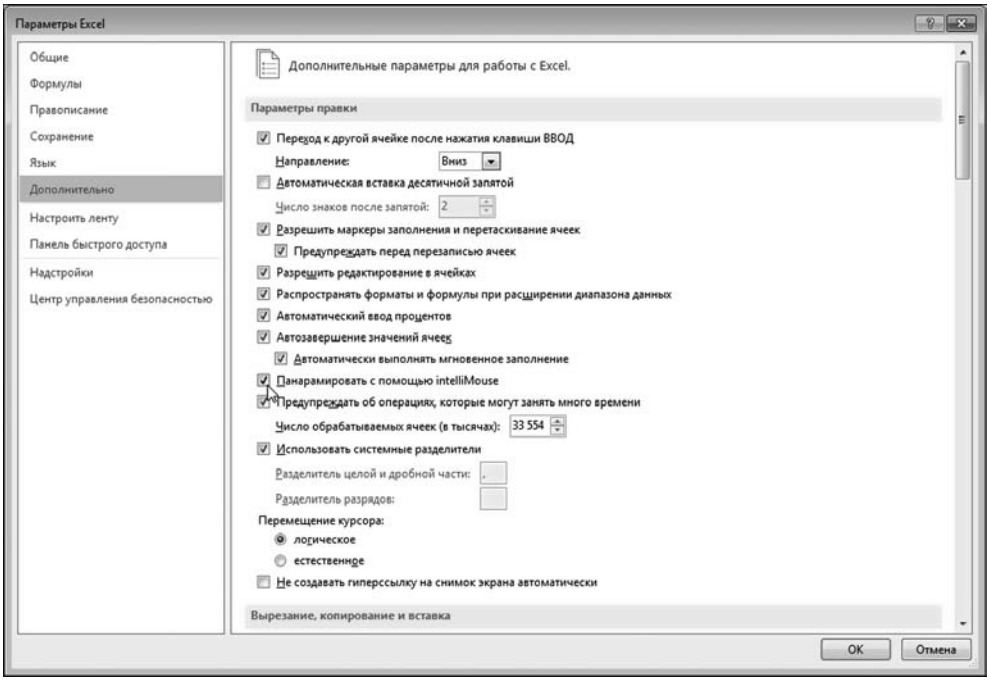


Рис. 2.20
Переход в режим изменения масштаба с помощью колесика мыши IntelliMouse

сначала нужно открыть вкладку **Вид**) рабочий лист приводится к нормальному виду.

Пиктограмма **Масштаб по выделенному** позволяет установить масштаб так, что в рабочей области отображается предварительно выделенный диапазон ячеек. Другими словами, если выделить диапазон ячеек и после этого щелкнуть на пиктограмме **Масштаб по выделенному**, то автоматически вычисляется и применяется такой масштаб, что рабочую область занимает выделенный диапазон. На рисунке 2.18 в рабочем документе выделен диапазон ячеек B2:E6, после чего выполняется щелок на пиктограмме **Масштаб по выделенному** в группе **Масштаб** вкладки **Вид**.

Результат представлен на рисунке 2.19.

Весьма удобен следующий нехитрый прием:

- выделяем диапазон ячеек;
- переходим в режим масштабирования по выделенному фрагменту (пиктограмма **Масштаб по выделенному**);
- выполняем необходимые манипуляции (например, вводим данные или просто просматриваем содержимое ячеек — в зависимости от потребностей);
- наконец, после всего этого возвращаемся к стандартному масштабу (пиктограмма **100%**).

Обладатели колесной мыши *IntelliMouse* могут воспользоваться еще одним удобным способом изменения масштаба: вращением колеса. Для этого необходимо перейти в соответствующий режим. Открываем вкладку **Файл**, на которой выбираем команду **Параметры** и в открывшемся окне настроек приложения **Параметры Excel** переходим в раздел **Дополнительно**. В этом разделе в группе **Параметры правки** есть опция **Панорамирование с помощью IntelliMouse**, для которой необходимо установить флажок (рис. 2.20).

После этого, вращая колесо мыши в одном направлении, масштаб отображения увеличиваем, вращая в другом, — уменьшаем (курсор мыши находится в рабочей области). Очень удобно.

ЦВЕТОВАЯ СХЕМА

— Это Жазель, француженка. Я признал ее по ноге.
— Нет, это не Жазель. Жазель была брунетка.
А эта — вся белая!

Из к/ф «Формула любви»

В Excel есть три цветовые схемы отображения рабочего окна, которые выбираются пользователем. Выбор цветовой схемы выполняется в окне настроек приложения **Параметры Excel**, в разделе **Общие**. В этом разделе есть раскрывающийся список **Тема Office** (в предыдущих версиях Excel список **Цветовая схема**). В версии Excel 2013 список содержит три позиции: **Белая**, **Светло-серая** и **Темно-серая** (в предыдущих версиях **Синяя**, **Серебристая** и **Черная**). Цветовая схема выбирается в этом списке (см. рис. 2.21).

Здесь и далее в книге использована схема **Светло-серая.** На рисунке 2.22 показано, как будет выглядеть рабочее окно при использовании цветовой схемы **Темно-серая.**

Цветовая схема определяет, в каком тоне, с каким доминирующим цветом отображаются в рабочем окне приложения элементы интерфейса. Выбор схемы — дело вкуса, не больше.

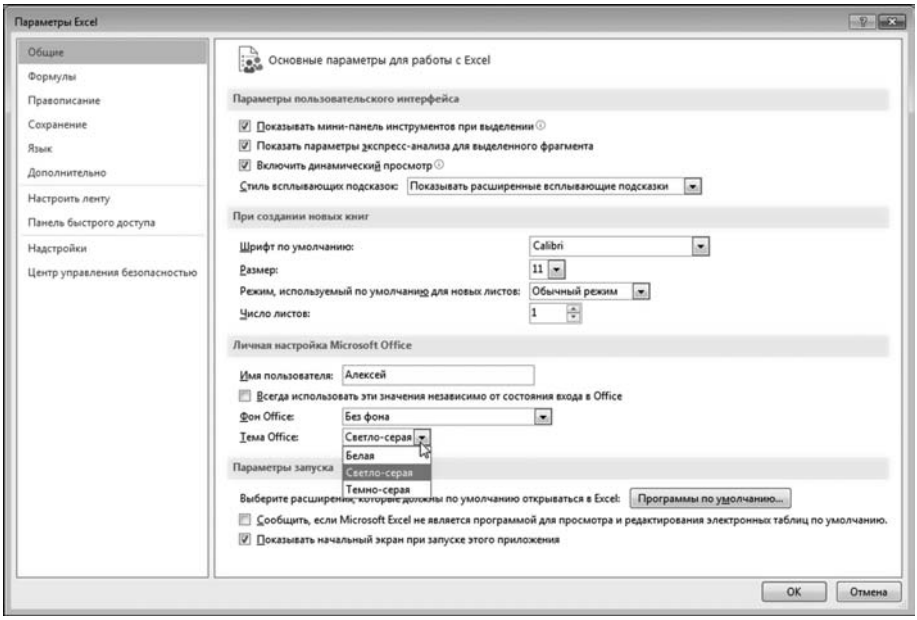


Рис. 2.21
Выбор цветовой схемы

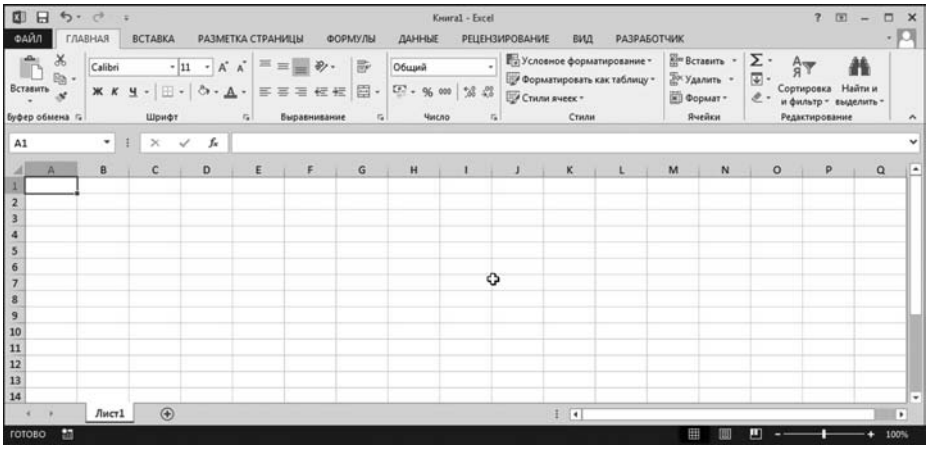


Рис. 2.22
Использована цветовая схема **Темно-серая**

ОПЕРАЦИИ С ЛИСТАМИ КНИГИ

Русалки оставляют свои хвосты и вливаются в коллектив!

Из к/ф «Старый знакомый»

Традиционно книга Excel содержит три рабочих листа. Листы в книге можно добавлять и удалять, изменять их названия (по умолчанию они называются **Лист1**, **Лист2** и т. д.). Можно также изменить настройки приложения так, что в новых книгах листов будет больше или меньше трех. Настройки количества листов в рабочей книге по умолчанию выполняются в окне настроек приложения **Параметры Excel** в разделе **Общие** (рис. 2.23).

В группе **При создании новых книг** в списке **Число листов** необходимо указать количество листов в новых рабочих книгах (которые создаются на основе стандартного шаблона). Обычно, как отмечалось, книга состоит из трех листов. Если установить значение равным, например, десяти, то новая книга будет содержать именно столько рабочих листов (рис. 2.24).

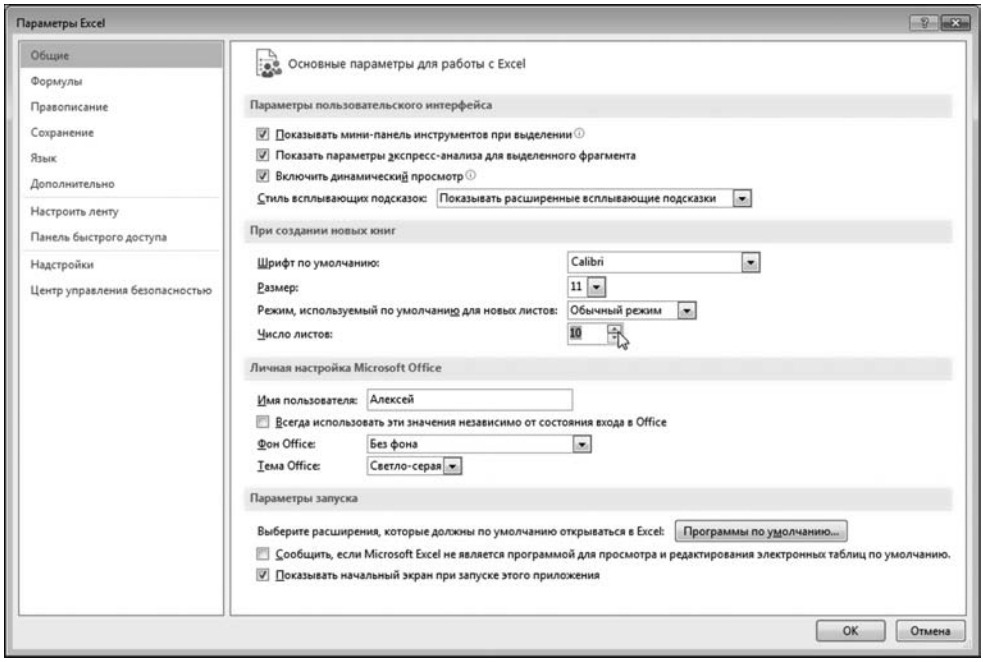


Рис. 2.23
Изменение количества рабочих листов в книге по умолчанию

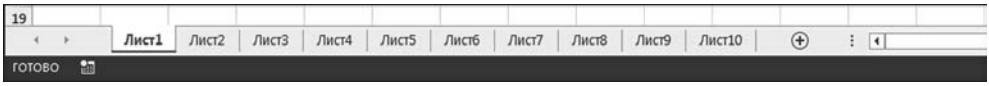
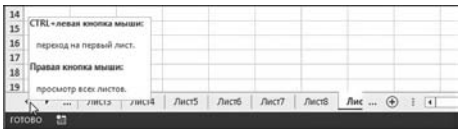
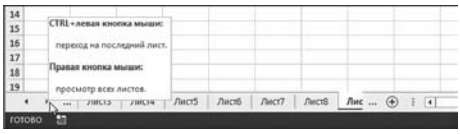


Рис. 2.24
Рабочая книга содержит по умолчанию десять рабочих листов

Таблица 2.9

Пиктограммы прокрутки листов книги

Пиктограмма	Назначение
	Переход в начало списка корешков рабочих листов (в Excel 2010 и более ранних версиях)
	Прокрутка списка корешков рабочих листов в направлении корешка первого листа (в Excel 2010 и более ранних версиях)
	Прокрутка списка корешков рабочих листов в направлении корешка последнего листа в (Excel 2010 и более ранних версиях)
	Переход в конец списка корешков рабочих листов (в Excel 2010 и более ранних версиях)
	Раскрывающийся список пиктограммы прокрутки корешков листов в направлении первого листа (в Excel 2013)
	Раскрывающийся список пиктограммы прокрутки корешков листов в направлении последнего листа (в Excel 2013)

При работе с книгами, содержащими большое количество листов, полезными будут пиктограммы прокрутки корешков рабочих листов. Панель с этими пиктограммами расположена слева от списка корешков листов. Надо отметить, что в Excel 2013, по сравнению с предыдущими версиями, произошли некоторые изменения. В Excel 2013 пиктограмм для работы с корешками листов стало меньше, но они стали функциональнее. Пиктограммы для Excel 2013 и более ранних версий приложения описаны в таблице 2.9.

Порядок листов в книге меняется очень просто: достаточно мышью перетащить корешок листа в нужное место, как показано на рисунке 2.25.

Курсор мыши наводится на корешок листа, нажимается левая кнопка мыши и, удерживая ее нажатой, корешок перетаскиваете куда нужно. Если при этом удерживать нажатой клавишу <Ctrl>, будет создана копия листа. В отличие от простого перемещения листа, при копировании в области курсора отображается маленький знак «плюс», как на рисунке 2.26.

Вставить новый рабочий лист в конец списка рабочих листов можно с помощью специальной пиктограммы, которая находится в самом конце списка корешков рабочих листов, как показано на рисунке 2.27.

На заметку

Для этой же цели используют комбинацию клавиш <Shift>+<F11>.

Основные операции с рабочими листами выполняются через команды контекстного меню. Чтобы увидеть его, щелкаем правой кнопкой мыши в области корешка рабочего листа (рис. 2.28).

Рис. 2.25
Перемещение листа

Рис. 2.26
При копировании листа необходимо удерживать нажатой клавишу <Ctrl>

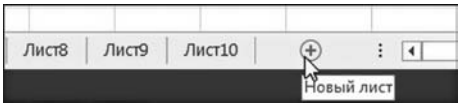


Рис. 2.27
Вставка нового листа с помощью специальной пиктограммы

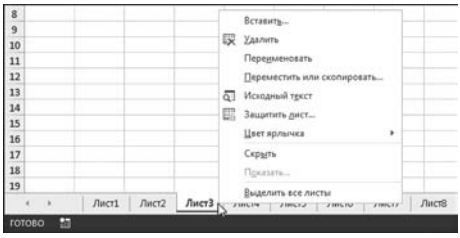


Рис. 2.28
Контекстное меню корешка рабочего листа

Рис. 2.29
В окне **Вставка** для вставки листа выбираем пиктограмму **Лист**

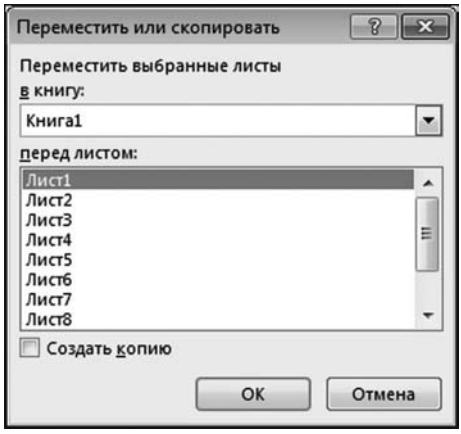


Рис. 2.30
В окне **Переместить или скопировать** задаются параметры перемещения и копирования листов

Таблица 2.10

Контекстное меню корешка рабочего листа

Команда	Назначение
Вставить	Вставка нового рабочего листа
Удалить	Удаление рабочего листа
Переименовать	Переименование рабочего листа
Переместить или скопировать	Перемещение или копирование рабочего листа
Исходный текст	Редактирование программного кода макросов рабочего листа
Защитить лист	Выбор режима и настроек защиты рабочего листа
Цвет ярлычка	Выбор цвета корешка рабочего листа
Скрыть	Скрытие рабочего листа
Показать	Отображение скрытого рабочего листа
Выделить все листы	Выделение корешков всех рабочих листов книги

Контекстное меню содержит весь джентльменский набор, необходимый для управления рабочими листами книги. Назначение команд контекстного меню описывается в таблице 2.10.

Например, если воспользоваться командой **Вставить**, откроется диалоговое окно **Вставка**, в котором для вставки нового листа в книгу следует выбрать пиктограмму **Лист** (рис. 2.29).

Выполнение команды **Переместить или скопировать** контекстного меню приводит к отображению одноименного диалогового окна, в котором задаются параметры перемещения или копирования рабочего листа (рис. 2.30).

В раскрывающемся списке **в книгу** выбирается рабочая книга, в которую копируется или перемещается рабочий лист. В списке **перед листом** выбирается лист книги, перед которым размещается перемещаемый или копируемый лист. По умолчанию выполняется перемещение листа. Для копирования необходимо установить флажок опции **Создать копию**.

Изменение названия листа также происходит просто: полезной окажется команда контекстного меню **Переименовать** или двойной щелчок в области корешка листа. В результате текст в корешке с названием листа выделяется для редактирования (рис. 2.31).

Команда контекстного меню **Защитить лист** предназначена для настройки параметров защиты рабочего листа. Параметры защиты настраиваются в окне **Защита листа**, показанном на рисунке 2.32.

Те же операции могут выполняться с помощью пиктограмм вкладки **Главная** ленты (группа **Ячейки**). Например, раскрывающийся список пиктограммы **Вставить** содержит команду вставки нового рабочего листа **Вставить лист** (рис. 2.33).

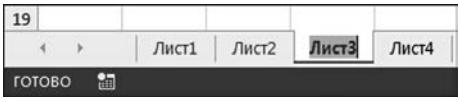


Рис. 2.31
Изменение названия листа

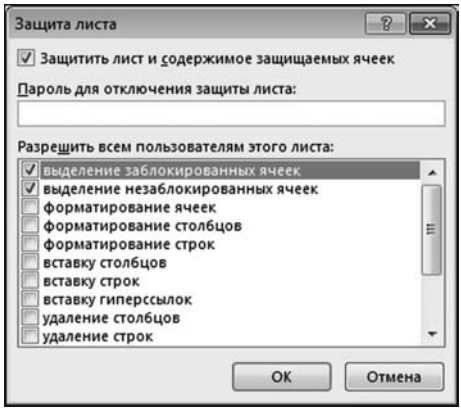


Рис. 2.32
В окне **Защита листа** выполняются настройки по защите рабочего листа

В раскрывающемся списке пиктограммы **Удалить** на вкладке **Главная** содержится команда **Удалить лист** для удаления рабочего листа (рис. 2.34).

Команды для скрытия и отображения рабочих листов, изменения их названий, перемещения и копирования, установки цвета корешка и настройки режима защиты содержатся в раскрывающемся списке пиктограммы **Формат** (рис. 2.35).

В данном случае четко проявляется базовый принцип приложений Microsoft Office: всегда имеется, по меньшей мере, несколько различных способов для выполнения базовых операций. В частности, скрыть рабочий лист (т. е. не удалять лист, а просто не отображать) можно с помощью команды меню **Скрыть или отобразить** в списке ко-

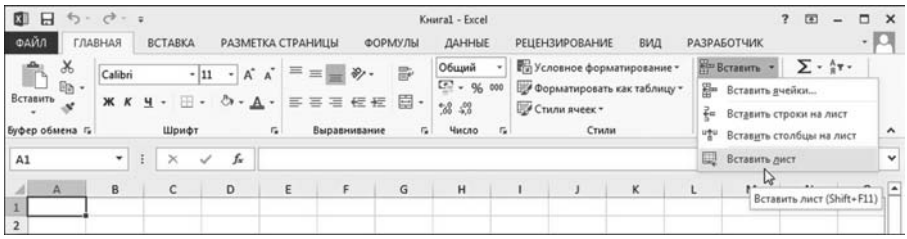


Рис. 2.33
В раскрывающемся списке пиктограммы **Вставить** на вкладке **Главная** содержится команда **Вставить лист** для вставки нового рабочего листа

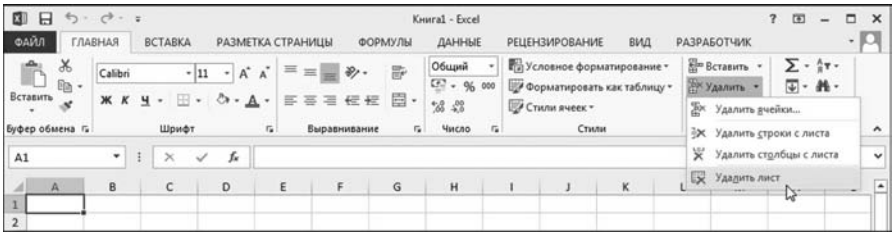


Рис. 2.34
Команда **Удалить лист** используется для удаления рабочего листа

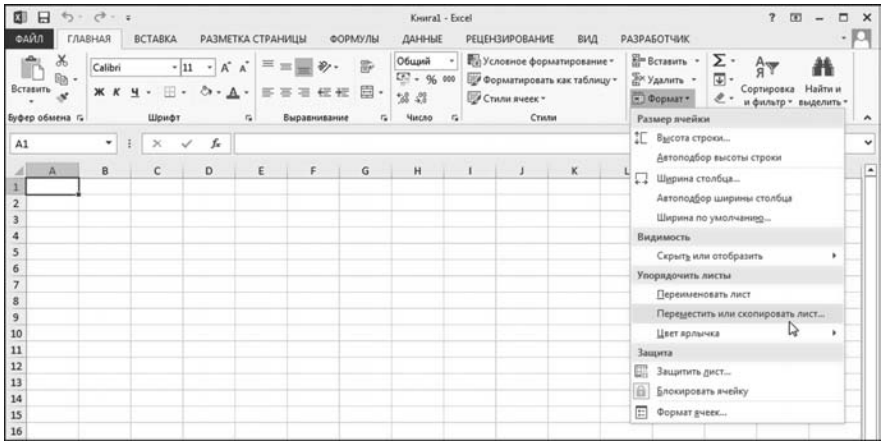


Рис. 2.35
В раскрывающемся списке пиктограммы **Формат** на вкладке **Главная** содержится ряд команд для работы с рабочими листами

манд пиктограммы **Формат** (рис. 2.35) или команды **Скрыть** контекстного меню (см. рис. 2.36).

В результате, хотя лист и не удаляется, на экране он не виден (рис. 2.37).

Вернуть лист на прежнее место можно с помощью все той же команды-меню **Скрыть или отобразить** в списке команд пиктограммы **Формат** (рис. 2.35) или команды **Показать** контекстного меню (рис. 2.38).

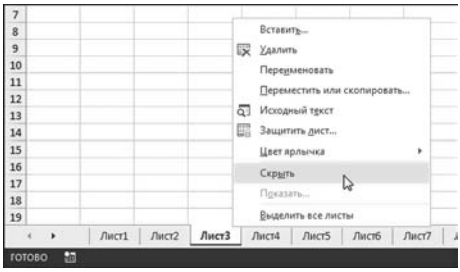


Рис. 2.36
Выбор в контекстном меню для листа **Лист3** команды **Скрыть** для сокрытия рабочего листа

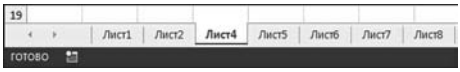


Рис. 2.37
Третий рабочий лист **Лист3** скрыт

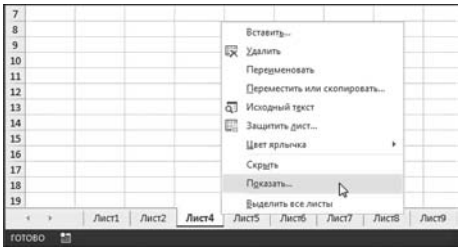


Рис. 2.38
В контекстном меню выбирается команда **Показать** для отображения скрытых листов

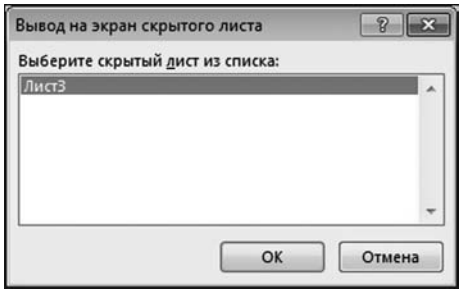


Рис. 2.39
Окно выбора скрытого рабочего листа для отображения

В последнем случае на экране появится окно **Вывод на экран скрытого листа**, в котором необходимо выбрать лист (из списка скрытых листов) для отображения (рис. 2.39).

Существует ряд других интересных возможностей, связанных с рабочими листами. Например, с помощью команды **Цвет ярлычка** можно раскрасить цветом корешки рабочих листов. Для этого в раскрывающейся цветовой палитре достаточно выбрать цвет для корешка листа.

СТРОКА СОСТОЯНИЯ

А что Вы на меня так смотрите, отец родной?
На мне узоров нет, и цветы не растут.

Из к/ф «Иван Васильевич меняет профессию»

Часто строкой состояния пренебрегают. А вместе с тем это очень эффективный и функциональный элемент интерфейса. Особенно если ее правильно настроить. Чтобы оценить, что же может отображаться в строке состояния, достаточно щелкнуть правой кнопкой мыши в области строки состояния. Вниманию читателя предстанет внушительное контекстное меню, как показано на рисунке 2.40.

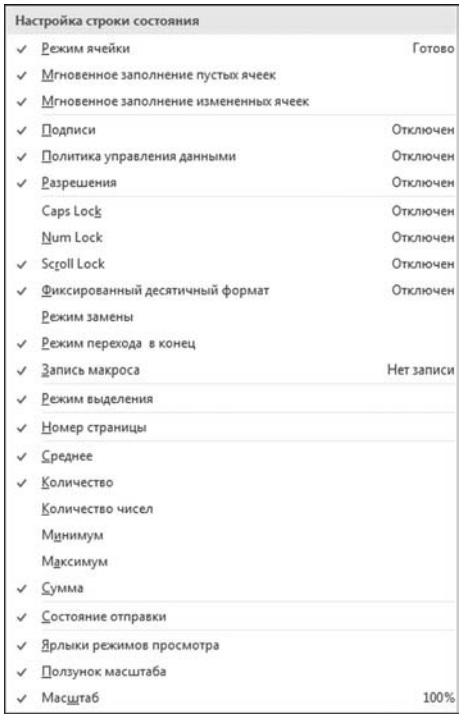


Рис. 2.40
Контекстное меню строки состояния

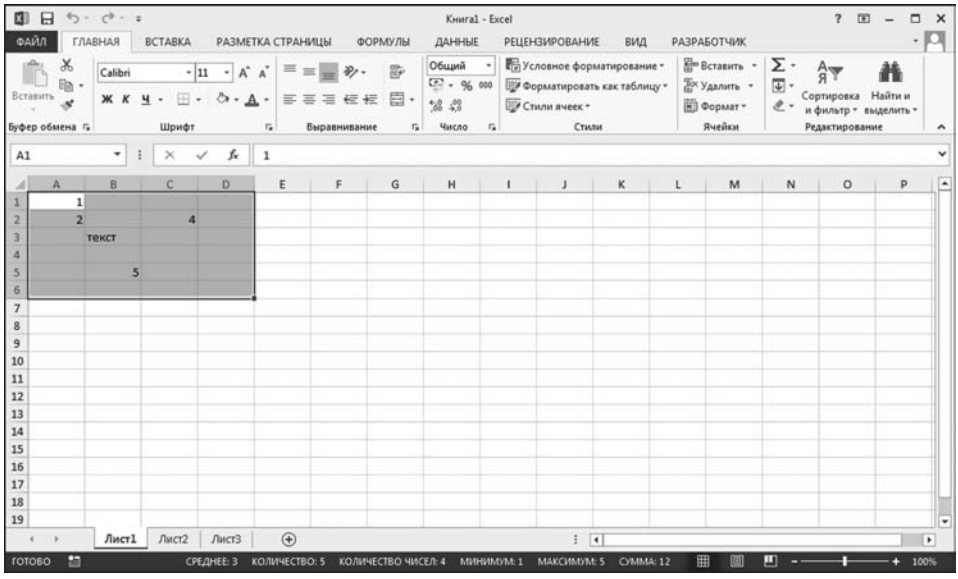


Рис. 2.41
В строке состояния отображается полезная информация

Контекстное меню содержит список параметров, показателей и индикаторов, которые отображаются в строке состояния. Среди них, кстати, индикатор масштаба и ползунок изменения масштаба. Устанавливая или отменяя флажки у позиций контекстного меню строки состояния, выполняют настройку строки. Например, если установить флажки в группе с командами **Среднее**, **Количество**, и т. д., то при выделении диапазона ячеек в рабочем листе в строке состояния будут отображаться значения соответствующих показателей (среднее значение в ячейках, количество ячеек со значениями, и т. д.). Возможная ситуация проиллюстрирована на рисунке 2.41.

В данном случае выделенный диапазон A1:D6 содержит всего пять непустых ячеек (четыре значения — числовые, одно — текстовое). В строке состояния для этого диапазона отображается среднее значение, количество ячеек со значениями, количество ячеек с числами, минимальное, максимальное значение, а также сумма числовых значений.

НАСТРОЙКА ПАНЕЛИ БЫСТРОГО ДОСТУПА

Если бы меня не подавляла бухгалтерия,
я бы потряс всю местную общественность.

Из к/ф «Старый знакомый»

На панели быстрого доступа размещаются пиктограммы для команд, которые используются наиболее часто. Есть команды, которые используют все (например, создание нового документа или открытие существующего, сохранение документа, вывод на печать и ряд других). Также у каждого пользователя имеется свой персональный набор любимых команд, которые в стандартной настройке приложения могут быть запрятаны очень далеко. Самый простой и очевидный пример — кнопки запусков макросов пользователя. Все это функциональное счастье удобно разместить на панели быстрого дос-

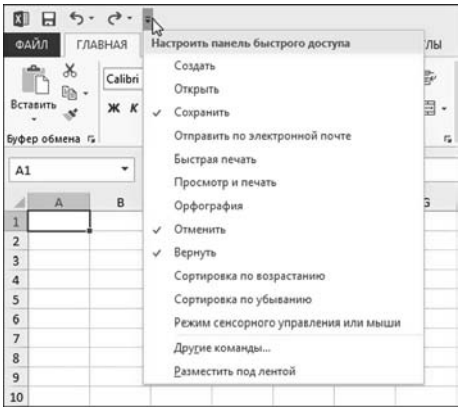


Рис. 2.42
Команды раскрывающегося списка панели быстрого доступа

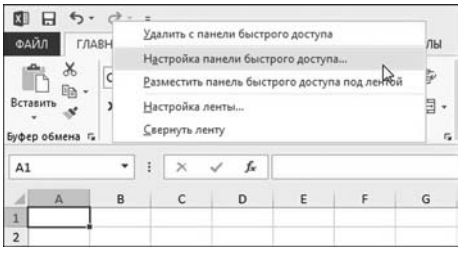


Рис. 2.43
Команды контекстного меню панели быстрого доступа

тупа. В первую очередь стоит обратить внимание на раскрывающийся список панели быстрого доступа. Чтобы раскрыть список следует щелкнуть на пиктограмме в виде направленной вниз стрелки в правой части панели быстрого доступа (рис. 2.42).

В верхней части списка представлен список команд, пиктограммы для которых можно разместить на панели быстрого доступа — достаточно лишь установить флажок у команды. Особняком стоят две последние команды списка. Команда **Другие команды** позволяет перейти к окну настроек приложения Excel для изменения параметров панели быстрого доступа. Команду **Разместить под лентой** используют для размещения панели быстрого доступа под лентой. Можно также воспользоваться контекстным меню панели быстрого доступа (рис. 2.43).

Интерес представляет команда **Настройка панели быстрого доступа**. После щелчка на ней открывается уже упомянутое выше окно настроек приложения **Параметры Excel**, открытое в разделе **Панель быстрого доступа** (рис. 2.44).

Основная область окна условно разделена на две части. Слева расположен раскрывающийся список **Выбрать команды из**, в котором выбирается категория команд, пиктограммы для которых предполагается добавлять на панель. Внизу под списком отображаются сами команды. В правой части имеется раскрывающийся список **Настройка панели быстрого доступа**, в котором задается область применения настроек для панели (для всех документов

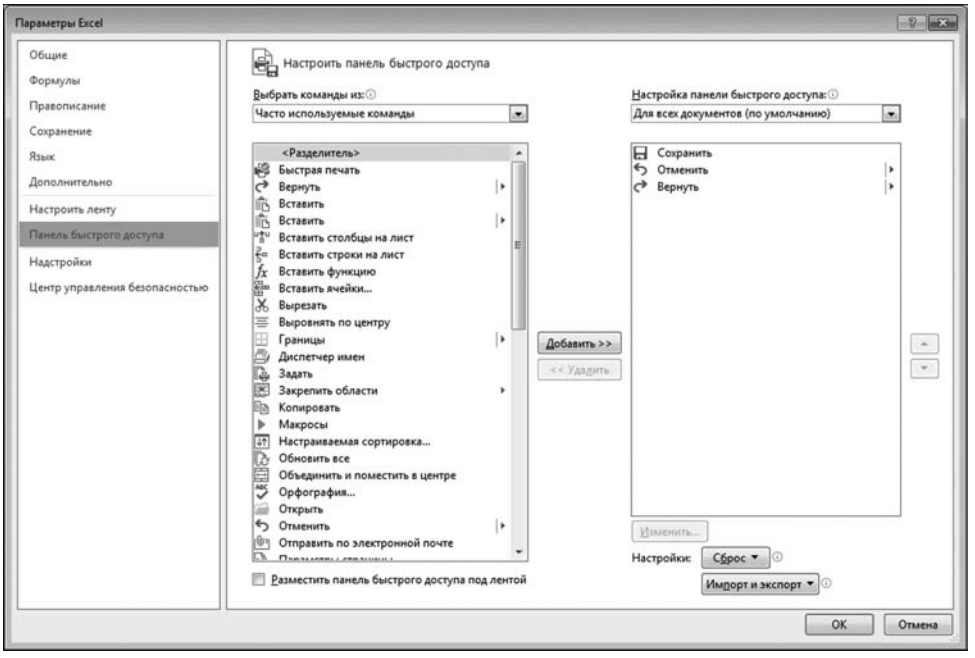


Рис. 2.44
Настройка панели быстрого доступа в окне **Параметры Excel**

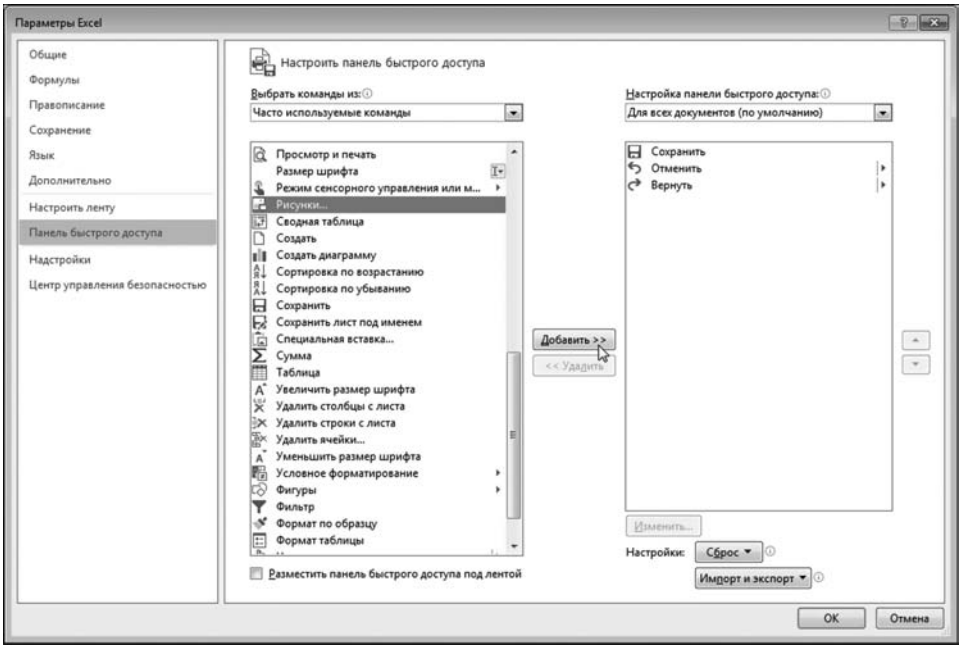


Рис. 2.45
Добавление кнопки на панель быстрого доступа

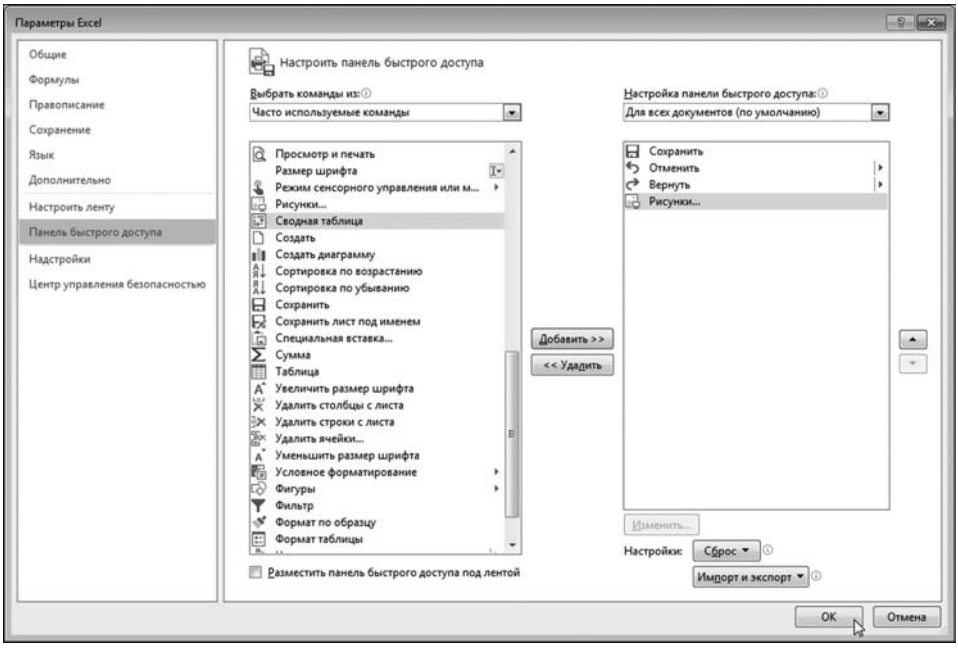


Рис. 2.46
Подтверждение выполненных настроек панели быстрого доступа

или только для активного документа). По умолчанию настройки являются глобальными, т. е. применяются ко всем документам.

Для добавления пиктограммы на панель быстрого доступа в левом списке выбирается добавляемая команда, после этого нажимаем кнопку **Добавить** между списками. На рисунке 2.45 на панель быстрого доступа добавляется пиктограмма команды вставки рисунка из файла.

Если все прошло нормально, команда появится в правом списке (рис. 2.46).

После подтверждения выполненных настроек, пиктограмма команды появляется на панели быстрого доступа (рис. 2.47).

Удаляются пиктограммы с панели так же просто, как и добавляются. Например, достаточно в области удаляемой пиктограммы раскрыть контекстное меню и выбрать команду **Удалить с панели быстрого доступа** (рис. 2.48).

Разумеется, можно для этой цели воспользоваться и окном настройки приложения. В этом случае в разделе **Панель быстрого доступа** в окне **Параметры Excel** (рис. 2.46) в правом списке выбирается удаляемая с панели пиктограмма и выполняется щелчок на кнопке **Удалить**.

Удобно то, что на панель быстрого доступа пиктограммы и даже целые группы пиктограмм можно добавлять прямо с вкладок ленты. Во-первых, это намного быстрее, чем искать нужные команды в списках окна настроек **Параметры Excel**. Во-вторых, на ленте нужные пиктограммы просто легче найти (конечно, если они там есть).

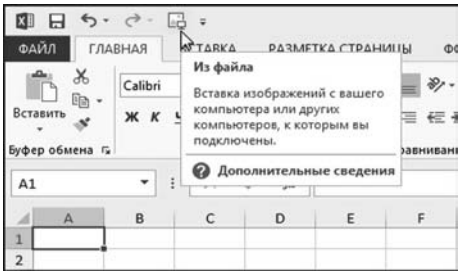


Рис. 2.47
На панель быстрого доступа добавлена кнопка

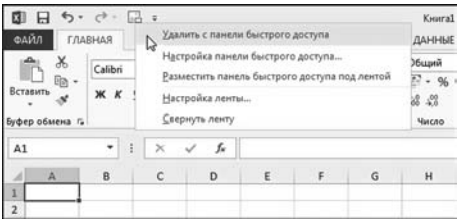


Рис. 2.48
Удаление кнопки с панели быстрого доступа

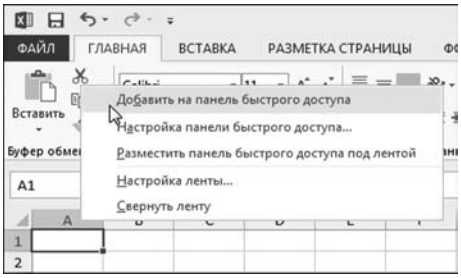


Рис. 2.49
Добавление кнопки ленты на панель быстрого доступа

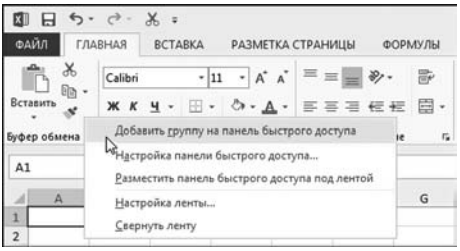


Рис. 2.50
Добавление группы ленты на панель быстрого доступа

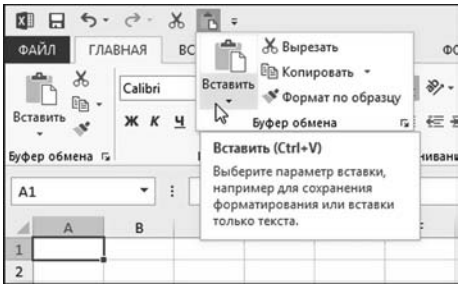


Рис. 2.51
На панель быстрого доступа добавлена группа

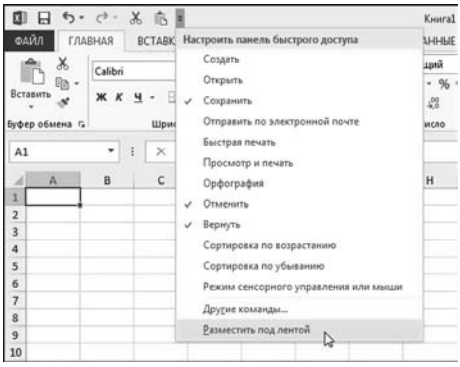


Рис. 2.52
Размещение панели быстрого доступа под лентой

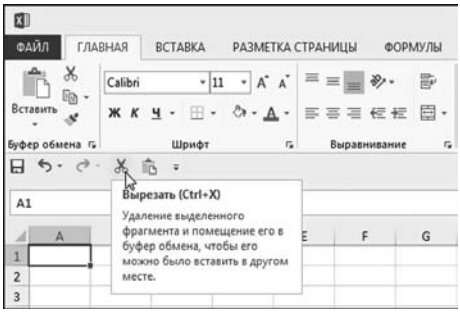


Рис. 2.53
Панель быстрого доступа размещена под лентой

На рисунке 2.49 показано, как на панель быстрого доступа добавляется пиктограмма с вкладки **Главная** ленты: на пиктограмму наводится курсор, открывается контекстное меню (щелчок правой кнопкой мыши), и в раскрывшемся списке выбирается команда **Добавить на панель быстрого доступа.**

Точно так же на панель добавляется группа пиктограмм: в области группы (но вне области пиктограмм) раскрывается контекстное меню и выбирается команда **Добавить группу на панель быстрого доступа** (рис. 2.50).

Результат добавления пиктограммы удаления данных в буфер обмена и группы **Буфер обмена** на панель быстрого доступа показан на рисунке 2.51.

Помимо содержимого панели быстрого доступа, можно настраивать и ее расположение. А именно, кроме обычного положения, панель быстрого доступа можно разместить под лентой. Чтобы изменить положение панели, воспользуемся командой **Разместить под лентой** из списка панели быстрого доступа (рис. 2.52).

Отметим, что в окне **Параметры Excel** (рис. 2.46) есть опция **Разместить панель быстрого доступа под лентой**, которая служит той же цели. Результат манипуляций с панелью быстрого доступа показан на рисунке 2.53.

Вернуть панель на место поможет команда **Разместить над лентой** (появляется вместо команды **Разместить под лентой**) из списка панели (рис. 2.54).

Наконец, сбросить пользовательские настройки панели быстрого доступа можно с помощью кнопки **Сброс** в окне настроек **Параметры Excel** (рис. 2.55).

Панель быстрого доступа нам понадобится при работе с макросами: если макрос используется часто, удобно размещать пиктограмму запуска макроса на панели быстрого доступа.

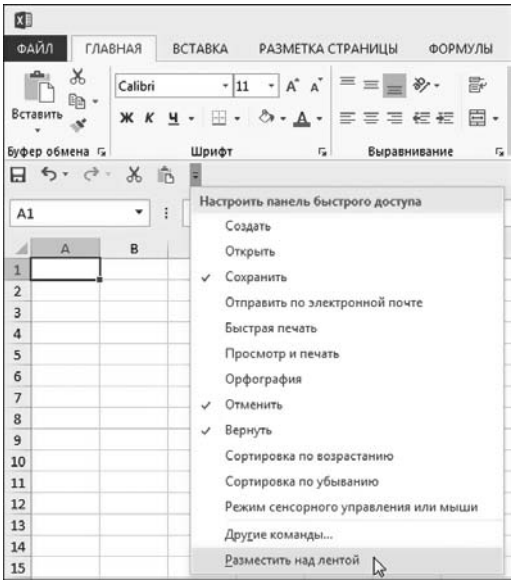


Рис. 2.54
Размещение панели
быстрого доступа над лентой

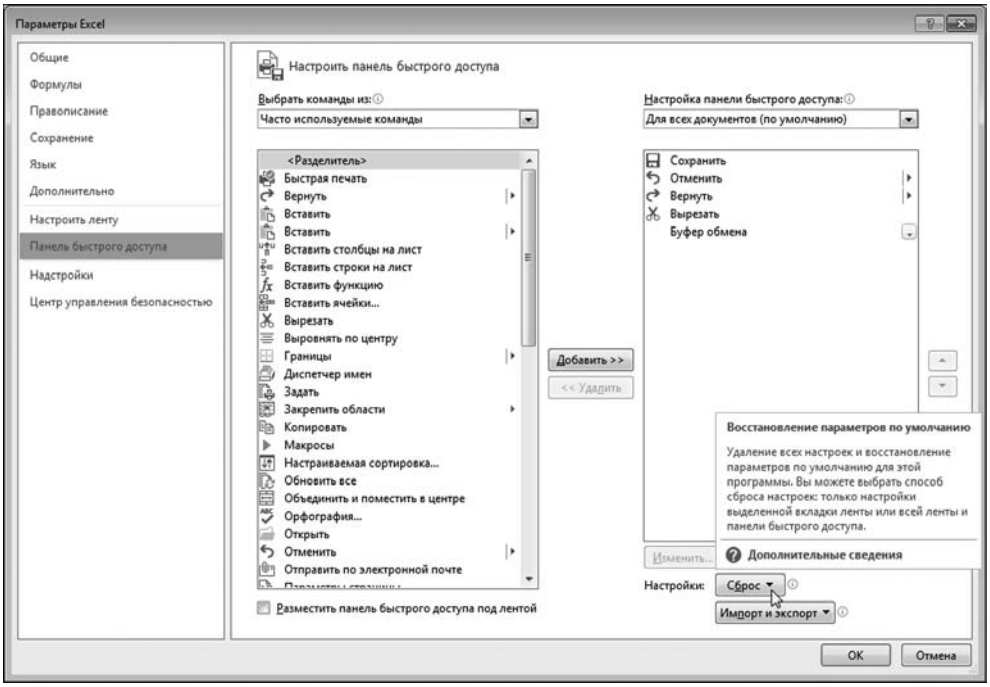


Рис. 2.55
Сброс настроек панели быстрого доступа

НАСТРОЙКА ЛЕНТЫ

Это вопрос очень деликатный и где-то очень огорчительный.
Меня лично это просто убивает.

Из к/ф «Гараж»

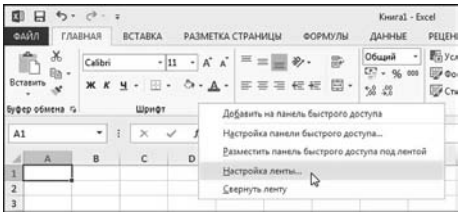


Рис. 2.56
В контекстном меню выбираем команду настройки ленты

Так же просто, как и панель быстрого доступа, настраивается лента приложения. Напомним, что лента появилась в версии Excel 2007, а возможность полноценно настраивать ленту — в версии Excel 2010.

Для настройки ленты используем команду **Настройка ленты** из контекстного меню (рис. 2.56).

На заметку

Обращаем внимание читателя, что в том же списке присутствует команда **Свернуть ленту**, с помощью которой на самом деле можно свернуть ленту. Кстати, развернуть ленту можно убрав флажок возле этой команды в контекстном меню для свернутой ленты.

Настройка ленты выполняется в разделе **Настройка ленты** окна **Параметры Excel** (рис. 2.57).

Рис. 2.57
Окно **Параметры Excel** открыто в разделе **Настройка ленты**

Окно откроется автоматически при выполнении команды **Настройка ленты** из контекстного меню, поэтому, строго говоря, можно просто открыть окно в нужном разделе без всяких команд контекстного меню. Как бы там ни было, дело все равно придется иметь с окном **Параметры Excel**.

Сам процесс настройки ленты очень напоминает настройку панели быстрого доступа: что-то куда-то нужно добавить или откуда-то удалить. Правда, вариантов здесь намного больше. В списке **Выбрать команды** выбирается группа команд для добавления. Внизу под списком отображаются команды — потенциальные кандидаты на добавление в ленту. В списке **Настройка ленты** выбирается категория вкладок, которые предполагается настраивать. Внизу в списке отображается структура ленты: вкладки, группы и пиктограммы. Кнопками **Добавить** и **Удалить**, которые находятся между списками, выполняется настройка. Кнопки **Создать вкладку**, **Создать группу** и **Переименовать** предназначены соответственно для создания пользователем вкладок и групп, а также для их переименования. Сброс настроек выполняется с помощью кнопки **Сброс**, а кнопка **Импорт-экспорт** предназначена для импорта и экспорта настроек. Две кнопки со стрелками в правой части окна **Параметры Excel** позволяют изменять порядок следования структурных элементов на ленте.

РЕЖИМЫ ВЫЧИСЛЕНИЙ И ИНДИКАЦИЯ ОШИБОК

Голова все может. В особенности,
если этого голова Великого Магистра. Я прав?

Из к/ф «Формула любви»

Здесь мы немножко поговорим о том, как все-таки Excel считает, т. е. вычисляет значения в ячейках. Хотя на первый взгляд может показаться, что с вычислениями в Excel все просто, это не совсем так. Точнее, если не вдаваться в подробности, то «правильный» ответ состоит в том, что действительно «все просто». Как только начинаешь вникать в детали, легкость обычно куда-то улетучивается.

По умолчанию в Excel установлен режим автоматического пересчета рабочего документа. Это означает, что если пользователь вносит изменения в содержимое ячеек документа, то все ячейки с формулами, которые напрямую или косвенно зависят от измененных ячеек, будут пересчитаны. Такой режим вполне логичен и оправдывает себя во многих практических случаях. Тем не менее иногда от него приходится отказываться. На причинах таких кардинальных решений останавливаться не будем — просто поверим, что такие случаи в жизни бывают.

Управление режимами вычислений выполняется по-разному. Например, можно воспользоваться пиктограммой **Параметры вычислений** в группе **Вычисление** на вкладке **Формулы**. Пиктограмма позволяет раскрыть список команд для выбора режима вычислений (рис. 2.58). Помимо режима автоматического пересчета рабочего листа, пользователю предоставляется возможность перейти в режим ручного пересчета документа или режим «почти автоматического» пересчета — в этом случае пересчитывается все, кроме таблиц данных.

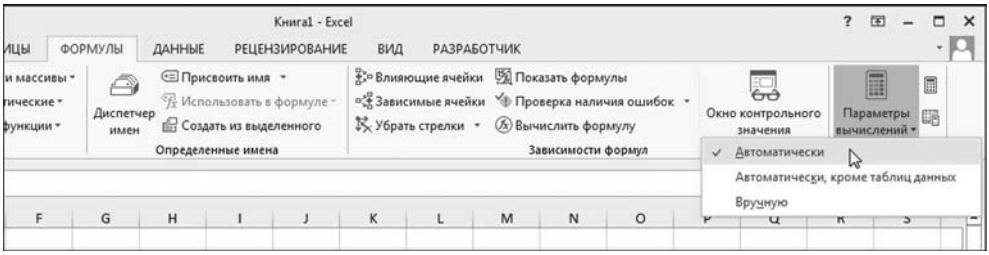


Рис. 2.58
Выбор режима вычислений с помощью пиктограммы
Параметры вычислений в группе **Вычисление** вкладки **Формулы**

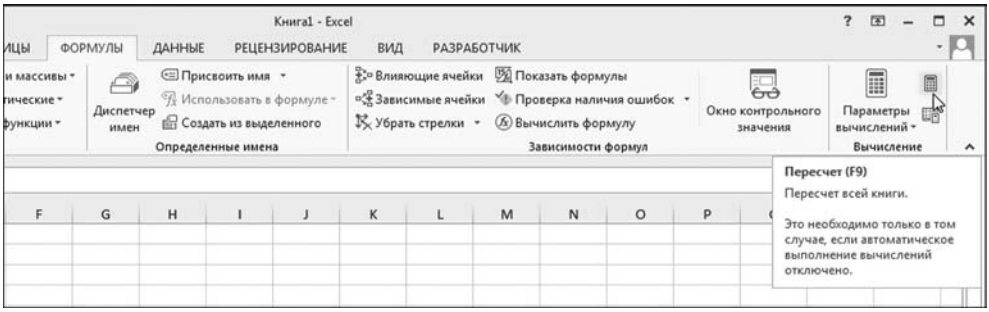


Рис. 2.59
Пересчет всей рабочей книги

На заметку

То, что здесь названо таблицей данных, в предыдущих версиях Excel называлось таблицей подстановок. Если коротко, то таблицы подстановок (таблицы данных в Excel 2010 и Excel 2013) позволяют получить набор расчетных значений, в которых варьируется один или два параметра. Для нас, для нашего дела, эти таблицы интереса не представляют.

Если мы переходим в режим «почти автоматических вычислений» (точнее, переводим в этот режим приложение), то пересчитывается в документе все, что не связано с таблицами данных. Все, что связано с таблицами данных, не вычисляется.

В режиме ручного пересчета документ придется пересчитывать каждый раз вручную — для пересчета всей рабочей книги щелкаем специальную пиктограмму в группе **Вычисление** или нажимаем клавишу <F9> (рис. 2.59).

Для пересчета только рабочего листа щелкаем другую пиктограмму, но тоже в группе **Вычисление** или нажимаем комбинацию клавиш <Shift>+<F9> (рис. 2.60).

Эти же режимы (имеется в виду режим автоматического/ручного пересчета документа) задаются в окне настроек приложения **Параметры Excel** в разделе **Формулы** (рис. 2.61).

Окно содержит несколько групп с опциями и переключателями. Некоторые из них представляют особый интерес. Режим вычислений устанавлива-

ется переключателем **Вычисления в книге**. У переключателя три положения, которые задают режим вычислений.

На заметку

Имеет смысл обратить внимание на опцию **Включить итеративные вычисления**. У опции интригующее название, хотя на практике все достаточно просто. Опция отвечает за вычисления по циклическим ссылкам, когда значение в вычисляемой ячейке зависит (прямо или косвенно) от значения в этой же

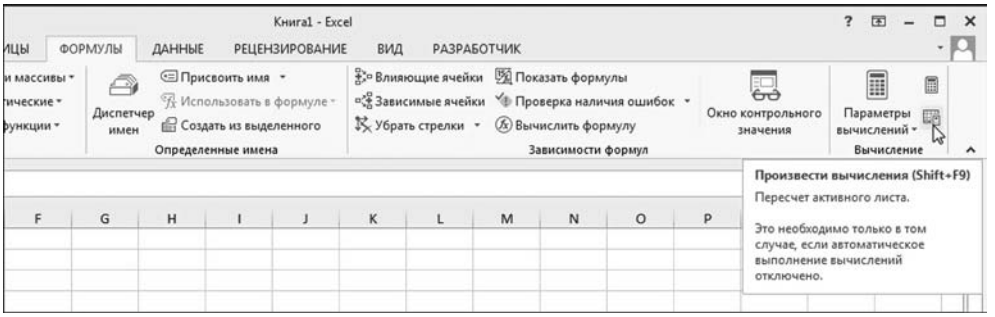


Рис. 2.60
Пересчет текущего рабочего листа

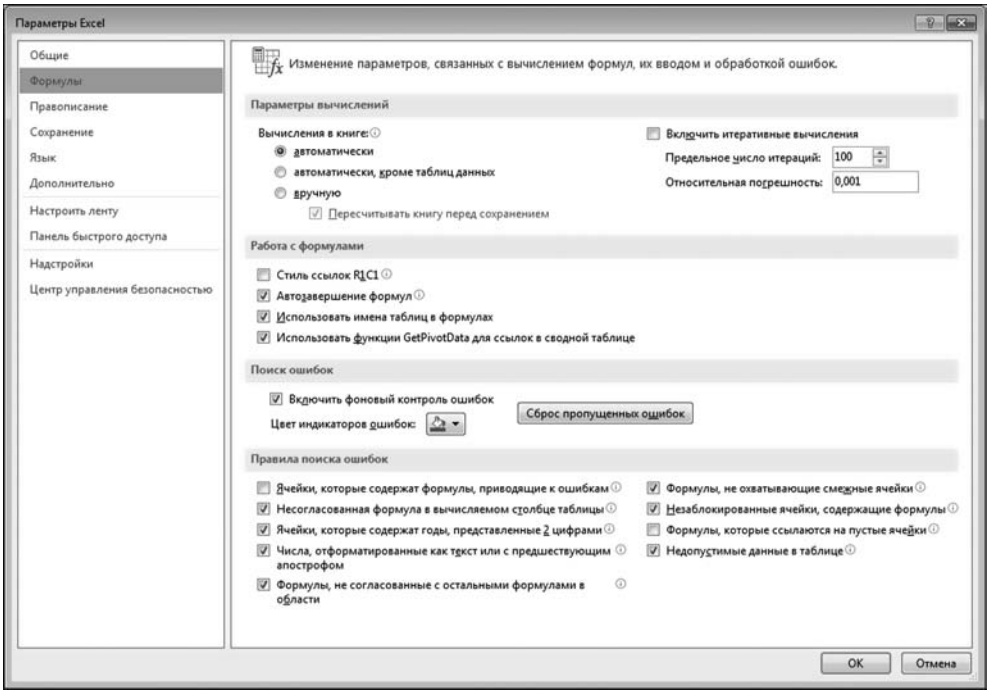


Рис. 2.61
Диалоговое окно **Параметры Excel** открыто в разделе **Формулы**

ячейке. По умолчанию такая ситуация называется циклической ссылкой. Если флажок опции **Включить итеративные вычисления** не установлен (а по умолчанию он не установлен), циклическая ссылка считается ошибкой. Обычно это так и есть — циклическая ссылка появляется в результате непродуманных действий пользователя. Если циклическая ссылка появилась в результате продуманных действий пользователя, то ошибкой она не является. Однако этот факт необходимо отметить флажком опции **Включить итеративные вычисления**.

Классическим примером ситуации, когда использование циклической ссылки себя полностью оправдывает, может быть процедура решения алгебраического уравнения методом последовательных итераций.

Также не стоит упускать из виду, что циклическая ссылка часто бывает непрямой, т. е. формула в ячейке не содержит ссылку на эту же ячейку, но при этом через цепочку ссылок в зависимых ячейках ссылается на себя же.

В разделе **Формулы** есть еще несколько полезных групп опций. Например, опция **Стиль ссылок R1C1** активируется для перехода в режим ссылок в формате R1C1.

На заметку

В режиме ссылок R1C1 ссылки на ячейки задаются в виде строка-столбец, т. е. указывается номер строки и номер столбца (а не имя столбца). Номер строки указывается после литеры **R**, а номер столбца — после литеры **C**. Например, адрес B3 в формате R1C1 будет выглядеть как R2C3.

Флажок опции **Автозавершение формул** лучше оставить — эта опция отвечает за режим отображения автоматической контекстной подсказки при вводе формул. Режим очень удобный и используется по умолчанию. При установленном флажке опции **Использовать имена таблиц в формулах** при работе с логическими таблицами ссылки на ячейки этих таблиц можно выполнять в «естественном» режиме, когда адрес ячейки задается именами заголовков строк и столбцов таблицы.

По умолчанию ячейки с ошибками отслеживаются и, кроме сообщения об ошибках, в левом верхнем углу у них отображается цветовая метка. За этот режим отвечает опция **Включить фоновый поиск ошибок**. При установленной опции в раскрывающемся списке-палитре **Цвет индикации ошибок** можно выбрать цвет для метки.

Опции в группе **Правила контроля ошибок** задают, фактически, правила контроля ошибок, т. е. те правила, по которым определяется, является ли та или иная ситуация ошибочной или нет. Другими словами, с помощью этой группы опций можно явно указать, что считать ошибкой, а что — нет. Названия у опций достаточно показательные, поэтому комментировать их не будем.

На заметку

Впадать в эйфорию по поводу, казалось бы, огромных перспектив по настройке режима контроля ошибок все же не стоит. Дело в том, что настройки в группе **Правила контроля ошибок** влияют лишь на то, будет в соответствующей ситуации для ячейки с «ошибкой» отображаться цветовая метка или нет.

ВЫВОД ДОКУМЕНТОВ НА ПЕЧАТЬ

Да погодите вы, голубчик, с портретом!
Дайте ему со скульптурой разобраться!

Из к/ф «Формула любви»

Чтобы распечатать документ, нужно, чтобы этот документ как минимум существовал. Если документ уже у нас в руках (в переносном смысле, конечно), то распечатать его в принципе просто — можно нажать комбинацию клавиш <Ctrl>+<P> или перейти к вкладке **Файл** и выбрать там раздел **Печать**. В результате появится окно вывода документа на печать, аналогичное тому, что представлено на рисунке 2.62.

В правой части окна показано, как будет выглядеть документ при выводе на печать. Другими словами, по картинке в правой части окна можно составить представление о том, как выполнена постраничная разбивка документа. Если нас все в этой ситуации устраивает, щелкаем на пиктограмме **Печать**. Это простой случай. Но, к сожалению, скорее всего, нас устраивает не все, и поэтому имеет смысл рассмотреть альтернативный случай — сложный.

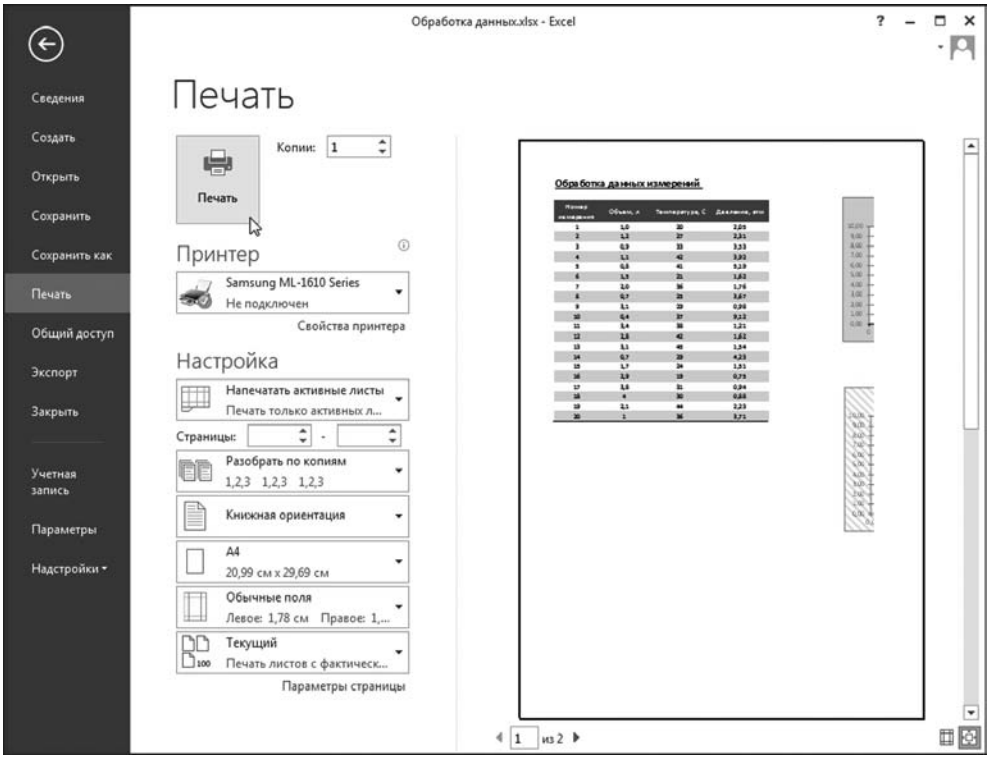


Рис. 2.62
Вывод документа на печать

На заметку

Сделаем несколько замечаний относительно настроек и режимов печати. С пиктограммами, имеющими исчерпывающе информативные названия, думается, читатель разберется без труда. Посмотрим, что произойдет, если щелкнуть на гиперссылке **Параметры страницы** в нижней части окна вывода на печать (рис. 2.62 или 2.63).

В результате открывается диалоговое окно **Параметры страницы**, в котором выполняются основные настройки, влияющие на способ вывода документа на печать. У окна четыре вкладки. На рисунке 2.64 показано окно, открытое на вкладке **Страница**.

Какие параметры можно настроить на этой вкладке? Важные. Например, ориентацию бумаги, а если точнее, то ориентацию для наложения данных рабочего документа на лист бумаги. Кроме того, здесь задаем масштаб отображения данных, качество печати, размер бумаги и прочие «технические» характеристики.

На вкладке **Поля** задаются размеры полей рабочего документа (рис. 2.65).

Вкладка **Колонтитулы** (рис. 2.66) предназначена для того, чтобы задать значения колонтитулов и выполнить прочие смежные настройки.

Вкладка **Лист** также не отстает в плане функциональности и полезности. Окно **Параметры страницы**, открытое на этой вкладке, показано на рисунке 2.67.

В первую очередь хочется обратить внимание на переключатель **Последовательность вывода страниц**, который задает способ, в котором страницы документа выводятся на печать. Здесь имеется в виду порядок (последовательность), в котором распечатываются страницы документа. Вариантов два: сначала сверху вниз и затем справа налево, или сначала слева направо и затем сверху вниз.

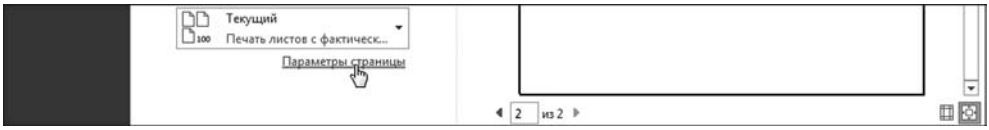


Рис. 2.63
Щелчок по гиперссылке **Параметры страницы** позволяет решить многие проблемы

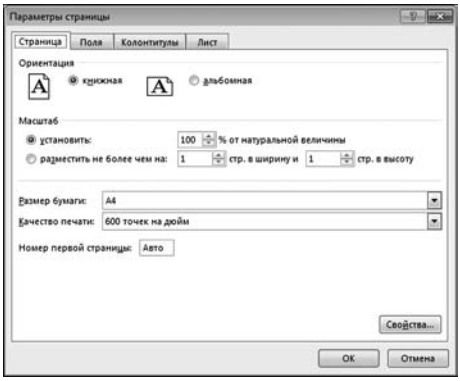


Рис. 2.64
Окно настроек **Параметры страницы** открыто на вкладке **Страница**

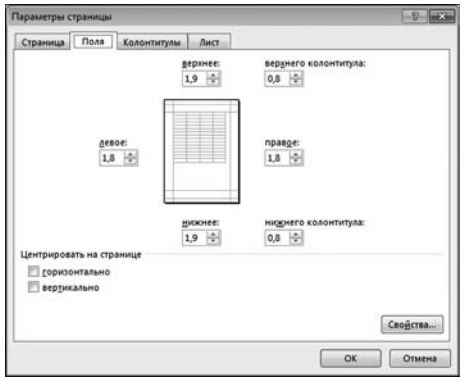


Рис. 2.65
Окно настроек **Параметры страницы** открыто на вкладке **Поля**

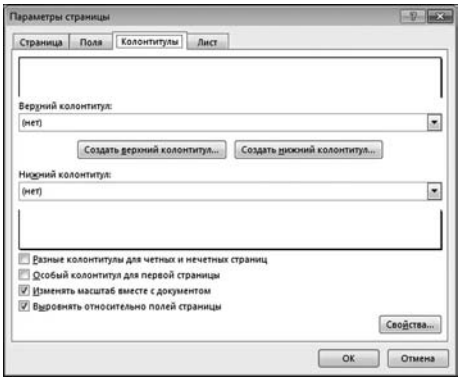


Рис. 2.66
Окно настроек **Параметры страницы** открыто на вкладке **Колонтитулы**

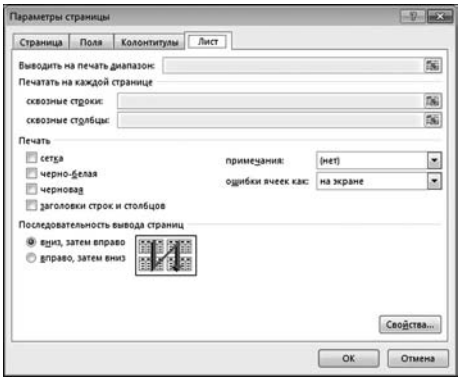


Рис. 2.67
Окно настроек **Параметры страницы** открыто на вкладке **Лист**

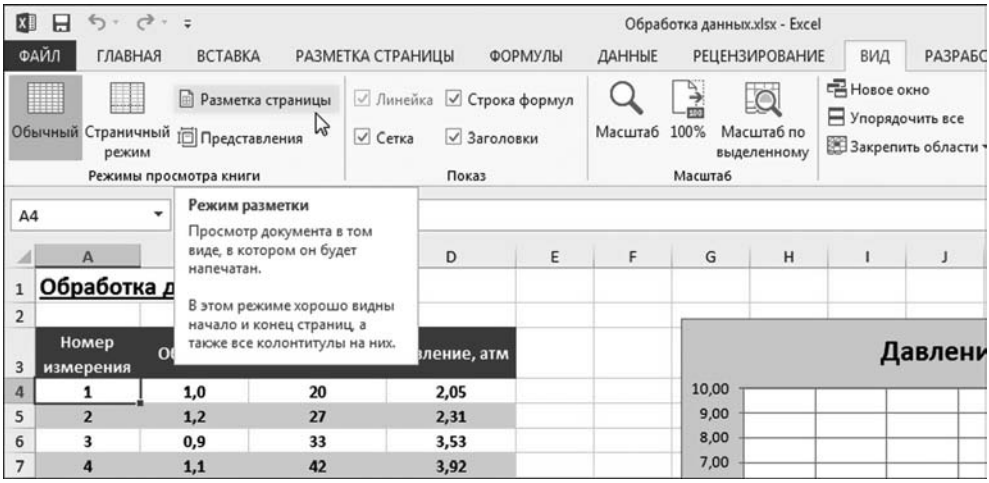


Рис. 2.68
Переход в режим разметки страниц

Группа опций **Печать** позволяет установить/отменить режим печати некоторых элементов рабочего листа — таких, например, как сетка, примечания, или задать способ вывода на печать ячеек с ошибками. Также можно задать диапазон ячеек, которые выводятся на печать.

Итак, возвращаемся к документу и открываем вкладку **Вид**. На этой вкладке в группе **Режимы просмотра книги** щелкаем пиктограмму **Разметка страницы** (рис. 2.68).

После этого документ переходит в режим разметки страниц и выглядит практически таким, каким он будет при выводе на печать — по крайней мере, в плане того, как данные рабочего листа размещаются по страницам. К сожалению, в данном конкретном случае мы фактически видим ту самую

унылую картину (рис. 2.69), как и ранее в режиме предварительного просмотра документа перед выводом на печать (рис. 2.62).

На заметку

Нас не устраивает то обстоятельство, что диаграммы попадают как раз на линию раздела страниц, поэтому на первой странице диаграммы присутствуют частично.

Мы пытаемся исправить ситуацию, для чего щелкаем пиктограмму **Страничный режим** (рис. 2.70).

При этом в документе выделяется (отображается) та область, что содержит данные, а границы раздела страниц отображаются пунктирной линией. На рисунке 2.70 эта пунктирная линия проходит совсем не так, как нам хотелось бы. Проблема решается просто — берем и перетаскиваем эту линию, как показано на рисунке 2.71.

После того как положение линии раздела страниц изменено, снова переходим в режим разметки страниц, для чего щелкаем пиктограмму **Разметка страницы** (см. рис. 2.72).

Вид документа в режиме разметки страницы внушает оптимизм — проблема действительно разрешилась (см. рис. 2.73).

Подтверждение успеха получаем и после перехода в режим предварительного просмотра документа перед выводом на печать. На рисунке 2.74 видна первая (из двух) страница документа в этом режиме.

На рисунке 2.75 в режиме предварительного просмотра показана вторая страница документа.

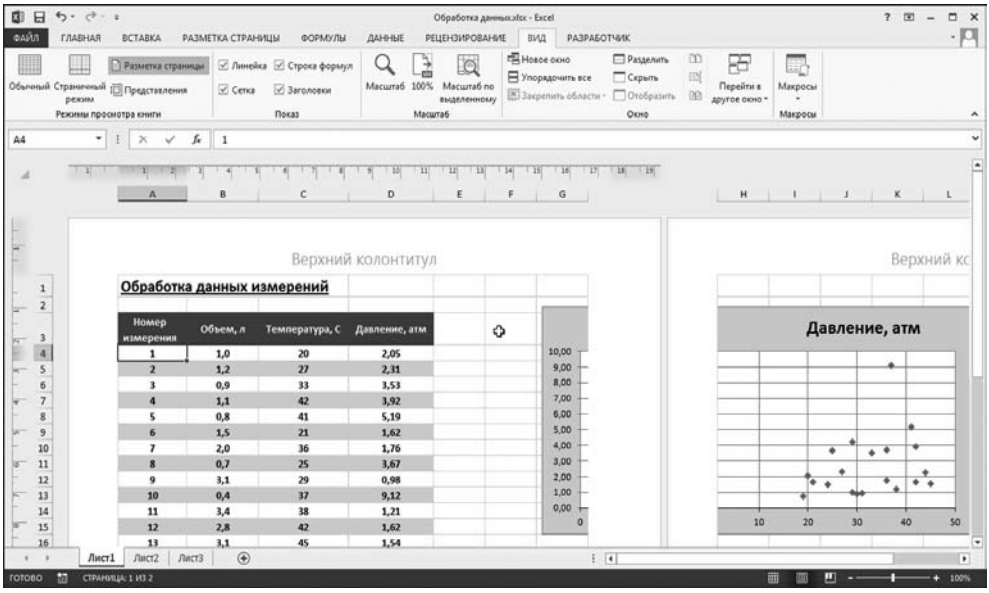


Рис. 2.69
Документ в режиме разметки страницы

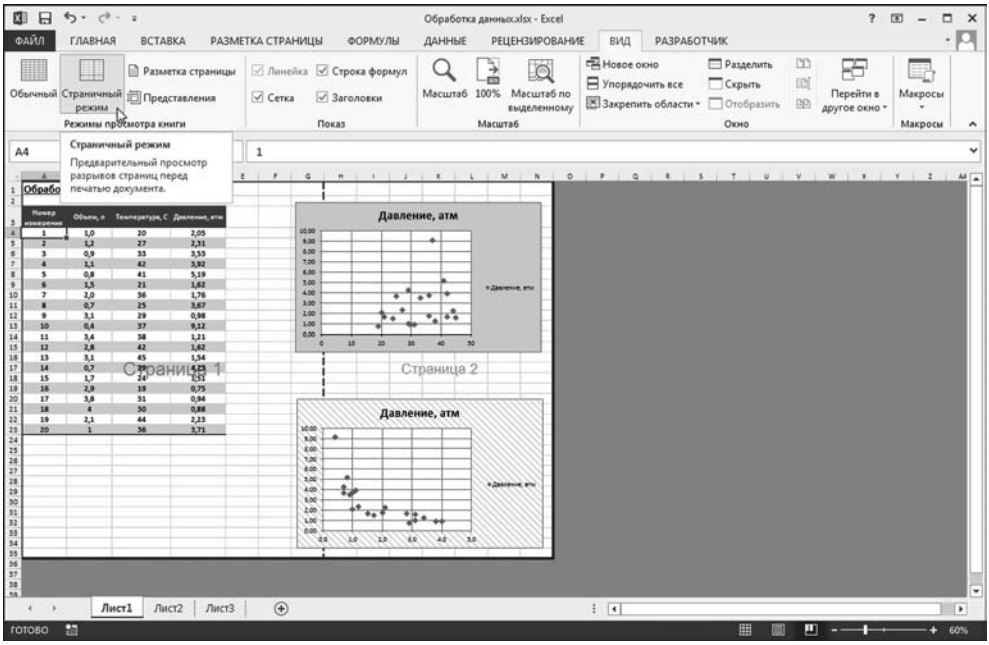


Рис. 2.70
Документ в режиме настройки страничного режима

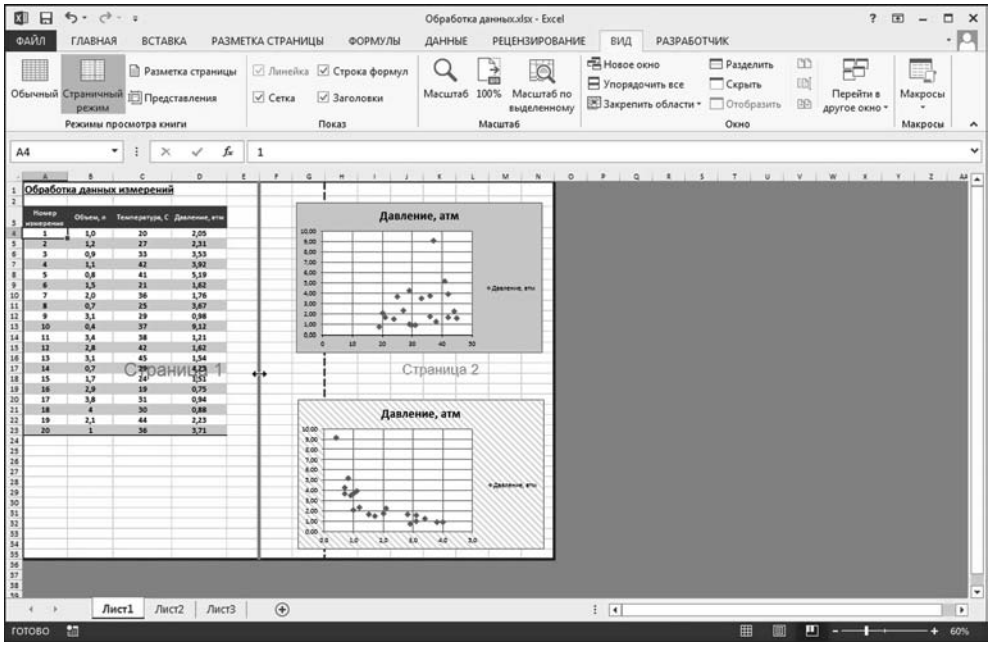


Рис. 2.71
Внесение изменений в настройки страничного режима

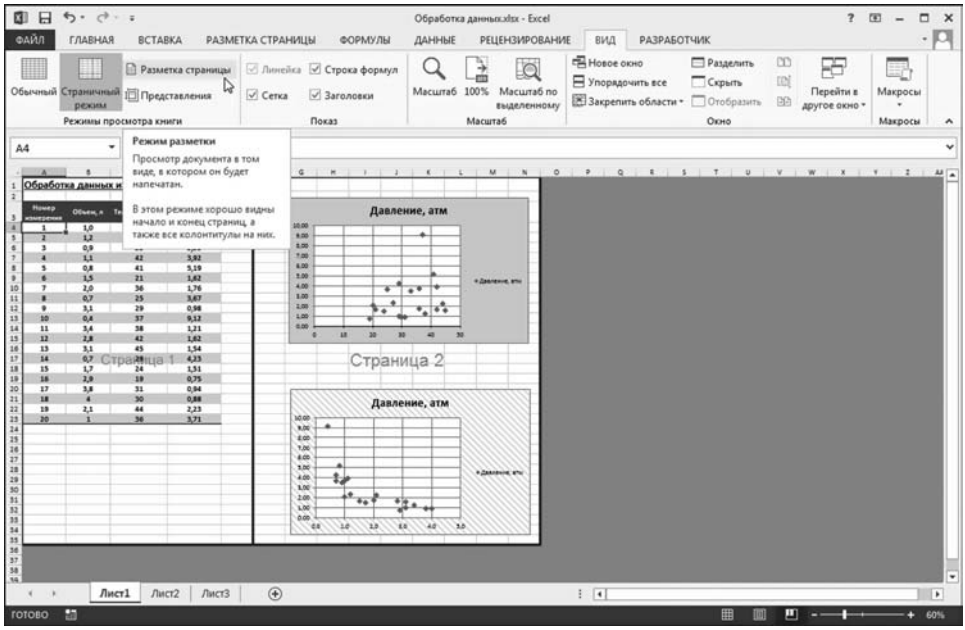


Рис. 2.72
В настройки страничного режима внесены изменения

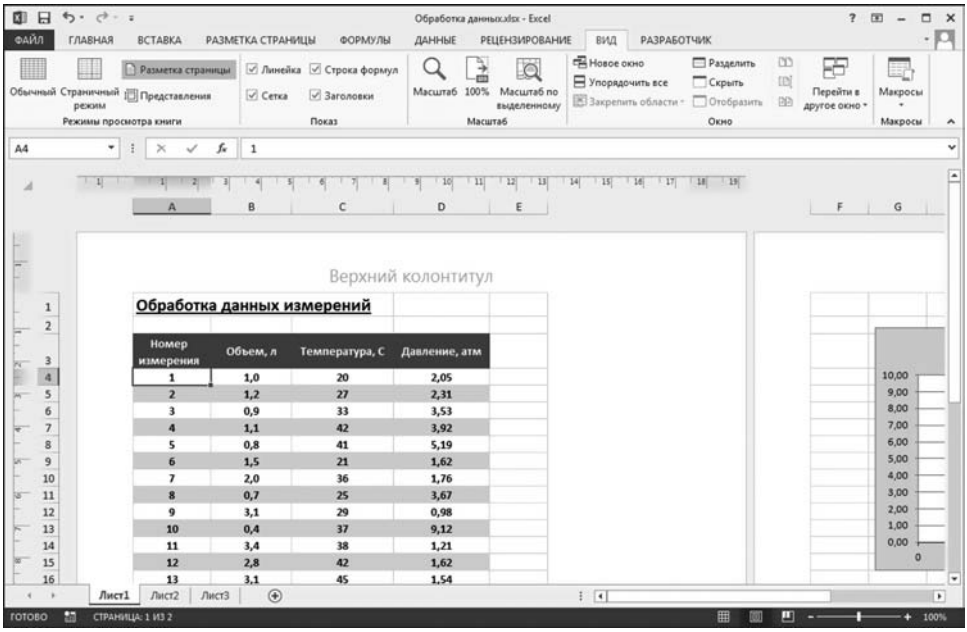


Рис. 2.73
Документ в режиме разметки страниц после внесения изменений
в настройки страничного режима

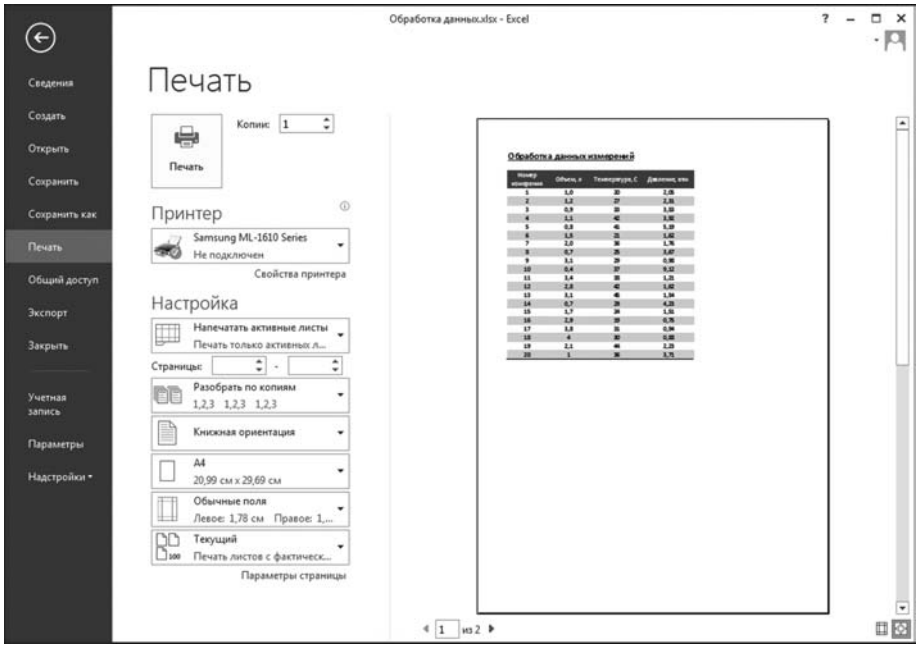


Рис. 2.74
Первая выводимая на печать страница

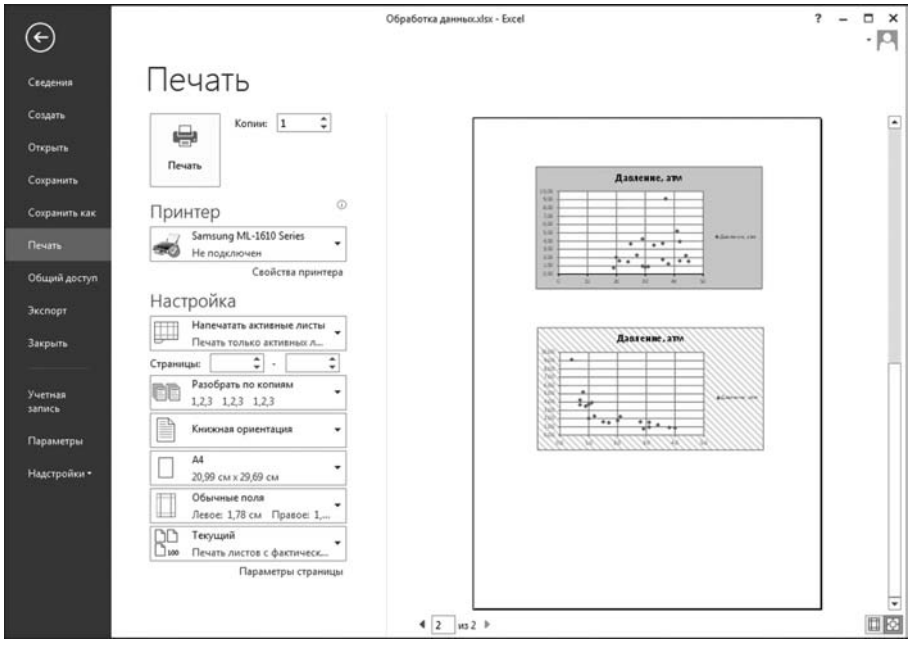


Рис. 2.75
Вторая выводимая на печать страница

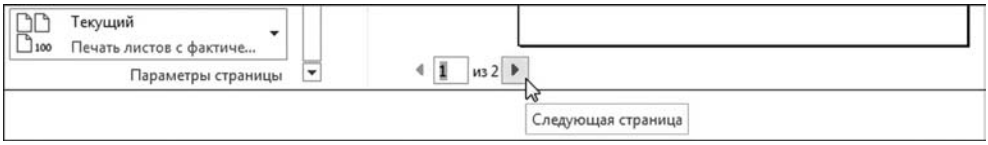


Рис. 2.76
Переключение между выводимыми на печать страницами документа

На заметку

Для переключения между страницами, выводимыми на печать, можно воспользоваться переключателем в нижней части окна вывода на печать, как это показано на рисунке 2.76.
Можно щелкнуть по стрелке для перехода к следующей/предыдущей странице или ввести номер страницы в специальном поле между мини-стрелками.

Можем печатать документ.

ГЛАВА 3

ФОРМАТИРОВАНИЕ И СТИЛИ

— Красиво плывут!
— Кто?
— Вон та группа, в полосатых купальниках.

Из к/ф «Полосатый рейс»

В этой главе речь пойдет о том, как различные данные представляются в рабочих документах Excel. В частности, будут обсуждаться форматы и стили. Вообще форматированию и применению стилей в Excel можно посвятить отдельную книгу. Мы остановимся только на наиболее важных моментах.

ПРИМЕНЕНИЕ И УДАЛЕНИЕ ФОРМАТОВ

Только не возражай мне.
И не нужно сцен.

Из к/ф «Иван Васильевич меняет профессию»

Кратко обсудим, какие настройки, касающиеся форматирования (т. е. внешнего вида) данных, применяются в Excel. В первую очередь имеются в виду настройки шрифта (такие, как тип, стиль, размер и цвет), способ выравнивания данных в ячейках по вертикали и горизонтали, ориентация текста в ячейках. Все эти настройки выполняются с помощью пиктограмм групп **Шрифт** и **Выравнивание** на вкладке **Главная** или с помощью мини-панели инструментов, которая появляется при раскрытии контекстного меню (имеет вид небольшой области с пиктограммами рядом с областью контекстного меню).

Принцип выполнения и применения настроек достаточно простой: предварительно выделяется ячейка или ячейки, к которым предполагается применить настройки, после чего эти настройки выполняются посредством пиктограмм вкладки **Главная**.

На заметку

Мы уже отмечали, что многие пиктограммы, в том числе и на вкладке **Главная** ленты приложения Excel стандартны и вполне «предсказуемы». Поэтому на многих очевидных моментах останавливаться не будем.

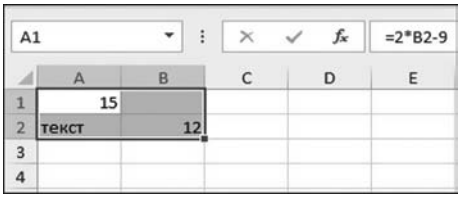


Рис. 3.1
Выделен диапазон ячеек
для форматирования

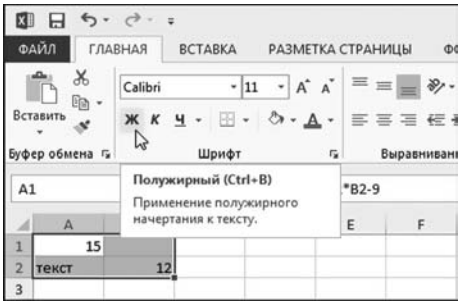


Рис. 3.2
Применение жирного шрифта

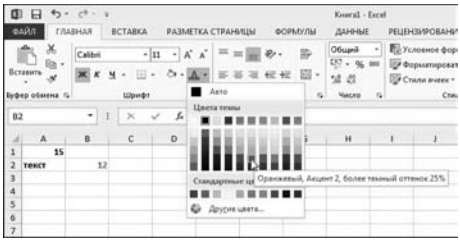


Рис. 3.3
Применение к данным
в ячейке выделения цветом

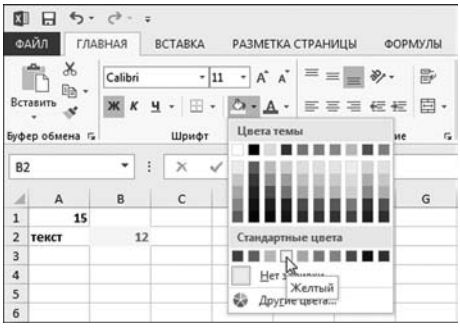


Рис. 3.4
Заливка цветом ячейки

На рисунке 3.1 представлена группа ячеек A1:B2 с разными данными (текст в ячейке A2, число 12 в ячейке B2, и формула =2*B2-9 в ячейке A1), которые выделены для применения форматирования.

Для начала применим к данным в выделенных ячейках жирный шрифт, для чего воспользуемся специальной пиктограммой в группе **Шрифт** вкладки **Главная** (рис. 3.2).

В результате данные в выделенных ячейках отображаются жирным шрифтом. Причем такой формат применяется даже к пустой ячейке B1. Если после его применения в ячейку B1 ввести данные, они будут отображаться жирным шрифтом. Далее выделяем ячейку B2 и изменяем цвет шрифта: в раскрывающемся списке-палитре выбираем цвет. При наведении курсора мыши на цветовую ячейку автоматически в рабочем листе отображается эффект от применения данного цвета (рис. 3.3).

Этот же режим (отображение эффекта от настроек при выборе пиктограммы параметров настройки) автоматически применяется во многих других случаях. Аналогично выполняется заливка фона ячейки цветом (рис. 3.4).

На рисунке 3.5 показано, как будут выглядеть числовые данные в ячейке B2 после применения жирного шрифта, изменения его цвета и заливки ячейки фоном.

Здесь же проиллюстрирован процесс выравнивания по центру (по горизонтали) данных в ячейке B2 (рис. 3.6), для чего полезной окажется пиктограмма в группе **Выравнивание** вкладки **Главная** (рис. 3.5).

На следующем этапе изменяем высоту ячеек во второй строке, для чего достаточно с помощью мыши в полосе нумерации строк переместить нижнюю границу для ячеек второго ряда (рис. 3.7).

Данная операция (изменение высоты ячейки) не относится к форматированию данных. В этом мы еще убедимся. Однако она позволяет с очевидностью обнаружить, что, кроме выравнивания данных по горизонтали, большое значение имеет способ выравнивания данных по вертикали. Этой цели служит набор специальных пиктограмм в группе **Выравнивание** вкладки **Главная** (рис. 3.8).

После выравнивания данных в ячейке B2 по вертикали по верхнему краю ячейки получаем результат, как на рисунке 3.9.

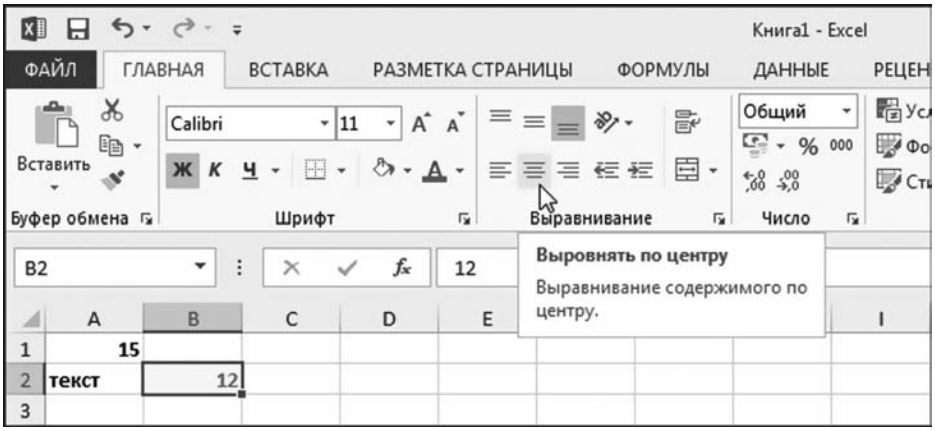


Рис. 3.5
Результат применения заливки и выравнивания по центру ячейки

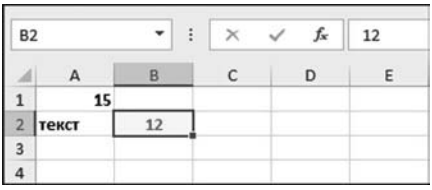


Рис. 3.6
Данные в ячейке выровнены по центру

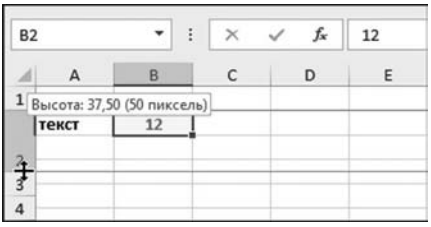


Рис. 3.7
Изменение высоты ячейки

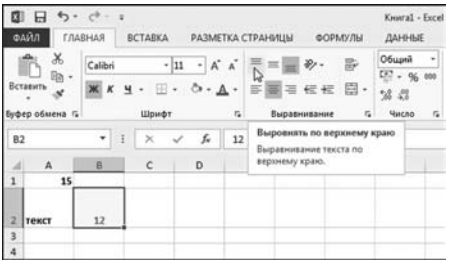


Рис. 3.8
Применение выравнивания по вертикали по верхней границе ячейки

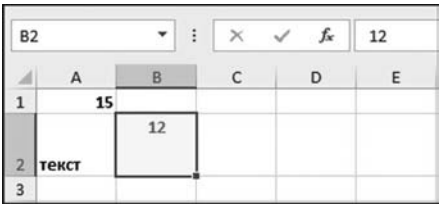


Рис. 3.9
Результат выравнивания данных по вертикали в ячейке по ее верхней границе

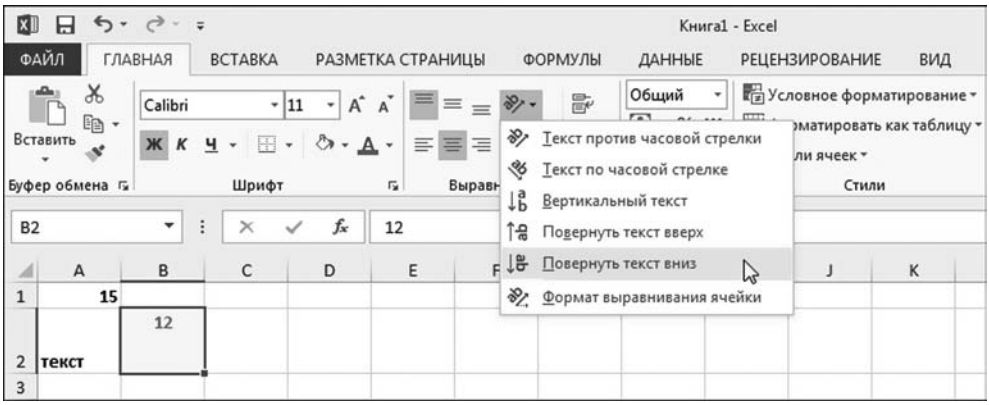


Рис. 3.10
Изменение ориентации текста в ячейке

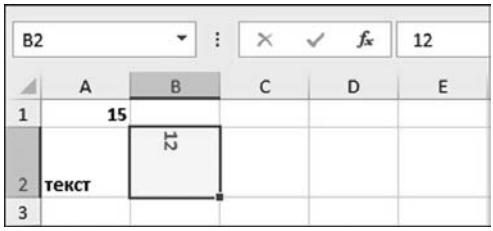


Рис. 3.11
Результат изменения ориентации текста в ячейке (текст направлен сверху вниз)

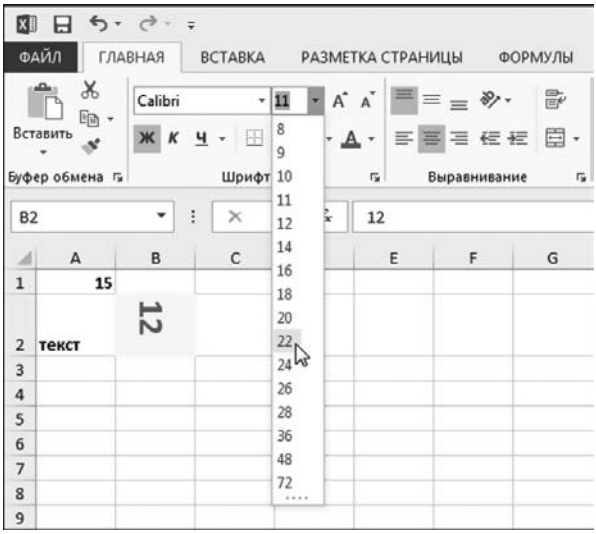


Рис. 3.12
Изменение размера шрифта

Также легко меняется ориентация текста в ячейке. Пиктограмма, предназначенная для решения такой задачи, находится на вкладке **Главная** в группе **Выравнивание** и представляет собой раскрывающийся список с командами, определяющими способ выравнивания данных в ячейке. В данном случае выбираем направление текста сверху вниз (рис. 3.10).

Результат показан на рисунке 3.11.

Здесь под текстом подразумевается значение, отображаемое в ячейке. Значение, в соответствии с выполненным форматированием, отображается по вертикали, сверху вниз, начиная от верхнего края ячейки с центрированием по горизонтали.

Настройки могут быть и не столь экзотическими. Например, на рисунке 3.12 показано, как изменяется размер шрифта для отображения данных в ячейке.

Не менее важной задачей, чем создание формата, является его распространение на другие ячейки. Другими словами, рассмотрим процедуру копирования форматов. Она принципиально отличается от копирования значений ячеек тем, что при копировании форматов значения в ячейках не меняются. Меняется только способ отображения этих значений.

Для копирования форматов на вкладке **Главная** в группе **Буфер обмена** есть пиктограмма с кисточкой (рис. 3.13).

Процедура копирования форматов следующая. Сначала выделяется ячейка или ячейки, формат которых будет копироваться. После этого щелкаем пиктограмму с кисточкой в группе **Буфер обмена**. Курсор мыши при наведении на ячейку документа примет вид, как на рисунке 3.14 (жирный крест с кисточкой).

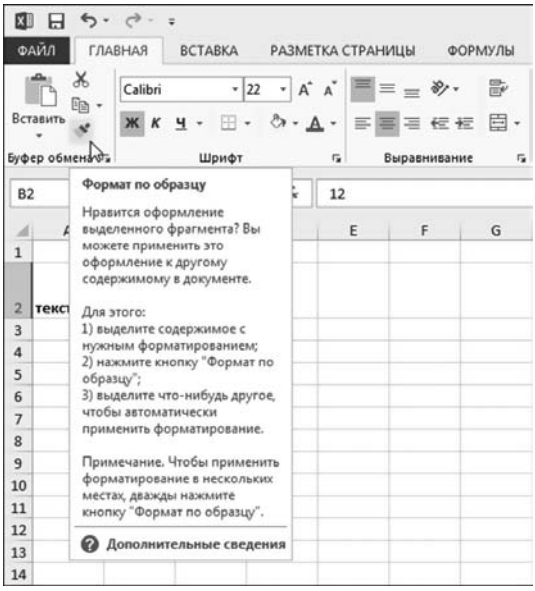


Рис. 3.13
Копирование формата

Рис. 3.14
Применение скопированного формата к ячейке

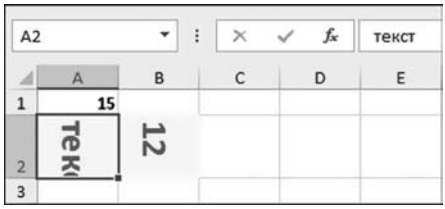


Рис. 3.15
Результат применения скопированного формата

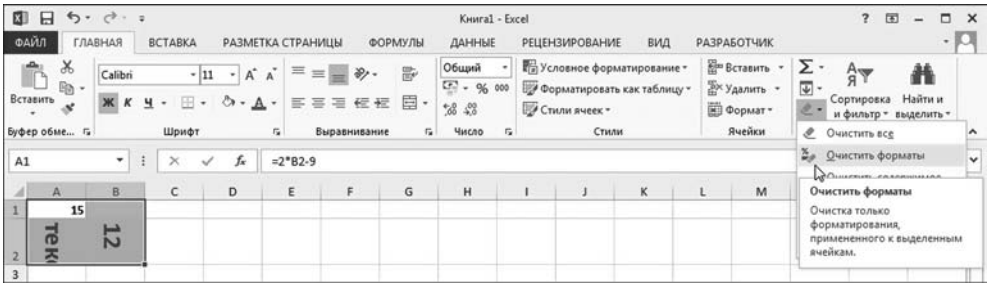


Рис. 3.16
Очистка формата

Щелкнув на ячейке или выделив диапазон ячеек, применяем к ним скопированный формат. Результат копирования формата из ячейки B2 в ячейку A2 показан на рисунке 3.15.

Здесь очень хорошо проявляется разница между *реальным значением* ячейки и *отображаемым* в ячейке *значением*. Важно понимать, что в общем случае это не одно и то же. Например, в ячейке A2 после применения формата отображается только часть слова текст, причем отображается по вертикали. Реальное значение ячейки не изменилось. Это все тот же текст. В последнем несложно убедиться, взглянув на отображаемое в строке формул значение (при выделенной ячейке A2).

Копирование формата из диапазона в диапазон ячеек имеет некоторые особенности. Во-первых, исходный диапазон должен быть связным, т. е. ячейки исходного диапазона должны находиться одна возле другой в выделенной прямоугольной области. Во-вторых, для конечных ячеек формат применяется по принципу взаимного соответствия: формат первой ячейки исходного диапазона применяется к первой ячейке конечного диапазона, формат второй ячейки исходного диапазона — ко второй ячейке конечного диапазона, и т. д. Если в исходном диапазоне ячеек меньше, чем в конечном, то перебор ячеек в конечном диапазоне осуществляется циклически.

Нередко бывает так, что применено огромное количество форматов, и только после этого приходит понимание, что делать этого не нужно. В таких случаях помогает очистка форматов. Для выполнения очистки форматов находим на вкладке **Главная** в группе **Редактирование** пиктограмму с изображением ластика (рис. 3.16).

В раскрывающемся списке много полезных команд. Нас интересует команда **Очистить форматы**. После ее выполнения получаем результат, как на рисунке 3.17.

Видим, что все примененное к ячейкам A1:B2 форматирование отменено. Высота ячеек второго столбца осталась неизменной. Как отмечалось, такие параметры, как высота и ширина ячейки не относятся к форматированию, хотя команды по настройке этих параметров скрыты за пиктограммой с названием **Формат** в группе **Ячейки**. Подобные параметры приходится настраивать отдельно.

Также отметим, что копирование форматов выполняется и с помощью команд контекстного меню. Для копирования формата через контекстное меню выделяем исходные ячейки с копируемым форматом и выполняем обычное копирование их содержимого в буфер обмена. После выделяем ячейки, к которым применяется копируемый формат, раскрываем контекстное меню и в списке команд этого меню выбираем пиктограмму с изображением кисти и знака процента, размещенную в группе **Параметры вставки** (рис. 3.18).

В результате производится копирование только форматов, значения конечных ячеек остаются неизменными.

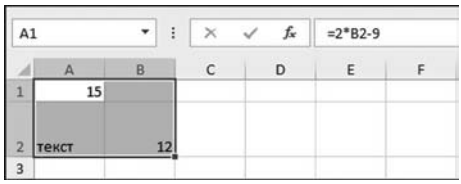


Рис. 3.17
Результат очистки форматов

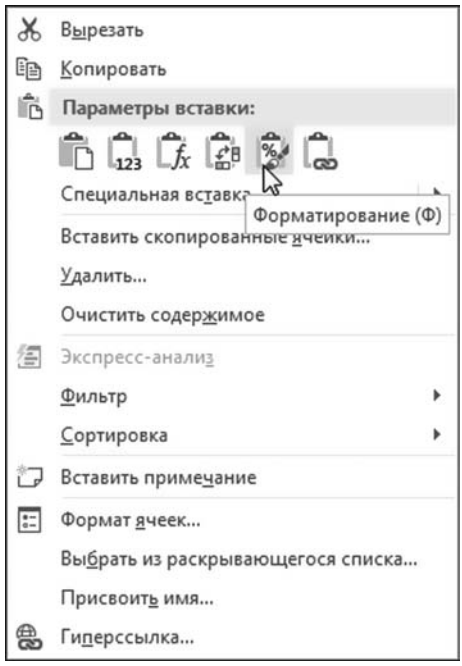


Рис. 3.18
Копирование формата с помощью контекстного меню

ФОРМАТЫ ЯЧЕЕК

Блондин — это Молчановский.
А Якин — талантливый!

Из к/ф «Иван Васильевич меняет профессию»

В Excel есть предопределенная группа встроенных форматов. Встроенные форматы, кроме того, автоматически применяются по умолчанию в зависимости от значения ячейки. Встроенные форматы по своему усмотрению применяются также пользователем для форматирования отдельных ячеек и групп ячеек. Кратко рассмотрим встроенные форматы Excel. Представление о форматах легко составить по диалоговому окну **Формат ячеек**, которое

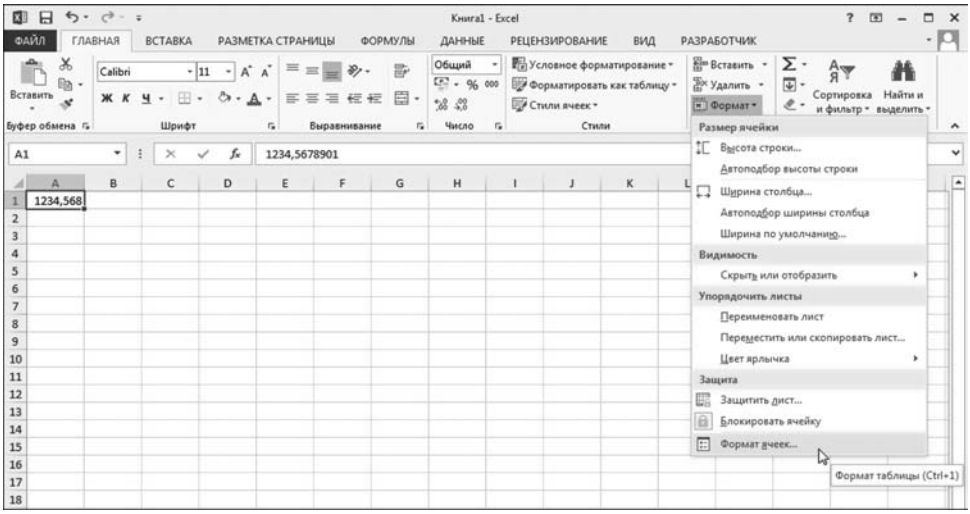


Рис. 3.19
Команда **Формат ячеек** позволяет открыть окно настройки формата

открывается одноименной командой контекстного меню или такой же командой из раскрывающегося списка пиктограммы **Формат** группы **Ячейки** на вкладке **Главная** (рис. 3.19).

В данном случае команда выполняется при активной ячейке A1 с числовым значением 1234,5678901. Это значение отображается в строке формул. В самой ячейке виден несколько урезанный вариант, но это *отображаемое* значение. Реальное значение, используемое при вычислениях, — в строке формул. Данное замечание важно, поскольку в диалоговом окне **Формат ячеек** при выборе типа числовых форматов автоматически отслеживается, как при том или ином формате выглядит отображаемое в ячейке значение. Что касается самого окна **Формат ячеек**, то оно, с одной стороны, достаточно простое, а с другой, содержит несколько вкладок с множеством утилит по выбору и настройке форматов. Вкладки диалогового окна **Формат ячеек** с выполняемыми на них настройками приведены и описаны в таблице 3.1. Предварительно отметим, что окно имеет шесть вкладок следующего назначения:

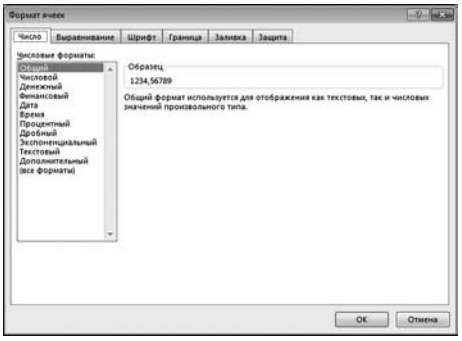
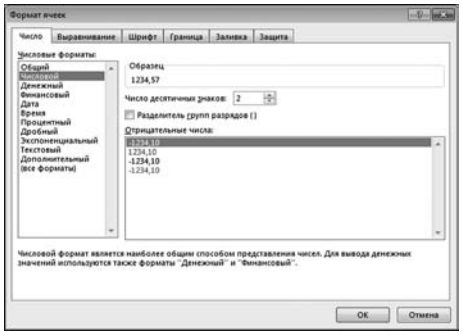
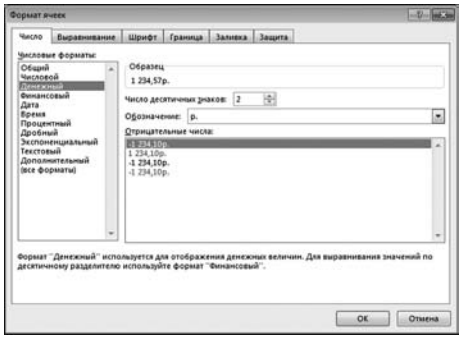
- вкладка **Число** предназначена для выбора и применения форматов к ячейкам базовых встроенных форматов (в основном числовых);
- вкладка **Выравнивание** содержит утилиты настроек способа позиционирования (ориентации, выравнивания) значений в ячейках;
- вкладка **Шрифт** позволяет задавать параметры шрифта для отображения данных в ячейке;
- вкладка **Граница** предназначена для определения типа и способа отображения границ ячейки;
- вкладка **Заливка** содержит палитру и ряд опций для выбора цвета и способа заливки фона ячейки;
- вкладка **Защита**, как несложно догадаться, содержит утилиты для настройки параметров защиты данных в ячейке.

Безусловный интерес представляет первая вкладка **Число** диалогового окна **Формат ячеек**. В левой части вкладки имеется список **Числовые форматы**, в котором выбирается тип встроенного формата. Справа в окне **Образец** показано, как при данном выбранном формате будет выглядеть значение в ячейке. Внизу приведен краткий комментарий по поводу формата. Также если допускает формат, могут отображаться дополнительные функциональные элементы настройки формата (опции, списки). Теперь перейдем к таблице 3.1.

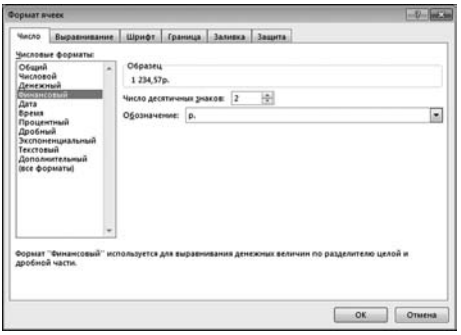
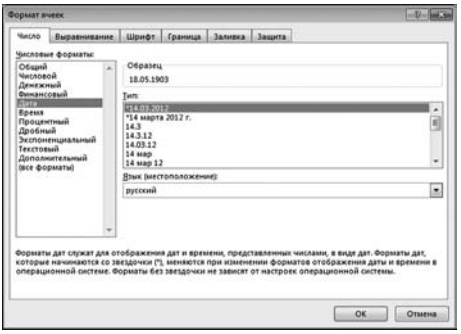
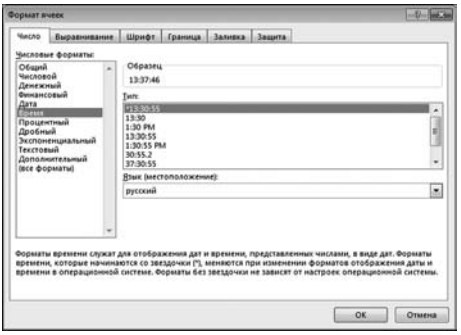
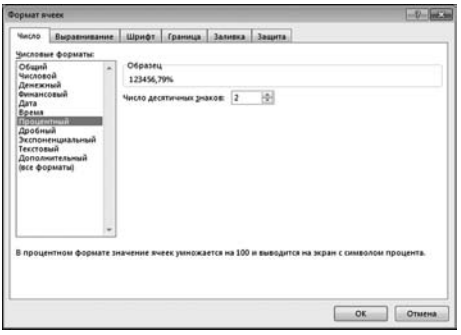
Еще раз обращаем внимание читателя, что применение того или иного формата не меняет значения ячейки, а только способ его отображения в ячейке.

Таблица 3.1

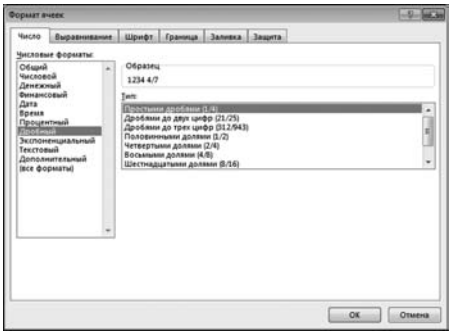
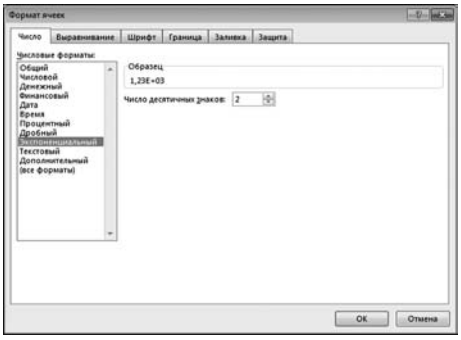
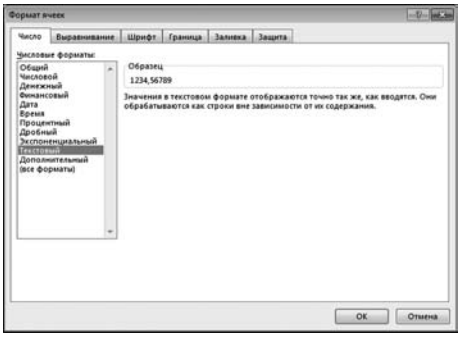
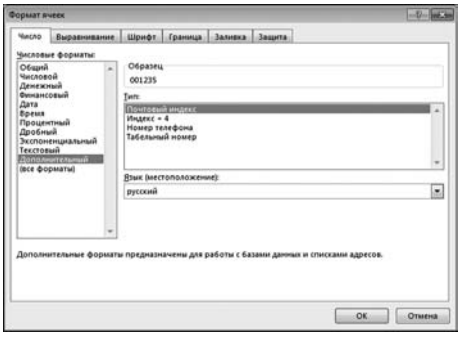
Диалоговое окно настроек формата Формат ячеек

Вид окна Формат ячеек	Описание
	Вкладка **Число**. В списке форматов **Числовые форматы** выбран формат **Общий**. Этот формат автоматически применяется к данным в ячейках (как числовым, так и текстовым). Для формата **Общий** характерно наименьшее искажение данных при отображении, если сравнивать с тем, в каком виде они в ячейку вводятся. В числах отображается до 10 цифр. Если число не помещается по ширине ячейки, оно при отображении округляется до нужного размера. Незначащий ноль в целой части отображается. Текст отображается в таком виде, как он введен в ячейку. Если в числе больше 12 цифр, оно представляется в научной нотации
	Вкладка **Число**. Формат **Числовой** предназначен для отображения числовых значений. Имеет ряд настроек. В первую очередь стоит обратить внимание на поле **Число десятичных знаков**, в котором указывается количество цифр, отображаемых в числе после десятичной запятой. Если установить флажок опции **Разделитель групп разрядов**, в представлении чисел используется разделитель тысячных разрядов (по умолчанию это пробел). В списке **Отрицательные числа** выбирают способ отображения отрицательных чисел (возможно выделение цветом и/или знаком «минус»)
	Вкладка **Число**. Формат **Денежный** применяется для отображения денежных единиц. В поле **Число десятичных знаков** указывается количество цифр в числе после десятичной запятой (значение по умолчанию **2**), в списке **Обозначение** выбирается тип денежной единицы (тип валюты), в списке **Отрицательные числа** — способ отображения отрицательных значений

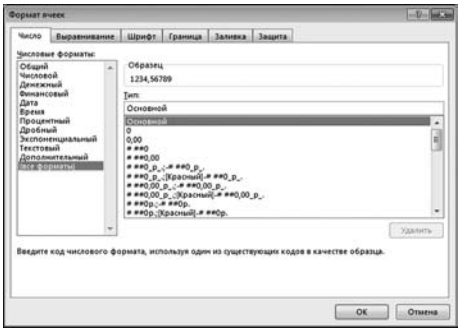
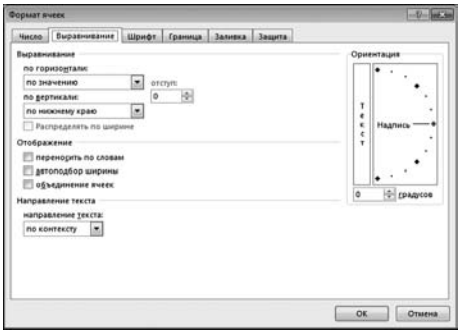
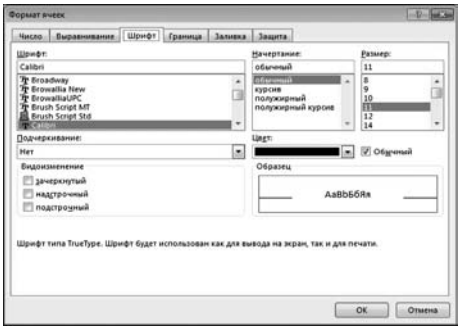
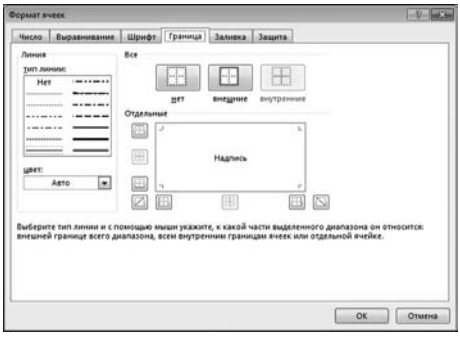
Продолжение табл. 3.1

Вид окна Формат ячеек	Описание
	Вкладка **Число**. От формата **Денежный** формат **Финансовый** отличается более скромными настройками и назначением: финансовый формат применяют для выравнивания денежных единиц по разделителю целой и дробной части. В формате **Финансовый** в поле **Число десятичных знаков** указывается количество отображаемых цифр после запятой, а в списке **Обозначение** задается тип валюты
	Вкладка **Число**. Формат **Дата** предназначен для отображения дат. В списке **Тип** выбирается маска для отображения даты, а в списке **Язык (местоположение)** задаются региональные настройки
	Вкладка **Число**. Формат **Время** предназначен для отображения в ячейке времени. В списке **Тип** указывается шаблон для отображения времени, а в списке **Язык (местоположение)** задается региональный язык
	Вкладка **Число**. Формат **Процентный** используется для обозначения числовых данных, выраженных в процентах. В поле **Число десятичных знаков** задается количество цифр после десятичной запятой

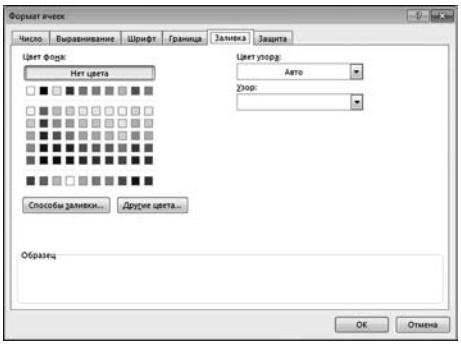
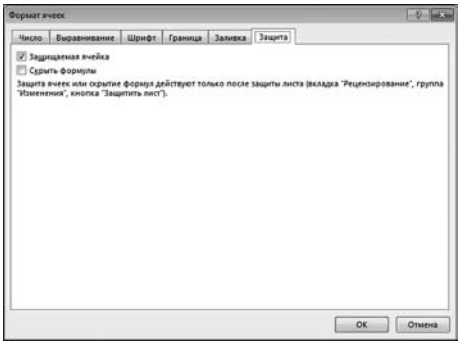
Продолжение табл. 3.1

Вид окна Формат ячеек	Описание
	Вкладка **Число**. Формат **Дробный** позволяет представлять нецелые значения в ячейках в виде рациональных дробей. В списке **Тип** выбирается шаблон для отображения чисел в ячейках. При отображении реальное значение ячейки округляется до ближайшего значения, соответствующего выбранному шаблону
	Вкладка **Число**. Формат **Экспоненциальный** позволяет отображать числовые значения в *научной нотации*: с мантиссой (число перед символом E) и показателем степени (число после символа E). Реальное значение получается умножением мантиссы на десять в соответствующей степени. Для формата в поле **Число десятичных знаков** задается количество цифр после десятичной запятой в мантиссе
	Вкладка **Число**. Формат **Текстовый** применяется для текстовых значений
	Вкладка **Число**. Формат **Дополнительный** содержит группу дополнительных форматов, наподобие формата почтового индекса. Тип формата выбирается в списке **Тип**, а в списке **Язык (местоположение)** задаются региональные настройки

Продолжение табл. 3.1

Вид окна Формат ячеек	Описание
	Вкладка **Число**. В списке форматов **Числовые форматы** выбрана позиция (**все форматы)**. В поле **Тип** с сопряженным списком задается шаблон формата. Этот режим, кроме того, используется при создании форматов пользователя
	Вкладка **Выравнивание**. На этой вкладке задаются параметры выравнивания и пространственной ориентации (направления), отображаемого в ячейке значения. Ориентация определяется с помощью графического индикатора **Ориентация** в правой части окна или поля **градусов** с указанием направления текста (в градусах). В группе **Выравнивание** задаются параметры выравнивания значения ячейки по вертикали и горизонтали. В группе **Отображение** задаются параметры отображения текста (режим переноса текста, автоматический подбор ширины текста по размеру ячейки, и т. д.). Направление текста можно изменить в списке **направление текста**, но это только для систем, поддерживающих письменность справа налево
	Вкладка **Шрифт**. На вкладке задаются основные параметры шрифта, используемого для отображения значений в ячейках. В частности, это тип шрифта (семейство, задается в поле-списке **Шрифт**), способ начертания (поле-список **Начертание**), размер (поле-список **Размер**), цвет (список **Цвет**), наличие и тип подчеркивания (список **Подчеркивание**), зачеркивания, режим подстрочных и надстрочных символов (группа опций **Видоизменение**)
	Вкладка **Граница**. На этой вкладке задается способ выделения границ ячейки или группы ячеек: наличие или отсутствие внешних и внутренних рамок (утилиты групп **Все** — для внешних и внутренних рамок в совокупности, и **Отдельные** — для каждой из рамок в отдельности), а также тип (список **тип линии**) и цвет линии рамок (список **цвет**)

Продолжение табл. 3.1

Вид окна Формат ячеек	Описание
	Вкладка **Заливка**. Основу вкладки составляет палитра **Цвет фона**, в которой выбирается цвет заливки. Если набор цветов по каким-то причинам кажется бледным, используем кнопку **Другие цвета**. Способ заливки определяем, нажав кнопку **Способы заливки**. Узор и цвет узора выбираются в одноименных раскрывающихся списках
	Вкладка **Защита**. Вкладка содержит всего две опции: **Защищаемая ячейка** и **Скрыть формулы**. Позволяют перейти в режим защиты ячеек (например, от выделения или форматирования ячеек — режим задается при установке защиты рабочего листа) и скрытия формул соответственно. Чтобы настройки вступили в силу, необходимо установить режим защиты рабочего листа

СОЗДАНИЕ ФОРМАТОВ

Не надо громких слов!
Они потрясают воздух, но не собеседника.

Из к/ф «Формула любви»

Если по каким-либо причинам ни один из встроенных форматов не нравится (хотя это и маловероятно), есть возможность создать собственный формат. Начнем с простого примера. Для этого в ячейку A1 вводим числовое значение 12,34, как показано на рисунке 3.20.

Затем открываем (при активной ячейке A1) диалоговое окно **Формат ячеек**, на вкладке **Число** (см. рис. 3.21).

В разделе **Числовые форматы** выбираем позицию **(все форматы)**, и в поле **Тип** вводим значение 000,000. Сразу замечаем, что в разделе **Образец** появляется странного вида число: 012,340. После применения к значению в ячейке данного формата (для этого достаточно щелкнуть по кнопке **ОК** диалогового окна **Формат ячеек**) именно так выглядит значение в ячейке A1 (см. рис. 3.22).

Рис. 3.20
Исходное форматирование

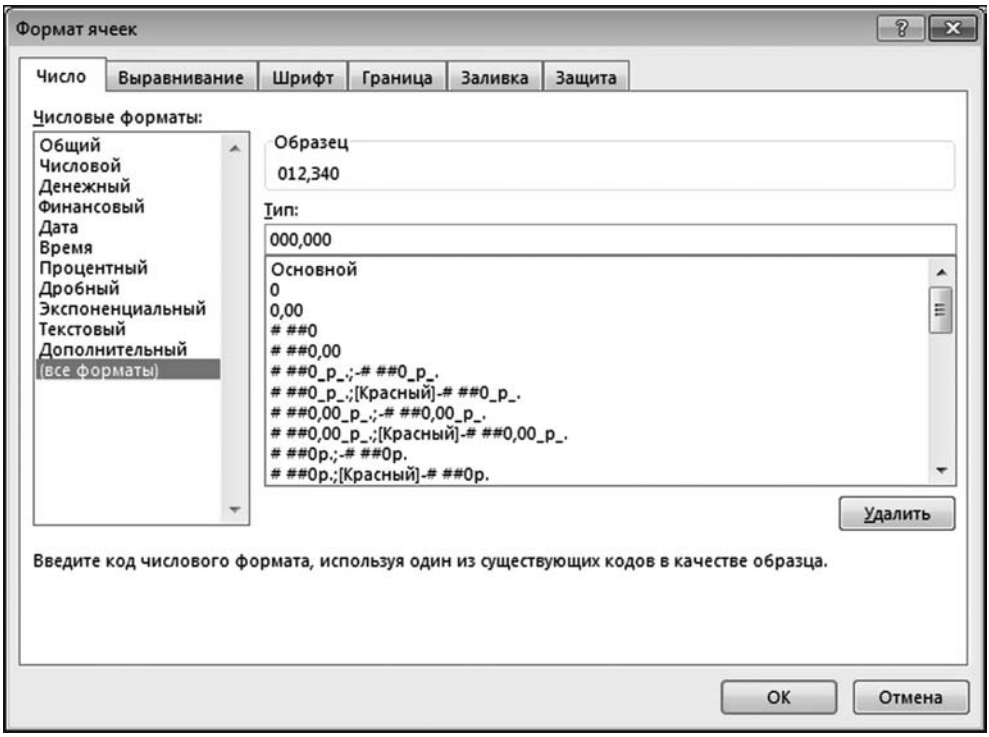


Рис. 3.21
Формат пользователя с шаблоном 000,000

Рис. 3.22
Результат применения формата с шаблоном 000,000

То, что вводилось в поле **Тип** — шаблон формата. В шаблоне формата 000,000 символ 0 означает, что в соответствующей позиции обязательно должна быть цифра. Запятая означает десятичную запятую. Таким образом, формат с шаблоном 000,000 означает, что при отображении числа в его целой части должно быть не меньше трех цифр и в дробной части также не меньше трех цифр. Если в числе цифр меньше, чем нужно, они дополняются незначащими нулями. У числа 12,34 в целой и дробной части по две цифры, поэтому в целой части при отображении числа в начале добавляется незначащий ноль, и еще один незначащий ноль в дробной части на последней (третьей) позиции. Если формат с шаблоном 000,000 применить к числу 1234,5678, то оно так и будет отображаться, т. е. как 1234,5678 (если ширина ячейки позволит).

Кроме нулей, в шаблонах форматов используются и другие символы. Их перечень с кратким описанием приведен в таблице 3.2.

Если в шаблон формата необходимо добавить статический (не зависящий от содержимого ячейки) текст, его заключают в двойные кавычки. Комби-

нируя разные символы в шаблонах форматов, легко добиться на первый взгляд до удивления странных результатов. Но это только на первый взгляд. Разберем некоторые простые примеры. Начнем с шаблона `0,* 0` (между звездочкой и последним нулем — пробел). Такой шаблон означает, что в целой части числа отображается, по крайней мере, одна цифра. В дробной части также хотя бы одна цифра. Поскольку после запятой в шаблоне формата имеется звездочка и после нее пробел, то после запятой и до дробной части при представлении числа пробелы будут добавляться, пока число не займет всю ширину ячейки. Этот простой факт означает также, что в дробной части отображается не просто не меньше, чем одна, а ровно одна цифра. В таблице 3.3 приведены примеры использования различных форматов.

Формат можно задавать отдельно для положительных, отрицательных, нулевых и текстовых значений. В этом случае шаблон формата состоит из четырех блоков, каждый из которых разделяется точкой с запятой. Первый блок — это шаблон для положительных чисел. Второй — для отрицательных значений. Третий — для нулевых значений и четвертый — для текста.

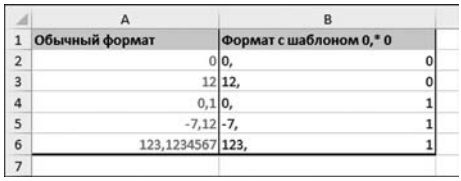
Таблица 3.2

Символы шаблонов форматов

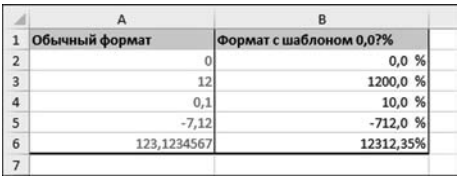
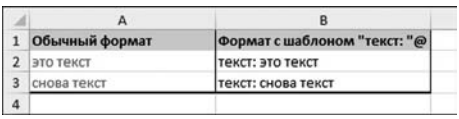
Символ шаблона	Назначение символа в шаблоне
0	Символ, обозначающий в шаблоне обязательное наличие цифры. Отсутствующие цифры заменяются незначащими нулями
#	Символ, обозначающий в шаблоне наличие цифры, если это только не незначащий ноль. Под незначащие нули позиция в представлении числа не отводится
?	Символ, обозначающий в шаблоне наличие цифры, если это только не незначащий ноль. Под незначащие нули в представлении числа отводится позиция
/	Символ дробной части. Слева от символа — шаблон числителя, а справа — шаблон знаменателя. Вся эта конструкция составляет шаблон дробной части числа. Шаблон целой части указывается в начале и отделяется от шаблона дробной части пробелом
E	Символ, который в шаблоне отделяет мантиссу числа от показателя степени. Слева от символа указывается шаблон мантиссы, а справа — шаблон показателя степени. Сразу после символа E указывается знак + или -. Если в шаблоне использована комбинация E+, все значения показателя степени отображаются со знаком (положительные со знаком + и отрицательные со знаком -). Комбинация E- в шаблоне означает, что со знаком отображаются только отрицательные значения показателя степени
%	Символ-инструкция преобразования числового значения к процентному формату. Алгоритм преобразования такой: число умножается на 100 и справа от числа отображается символ процента %
\	Символ-инструкция, используемый при вводе букв (символ \ предшествует букве, которая добавляется в представление значения ячейки)
_	Символ подчеркивания используется в шаблонах форматов для обозначения пробела. Ширина пробела определяется шириной следующего после пробела символа. Этот символ специально указывается для того, чтобы определить ширину пробела, и в ячейке не отображается
@	Символ, обозначающий текстовое значение ячейки
*	Символ-инструкция повторения. Символ, указанный после звездочки в шаблоне, повторяется до полного заполнения ячейки

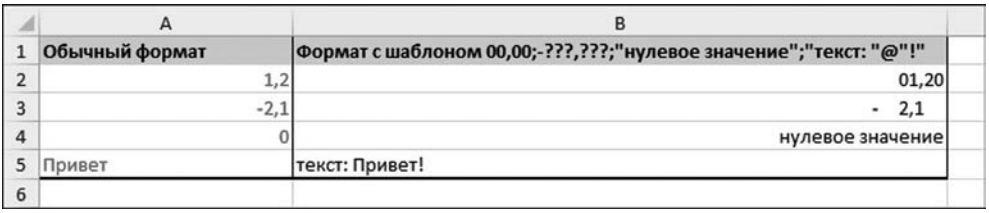
Таблица 3.3

Использование различных форматов

Шаблон	Пример использования	Описание
0,* 0		Отображается, по меньшей мере, одна цифра в целой части, десятичная запятая, ячейка по ширине заполняется пробелами и на последней позиции одна цифра в десятичной части
\Ч_w#,??		Отображается буква Ч, пробел шириной в букву w, после чего число. В числе нулевая целая часть не отображается. В десятичной под две цифры выделяются позиции
0" "#/##		В этом дробном формате в целой части отображается, по меньшей мере, одна цифра, затем пробел, и дробная часть в виде рациональной дроби. В числителе не больше одной цифры, в знаменателе — не больше двух. Место под недостающие цифры не выделяется
0" "#/#		Дробный формат: в целой части отображается, по меньшей мере, одна цифра, затем пробел и дробная часть в виде рациональной дроби. В числителе и знаменателе не больше одной цифры. Место под недостающие цифры не выделяется
0" "?/??		В этом дробном формате в целой части отображается, по меньшей мере, одна цифра, затем пробел и дробная часть в виде рациональной дроби. В числителе не меньше одной цифры, в знаменателе — не больше двух. Под недостающие цифры выделяется место
0" "?/?		Дробный формат: в целой части отображается, по меньшей мере, одна цифра, затем пробел и дробная часть в виде рациональной дроби. В числителе и знаменателе не больше одной цифры. Под недостающие цифры выделяется место
0" "0/00		В этом дробном формате в целой части отображается, по меньшей мере, одна цифра, затем пробел и дробная часть в виде рациональной дроби. В числителе не меньше одной цифры, в знаменателе — не больше двух. Незначащие нули отображаются и в числителе, и в знаменателе

Продолжение табл. 3.3

Шаблон	Пример использования	Описание
0,???E+0		Формат с мантиссой и показателем степени. В мантиссе в целой части отображается не меньше одной цифры (в том числе и незначащий нуль), а в десятичной выделяется место для трех цифр. Показатель степени отображается со знаком и содержит, по меньшей мере, одну цифру, включая незначащий нуль
0,0?%		Процентный формат, в котором в целой части отображается, по меньшей мере, одна цифра (в том числе незначащий нуль). В дробной части отображается, по меньшей мере, одна цифра, а место выделяется, по меньшей мере, под две цифры
"текст: "@		Текстовый формат, в котором отображается слово текст и через двоеточие и пробел текстовое значение ячейки



	A	B
1	Обычный формат	Формат с шаблоном 00,00;-???,???;"нулевое значение";"текст: "@"!"
2	1,2	01,20
3	-2,1	- 2,1
4	0	нулевое значение
5	Привет	текст: Привет!
6		

Рис. 3.23
Результат применения шаблона 00,00;-???,???;"нулевое значение";"текст: "@"!"

Например, формат с шаблоном 00,00;-???,???;"нулевое значение";"текст: "@"!" означает, что положительные числа отображаются в формате с шаблоном 00,00. При отображении отрицательных чисел используется формат -???,???. К ячейкам с нулевым значением применяется формат "нулевое значение". Наконец, текстовые значения отображаются в формате "текст: "@"!". Результат применения такого формата показан на рисунке 3.23.

Кроме формального способа представления значений, в шаблоне по желанию также задается цвет шрифта. Цвет указывается для каждого блока в квадратных скобках (в начале блока). Допускается использование следующих ключевых слов для определения цвета шрифта: Белый, Голубой, Желтый, Зеленый, Красный, Розовый, Синий и Черный. Например, формат с шаблоном [Синий]0,0;[Зеленый]-0,0E+0;[Красный]"ноль";[Желтый]@ означает, что положительные числа отображаются *синим* цветом в формате 0,0. Отрицательные числа отображаются *зеленым* цветом в формате -0,0E+0. Нулевые значения отображаются в формате "ноль" *красным* цветом (т. е. вместо нулевых значений отображается красного цвета надпись *ноль*). Текст отображается

желтым цветом в формате @ (отображается текстовое содержимое ячейки). Результат применения такого формата представлен на рисунке 3.24.

Совершенно не обязательно в шаблоне формата указывать все четыре блока. Блоков может быть от одного до четырех. Общие правила применения шаблонов форматов в блоках следующие:

- если шаблон состоит из одного блока, этот шаблон применяется ко всем данным (положительным, отрицательным и нулевым числам, а также к тексту);
- если шаблон состоит из двух блоков, то шаблон первого блока применяется к положительным и нулевым числам, а шаблон второго блока применяется к отрицательным значениям;
- в шаблоне из трех блоков первый блок определяет формат положительных чисел, второй блок — формат отрицательных чисел, и третий блок — формат нулевых значений;
- в шаблоне из четырех блоков первый блок определяет формат положительных чисел, второй — отрицательных, третий — нулевых значений, четвертый — текста.

В первые два блока шаблона можно добавлять специальный *код-условие*. Условие заключается в квадратные скобки, и в нем используются следующие знаки (операторы) сравнения: > (больше), < (меньше), >= (больше или равно), <= (меньше или равно), = (равно), <> (не равно). Например, условие [>10] означает, что формат применяется к числам, которые больше 10.

Формат из нескольких блоков с условиями применяется по следующей схеме. Проверяется условие первого блока. Если оно истинно (т. е. если выполняется), то ячейка форматируется в соответствии с шаблоном первого блока. Если условие не выполняется, проверяется условие второго блока.

	A	B
1	Обычный формат	Шаблон [Синий]0,0;[Зеленый]-0,0E+0;[Красный]"ноль";[Желтый]@
2	1,2	1,2
3	-2,1	-2,1E+0
4	0	ноль
5	текст	текст
6		

Рис. 3.24
Результат применения формата с шаблоном
[Синий]0,0;[Зеленый]-0,0E+0;[Красный]"ноль";[Желтый]@

	A	B
1	Обычный формат	Шаблон [Синий][>=10]"большое число";[Красный][<0]"меньше нуля";[Черный]0,???;[Зеленый]"текст"
2	1,2	1,2
3	12	большое число
4	-2,1	меньше нуля
5	0	0,
6	текстовое значение	текст
7		

Рис. 3.25
Применение формата с шаблоном [Синий][>=10]"большое число";
[Красный][<0]"меньше нуля";[Черный]0,???;[Зеленый]"текст"

Нетрудно догадаться, что, если условие второго блока истинно, к ячейке применяется формат с шаблоном второго блока. Если условие и второго блока не выполнено, к ячейке с числовым значением применяется формат с шаблоном третьего блока. Четвертый блок используется для форматирования ячеек с текстовыми значениями. В качестве иллюстрации рассмотрим формат с шаблоном [Синий][>=10]"большое число";[Красный][<0]"меньше нуля";[Черный]0,???;[Зеленый]"текст". Результат применения такого формата к ячейкам с разными значениями показан на рисунке 3.25.

В соответствии с форматом первый блок шаблона применяется для ячеек, числовые значения которых не меньше 10. В этом случае вместо числового значения синим цветом отображается фраза большое число. Для ячеек с отрицательными числами (которые меньше нуля) применяется формат, в соответствии с которым красным цветом отображается фраза меньше нуля. Во всех прочих случаях черным цветом отображается число-значение ячейки. Применяется формат с шаблоном 0,???: обязательная цифра в целой части и минимум три позиции для цифр в десятичной части (хотя сами цифры могут и не отображаться). Текст отображается зеленым цветом. Точнее, вместо реального текстового значения ячейки отображается слово текст.

На заметку

Фактически, описанный выше способ форматирования относится к применению условного формата, т. е. формата, который различен для ячеек с разными значениями. В Excel существует более простой и эффективный способ создания условного форматирования.

В заключение раздела об обычном (не условном) форматировании, хочется обратить внимание читателя на группу пиктограмм **Число** на вкладке **Главная**. С помощью пиктограмм вкладки основные описанные выше форматы и дополнительные настройки к ним применяются быстро и безболезненно. Пиктограммы вкладки описаны в таблице 3.4.

Кроме того, группа **Число** имеет активную метку (в правом нижнем углу). Щелчок на метке группы приводит к открытию окна **Формат ячеек**.

Таблица 3.4

Пиктограммы группы Числа

Пиктограмма	Описание
Общий	Пиктограмма с раскрывающимся списком основных числовых форматов
	Пиктограмма со списком наиболее популярных финансовых форматов
%	Пиктограмма для применения процентного формата
000	Пиктограмма для применения режима использования разделителя тысяч
,0 ,00	Щелчок на пиктограмме приводит к тому, что в представлении числа увеличивается на одну количество отображаемых после запятой цифр
,00 ,0	Щелчок на пиктограмме приводит к тому, что в представлении числа уменьшается на одну количество отображаемых после запятой цифр

УСЛОВНОЕ ФОРМАТИРОВАНИЕ

Вас не смущает такая постановка вопроса?
Уж больно Западом отдает.

Из к/ф «Старый знакомый»

Для создания условных форматов на вкладке **Главная** в разделе **Стили** есть пиктограмма **Условное форматирование.** Пиктограмма представляет собой раскрывающийся список с командами и меню (рис. 3.26).

Для применения условного формата выделяем диапазон ячеек, к которым будет применяться формат, после чего щелкаем пиктограмму **Условное**

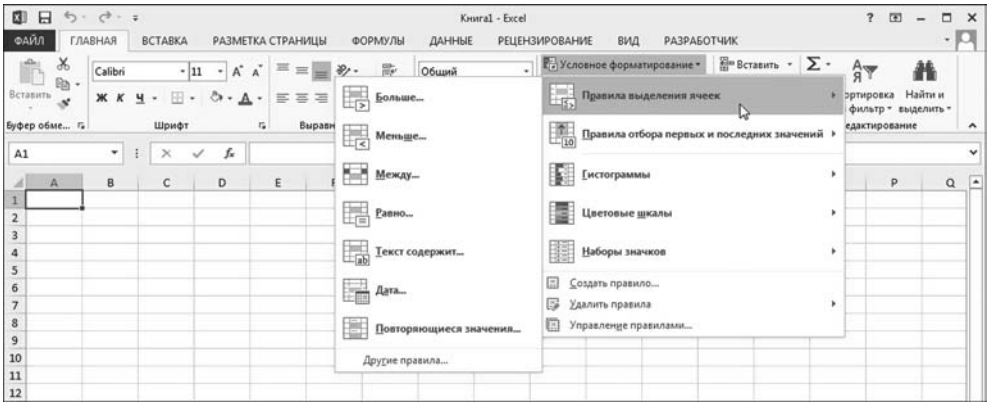


Рис. 3.26
Содержимое пиктограммы **Условное форматирование**

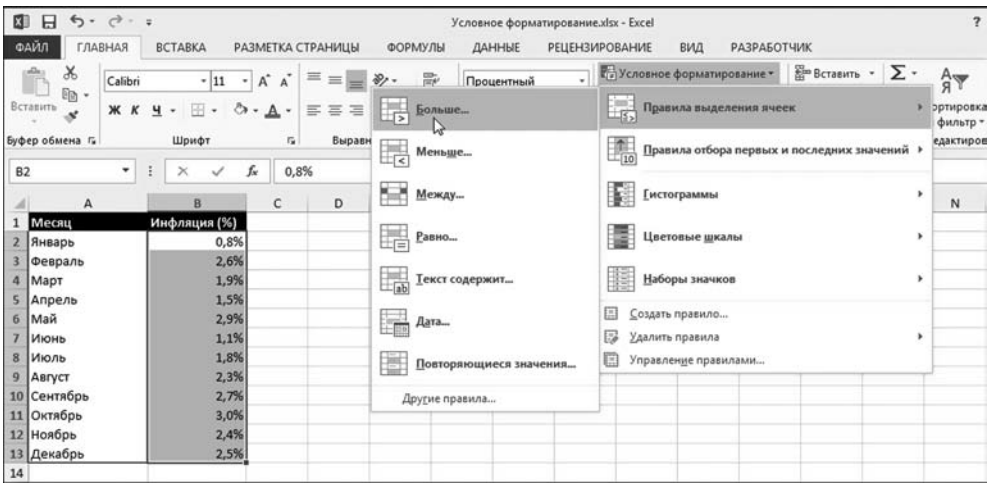


Рис. 3.27
Применение условного форматирования: выделение ячеек, значения которых больше некоторой величины

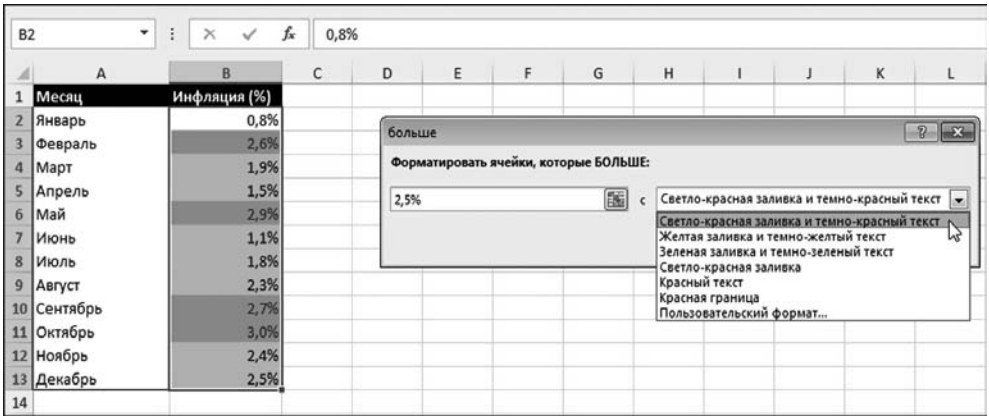


Рис. 3.28
Настройка параметров условного форматирования

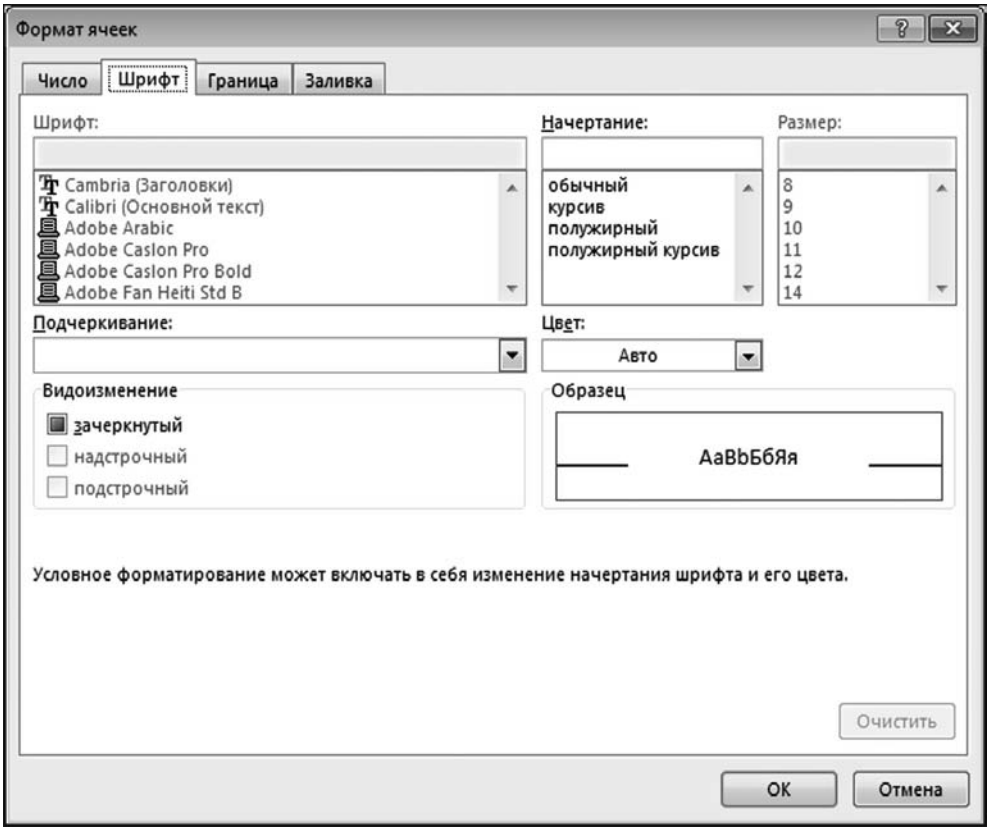


Рис. 3.29
Окно настроек формата ячейки (активны не все опции)

форматирование. В данном случае воспользуемся данными об уровне инфляции по месяцам на протяжении года. Выделим форматированием ячейки, содержащие значения, большие 2,5%. Для этого выделяем диапазон ячеек B2:B13, содержащих числовые данные, а затем в команде-меню **Правила выделения ячеек** выбираем команду **Больше** (рис. 3.27).

В результате открывается диалоговое окно для настройки параметров условного форматирования (рис. 3.28).

В левом поле окна настройки указывается граничное числовое значение или ссылка на ячейку (если значение считывается из ячейки). В правом раскрывающемся списке выбирается способ выделения ячеек с применяемым форматом. Способ форматирования выбираем из списка или создаем собственный формат, для чего в раскрывающемся списке выбираем команду **Пользовательский формат**. В результате откроется уже знакомое нам окно **Формат ячеек**, правда, с ограниченными функциональными возможностями (рис. 3.29).

В этом окне выполняются необходимые настройки. В данном случае формат такой: заливка красного цвета и полужирный шрифт. Результат применения условного формата к документу показан на рисунке 3.30.

К одним и тем же данным применимо несколько условных форматов. Другими словами, после применения первого условного формата применим еще один. Для этого снова выделяем диапазон ячеек B2:B13 и в списке команд пиктограммы **Условное форматирование** выбираем команды **Правила отбора первых и последних значений ▷ Ниже среднего** (рис. 3.31).

Данное форматирование применяется к ячейкам, значения которых меньше среднего значения по диапазону данных. При этом среднее значение вычисляется автоматически — вычислять его в явном виде нет необходимости. Окно настройки формата показано на рисунке 3.32.

Выбираем предопределенный формат: темно-зеленый текст на фоне зеленой заливки. Результат применения формата показан на рисунке 3.33.

	A	B	C
1	Месяц	Инфляция (%)	
2	Январь	0,8%	
3	Февраль	2,6%	
4	Март	1,9%	
5	Апрель	1,5%	
6	Май	2,9%	
7	Июнь	1,1%	
8	Июль	1,8%	
9	Август	2,3%	
10	Сентябрь	2,7%	
11	Октябрь	3,0%	
12	Ноябрь	2,4%	
13	Декабрь	2,5%	
14			

Рис. 3.30
Результат применения условного форматирования: выделены ячейки со значениями, большими 2,5%

Таким образом, как и ранее, ячейки со значениями, большими 2,5%, выделяются жирным шрифтом и красным фоном, а ячейки со значением меньше среднего выделяются зеленым фоном и темно-зеленым шрифтом.

Хотя в данном случае форматы не накладываются, в принципе не исключено, что один формат «перекроет» другой, т. е. к одним и тем же ячейкам формально применяется сразу несколько форматов (поскольку значения в ячейках удовлетворяют условиям для применения этих форматов). Для разрешения таких и не только таких ситуаций в списке команд пиктограммы **Условное форматирование** имеется команда **Управление правилами** (см. рис. 3.34).

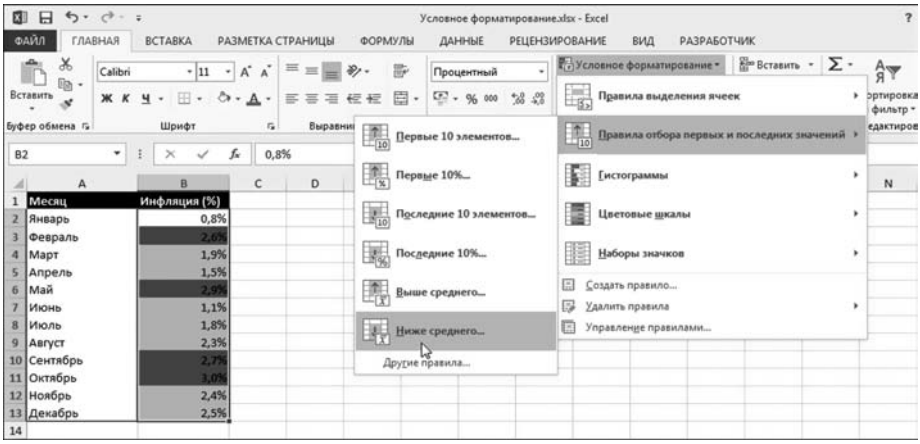


Рис. 3.31
Применение еще одного условного формата:
выделяются ячейки со значениями ниже среднего

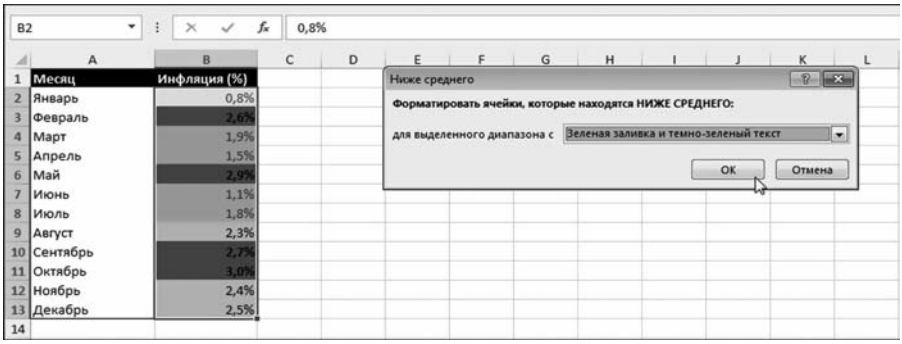
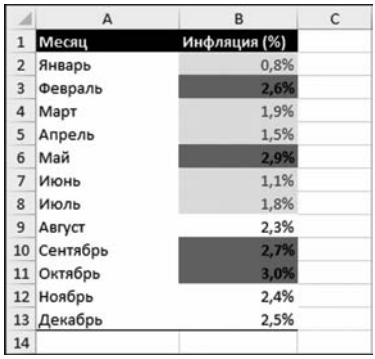


Рис. 3.32
Настройка параметров условного форматирования
для выделения ячеек со значениями ниже среднего



	A	B	C
1	Месяц	Инфляция (%)	
2	Январь	0,8%	
3	Февраль	2,6%	
4	Март	1,9%	
5	Апрель	1,5%	
6	Май	2,9%	
7	Июнь	1,1%	
8	Июль	1,8%	
9	Август	2,3%	
10	Сентябрь	2,7%	
11	Октябрь	3,0%	
12	Ноябрь	2,4%	
13	Декабрь	2,5%	
14			

Рис. 3.33
Результат применения двух условных форматов:
выделены ячейки со значениями больше 2,5% и со значениями ниже среднего

С ее помощью открываем окно **Диспетчер правил условного форматирования**, с помощью которого создаются новые правила (форматы), удаляются правила (форматы), изменяется порядок применения форматов и режим их применения. Окно показано на рисунке 3.35.

В частности, в списке **Показать правила форматирования для** выбирается лист книги с диапазоном ячеек для применения форматирования или активный диапазон (ячейка). Пиктограммы **Создать правило**, **Изменить правило** и **Удалить правило** предназначены соответственно для создания, изменения и удаления правил. Сами правила списком отображаются в основной части окна. В разделе **Формат** видно, как выглядит отформатированное значение. В разделе **Применяется к** отображается диапазон ячеек, к которым применяется соответствующий формат. Если установить флажок опции **Остановить, если истина**, то после применения этого формата (правила) другие форматы (правила) к уже отформатированным ячейкам применяться не будут. Порядок применения правил определяется их последовательностью в общем списке правил. Изменить порядок правил в списке (и соответственно порядок применения правил) позволяют пиктограммы со стрелками сверху над списком правил.

Интерес представляет процедура изменения правила. Для этого правило выделяем и щелкаем пиктограмму **Изменить правило**. Откроется окно **Изменение правила форматирования**, как показано на рисунке 3.36.

В верхней области окна находится список для выбора типа правила. Вид нижней части окна зависит от того, что выбрано в этом списке. Но вообще там находятся опции для выполнения настроек формата. В данном случае с помощью кнопки **Формат** открывается соответствующее диалоговое окно, в котором настраиваются параметры формата, а в раскрывающихся списках и поле определяется условие, при выполнении которого применяется формат (правило).

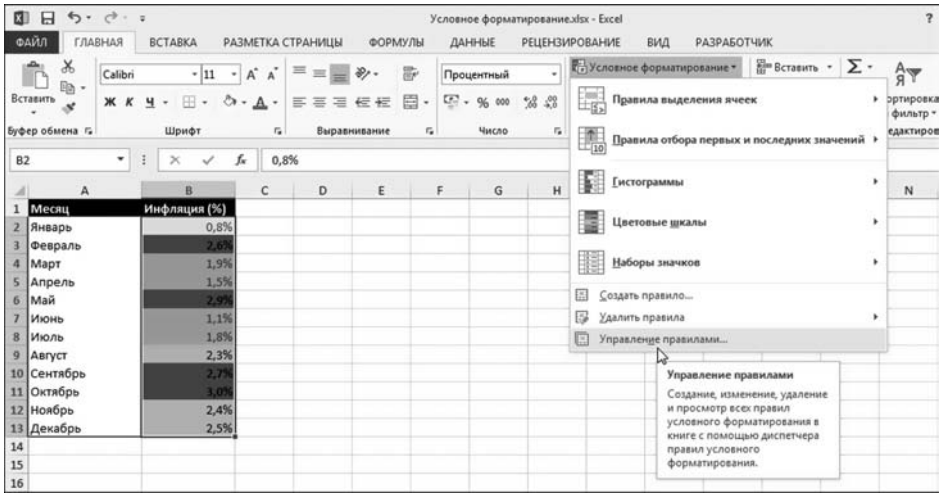


Рис. 3.34
Управление правилами условного форматирования

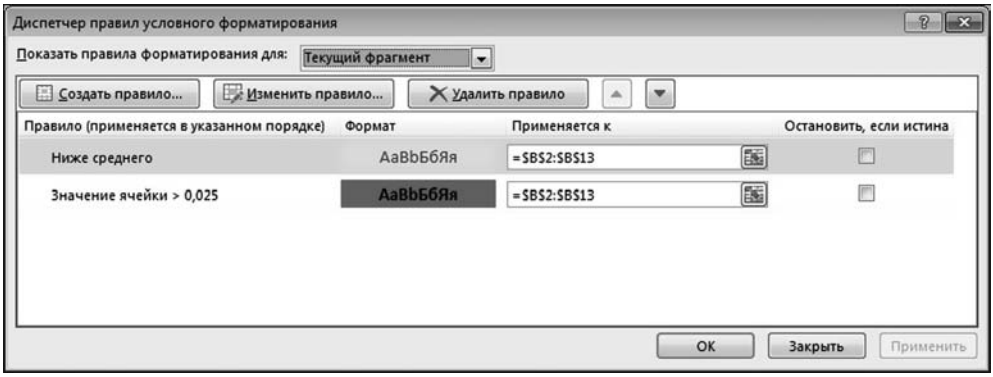


Рис. 3.35
Окно диспетчера правил условного форматирования

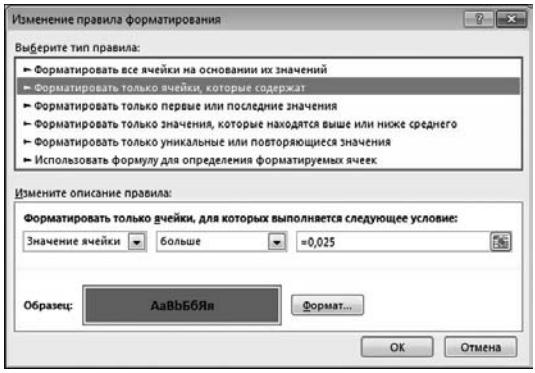


Рис. 3.36
Окно изменений правил условного форматирования

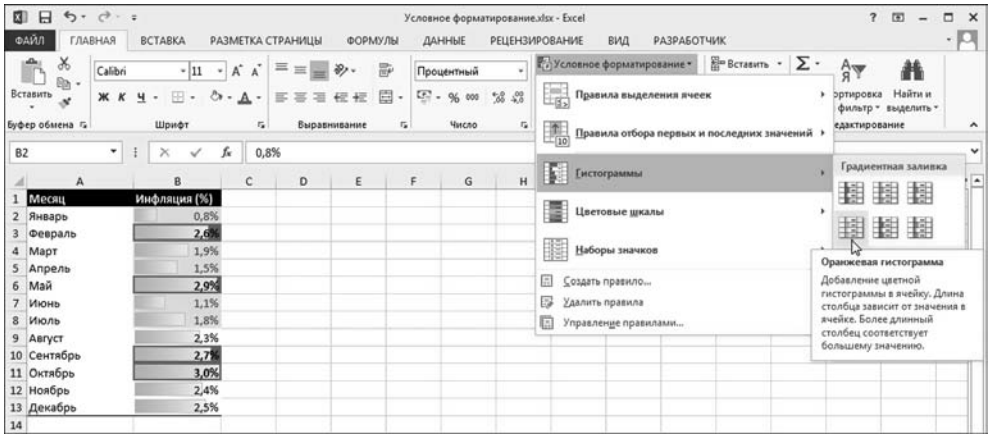


Рис. 3.37
Условный формат на основе гистограммы

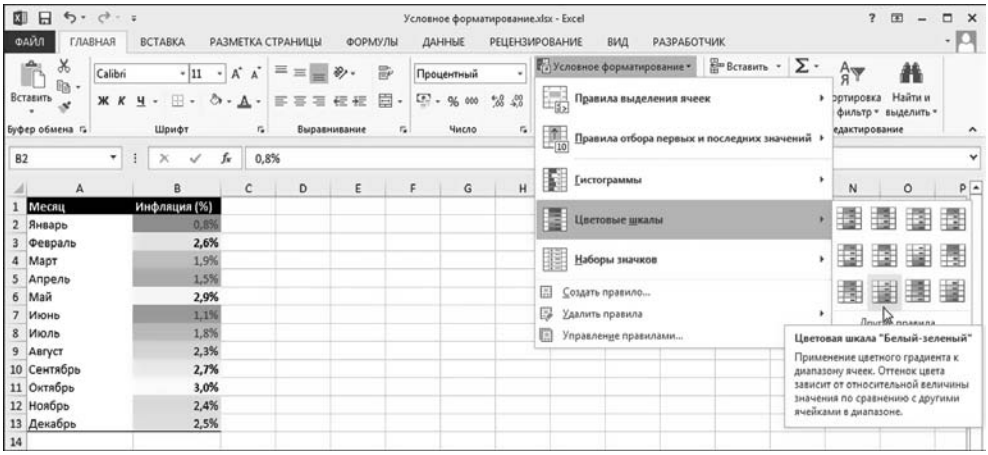


Рис. 3.38
Условный формат с градиентной закраской

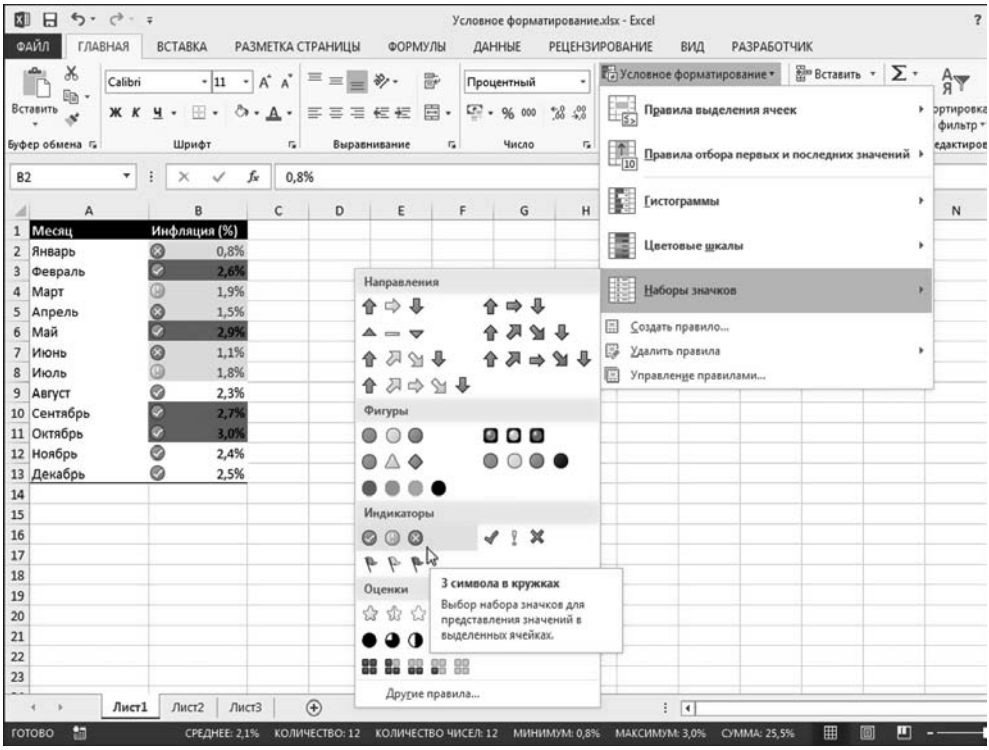


Рис. 3.39
Условный формат с использованием значков

Хочется отметить еще несколько красивых способов условного форматирования. На рисунке 3.37 показано условное форматирование с использованием гистограммы.

Такой формат применяется командой **Гистограммы** (с последующим выбором конкретного типа форматирования) в списке команд пиктограммы **Условное форматирование**. В этом формате на фоне ячейки отображаются столбики гистограммы, в соответствии с числовыми значениями ячейки.

Командой **Цветовые шкалы** выбирается условный формат, которым для заливки ячеек используется градиентная раскраска. На рисунке 3.38 выбран режим, при котором ячейки заливаются оттенками зеленого цвета (от белого до темно-зеленого) — чем меньше значение в ячейке, тем зеленее фон заливки.

Очень «веселый» формат представлен в документе на рисунке 3.39.

За этот формат отвечает команда **Набор значков** в списке команд пиктограммы **Условное форматирование**. В ячейке, в зависимости от значения, отображается тот или иной значок. Тип значков выбирается из специальной палитры. Режим применения значков настраивается в окне настройки правил условного форматирования. Окно, напомним, открывается с помощью команды **Управление правилами**.

СТИЛИ

Лично мы знакомимся с Вами сейчас,
а с моим творчеством Вы будете иметь возможность
познакомиться через некоторое время.

Из к/ф «Старый знакомый»

В Excel достаточно много встроенных стилей, которые позволяют форматировать ячейки с данными. На практике для повседневной работы этих стилей более чем достаточно. Применяется встроенный стиль просто — выделяем диапазон ячеек и на вкладке **Главная** в группе **Стили** раскрываем список стилей пиктограммы **Стили ячеек** (см. рис. 3.40).

В этом случае все ячейки диапазона выделяются в одном стиле. Можно к разным ячейкам логической таблицы применять разные стили. Но есть еще более простой способ — применить *стиль таблицы*. Откровенно говоря, стили таблиц имеют более широкое назначение, чем просто выделение разным способом строк ячеек, но в данном случае это не очень принципиально. Итак, раскрываем список пиктограммы **Форматировать как таблицу** в группе **Стили** и выбираем подходящий способ выделения таблицы (см. рис. 3.41).

Здесь и далее под таблицей мы подразумеваем диапазон ячеек, объединенных на уровне логических связей. Такую таблицу будем называть *логической* (в отличие от структурной таблицы, к которой относится и весь рабочий лист). После этого откроется вспомогательное диалоговое окно **Создание таблицы**, в котором просто необходимо указать диапазон ячеек, который мы называем таблицей (см. рис. 3.42).

Результат применения такого «табличного» стиля к диапазону ячеек представлен в документе на рисунке 3.43.

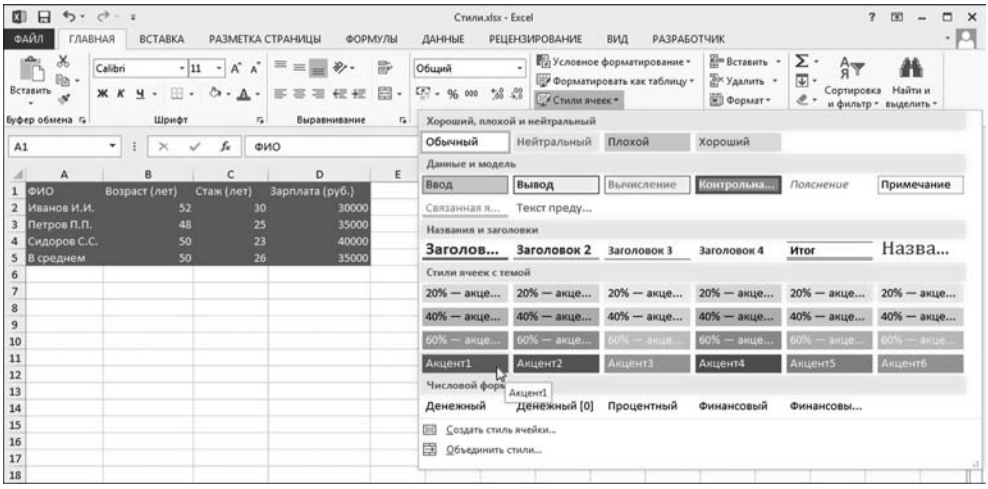


Рис. 3.40
Применение стиля к ячейкам

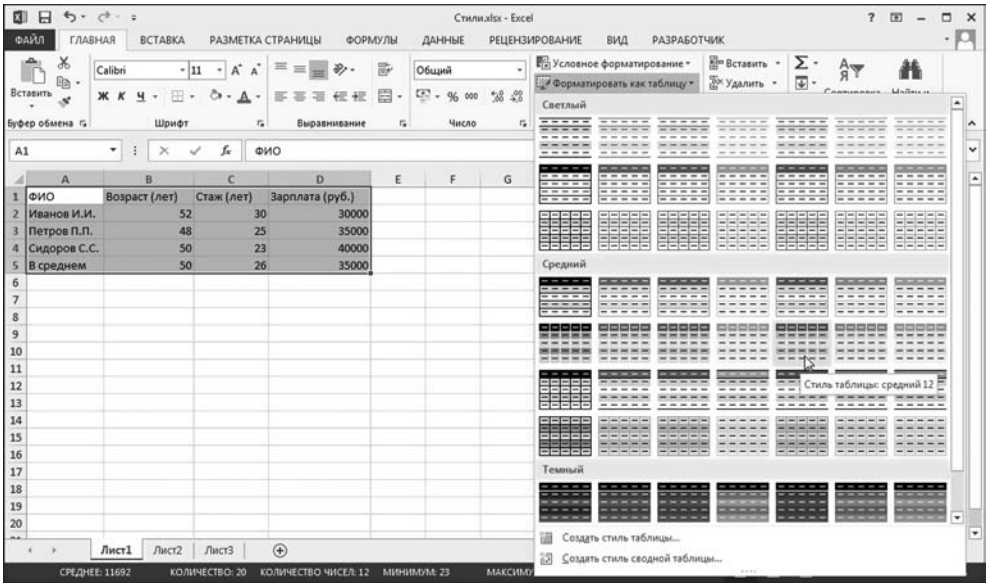


Рис. 3.41
Применение встроенного стиля таблицы

Важно то, что заголовок таблицы очень функциональный. Теперь ячейки заголовка — раскрывающиеся списки, с помощью которых можно скрыть или отобразить строки таблицы или выполнить сортировку ее элементов (рис. 3.44).

Несложно создать собственный стиль. Для этого используем команду **Создать стиль ячейки** в списке команд пиктограммы **Стили ячеек** (см. рис. 3.45).

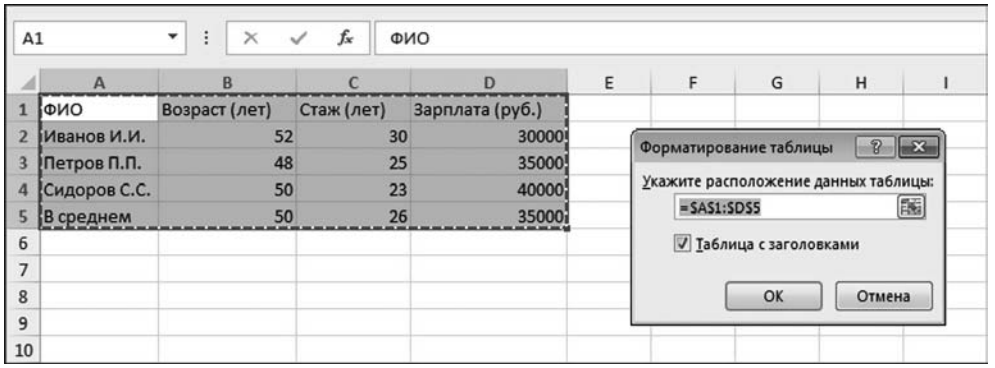


Рис. 3.42
Определение диапазона ячеек для выделения в качестве логической таблицы

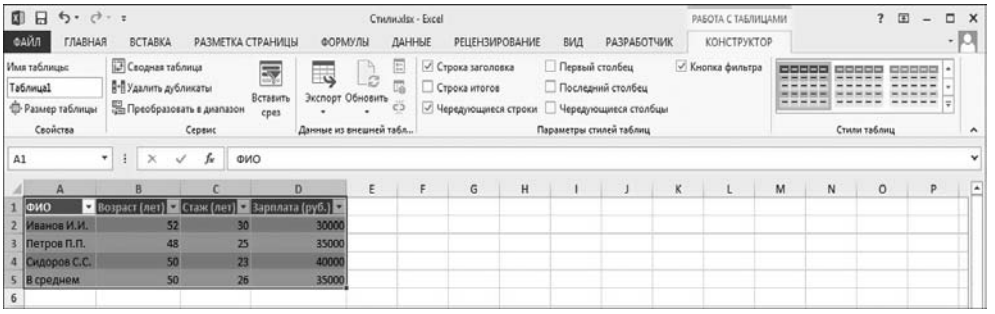


Рис. 3.43
Результат применения встроенного стиля таблицы

В окне **Стиль**, которое открывается при выборе указанной команды, задаются основные параметры создаваемого стиля (см. рис. 3.46).

Название стиля указываем в поле **Имя стиля**. После создания стиля это имя отображается в списке команд пиктограммы **Стили ячеек**. Практически также создается стиль таблицы: выбираем команду **Создать стиль таблицы** из списка команд пиктограммы **Форматировать как таблицу** (см. рис. 3.47).

Правда, окно настроек стиля таблицы (см. рис. 3.48) в этом случае более замысловатое, чем в предыдущем.

Хотя на самом деле ничего сложного здесь нет — выбирается тип элемента и задается способ его форматирования.

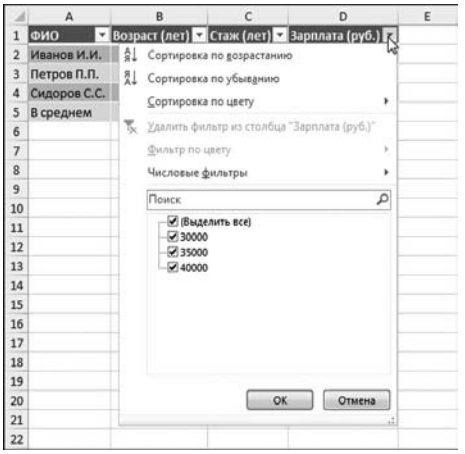


Рис. 3.44
Функциональные заголовки таблицы

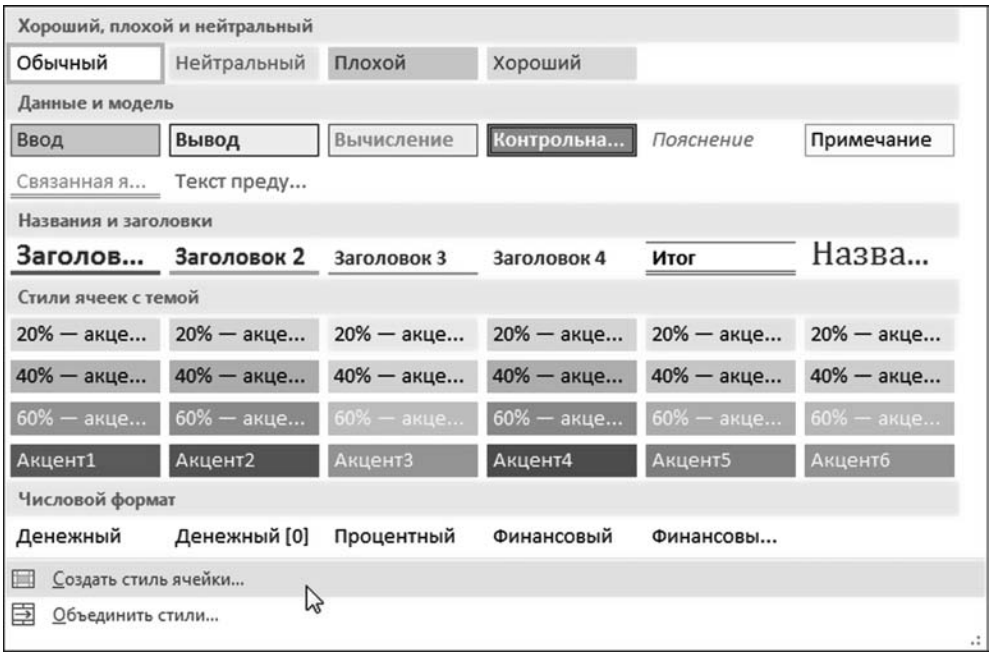


Рис. 3.45
Для создания стиля используем команду **Создать стиль ячейки**

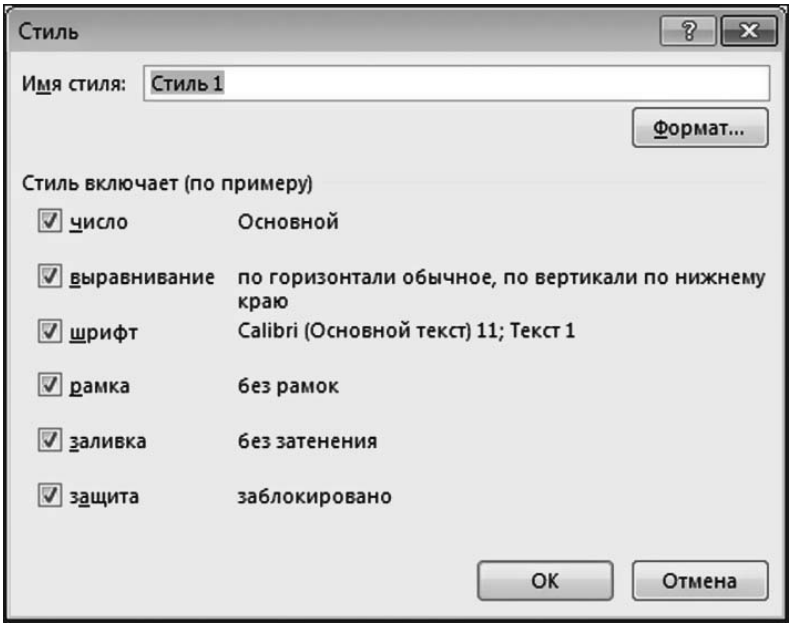


Рис. 3.46
Окно настроек параметров создаваемого стиля

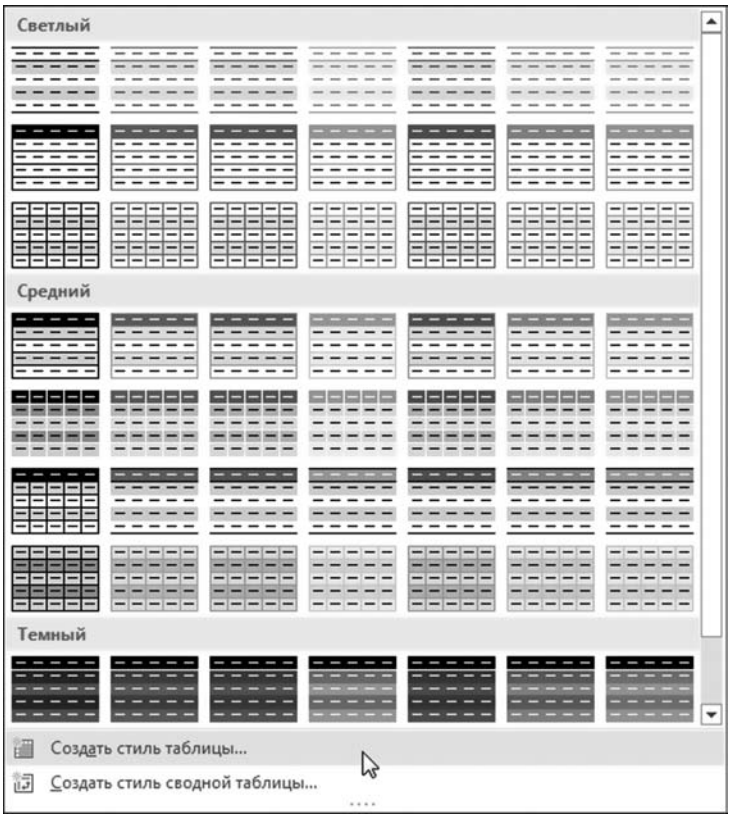


Рис. 3.47
Для создания стиля таблицы используем команду **Создать стиль таблицы**

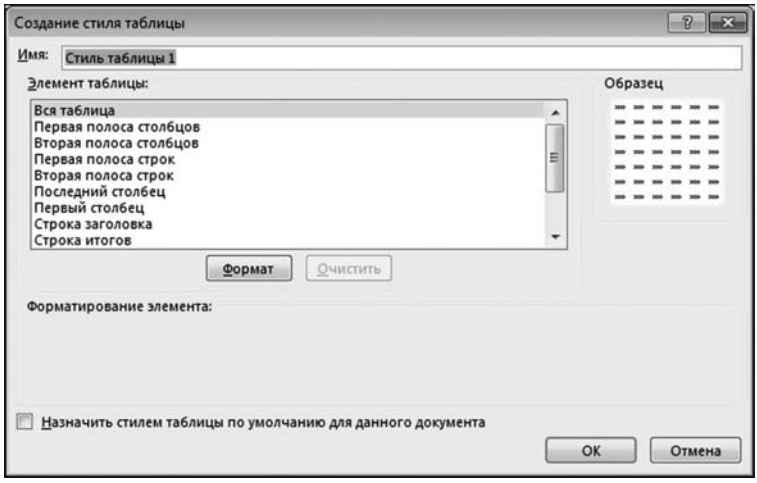


Рис. 3.48
Окно настройки параметров создаваемого стиля таблицы

ГЛАВА 4 НАСТРОЙКИ, РЕЖИМЫ И ПОЛЕЗНЫЕ УТИЛИТЫ

Внимание!
Приготовиться фанфаристам, морякам, цыганам.
Нептун, кончайте курить!

Из к/ф «Старый знакомый»

В Excel можно выполнять нетривиальные вычисления. Более того, в Excel можно вычислять не только легко и быстро, но еще и «комфортно». Методам создания «комфорта» посвящена эта глава. А именно, здесь описываются некоторые концептуально важные настройки приложения Excel, которые выполняются пользователем и влияют на приложение в целом: вид и способ отображения рабочей области, параметры создаваемых документов и ряд других возможностей.

РАЗМЕР ЯЧЕЕК

Тут что важно?
Тут все важно!

Из к/ф «Тридцать три»

Мы уже знаем, что размеры ячеек в рабочих листах можно менять. Обычно размер ячеек меняют для того, чтобы данные в ячейках отображались компактно и красиво.

На заметку

Размеры ячейки — это ее высота и ширина. Следует понимать, что если мы изменяем высоту (ширину) ячейки, то соответствующим образом изменяется высота (ширина) всех ячеек строки (столбца). Ширину и высоту ячеек обычно измеряют в пикселях. По умолчанию ширина ячейки составляет 64 пикселя, а высота — 20 пикселей.

Самый простой способ изменить размеры ячеек — сделать это своими руками (и, разумеется, мышью). Для этого достаточно навести курсор мыши в область полосы нумерации строк или именования столбцов и перетащить мышью соответствующую границу, как показано на рисунке 4.1.

На заметку

При перетаскивании границы ячеек автоматически отображается подсказка с текущим значением ширины/высоты ячейки.

В данном случае изменяется ширина ячеек в столбце D. Результат представлен на рисунке 4.2.

Разумеется, изменить геометрические параметры можно и по-другому. Полезными при этом оказажутся команды в раскрывающемся списке пиктограммы **Формат**, группы **Ячейки** на вкладке **Главная** ленты. На рисунке 4.3 в документе выделена ячейка B3, а в списке команд пиктограммы **Формат** выбрана команда **Ширина столбца**.

В результате открывается диалоговое окно **Ширина столбца**, в котором можно указать размер (в сантиметрах) для ширины ячейки. Окно **Ширина столбца** на фоне рабочего документа показано на рисунке 4.4.

В единственном поле диалогового окна **Ширина столбца** указываем размер для ширины столбца B в данном случае. После щелчка кнопки **OK** изменения вступят в силу.

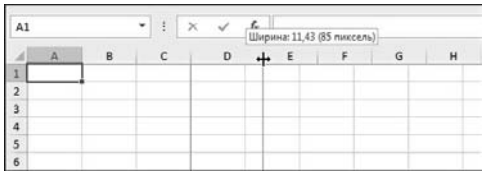


Рис. 4.1
Изменяем ширину ячейки с помощью мыши

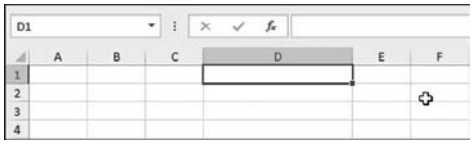


Рис. 4.2
Изменена ширина ячеек в столбце D

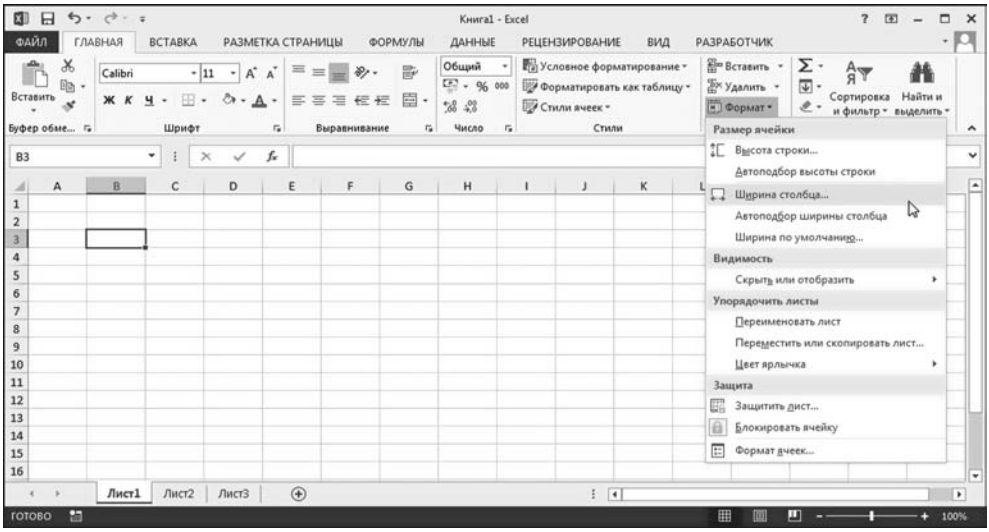


Рис. 4.3
Изменение ширины/высоты ячеек с помощью команд пиктограммы **Формат**

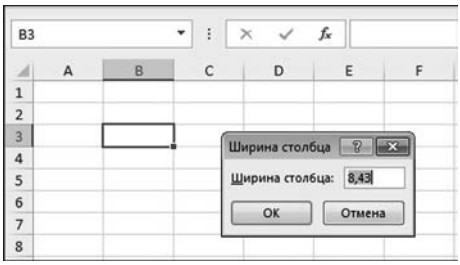


Рис. 4.4
Ширина столбца указывается в поле **Ширина столбца** одноименного окна

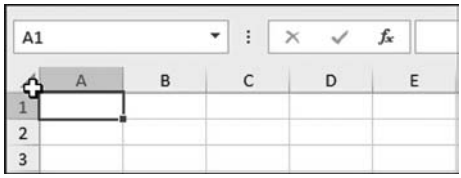


Рис. 4.5
Выделение всех ячеек рабочего листа

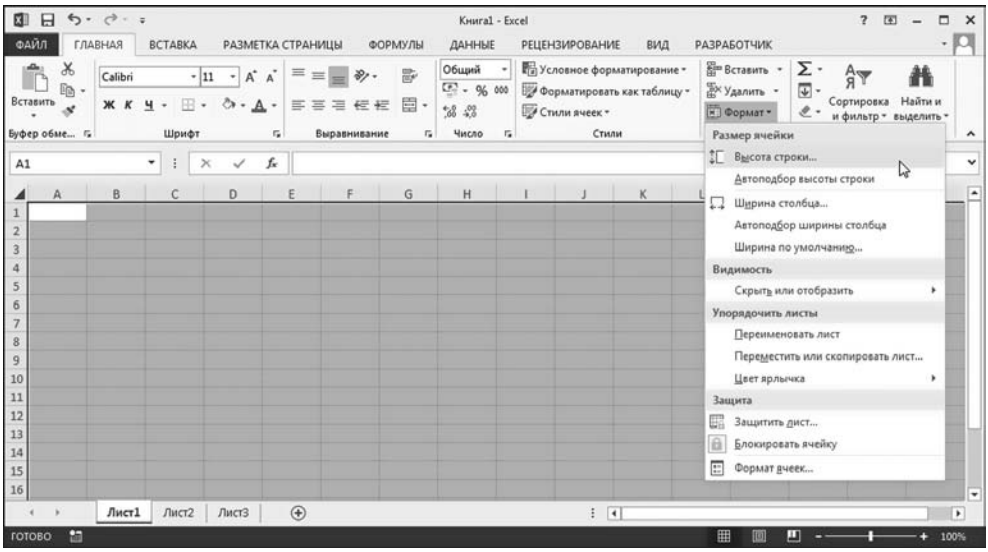


Рис. 4.6
Выбор команды **Высота строки** в режиме выделения всех ячеек рабочего листа

Аналогичным образом изменяется высота ячеек — для этого достаточно воспользоваться командой **Высота строки** в списке команд **Формат** (рис. 4.3). Есть там и другие полезные команды. Например, команды **Автоподбор высоты строки** и **Автоподбор ширины столбца** позволяют подобрать линейные размеры ячеек в соответствии с их содержимым. Вызов команды **Ширина по умолчанию** приводит к открытию уже знакомого нам диалогового окна **Ширина ячеек**. Теперь указанный параметр ширины будет применен для всех ячеек рабочего листа — кроме тех ячеек, для которых размер устанавливался вручную.

На заметку

Команда **Ширина по умолчанию** достаточно коварная. Чтобы убедиться в последнем, желающие могут проделать такой нехитрый эксперимент. Действие

первое: изменяем ширину какого-нибудь столбца. Действие второе: возвращаем ширину этого столбца к исходному значению (по умолчанию). Действие третье: используем команду **Ширина по умолчанию** для изменения ширины всех ячеек рабочего листа. Результат: ширина ячеек того столбца, с которым мы проводили манипуляции, не изменится.

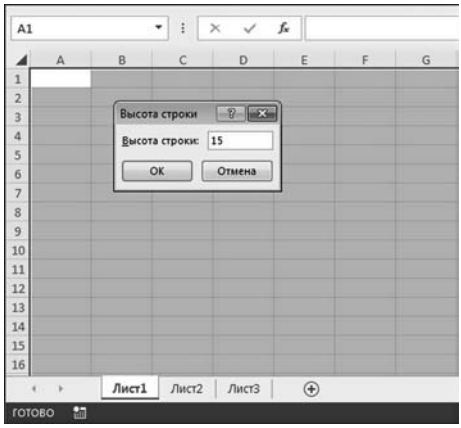


Рис. 4.7
Изменение высоты ячеек рабочего листа

Чтобы изменить высоту/ширину всех ячеек рабочего листа, можно поступить так: выделяем весь рабочий лист и затем изменяем высоту/ширину ячеек. На рисунке 4.5 показано, как щелчком мыши в области пересечения полос именования столбцов и нумерации строк выделяем всю область рабочего листа.

После того как ячейки рабочего листа выделены, в списке команд пиктограммы **Формат** выбираем позицию **Высота строки** (рис. 4.6).

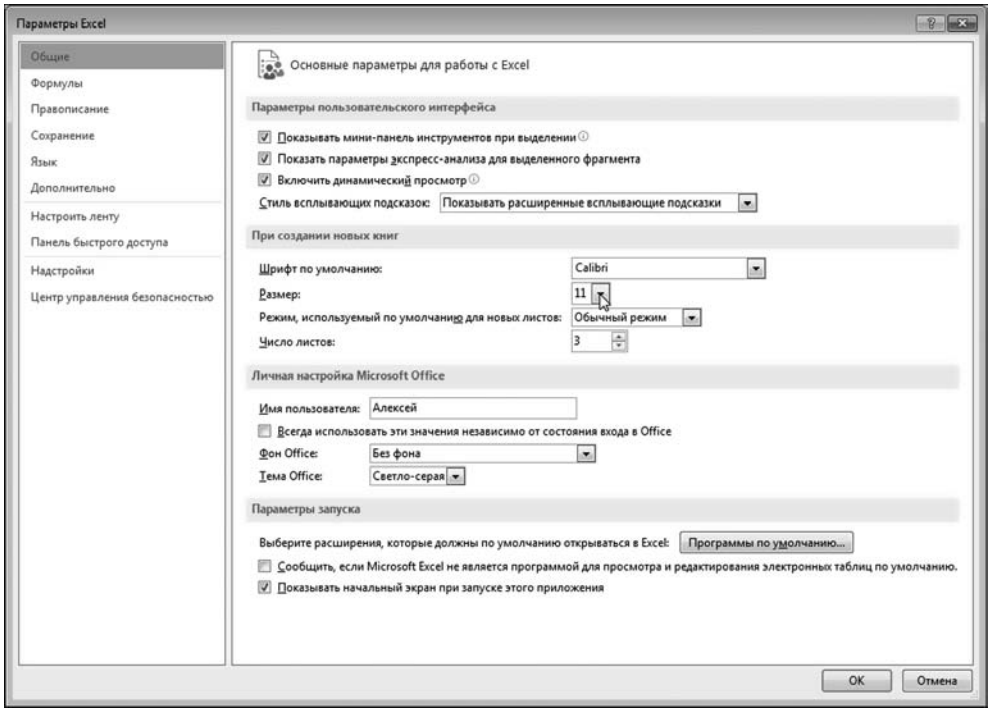


Рис. 4.8
Настройка параметров шрифта по умолчанию

Появляется диалоговое окно **Высота строки** с одноименным полем ввода, в котором указывается высота строки (в сантиметрах) — как показано на рисунке 4.7.

После этого высота всех ячеек рабочего листа станет такой, как мы указали в окне **Высота строки**.

На заметку

Откровенно говоря, размер ячеек по умолчанию определяется настройками шрифта по умолчанию. В первую очередь, это размер шрифта. Шрифт по умолчанию задается в разделе **Общие** диалогового окна настройки приложения **Параметры Excel** (рис. 4.8).

Если изменить, например, предлагаемый по умолчанию размер 11 шрифта на больший, то при создании нового документа увеличатся и размеры ячеек в рабочем документе.

СКРЫТИЕ СТРОК И СТОЛБЦОВ

Были демоны. Мы этого не отрицаем.
Но они самоликвидировались.
Так что прошу эту глупую панику прекратить.

Из к/ф «Иван Васильевич меняет профессию»

Ячеек в рабочем листе много. Даже не так — их очень много. Редкому интеллектуалу удается задействовать в работе все эти ячейки. Возникает вопрос: что делать с «излишками»? Самый простой выход состоит в том, чтобы скрыть ненужные или временно не используемые ячейки (точнее, строки или столбцы ячеек). Как и большинство других процедур в Excel, это

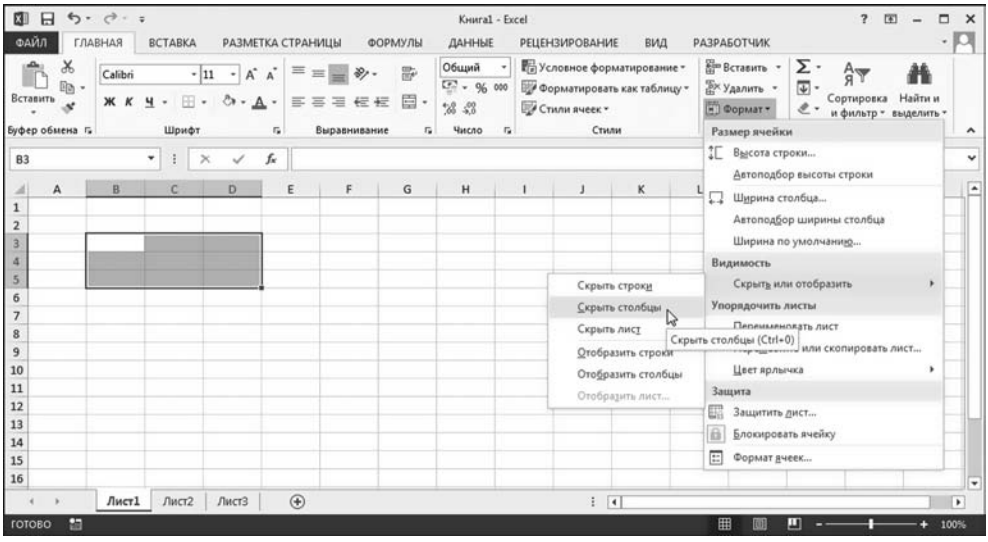


Рис. 4.9
В документе выделен диапазон ячеек B3:D5

действие может выполняться разными способами. Внимательно изучим документ, представленный на рисунке 4.9.

В этом документе мы предусмотрительно выделили диапазон ячеек B3:D5. После этого мы в списке команд пиктограммы **Формат** выбираем позицию **Скрыть или отобразить ▷ Скрыть столбцы** (если мы хотим скрыть столбцы, а если бы хотели скрыть строки, то выбрали бы подменю **Скрыть строки**). В результате столбцы B, C и D будут скрыты. Документ примет вид, как на рисунке 4.10.

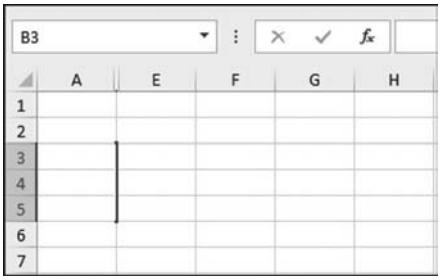


Рис. 4.10
В документе скрыты столбцы B, C и D

О том, что в области отображения не хватает трех столбцов, свидетельствует лишь утолщение на границе раздела столбцов A и E в полосе именования столбцов.

На заметку

В поле имен указана ячейка B3, которая принадлежит одному из скрытых столбцов. Сам столбец не виден. От выделенного ранее диапазона B3:D5 осталось лишь воспоминание в виде толстой линии на три ячейки. Также в полосе нумерации строк отмечены строки с 3-й по 5-ю, как это бывает при выделении диапазонов ячеек. Ничего удивительного в этом нет, поскольку скрытие строк/столбцов не означает, что соответствующий диапазон ячеек не может быть выделен.

Ту же процедуру (имеется в виду скрытие строк или столбцов) можно проделать с помощью команд контекстного меню. Покажем, как скрыть строки с 3-й по 5-ю включительно. Для этого соответствующие строки выделяем и щелчком правой кнопки мыши раскрываем контекстное меню. Нас интересует команда **Скрыть** (рис. 4.11).

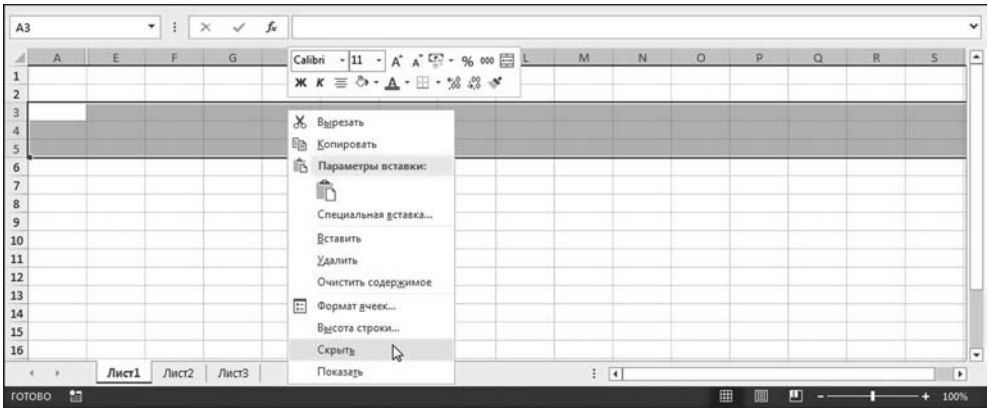


Рис. 4.11
Скрытие строк с помощью команд контекстного меню

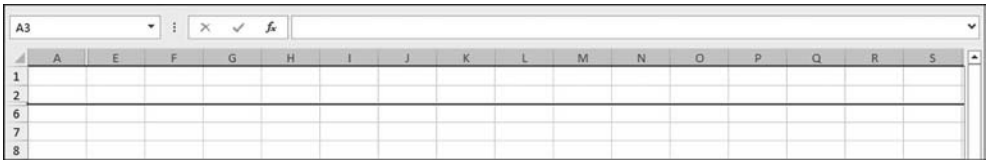


Рис. 4.12
В документе скрыты строки с 3-й по 5-ю включительно (и столбцы B, C и D)

На заметку

Чтобы выделить строку (столбец) наводим курсор мыши на полосу нумерации строк (именования столбцов) — курсор при этом примет вид компактной стрелки, направленной вдоль строки (столбца) — и щелкаем левой кнопкой мыши.

На рисунке 4.12 показан результат — в рабочем документе скрыты три строки. Три столбца в этом же документе были скрыты еще раньше.

Таким образом, мы спрятали в рабочем документе столбцы B, C и D, а также строки 3, 4 и 5. В результате диапазон ячеек B3:D5 полностью скрыт. Тем не менее, доступ к ячейкам этого диапазона получить можно. Для этого в поле имен вводится адрес ячейки (например, B3) или диапазона ячеек (например, B3:D5). На рисунке 4.13 показана подобная ситуация.

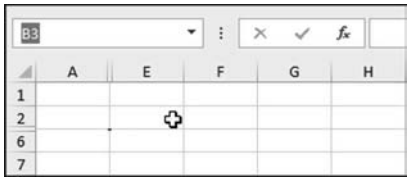


Рис. 4.13
В поле имен вводится адрес ячейки из скрытого диапазона

О том, что выделена ячейка (а именно ячейка B3) свидетельствует только жирная точка на пересечении линий вдоль скрытых столбцов и строк. Значение выделенной ячейки отображается в строке формул. В данном случае там пусто, поскольку соответствующая ячейка пустая. При необходимости значение для выделенной ячейки в скрытом диапазоне можно ввести в строку формул.

Отображаются скрытые ячейки тоже исключительно просто. На рисунке 4.14 выделены столбцы ячеек с A по E включительно. Очевидно, что между этими ячейками находятся скрытые столбцы B, C и D. Для отображения скрытых столбцов в контекстном меню выбираем команду **Показать**.

Также можем воспользоваться командами подменю **Скрыть или отобразить** в списке команд пиктограммы **Формат** группы **Ячейки** на вкладке **Главная** ленты (рис. 4.15).

Разумеется, есть и более простые способы скрыть ячейки. Чтобы надежно спрятать строки, перетаскиваем в области нумерации строк границу нижней строки, пока строки не сложатся «гармошкой». Такой процесс проиллюстрирован на рисунке 4.16.

Фактически, речь идет о том, чтобы установить для скрываемых строк нулевое значение для высоты. Аналогичным образом можем скрыть столбцы в рабочем документе — путем перетаскивания границ столбцов в области полосы именования столбцов.

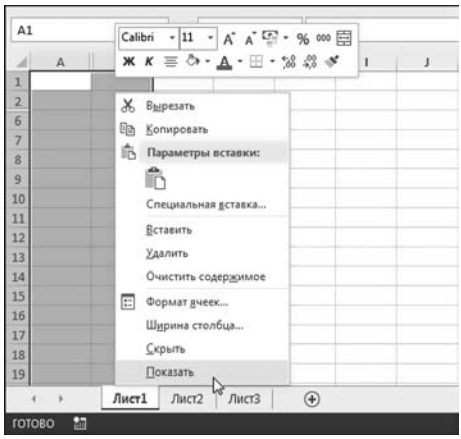


Рис. 4.14
Отображение скрытых столбцов с
помощью команды контекстного меню

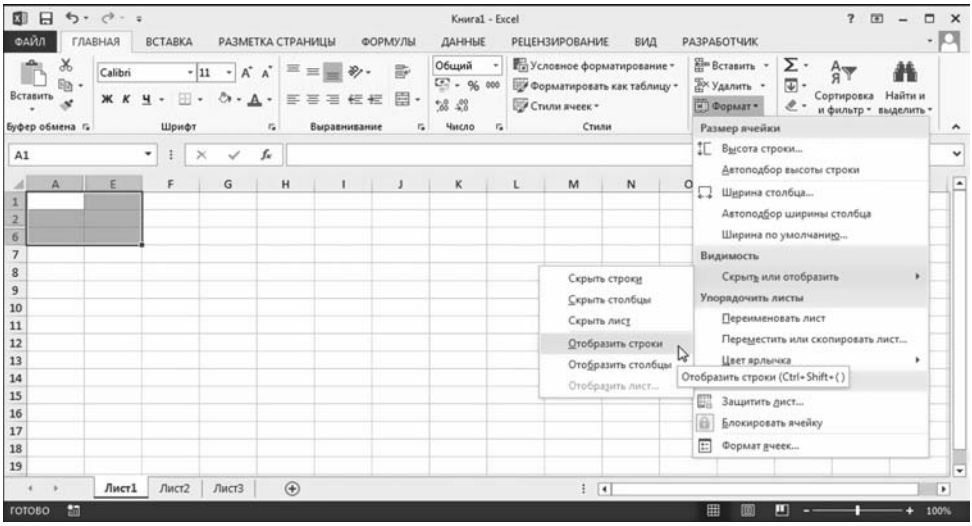


Рис. 4.15
Отображение скрытых строк с помощью команды
пиктограммы **Формат** на вкладке **Главная** ленты

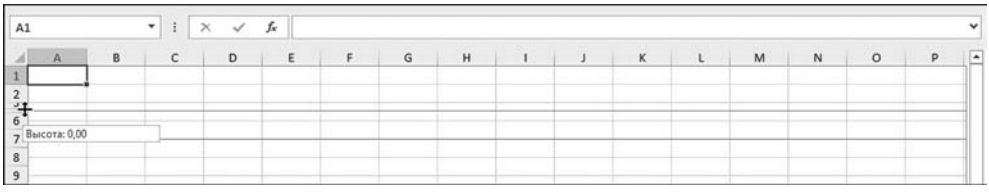


Рис. 4.16
Скрытие строк путем перетаскивания границы в области полосы нумерации строк

НАСТРОЙКА ВИДА РАБОЧЕЙ ОБЛАСТИ

«Аве нове ностра алес»,
что означает «ежели один человек
построил, другой завсегда разобрать может».

Из к/ф «Формула любви»

Здесь мы обсудим некоторые приемы, которые позволяют сделать рабочий документ красивым, а работу с ним — приятной. В первую очередь остановимся на настройках, которые в определенном смысле влияют на наше восприятие приложения Excel. Интересная подборка утилит представлена на вкладке **Вид** ленты. Обратим внимание на группу **Показ**, в которой есть четыре опции — простые, но, вместе с тем, полезные (рис. 4.17).

Назначение опций описано в таблице 4.1.

В обычном режиме доступны три из четырех опций. Все флажки по умолчанию установлены. Опции позволяют отображать/скрывать основную сетку в рабочем документе, строку формул и полосы нумерации строк и именования столбцов. На рисунке 4.18 показано, как будет выглядеть документ, если отменить все, что можно отменить.

Вид у документа удручающий и, как говорится, на любителя. Зачем нужен такой режим — вопрос отдельный. Например, кому-то не нравится разбитый на ячейки документ. Хотя здесь следует отметить, что даже если сетка не отображается, документ все равно состоит из ячеек. Поэтому такой эффект чисто визуальный. Есть другой режим — режим разбивки на страни-

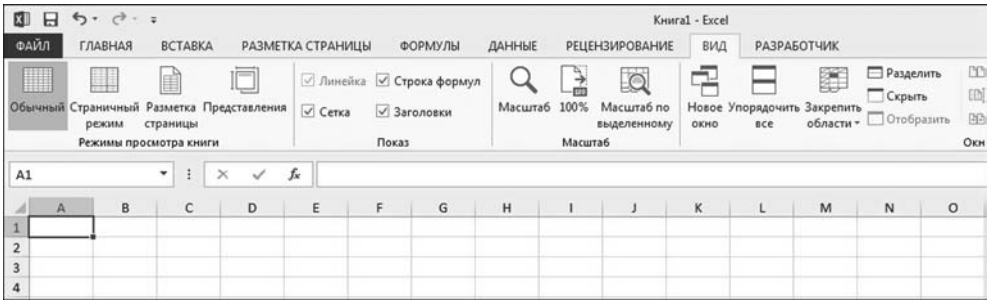


Рис. 4.17
Лента приложения открыта на вкладке **Вид** с группой **Показ**

Таблица 4.1

Назначение опций группы Показ вкладки Вид

Опция	Назначение
Линейка	Опция активна только в режиме разбивки на страницы. Предназначена для отображения масштабной линейки по границам страницы
Сетка	Опция предназначена для отображения основной сетки (по границам ячеек) в рабочем документе
Строка формул	Опция предназначена для отображения строки формул
Заголовки	Опция предназначена для отображения полосы именования столбцов и полосы нумерации строк

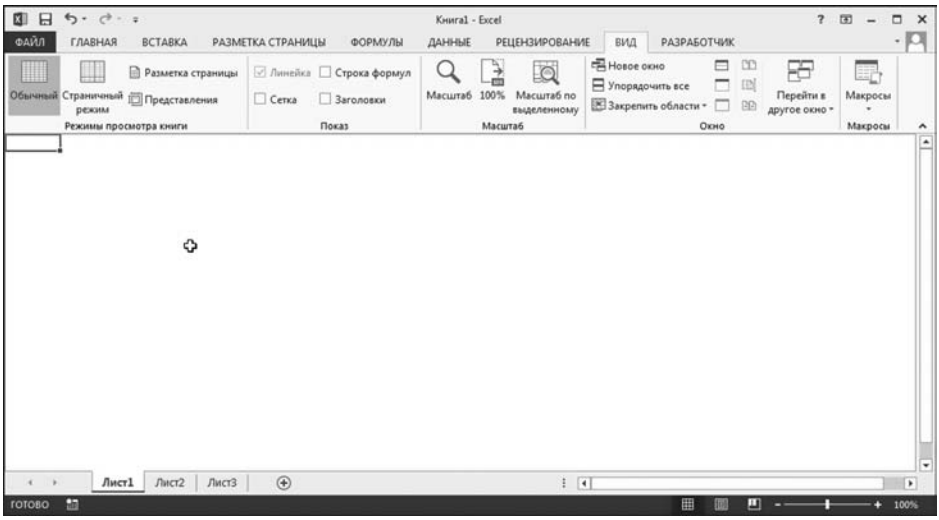


Рис. 4.18
Вид рабочего документа без отображения сетки,
строки формул, полос нумерации строк и именования столбцов

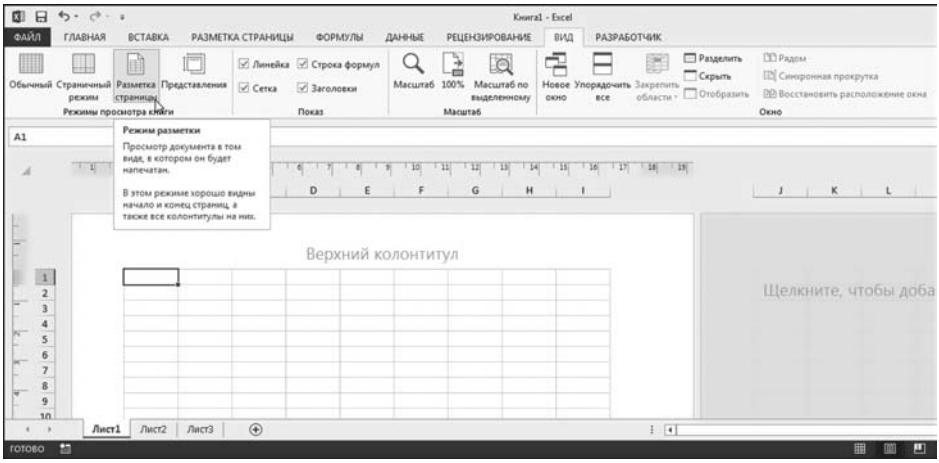


Рис. 4.19
Документ в режиме разбивки на страницы

цы. Перейти в этот режим можно с помощью пиктограммы **Разметка страницы** (вторая слева пиктограмма в группе **Режимы просмотра книги** вкладки **Вид**), как показано на рисунке 4.19.

В этом режиме документ разбивается на страницы, и создается иллюзия, будто работаем с текстовым редактором вроде Word — только область рабочего листа разграфлена сеточкой. В данном режиме активна опция **Линейка** в группе **Показ**. Флажок у опции означает отображение масштабной линейки вдоль верхней и левой границ активной страницы. Удобство режима еще и в

том, что можно сразу задать, например, колонтитулы для страницы. Хотя все эти настройки достаточно условны, поскольку имеют непосредственное отношение лишь к тому, как документ выглядит при выводе на печать.

Полезный прием часто применяется в случае, когда приходится работать с большими (в плане количества используемых ячеек) документами. В области экрана монитора помещается не очень большой диапазон ячеек, поэтому неудобно обращаться к ячейкам, находящимся в разных частях документа. Решение проблемы заключается в том, чтобы компактно и элегантно зафиксировать в области видимости те диапазоны ячеек, которые используются в работе. Добиваемся нужного результата с помощью разбивки рабочего листа на области и закрепления этих областей.

На вкладке **Вид** в группе **Окно** есть пиктограмма-раскрывающийся список **Закрепить области** (рис. 4.20).

Список команд вкладки содержит три позиции: **Закрепить области**, **Закрепить верхнюю строку** и **Закрепить первый столбец**. На рисунке 4.20 выбрана команда **Закрепить верхнюю строку**. Закрепление первой (верхней) строки означает переход в режим, при котором прокрутка (вдоль вертикали) документа не приводит к смещению первой строки — другими словами, все строки прокручиваются, кроме первой. Аналогично обстоит дело с закреплением первого столб-

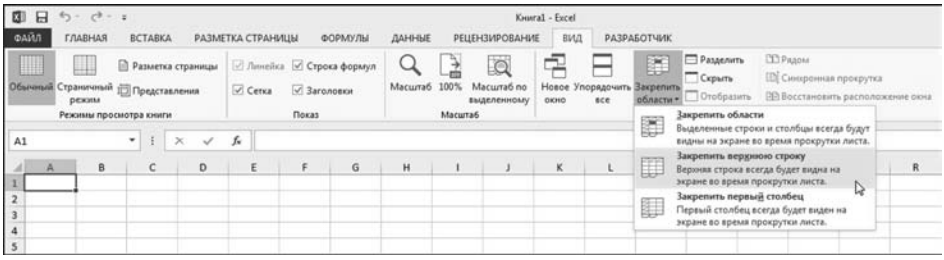


Рис. 4.20
Закрепление верхней строки

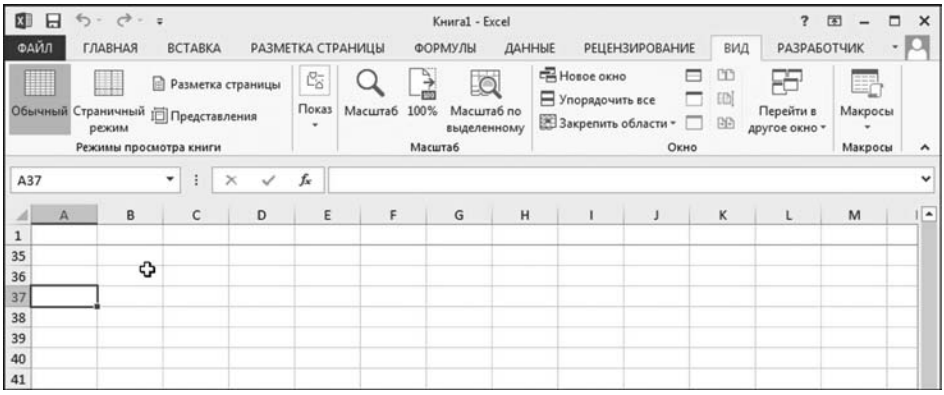


Рис. 4.21
В документе закреплена верхняя строка

ца. Закрепить первый столбец можно с помощью команды **Закрепить первый столбец** в списке команд пиктограммы **Закрепить области**. На рисунке 4.21 показано, как будет выглядеть документ с закрепленной верхней строкой.

На заметку

Закрепить можно или верхнюю строку, или первый столбец. После того как область (строка или столбец) закреплена, в раскрывающемся списке команд пиктограммы **Закрепить области** появляется команда **Снять закрепление областей**, с помощью которой отменяется режим закрепления областей.

Стоит обратить внимание, что на рисунке 4.21 под верхней строкой отображается сразу 35-я строка. Причем строки со 2-й по 34-ю не скрыты, а просто не отображаются. О том, что верхняя строка закреплена, свидетельствует тонкая вертикальная линия по нижней границе верхней строки.

Закрепить можно не только верхнюю строку или первый столбец, а нечто большее. На рисунке 4.22 в рабочем документе выделена ячейка D5, а в списке команд пиктограммы **Закрепить области** выбрана одноименная пиктограмма.

В результате закрепляется область по границе диапазона ячеек A1:C4. Границы закрепленной области выделяются по вертикали и горизонтали тонкими линиями (см. рис. 4.23). Особенность ситуации в том, что прокрутка области документа не смещает диапазон ячеек A1:C4. При этом прокручиваются столбцы справа от столбца C и строки снизу под 4-й строкой — как, например, видно из рисунка 4.23.

Полезной станет пиктограмма **Разделить** в группе **Окно** вкладки **Вид** (см. рис. 4.24).

Эта пиктограмма является переключателем и позволяет отображать/отменять границы разбивки на области (без закрепления).

На заметку

Если документ разделить на области и не закреплять их, можно будет в этих областях просматривать разные блоки документа. Правда, удовольствие достаточно сомнительное. При этом у смежных областей совпадают строки, а у расположенных одна под другой областей совпадают столбцы.

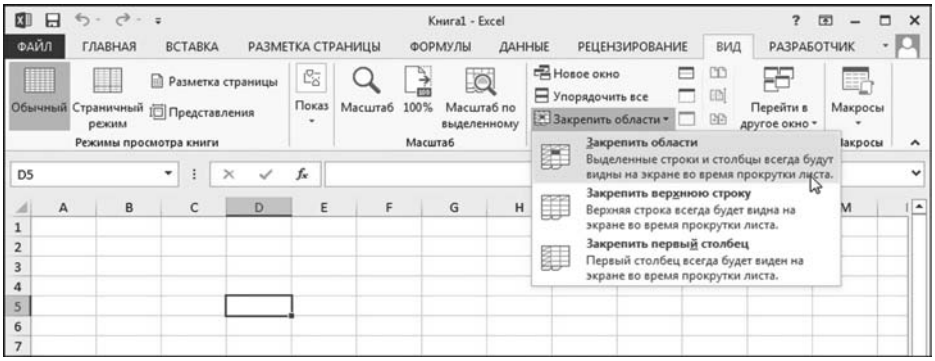


Рис. 4.22
Закрепление области в рабочем документе

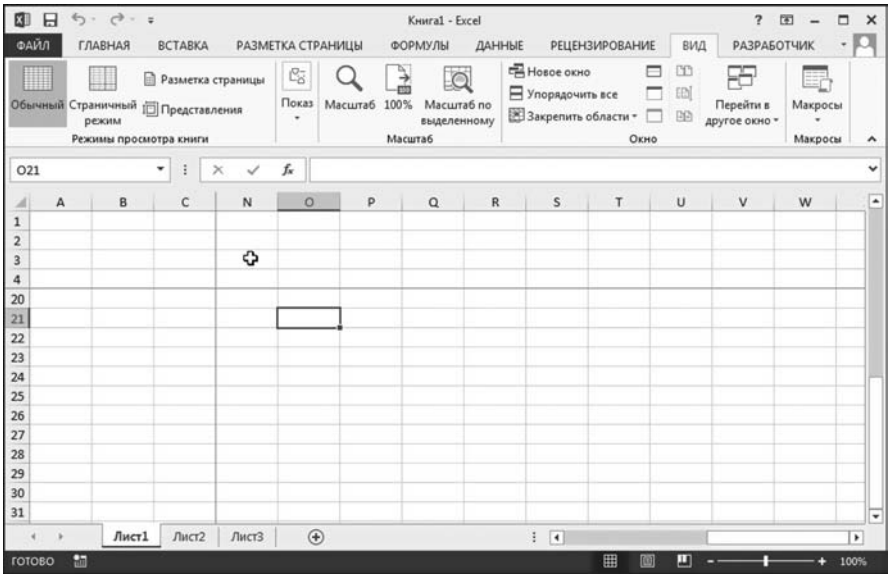


Рис. 4.23
Документ с закрепленной областью по границе диапазона ячеек A1:C4

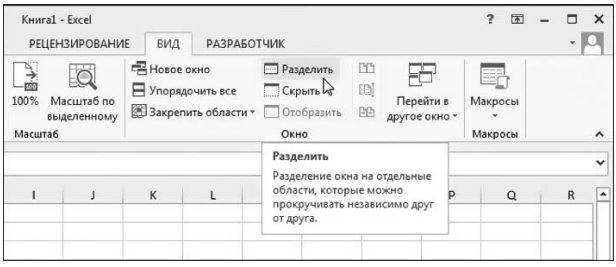


Рис. 4.24
Разделение окна на области с помощью пиктограммы **Разделить**

Процесс перемещения горизонтальной границы области закрепления выглядит примерно так, как проиллюстрировано на рисунке 4.25.

На рисунке 4.26 видим, как может сложиться ситуация при перемещении вертикальной границы области закрепления.

Следует иметь в виду, что наличие границ в том виде, как это было показано выше, не означает закрепления области. Чтобы закрепить область, необходимо еще выполнить команду **Закрепить область** в списке команд пиктограммы с таким же названием.

Убрать границы незакрепленной области можно, перетащив их за пределы рабочей области, — горизонтальную вверх, а вертикальную влево. Поможет и двойной щелчок на незакрепленной границе. Следует также учесть, что в левой верхней части рабочего документа совсем не обязательно закрепляется область с левой верхней ячейкой A1. Схема действий такая. Во-пер-

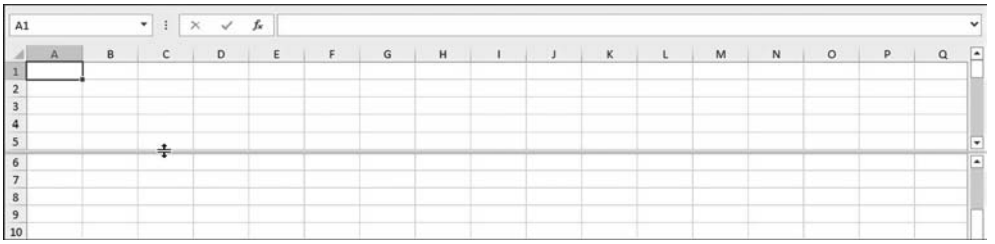


Рис. 4.25
Перемещение горизонтальной границы области закрепления

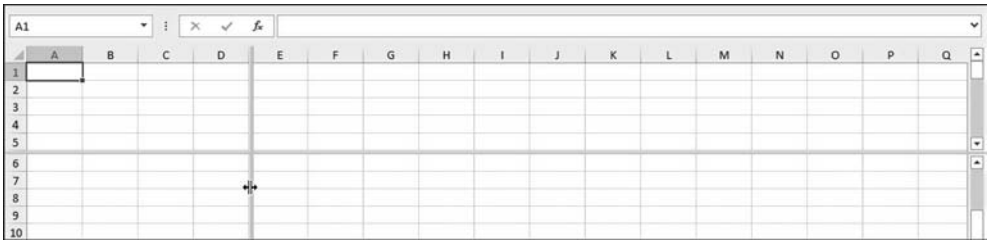


Рис. 4.26
Перемещение вертикальной границы области закрепления

вых, устанавливаются границы области закрепления, но закрепление не выполняется. Затем определяем закрепляемую область: например, выделяем ячейку в левом верхнем квадрате и прокруткой вдоль вертикали и горизонтали выбираем нужную область для закрепления.

На заметку

В предыдущих версиях Excel закрепить область можно и более прозаичным способом — с помощью специальных ползунков, расположенных на границах полос прокрутки документа. На соответствующий ползунок следует навести курсор мыши (курсор при этом принимает вид двунаправленной стрелки) и перетащить соответствующую границу в нужное место. На рисунке 4.27 проиллюстрирован процесс захвата ползунка для перемещения горизонтальной границы области закрепления.

Как захватить ползунок для перемещения вертикальной границы области закрепления, показано на рисунке 4.28.

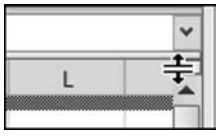


Рис. 4.27
Захват ползунка для перемещения горизонтальной границы области закрепления

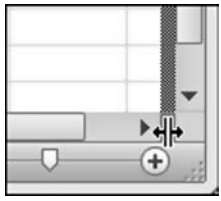


Рис. 4.28
Захват ползунка для перемещения вертикальной границы области закрепления

УПРАВЛЕНИЕ ОКНАМИ

Я здесь планировал спортивную группу на подъемном кране — современное прочтение индустриальной темы. А к этой самодеятельности дворца культуры я никакого отношения не имею!

Из к/ф «Старый знакомый»

Несколько замечаний сделаем по поводу того, как управляться сразу с несколькими рабочими окнами приложения. Ничего сложного в этом нет, просто есть несколько полезных пиктограмм, на которые стоит обратить внимание. Например, хотим мы открыть несколько копий одного документа. Зачем это нужно? Иногда бывает быстрее переключаться между рабочими окнами одного и того же документа, чем пролистывать этот документ в поисках нужной ячейки. Так или иначе, решаем создать копию текущего рабочего документа. Для этого используем пиктограмму **Новое окно** (рис. 4.29).

В результате один и тот же документ предстает в нескольких «ипостасях». Причем «ипостасей» может быть достаточно много. Выбрать нужную версию можем с помощью раскрывающегося списка пиктограммы **Перейти в другое окно** (рис. 4.30).

В раскрывающемся списке отображаются названия всех открытых документов, с учетом их кратности, так сказать.

Если рабочих документов открыто очень много, часть окон можно скрыть. Для этого используем пиктограмму **Скрыть** в группе **Окно** вкладки **Вид** (рис. 4.31).

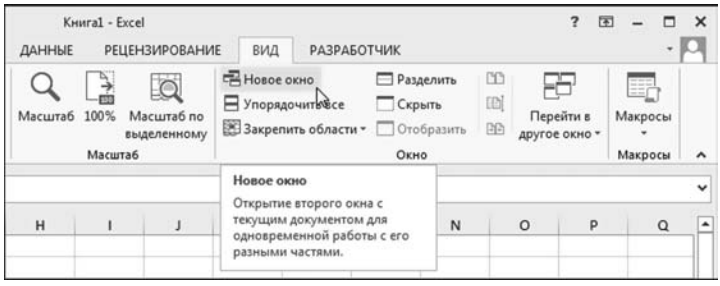


Рис. 4.29
Создание копии рабочего документа

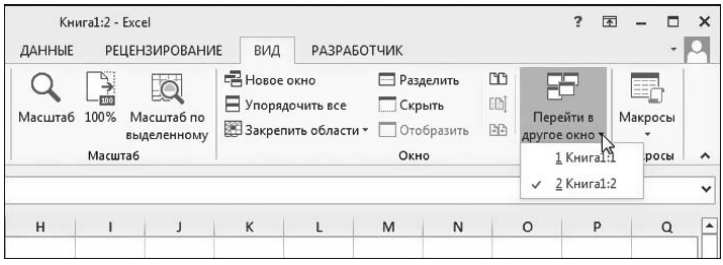


Рис. 4.30
Работа с несколькими окнами

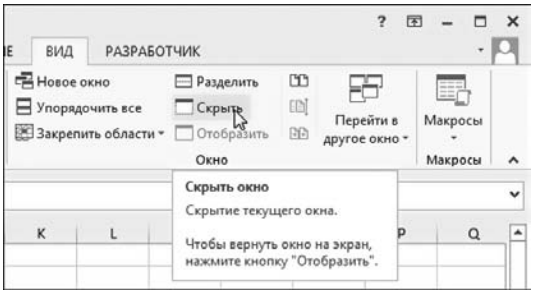


Рис. 4.31
Чтобы скрыть/отобразить текущее рабочее/скрытое окно используем пиктограмму **Скрыть**/**Отобразить**

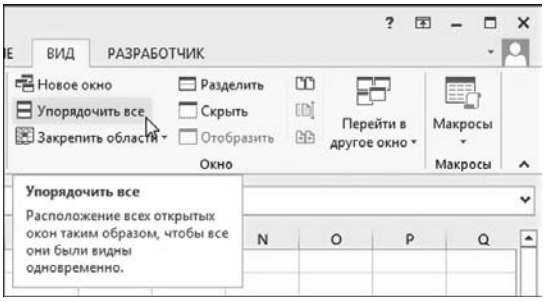


Рис. 4.32
Для упорядочивания рабочих окон используем пиктограмму **Упорядочить все**

Щелчок на пиктограмме **Скрыть** приводит к тому, что текущий рабочий документ будет скрыт (но не закрыт). Если имеются скрытые документы, активна пиктограмма **Отобразить** (рис. 4.31), которая позволяет отображать скрытые окна. При щелчке на этой пиктограмме открывается дополнительное окно, в котором представлен список скрытых документов. В диалоговом окне следует выбрать те документы, окна которых хотим отобразить (отменить режим их скрытия).

Для упорядочивания открытых окон используем пиктограмму **Упорядочить все** (рис. 4.32).

Откроется диалоговое окно **Расположение окон** (рис. 4.33), в котором с помощью переключателей задаем способ упорядочивания (расположения на экране) окон.

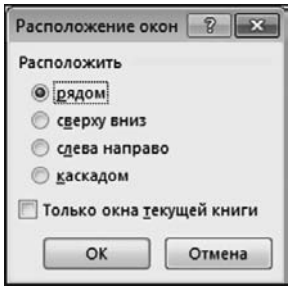


Рис. 4.33
Способ упорядочивания окон выбираем в окне **Расположение окон**

На заметку

Есть мнение, что Юлий Цезарь мог делать до семи разных дел одновременно. Сложно сказать, сколько рабочих окон он смог бы «держать под контролем», но для обычного пользователя двух-трех окон более чем достаточно.

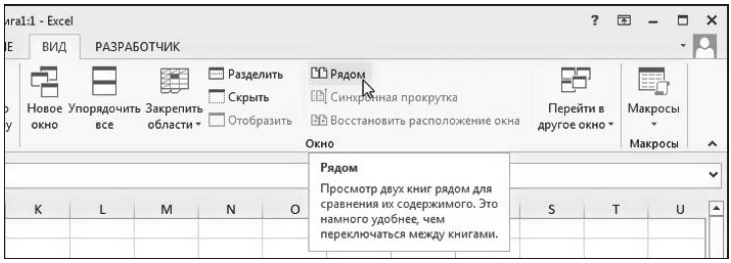


Рис. 4.34
Переходим в режим сравнения рабочих документов

В некоторых случаях возникает необходимость сравнить два документа. По умолчанию подразумевается, что первый из двух сравниваемых документов — это текущий рабочий документ. Чтобы выбрать второй документ для сравнения, щелкаем пиктограмму-переключатель с изображением двух смежных листов. Пиктограмма находится в верхнем ряду в центральной части группы **Окно** вкладки **Вид** (рис. 4.34).

Если открытых документов (окон документов) больше двух, переход в режим сравнения предваряется открытием диалогового окна для выбора второго документа для сравнения. Когда в разработке всего два окна, второе окно для сравнения определяется автоматически.

На заметку

Если открыт всего один документ, то пиктограмма-переключатель перехода в режим сравнения неактивна.

В режиме сравнения удобно сопоставлять содержимое двух документов.

На заметку

После того, как документы для сравнения выбраны, становятся активными еще две пиктограммы, сразу под пиктограммой перехода в режим сравнения документов. С помощью этих пиктограмм можно перейти в режим синхронной прокрутки документов или восстановить исходное положение окон сравниваемых документов (одно окно под другим).

ДОБАВЛЕНИЕ ФОНА И ДРУГИЕ НАСТРОЙКИ

Ну что ж, придется повысить напряжение.
Опасно? Конечно, опасно!
Но риск, как говорится...

Из к/ф «Иван Васильевич меняет профессию»

В рабочий документ (рабочий лист) можно добавить фоновый рисунок. На вкладке **Разметка страницы** в группе **Параметры страницы** находим пиктограмму **Подложка** (рис. 4.35).

После щелчка на этой пиктограмме открывается системное диалоговое окно **Вставка картинок** (рис. 4.36), в котором указываем, откуда будет загружаться изображение.

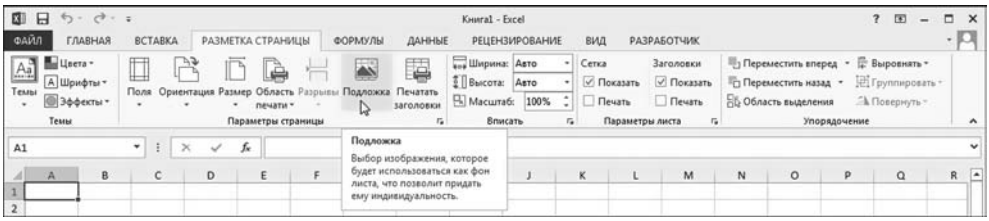


Рис. 4.35
Добавление фона (подложки) для рабочего документа (рабочего листа)

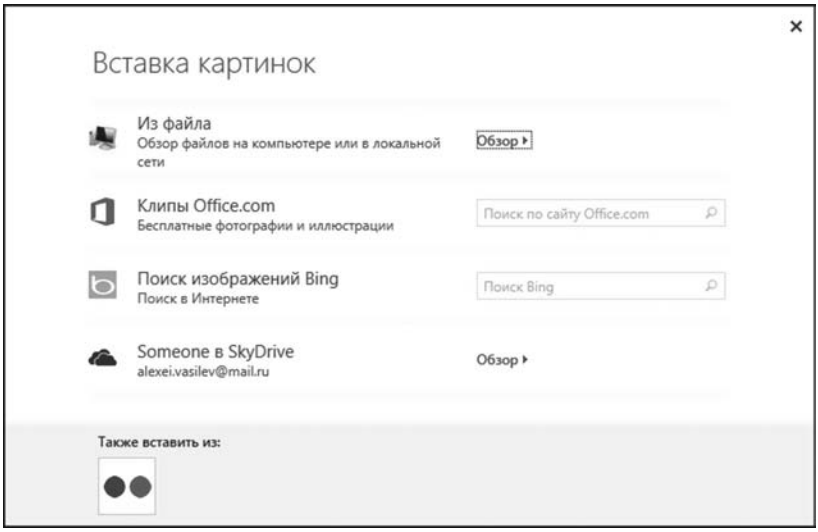


Рис. 4.36
Диалоговое окно **Вставка картинок**
для определения места загрузки изображения для подложки

Если мы решаем загружать изображение из файла на компьютере, щелкаем в окне **Вставка картинок** пиктограмму **Из файла**, в результате чего открывается еще одно диалоговое окно для выбора графического файла (см. рис. 4.37).

Сделав выбор, получаем чудесный рабочий документ, в подложке которого теперь чудесный фоновый рисунок (см. рис. 4.38).

При этом пиктограмма **Подложка** незаметно превращается в пиктограмму **Удалить фон** (рис. 4.38). Несложно догадаться, что щелчок на этой пиктограмме приведет к удалению фонового рисунка.

На заметку

Фоновый рисунок не следует путать с графическим объектом-рисунком, добавляемым в рабочий документ. Это далеко не одно и то же.

Ряд полезных настроек выполняется в окне **Параметры Excel**. На рисунке 4.39 это окно раскрыто в разделе **Дополнительно**.

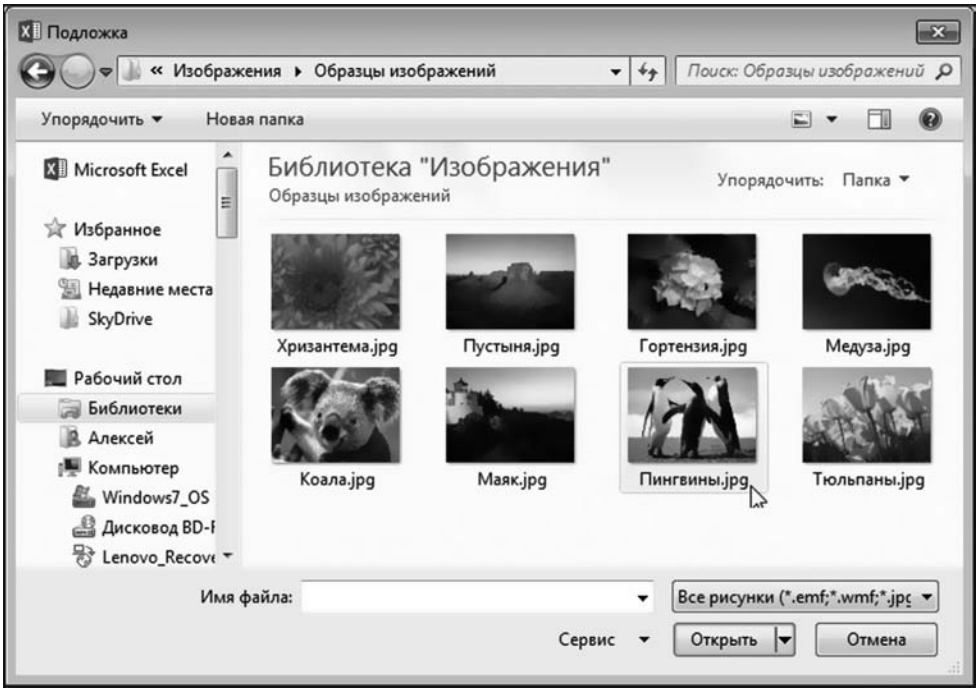


Рис. 4.37
Выбор изображения для фона (подложки) рабочего листа

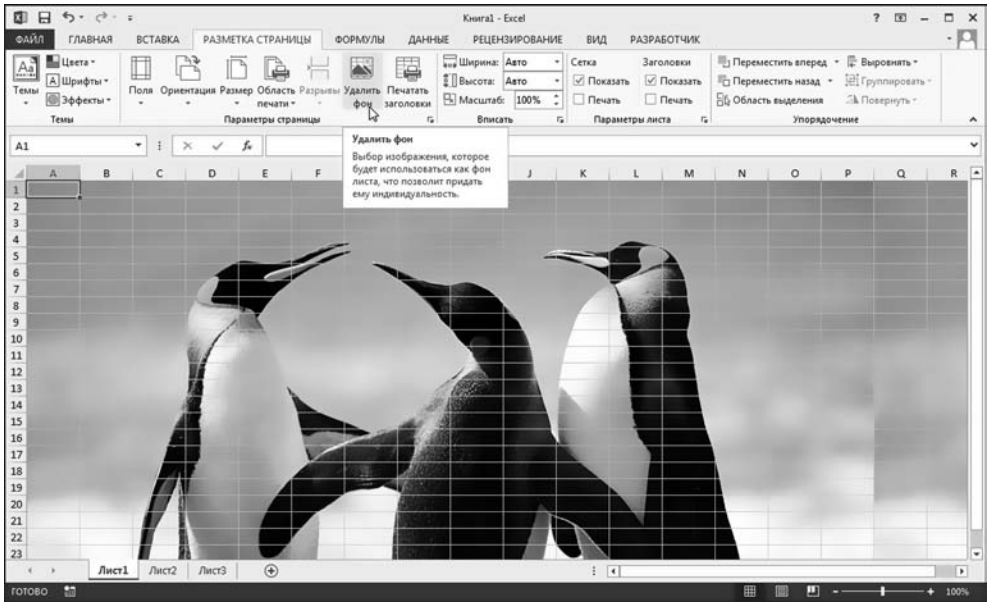


Рис. 4.38
Рабочий лист с фоновым изображением

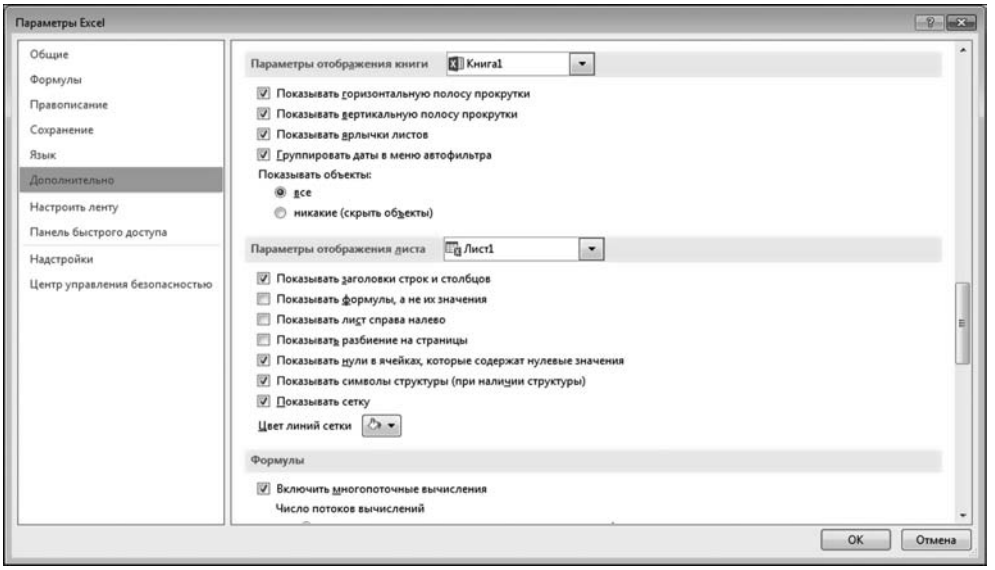


Рис. 4.39
Окно настроек приложения **Параметры Excel** открыто в разделе **Дополнительно**

Здесь содержится множество утилит, позволяющих выполнять настройки внешнего вида рабочего окна приложения. В частности, можем поупражняться в отображении/скрытии сетки в рабочей области, заголовок строк/столбцов и т. д. Полезной в некоторых случаях может быть опция-палитра выбора цвета линий сетки **Цвет линий сетки** (рис. 4.39).

На заметку

Эффект от заливки цветом ячеек рабочего листа и изменения цвета линий сетки может быть поистине поразительным.

Повлиять на общий способ отображения рабочих документов (в первую очередь, на тип и размер шрифтов, используемых для отображения данных в рабочей области и подписей заголовков строк и столбцов) можно с помощью пиктограмм группы **Темы** на вкладке **Разметка страницы**.

СОЗДАНИЕ ПРИМЕЧАНИЙ

— А если я Вам за такие слова всю амбицию разобью?
— Попробуйте!

Из к/ф «Безумный день инженера Баркасова»

Примечание — это как стикер с напоминанием, который мы клеим на холодильник. Но поскольку разработчиками Excel холодильник не предусмотрен, то примечание «клеится» на ячейку (прикрепляется к ячейке). Как и в случае с мини-напоминаниями, примечание должно быть кратким и в то же время информативным.

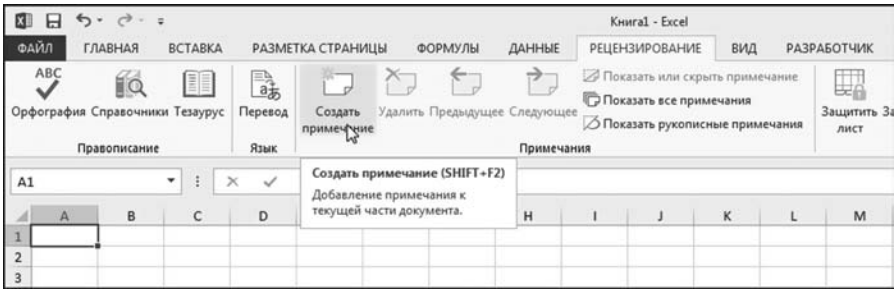


Рис. 4.40
Создание примечания с помощью пиктограммы **Создать примечание** на вкладке **Рецензирование**

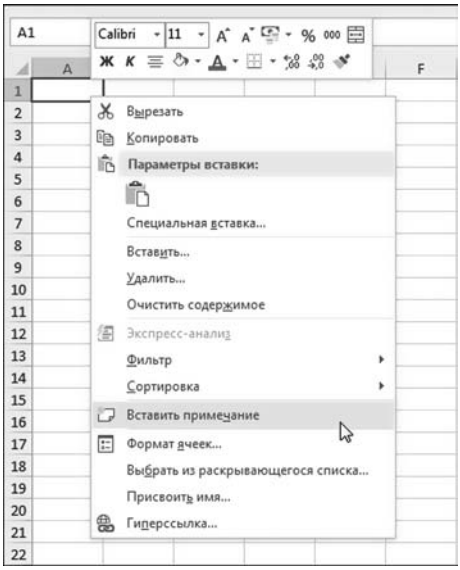


Рис. 4.41
Вставка примечания с помощью команды контекстного меню

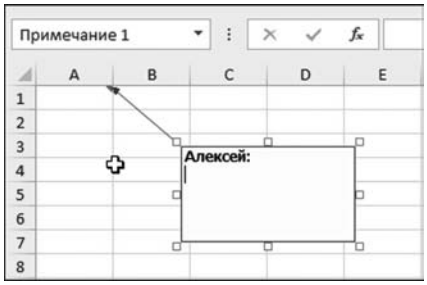


Рис. 4.42
Вставка примечания: необходимо ввести текст примечания

Создать примечание достаточно просто. Выделяем ячейку, которую собираемся «осчастливить» примечанием, и щелкаем «правильную» пиктограмму (или выбираем «правильную» команду из контекстного меню). «Правильной» является пиктограмма **Создать примечание** в группе **Примечание** на вкладке **Рецензирование** (рис. 4.40).

Не менее «правильная» и команда **Вставить примечание** в контекстном меню (рис. 4.41).

В любом случае в документ (в ту ячейку, что была выделена при добавлении примечания) вставляется примечание, которое по умолчанию имеет вид прямоугольной области с однородным фоном, текстом примечания (этот текст должен ввести пользователь) и стрелкой, указывающей на ячейку, к которой это примечание прикреплено (рис. 4.42).

Сама ячейка при этом получает в качестве «приза» красную отметку в левом верхнем углу (индикатор примечания). Индикатор примечания позволяет, окинув взглядом область рабочего документа, сразу выявить ячейки с примечаниями.

На заметку

Индикатор примечания отображается только в том случае, если приложение

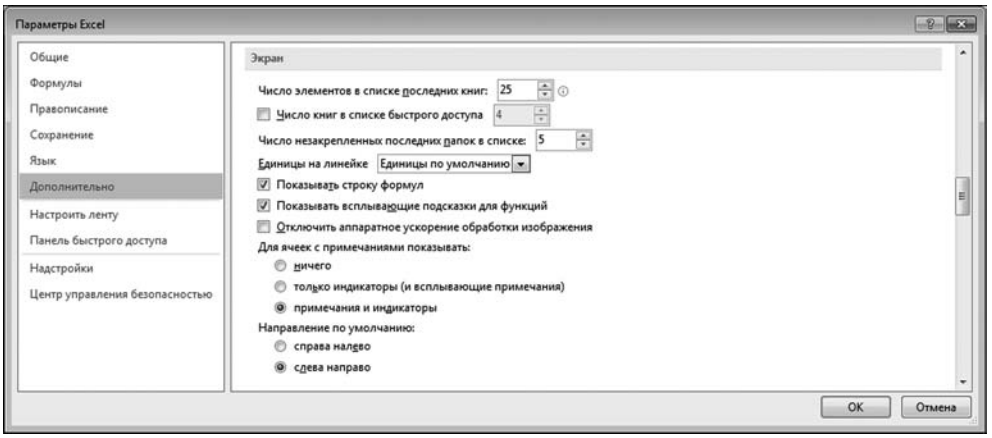


Рис. 4.43
Режим отображения примечаний и индикаторов примечаний

находится в режиме отображения индикаторов. По умолчанию используется именно такой режим. Но при желании в настройки отображения примечаний можно внести изменения. На рисунке 4.43 показано окно настроек приложения **Параметры Excel**, открытое в разделе **Дополнительно**.

Нас интересует секция **Экран** в той ее части, где размещен переключатель **Для ячеек с примечаниями показывать** на три положения. Если переключатель установлен в положение **ничего**, то в рабочем документе не будут отображаться не только индикаторы примечаний, но и сами примечания. Положение переключателя **только индикаторы (и всплывающие примечания)** означает, что индикаторы примечаний будут отображаться всегда, а примечания — при наведении курсора мыши на соответствующую ячейку. Наконец, если переключатель установлен в положение **примечания и индикаторы**, в документе отображаются и индикаторы, и примечания. Последний режим, как отмечалось, является режимом по умолчанию.

Например, если документ находится в режиме отображения индикаторов и всплывающих примечаний, то примечание отображается только при наведении курсора мыши над областью ячейки с примечанием (рис. 4.44).

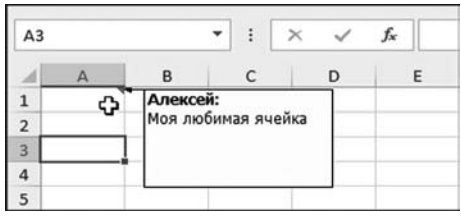


Рис. 4.44
Примечание отображается при наведении курсора на ячейку с этим примечанием

Для переключения режимов не обязательно прибегать к помощи окна настроек **Параметры Excel**. Полезными будут пиктограммы **Показать или скрыть примечание** (активна, если выделена ячейка с примечанием — скрытым или отображаемым) и **Показать все примечания**. Последняя пиктограмма является переключателем (т. е. может быть в двух состояниях — нажатом и не нажатом). С ее помощью легко переходим в режим, при котором отображаются все примечания (см. рис. 4.45).

Для удаления примечания можем воспользоваться пиктограммой **Удалить** в группе **Примечания** вкладки **Рецензирование** (рис. 4.46).

При этом должна быть выделена ячейка с примечанием. Наконец, перейти в режим настройки формата примечания можно так: выделяем примечание и в контекстном меню выбираем команду **Формат примечания** (рис. 4.47).

Открывается диалоговое окно **Формат примечания**, в котором можно настроить практически все (рис. 4.48).

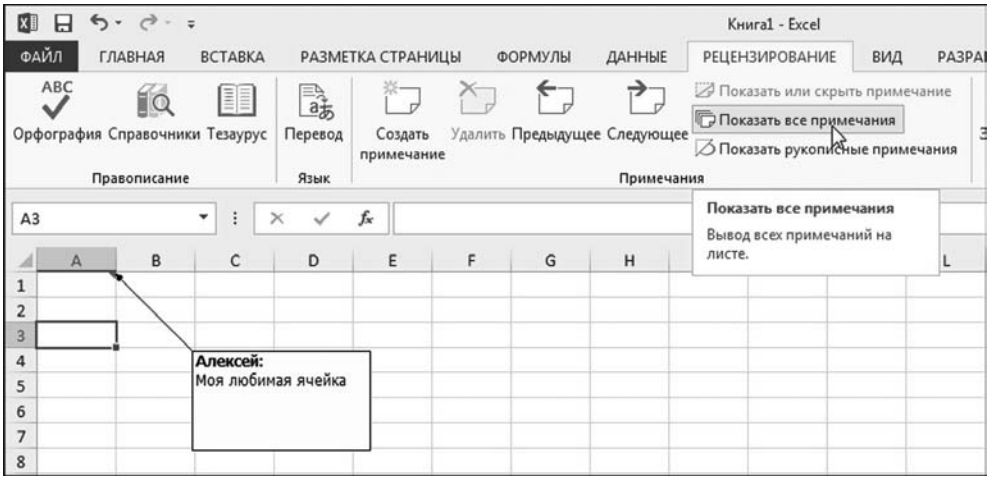


Рис. 4.45
Переход в режим постоянного отображения примечаний

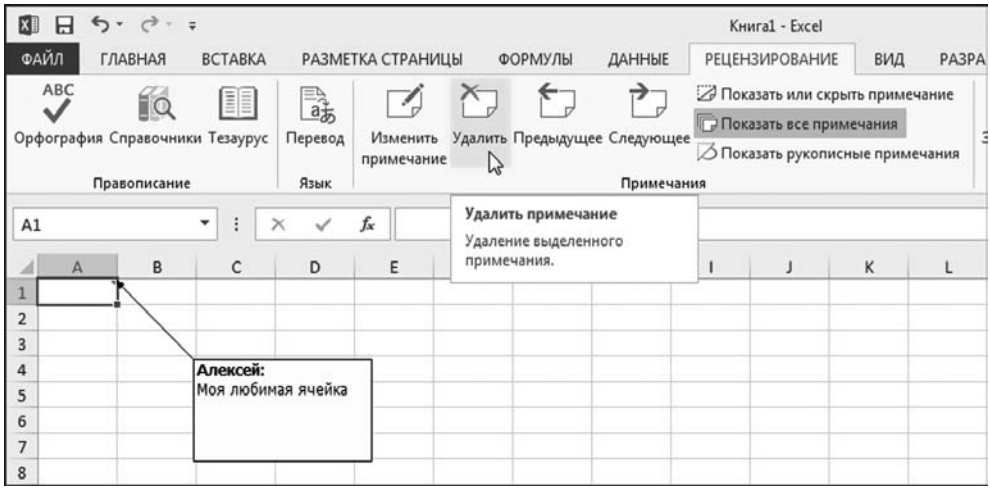


Рис. 4.46
Удаление примечания с помощью пиктограммы **Удалить** в группе **Примечания** вкладки **Рецензирование**

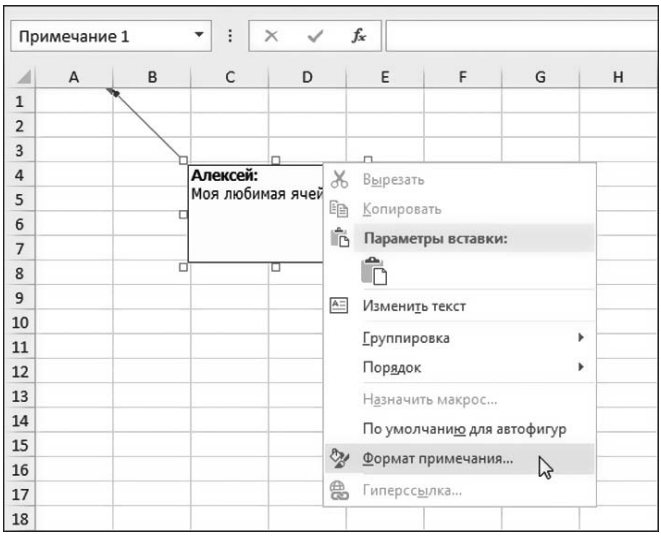


Рис. 4.47
Переход в режим редактирования и настройки параметров примечания

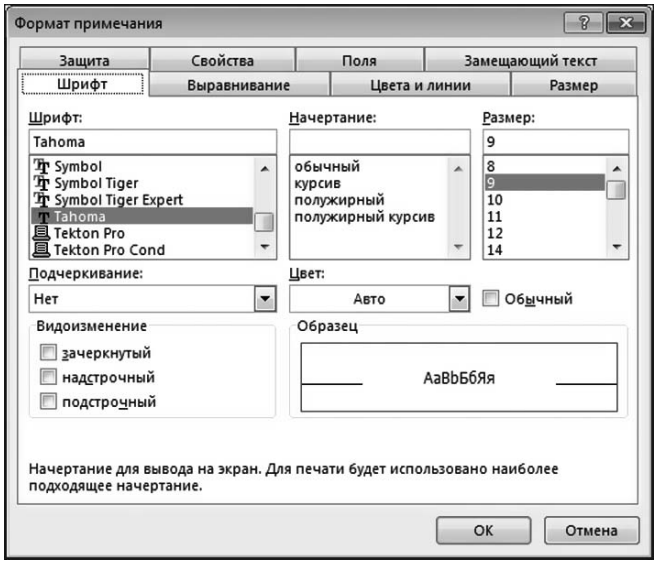


Рис. 4.48
Диалоговое окно **Формат примечания** для определения параметров примечания

На заметку

На вкладке **Рецензирование** в группе **Примечания** есть и другие пиктограммы, которые позволяют переходить в режим редактирования текста примечания или перебирать примечания в документе. Не стоит также пренебрегать и командами контекстного меню.

СРЕДСТВА ЗАЩИТЫ

— Маргадон, почему открыта дверь?
— Эскьюз ми, Магистр!
— Что «эскьюз ми»?
— Варварский обычай — ключи раздают, а замков нет!

Из к/ф «Формула любви»

В наш век всеобщей компьютерной грамотности и повального увлечения сбором информации защита документации становится как нельзя более актуальной. Разработчики Excel предусмотрительно предоставили нам такую возможность. Этой цели служит группа **Изменения** на вкладке **Рецензирование**. Интрига завязывается при щелчке на пиктограмме **Защитить лист** (рис. 4.49).

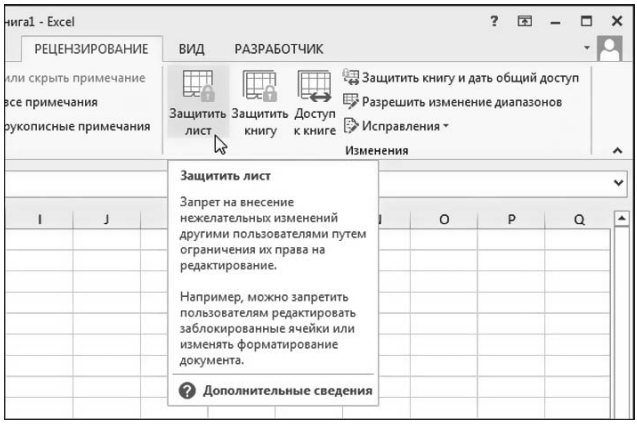


Рис. 4.49
Переход в режим защиты рабочего листа

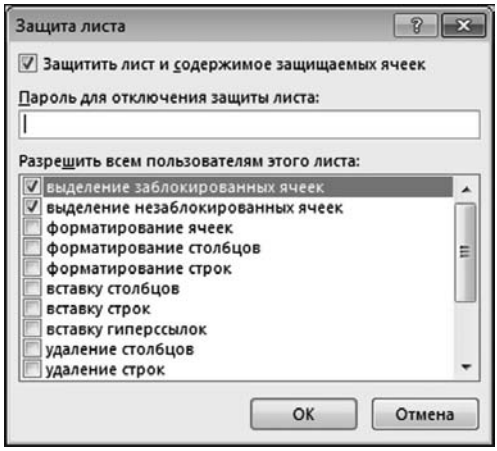


Рис. 4.50
Окно настройки параметров защиты рабочего листа

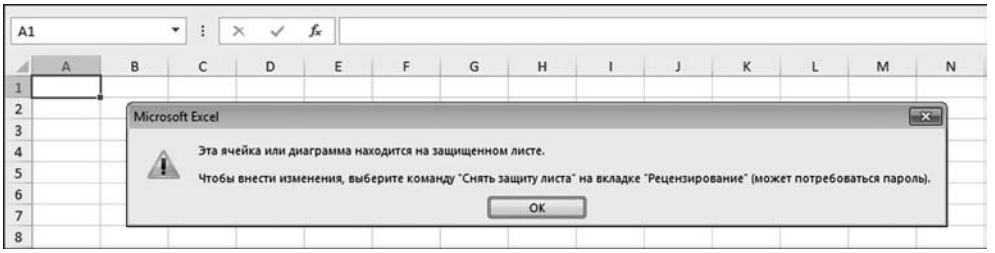


Рис. 4.51
Неудачная попытка изменить значение защищенной ячейки

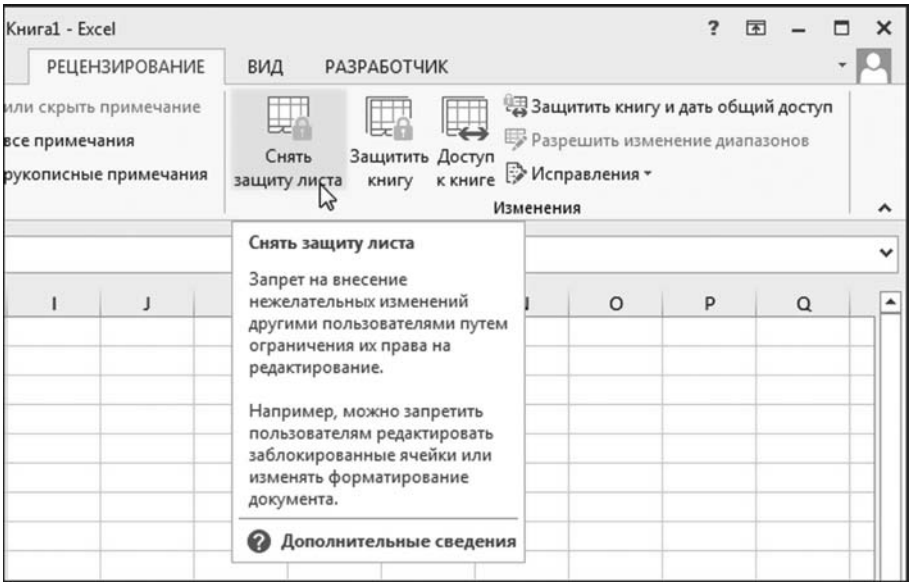


Рис. 4.52
Отмена режима защиты рабочего листа

Этот шаг означает попытку перейти в режим защиты рабочего листа. В результате открывается диалоговое окно **Защита листа** (рис. 4.50), в котором задаются параметры режима защиты документа.

В окне можно задать пароль на случай, если кто-то захочет отменить режим защиты рабочего листа. Там также есть внушительный список опций, которые позволяют определить разрешенные действия с защищенными ячейками. В данном случае мы разрешили выделение защищенных и незащищенных ячеек. В результате ячейки документа можно выделять, но нельзя изменить их значение. На рисунке 4.51 показано, что произойдет, если попытаться изменить значение защищенной ячейки.

Проще говоря, при попытке изменить значение ячейки, ситуация предается огласке с выводом сообщения о том, что значение ячейки изменить нельзя.

Если документ защищен, пиктограмма **Защитить лист** меняется на пиктограмму **Снять защиту листа** (рис. 4.52). Этой пиктограммой можем воспользоваться для отмены режима защиты рабочего листа.

На заметку

Дополняют картину прочие пиктограммы вкладки **Изменения**, с помощью которых процесс защиты рабочего документа становится легким и понятным.

СОЗДАНИЕ ГИПЕРССЫЛОК

— Интурист хорошо говорит.
– А что он говорит? Конкретно, что?
— А пес его знает! Феденька, надо бы переводчика!

Из к/ф «Иван Васильевич меняет профессию»

Обычный документ Excel может иметь достаточно сложную функциональную структуру в том смысле, что бывает непросто понять, что, как и в какой последовательности вычисляется. В некотором смысле гиперссылки могут усугубить ситуацию. Вместе с тем, гиперссылки — это гибкий и красивый элемент рабочего документа. Именно поэтому их все так любят.

С прикладной точки зрения гиперссылки устанавливают связи между разными документами или частями одного документа. Щелчок на гиперссылке позволяет выполнить быстрый переход к другому документу или, если мы говорим об Excel, к определенной ячейке в документе.

Создавать гиперссылки просто, в чем мы сейчас и убедимся. Сделаем это с помощью небольшого примера. А именно, выделяем ячейку A1 и затем щелкаем пиктограмму **Гиперссылка** в группе **Ссылки** на вкладке **Вставка** (рис. 4.53).

Можно пойти и другим путем, призвав на помощь контекстное меню. На рисунке 4.54 проиллюстрирован процесс создания гиперссылки в ячейке A1 с использованием команды контекстного меню **Гиперссылка**.

Так или иначе, откроется диалоговое окно **Вставка гиперссылки**, в котором задаются все основные параметры гиперссылки. Это диалоговое окно представлено на рисунке 4.55.

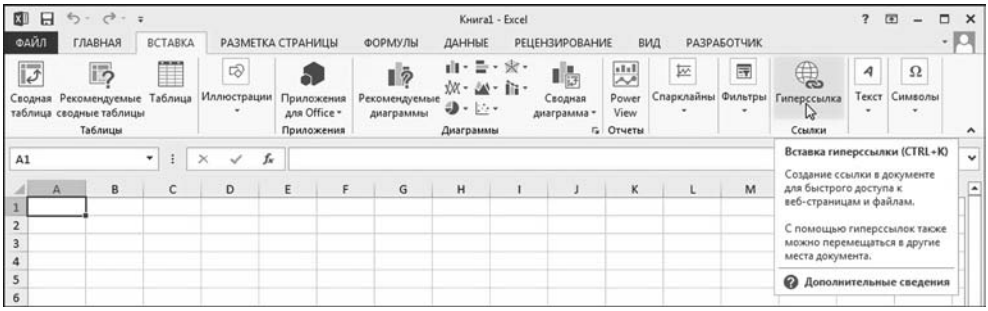


Рис. 4.53
Создание гиперссылки с помощью пиктограммы **Гиперссылка**

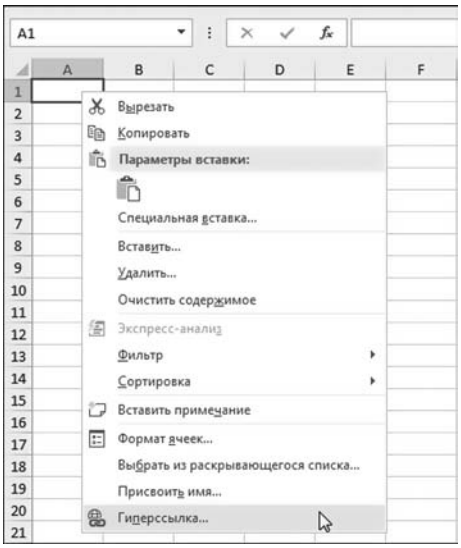


Рис. 4.54
Создание гиперссылки с помощью команды контекстного меню

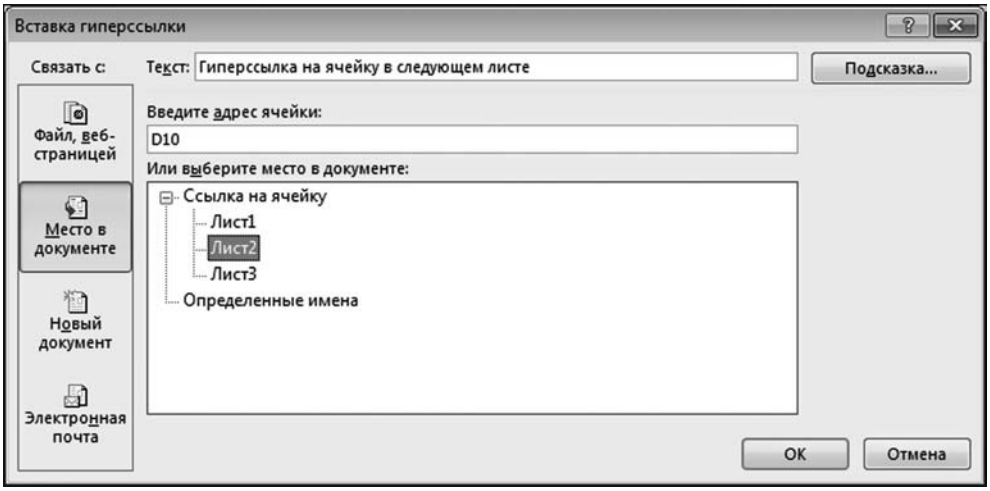


Рис. 4.55
Создание гиперссылки в окне **Вставка гиперссылки**

Окно **Вставка гиперссылки** содержит достаточно большое количество функциональных элементов, позволяющих определить тип и свойства создаваемой гиперссылки. В поле **Текст** указывается непосредственно текст гиперссылки. Мы вводим в это поле текст Гиперссылка на ячейку в следующем листе. Группа **Связать с** содержит четыре пиктограммы-переключателя. Вид центральной части диалогового окна **Вставка гиперссылки** изменяется в зависимости от того, какая пиктограмма нажата. Здесь мы создаем

гиперссылку на ячейку в том же документе, что и сама гиперссылка. Поэтому у нас нажата кнопка-пиктограмма **Место в документе** (рис. 4.55). В этом случае в поле **Введите адрес ячейки** указываем адрес ячейки (ячейка D10), на который выполняется гиперссылка. Мы создаем гиперссылку на ячейку D10 во втором листе Лист2 (гиперссылка находится в первом листе Лист1). Лист выбираем в области **Или выберите место в документе**. В принципе, гиперссылка готова.

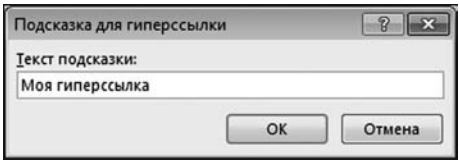


Рис. 4.56
Окно **Подсказка для гиперссылки** для ввода текста подсказки по гиперссылке

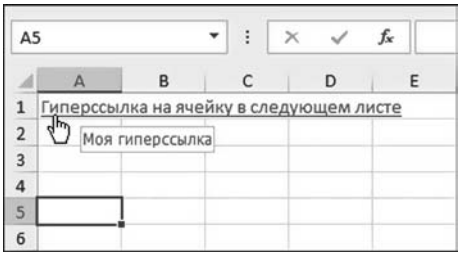


Рис. 4.57
Гиперссылка в рабочем документе (в ячейке A1)

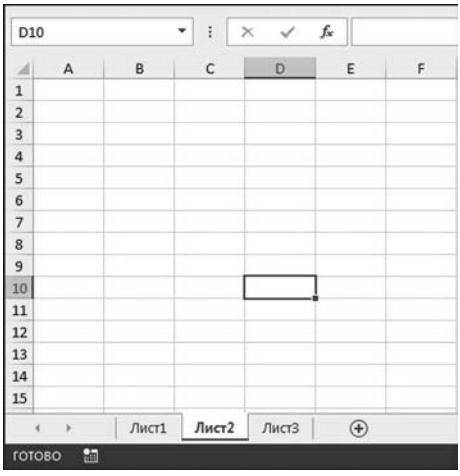


Рис. 4.58
Выполнен переход по гиперссылке к ячейке D10 листа Лист2

На заметку

Для гиперссылки можно создать подсказку. Подсказка отображается при наведении курсора мыши на гиперссылку. Чтобы ввести текст подсказки для гиперссылки следует щелкнуть на кнопке **Подсказка** в окне **Вставка гиперссылки** (рис. 4.55). В результате открывается диалоговое окно **Подсказка для гиперссылки**, представленное на рисунке 4.56.
В поле **Текст подсказки** вводится текст для подсказки по гиперссылке. Мы в данном случае для подсказки используем фразу Моя гиперссылка.

На рисунке 4.57 показано, как будет выглядеть созданная нами гиперссылка в рабочем документе.

Это специально отформатированный (синий с подчеркиванием) текст в ячейке A1, при наведении на который курсор мыши принимает классический вид ладони, а также появляется контекстная подсказка с уже знакомым нам текстом. Щелчок на гиперссылке приводит к тому, что осуществляется переход к ячейке D10 в листе Лист2 (рис. 4.58).

На заметку

Помимо гиперссылок на ячейки в том же документе, можно создавать гиперссылки на диапазоны ячеек, на новый или существующий документ, или, например, на страницу в Интернете.

В уже созданную гиперссылку можно вносить изменения. Для этого выделяем ячейку с гиперссылкой и в контекстном меню выбираем команду **Изменить гиперссылку** (рис. 4.59).

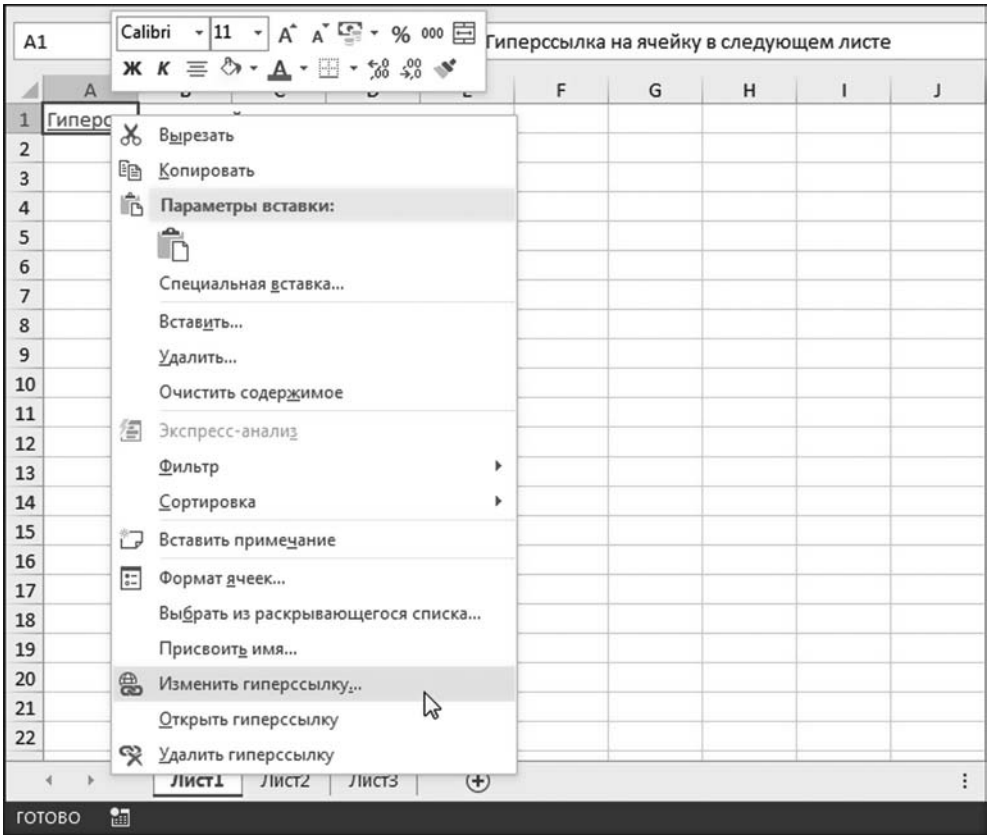


Рис. 4.59
Изменение гиперссылки

На заметку

В том же контекстном меню, кроме того, можно увидеть команду **Удалить гиперссылку**, с помощью которой, как несложно догадаться, гиперссылка удаляется из документа. При удалении гиперссылки текстовое значение гиперссылки остается значением ячейки.

В результате откроется диалоговое окно, с помощью которого вносятся изменения в настройки гиперссылки.

ЧАСТЬ ВТОРАЯ

ВЫЧИСЛЕНИЯ И ПРЕДСТАВЛЕНИЕ РЕЗУЛЬТАТОВ

ГЛАВА 5

МЕТОДЫ ВЫЧИСЛЕНИЙ И ОБРАБОТКА ДАННЫХ

Я делаю по-другому.
В моем чердаке только необходимые мне инструменты.
Их много, но они в идеальном порядке и всегда под рукой.
А лишнего хлама мне не нужно!

Из к/ф «Приключения Шерлока Холмса и доктора Ватсона»

Наиглавнейшее назначение приложения Excel — обработка (в широком смысле этого слова) всевозможных данных. Это могут быть как данные непосредственно в ячейках рабочего документа Excel, так и внешние данные, к которым выполняется запрос. Здесь, в этой главе, мы рассмотрим некоторые утилиты приложения, а также приемы/режимы работы, которые позволяют наиболее эффективно проводить вычисления и на разном уровне манипулировать данными.

На заметку

Если до этого мы основное внимание уделяли тому, *как* данные появляются в ячейках рабочего документа, то сейчас мы плавно переходим к вопросу о том, *зачем* они там появляются, и что потом с этими данными *можно* делать. Что с данными *нужно* делать — это вопрос несколько иного рода, и ответ на него зависит во многом от потребностей пользователя и специфики задач, которые перед ним стоят.

АВТОМАТИЧЕСКОЕ ЗАПОЛНЕНИЕ ЯЧЕЕК

— Есть опасение, что люди будут плохо усваивать
этот справочный материал.
— Почему Вы так думаете?
— Слишком беглое знакомство получается!

Из к/ф «Старый знакомый»

Нередко заполнение диапазонов ячеек выполняется в автоматическом режиме. В таких случаях говорят об *автоматическом заполнении ячеек*. К автоматическому заполнению диапазонов ячеек прибегают в случаях, когда вводимые в ячейки диапазона значения образуют очевидную последовательность или могут быть получены путем последовательного копирования формул (как правило, с относительными ссылками) из одной ячейки (диапазона) в другую ячейку (диапазон). При таком подходе заполняется одна или несколько ячеек диапазона, после чего на основе этих ячеек заполняются прочие ячейки диапазона. С формальной точки зрения автоматическое заполнение выполняется дос-

таточно просто: выделяются базовые ячейки, на основе которых заполняются все остальные, курсор мыши наводится на маркер автоматического заполнения, маркер фиксируется мышью (удерживается нажатой левая кнопка мыши), и рамка выделения растягивается на весь диапазон. В результате происходит автоматическое заполнение всего диапазона. Если автоматическое заполнение осуществляется на основе базовой ячейки с формулой, то предугадать результат достаточно просто: формула из базовой ячейки последовательно копируется в прочие ячейки диапазона, хотя и в этом случае возможны нетривиальные ситуации. При автоматическом заполнении диапазона ячеек на основе базовых ячеек с числами или текстом интрига сохраняется дольше. По большому счету, вопрос сводится к тому, сможет или не сможет приложение Excel «угадать» логику заполнения ячеек диапазона. Нередко приходится вмешиваться в процесс и корректировать его.

Рассмотрим наиболее простые случаи, к которым, несомненно, относится заполнение ячеек числовыми значениями. На рисунке 5.1 выделен диапазон ячеек A1:A2. В ячейку A1 введено числовое значение 1, а в ячейку A2 введено числовое значение 3.

Захватываем маркер заполнения и растягиваем рамку выделения так, чтобы она охватывала диапазон ячеек A1:A10, как показано на рисунке 5.2.

В результате в ячейки диапазона A3:A10 вносятся числовые значения, которые образуют арифметическую прогрессию (рис. 5.3).

Каждое следующее значение получается увеличением предыдущего на два. Причина в том, что при автоматическом заполнении по умолчанию предполагается, что значения ячеек образуют арифметическую прогрессию, и начальные ячейки содержат первые члены этой

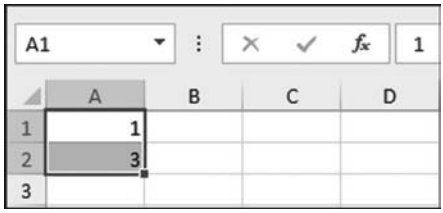


Рис. 5.1
Выделен диапазон ячеек A1:A2 с числовыми значениями

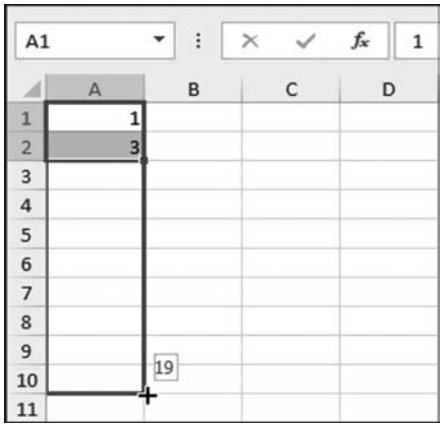


Рис. 5.2
Выделение диапазона A1:A10 для автоматического заполнения ячеек

Рис. 5.3
Результат автоматического заполнения ячеек A1:A10

прогрессии. В данном случае первое значение в прогрессии равно 1, а второе равно 3. Поэтому шаг составляет 3-1=2. Важно, что ячейки заполняются числами, не формулами. Другими словами, если после автоматического заполнения внести изменения в ячейку A1, то значения прочих ячеек диапазона A1:A10 не изменятся.

Далее представлена похожая ситуация, но только в качестве начальных ячеек выделен диапазон ячеек A1:B2. Предварительно в эти ячейки введены числовые значения (рис. 5.4).

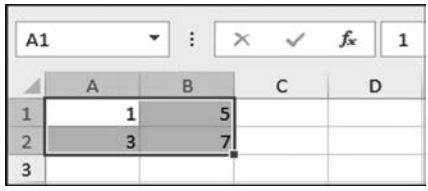


Рис. 5.4
Выделен диапазон ячеек A1:B2 с числовыми значениями

Рис. 5.5
Выделение диапазона A1:G2 для автоматического заполнения ячеек

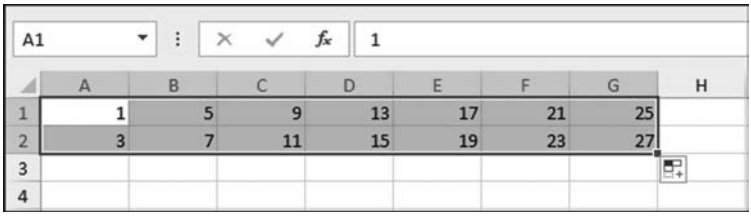


Рис. 5.6
Результат автоматического заполнения ячеек A1:G2

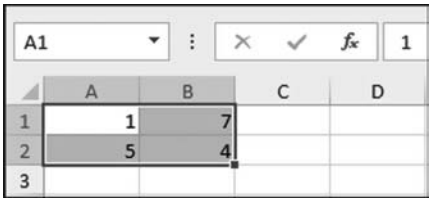


Рис. 5.7
Начальные ячейки для автоматического заполнения

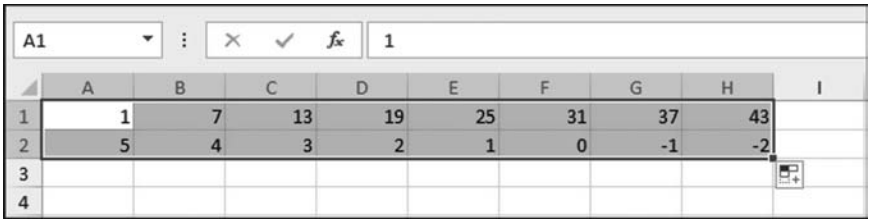


Рис. 5.8
Заполнение диапазона A1:H2

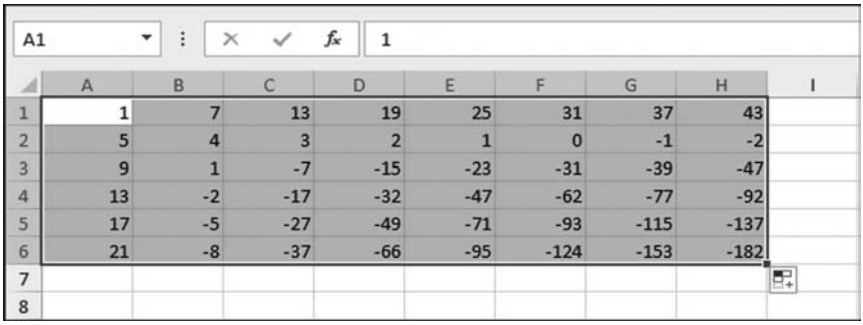


Рис. 5.9
Заполнение диапазона A1:H6

Захватываем маркер выделения и выделяем диапазон A1:G2 (рис. 5.5). Результат показан на рисунке 5.6.

В этом случае получаем две арифметические прогрессии — в каждой строке своя последовательность чисел. Если необходимо заполнить ячейки сразу по столбцам и строкам, то такое заполнение выполняется в два этапа, последовательно по строкам и по столбцам (или по столбцам и по строкам). Например, на основе значений ячеек диапазона A1:B2 заполним ячейки диапазона A1:H6. Выделенные ячейки диапазона A1:B2 с начальными значениями показаны на рисунке 5.7.

Сначала заполняем, например, диапазон ячеек A1:H2, как показано на рисунке 5.8. После этого, на основе ячеек A1:H2 заполняем диапазон ячеек A1:H6 (рис. 5.9).

Схема заполнения ячеек следующая: сначала строится арифметическая прогрессия для каждой из двух строк в диапазоне A1:H2, а затем арифметическая прогрессия строится для каждого из столбцов диапазона ячеек A1:H6. Если сначала заполнить диапазон ячеек A1:B6, а затем на его основе диапазон A1:H6, получим точно такой же результат.

Результат автоматического заполнения ячеек может быть достаточно неожиданным. Рассмотрим рисунок 5.10. Здесь выделен диапазон ячеек A1:A3 со значениями 1, 4 и 9 в ячейках A1, A2 и A3 соответственно.

Методом автоматического заполнения заполняем ячейки A1:A6. Результат показан на рисунке 5.11. Совершенно очевидно, что числа в ячейках A1:A3 — это квадраты натуральных чисел 1, 2 и 3 соответственно. Можно

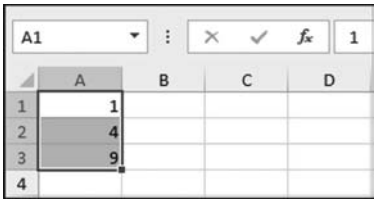


Рис. 5.10
В качестве начального выбран диапазон-столбик из трех ячеек

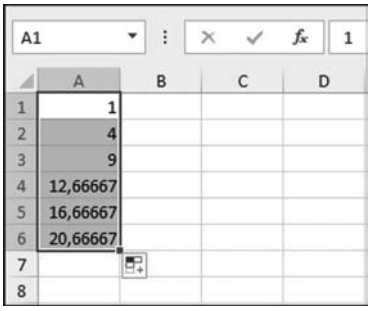


Рис. 5.11
Результат автоматического заполнения диапазона на основе диапазона-столбика из трех ячеек

было бы ожидать, что в ячейках A4:A6 появятся квадраты чисел 4, 5 и 6 соответственно, т. е. числа 16, 25 и 36. Но мы их там не увидим. Вместо этого появляются очень несимпатичные и совершенно нецелые числа. Объяснение простое. Состоит в том, что по умолчанию на основе данных в ячейках методом наименьших квадратов строится линейная регрессионная модель. Для данного конкретного случая числа с порядковым номером $n > 3$ вычисляются по формуле $4n - 10/3$. В ячейке A4 число с номером 4, а его значение равно (приближенно) 12,66667. Для ячейки A5 значение вычисляется по приведенной выше формуле для значения $n = 5$, и получаем 16,66667. Значение в следующей ячейке A6 будет 20,66667.

В Excel методом автоматического заполнения можно создавать не только линейную регрессию или арифметическую прогрессию. Приложение «узнает», например, и геометрическую прогрессию. Проиллюстрируем это. Рассмотрим документ на рисунке 5.12.

В этом документе в ячейки A1:A3 введены значения 1, 3 и 9, а выделен диапазон ячеек A1:A6. Числа 1, 3 и 9 представляют собой первые три члена геометрической прогрессии с начальным значением 1 и знаменателем 3. Ячейки A1:A6 заполним в соответствии с правилом формирования элементов геометрической прогрессии. Для этого при выделенном диапазоне A1:A6 на вкладке **Главная** в группе **Редактирование** щелкаем пиктограмму **Заполнить** (рис. 5.12), в результате чего открывается список, в котором выбираем команду **Прогрессия**. Откроется одноименное диалоговое окно, в котором выполняются настройки (рис. 5.13).

Диалоговое окно **Прогрессия** имеет несколько элементов управления. Переключатели группы **Расположение** устанавливаем в положение **по столбцам**. Переключатели группы **Тип** устанавливаем в положение **геометрическая**. Также устанавливаем флажок опции **Автоматическое определение шага** для того, чтобы параметры геометрической регрессии определялись автоматически. После подтверждения выполненных настроек (щелчок на кнопке **ОК**) получаем результат, как на рисунке 5.14.

Все вычислено правильно, диапазон ячеек заполнен.

На заметку

Обращаем внимание, что при заполнении ячеек с использованием настроек диалогового окна **Прогрессия** в общем случае есть возможность явно задавать шаг для формирования прогрессии, а также работать с датами — ситуация, которая довольно часто встречается на практике.

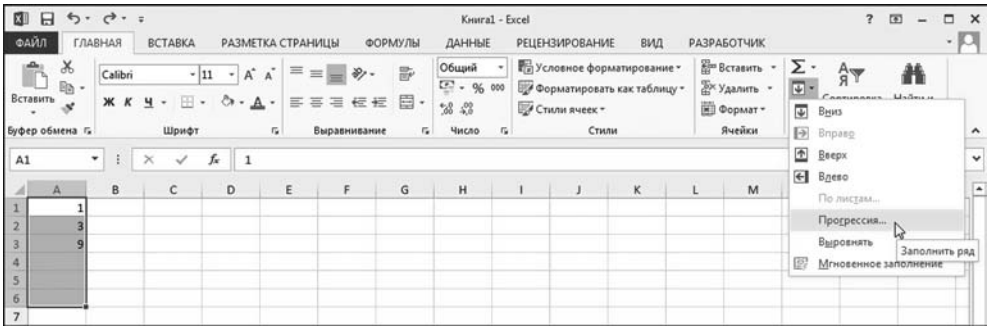


Рис. 5.12
Заполнение ячеек на основе геометрической прогрессии

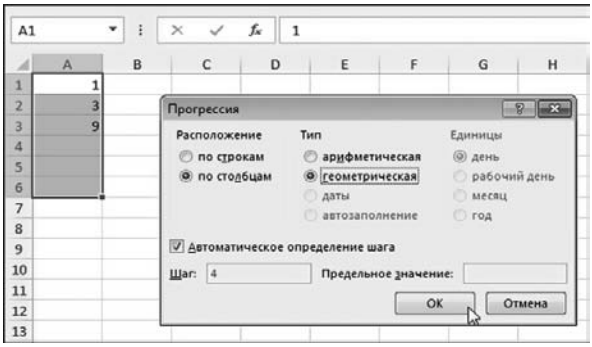


Рис. 5.13
Диалоговое окно **Прогрессия**
на фоне выделенного диапазона ячеек

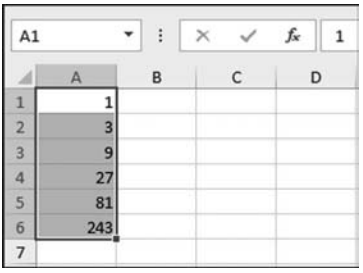


Рис. 5.14
Результат заполнения
диапазона ячеек

КОПИРОВАНИЕ ФОРМУЛ И ЗНАЧЕНИЙ

— Хотелось бы в общих чертах понять, что ему нужно.
— Да понять его, надежа-царь, не мудрено.

Из к/ф «Иван Васильевич меняет профессию»

Особенности копирования формул в Excel мы уже обсуждали. Здесь лишь кратко напомним основные моменты и сделаем ряд замечаний. Собственно, при копировании формул первоочередное значение имеет способ преобразования ссылок в копируемых формулах. Ссылки, в свою очередь (и мы это уже знаем), могут быть:

- абсолютными;
- относительными;
- смешанными.

На заметку

С формальной точки зрения абсолютная ссылка от относительной отличается наличием символов $ перед именем столбца и номером строки. Другими

словами, абсолютная ссылка содержит два символа $. Смешанная ссылка является абсолютной только по строке или только по столбцу. Смешанная ссылка содержит только один символ $.

Фактически, разница между относительными и абсолютными ссылками проявляется только при копировании формул. Здесь мы рассмотрим некоторые специальные вопросы, связанные с копированием формул, а если точнее, то содержимого ячеек с формулами.

Пикантность ситуации состоит в том, что если в ячейке значение вычисляется по формуле, то интерес может представлять и формула, и значение или что-то еще. Другими словами, очень может статься, что из ячейки с формулой нужно скопировать не формулу, а значение (числовое или текстовое), или загадочное *что-то еще*. Покажем, как это происходит, на простом примере. Обратимся к документу на рисунке 5.15.

В ячейки A1, B1 и C1 введены натуральные числа 1, 2 и 3 соответственно. В ячейку A2 введена формула =$A$1+3*B1. Для ячейки применено форматирование с выделением жирным шрифтом, горизонтальным выравниванием содержимого по центру ячейки, заливкой области ячейки черным цветом и применением белого цвета для шрифта. Ячейку A2 выделяем и копируем содержимое ячейки в буфер обмена с помощью пиктограммы группы **Буфер обмена** на вкладке **Главная** (рис. 5.15). После этого содержимое буфера обмена копируем в ячейку B2 (рис. 5.16).

Копирование выполняется с помощью пиктограммы-раскрывающегося списка. Это центральная пиктограмма в группе **Буфер обмена** и ее сложно не заметить. Если щелкнуть в нижней части пиктограммы (в области стрелки), раскроется меню из пиктограмм, представленное на рисунке 5.17.

Каждая из пиктограмм соответствует определенному режиму вставки содержимого буфера обмена. Среди этих режимов и тот, при котором в конечную ячейку копируется числовое значение из исходной ячейки. Назначение пиктограмм и соответствующие им режимы описаны в таблице 5.1.

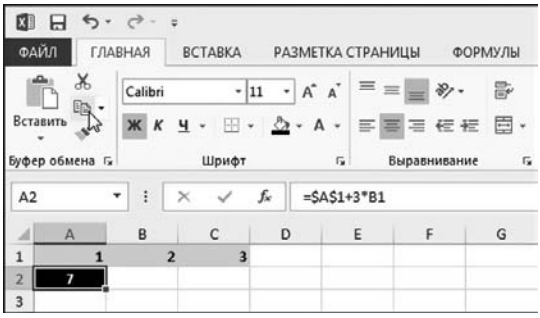


Рис. 5.15
Документ перед копированием значений ячеек

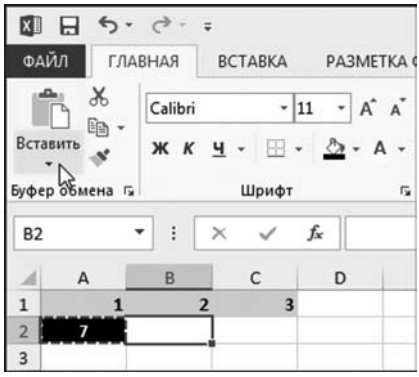


Рис. 5.16
Копирование содержимого буфера обмена

На заметку

У каждого рабочего листа есть так называемая графическая подложка. Именно на эту подложку добавляются графические изображения, диаграммы и некоторые другие объекты. Два режима вставки, при которых из буфера обмена в рабочий лист вставляется изображение, подразумевают вставку изображения именно на графическую подложку рабочего листа.

В раскрывающемся списке с пиктограммами в нижней части раскрывающегося окна есть команда **Специальная вставка**. После щелчка на этой команде открывается диалоговое окно, в котором выполняется настройка параметров вставки (см. рис. 5.18).

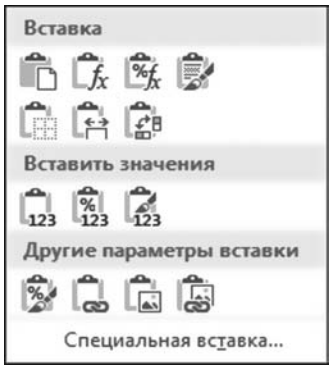


Рис. 5.17
Пиктограммы меню вставки содержимого буфера обмена

Таблица 5.1

Пиктограммы вставки содержимого буфера обмена

Пиктограмма	Режим вставки
	Используемый по умолчанию режим вставки. Вставляется содержимое, и применяется форматирование исходной ячейки. Если исходная ячейка содержит формулу — вставляется формула, если исходная ячейка содержит число — вставляется число, если содержит текст — вставляется текст. Такая вставка будет выполнена, если сразу щелкнуть на пиктограмме вставки содержимого буфера обмена
	Вставка в итоговую ячейку только формулы. При этом сохраняется форматирование итоговой ячейки
	Режим вставки формулы и применения числового формата исходной ячейки
	Вставка содержимого исходной ячейки с сохранением форматирования итоговой ячейки
	Режим вставки содержимого исходной ячейки без выделения границ ячейки
	Режим вставки, при котором сохраняется ширина исходной ячейки
	Режим применяется для копирования/вставки содержимого диапазона ячеек. Содержимое из буфера обмена вставляется с одновременным транспонированием
	Режим вставки только числового значения, даже если исходная ячейка содержит формулу
	Режим вставки числового значения с применением числового формата исходной ячейки
	Вставка числового значения с применением формата исходной ячейки
	К конечной ячейке применяется формат исходной ячейки. Вставка содержимого буфера обмена не происходит
	Вставка ссылки на ячейку, из которой было скопировано содержимое в буфер обмена. Фактически, вставляется формула с одной абсолютной ссылкой на исходную ячейку
	Режим вставки рисунка (изображения ячейки) — вставляется не в отдельную ячейку, а непосредственно в рабочий лист
	Вставка изображения (рисунка) исходной ячейки с одновременной ссылкой на эту ячейку, что обеспечивает автоматическое обновление изображения в итоговой ячейке при изменении значения в исходной ячейке. Вставка выполняется в рабочий лист

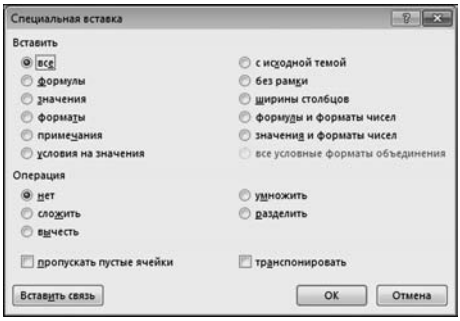


Рис. 5.18
Диалоговое окно выбора параметров специальной вставки

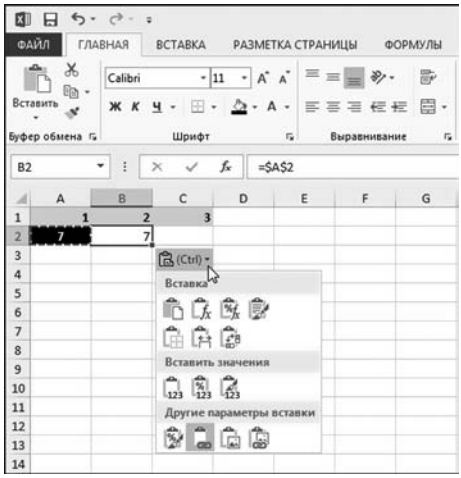


Рис. 5.19
Вставлена ссылка на ячейку и раскрыто контекстное меню параметров вставки

Окно **Специальная вставка** содержит ряд переключателей и опций, с помощью которых задаются основные параметры вставки содержимого буфера обмена. А именно, окно содержит переключатель **Вставить** с огромным (как для переключателя) количеством положений, также переключатель **Операция** на пять положений и две опции. Также в окне есть три кнопки — помимо банальных **ОК** и **Отмена** еще и кнопка **Вставить связь** для вставки связи на ячейку.

Положение переключателя **Вставить** определяет, какие данные вставляются в итоговую ячейку из буфера обмена. Названия положений этого переключателя достаточно красноречивы и говорят сами за себя. Что касается переключателя **Операция**, то он достаточно уникален и позволяет при вставке значения из буфера обмена одновременно выполнять (или не выполнять) один из четырех арифметических операторов: сложение, вычитание, умножение и деление. Это бинарные операции и для их выполнения необходимо два значения. Первое (первый операнд выражения) — это значение в той ячейке, в которую копируется значение. Второе (второй операнд) — значение в буфере обмена.

На заметку

Сценарий происходящего при вставке содержимого из буфера обмена (при вставке с помощью опций диалогового окна **Специальная вставка**) определяется положением обеих переключателей (**Вставка** и **Операция**), и эффект может быть удивительным, а иногда и странным. Например, в рассматриваемом примере в ячейку A2 введена формула =$A$1+3*B1. Если в ячейку B2 введено числовое значение 2 и в эту ячейку копируется (через буфер обмена) значение ячейки A2 с переключателем **Вставка** в положении **все** и переключателем **Операция** в положении **разделить**, то в результате в ячейке B2 будет значение =2/($A$1+3*C1). Объяснение достаточно простое. Исходная ячейка содержит формулу, поэтому результатом будет формула. Выражение в формуле получается делением текущего значения конечной ячейки на формулу, содержащуюся в исходной ячейке. При этом формула исходной ячейки преобразуется в соответствии с правилами преобразования абсолютных и относительных ссылок.

На рисунке 5.19 показан результат вставки содержимого буфера обмена с помощью кнопки **Вставить связь** диалогового окна **Специальная вставка**.

Сразу после вставки в правом нижнем углу от итоговой ячейки (диапазона ячеек) отображается пиктограмма раскрывающегося списка выбора режимов вставки. На рисунке 5.19 этот список раскрыт. С его помощью уже после вставки содержимого буфера обмена можно изменить режим вставки.

На заметку

В Excel ячейки (содержимое) можно не только копировать, но и перемещать. Действие это уникально как по внешним эффектам, так и по результату. Вкратце выглядит процедура перемещения ячеек следующим образом. Выделяем диапазон ячеек (с данными), наводим мышь на границу рамки выделения диапазона (курсор примет классический «стрелочный» вид с пиктограммой из четырех разнонаправленных стрелок) и, удерживая левую кнопку мыши нажатой, перемещаем ячейки по рабочему листу к пункту назначения. Если «пункт назначения» данных не содержит, все пройдет тихо и незаметно. Если данные есть, появится запрос, что с ними делать. Прелесть в том, что все ссылки (и на перемещаемый диапазон, и в перемещаемом диапазоне) меняются автоматически так, как надо — структура функциональных связей в документе не меняется.

Есть еще одно крайне немаловажное обстоятельство, которое следует иметь в виду при работе с формулами. Речь идет о так называемом режиме ссылок в формате R1C1. Чтобы перейти в соответствующий режим, на вкладке **Файл** выбираем пункт **Параметры Excel** и в открывшемся диалоговом окне настройки параметров приложения в разделе **Формулы** устанавливает флажок опции **Стиль ссылок R1C1** (рис. 5.20).

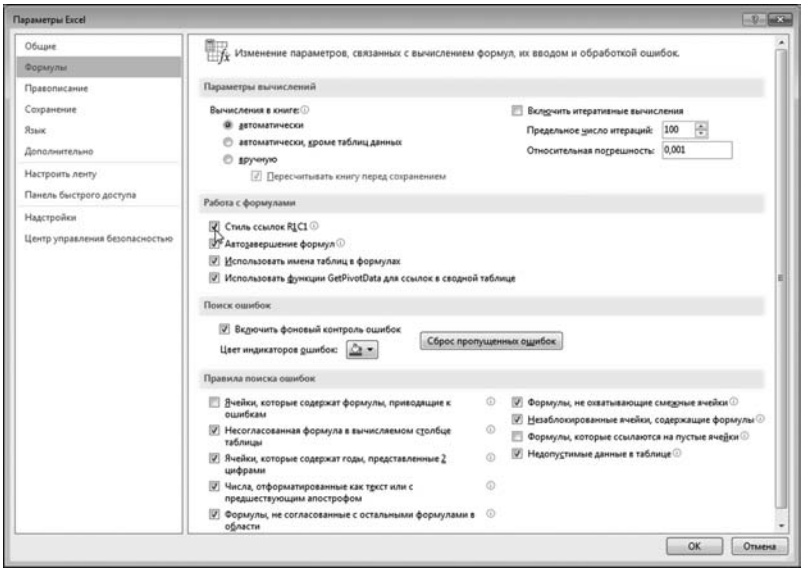


Рис. 5.20
Переход в режим ссылок **R1C1**

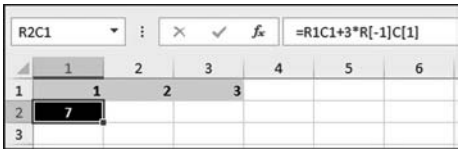


Рис. 5.21
Формулы со ссылками в формате R1C1

После этого адресация ячеек будет выполняться в несколько ином виде. А именно, для определения адреса ячейки указывается номер строки и номер столбца, на пересечении которых находится ячейка. Номер строки указывается после литеры R, а номер столбца указывается после литеры C. Например, ссылка R2C3 означает ячейку, находящуюся на пересечении 2-й строки и 3-го столбца, т. е. ячейку C2 в обычном представлении.

Ссылка в формате **R1C1** является абсолютной. Это собственно и так понятно, поскольку если заданы порядковый номер строки и столбца ячейки, то ее положение определено однозначно. Чтобы задать в формате **R1C1** относительную ссылку, индексы для строки и столбца указываются в квадратных скобках. При этом строка и столбец (их порядковый номер) отсчитываются от ячейки, в которую введена формула со ссылкой. Отрицательное значение индекса строки означает направление вверх, положительное — вниз. Отрицательное значение индекса столбца означает направление влево, а положительное — вправо. На рисунке 5.21 показан рассматривавшийся ранее документ, но теперь со ссылками в режиме **R1C1**.

Формула в ячейке A2, которая ранее выглядела как =$A$1+3*B1, теперь принимает вид =R1C1+3*R[-1]C[1]. Ссылка R1C1 означает ячейку, находящуюся в первой строке и первом столбце. Ссылка R[-1]C[1] означает ячейку, находящуюся на одну строку выше и на один столбец правее по отношению к текущей ячейке (т. е. ячейке с формулой, в которой содержится ссылка).

На заметку

Обратите внимание, что в режиме **R1C1** столбцы рабочего листа нумеруются цифрами, а не буквами, как до этого.

Особая прелесть ссылок в формате **R1C1** состоит в том, что при копировании ссылки (и абсолютные, и относительные) в этом формате не меняются. Чтобы понять, почему так происходит, достаточно вспомнить, что означают цифры в коде формата **R1C1**.

На заметку

Нулевой индекс в коде формата **R1C1** для относительной ссылки может не указываться. Квадратные скобки тоже не указываются. Например, ссылка R[0]C[2] может быть записана как RC[2].

Откровенно говоря, режим **R1C1** не очень популярен, несмотря на очевидные преимущества. Хотя в любом подходе есть свои недостатки.

На заметку

При переходе в режим **R1C1** все ссылки рабочих документов преобразуются к соответствующему формату автоматически. Если выйти из этого режима (отменить флажок опции **Стиль ссылок R1C1** в разделе **Формулы** окна **Параметры Excel**), все ссылки будут преобразованы к стандартному формату «строка-столбец».

ЧИСЛОВЫЕ ФОРМУЛЫ И ЦИКЛИЧЕСКИЕ ССЫЛКИ

Видел чудеса техники, но такого...

Из к/ф «Иван Васильевич меняет профессию»

Обычно формулы содержат ссылки на ячейки — обычно, но не всегда. В последнем случае говорят о числовых формулах. Хотя на первый взгляд может показаться, что смысла в использовании таких формул нет, все же иногда они бывают полезными. Например, мы заполняем диапазон ячеек последовательностью натуральных чисел. Если мы на основе этого диапазона станем заполнять (методом автоматического заполнения) и другие ячейки, последовательность чисел будет «продолжена». Рассмотрим небольшой пример. На рисунке 5.22 показан документ с двумя диапазонами ячеек: один заполнен числами, а другой заполнен числовыми формулами.

Ячейки A2, A3 и A4 заполнены соответственно числами 1, 2 и 3. Ячейки B2, B3 и B4 заполнены числовыми формулами =1, =2 и =3. После этого применяем процедуру автоматического заполнения диапазона ячеек A2:A7 на основе ячеек диапазона A2:A4 (рис. 5.23).

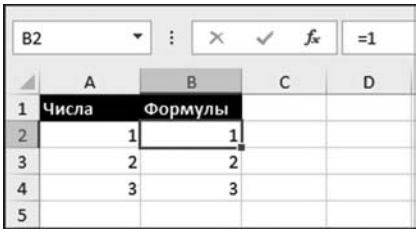


Рис. 5.22
Документ с числовыми формулами

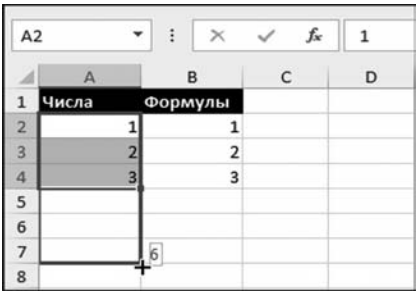


Рис. 5.23
Автоматическое заполнение числового диапазона

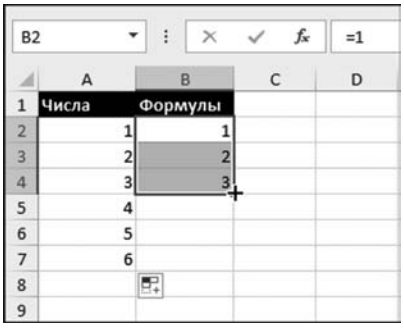


Рис. 5.24
Автоматические заполнение диапазона с числовыми формулами

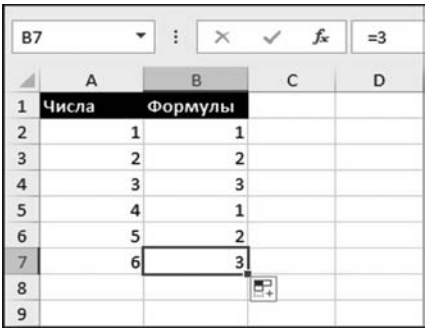


Рис. 5.25
Результат заполнения числового диапазона и диапазона с числовыми формулами

В данном случае ничего непредвиденного не происходит — диапазон ячеек A2:A7 в результате заполнен натуральными числами от 1 до 6. Этот эффект вполне ожидаем. Затем выполняем автоматическое заполнение диапазона ячеек B2:B7 на основе диапазона ячеек B2:B4 с числовыми формулами (см. рис. 5.24).

Результат этой процедуры представлен на рисунке 5.25.

Заполнение ячеек происходит циклически, в результате чего диапазон B2:B7 заполнен повторяющимися числовыми формулами. Почему так происходит? Ответ прост. Состоит он в том, что хотя в формулы входят только числа, это все же формулы, и обрабатываются они как формулы.

Эффектный механизм вычислений связан с использованием *циклических ссылок*. Циклическая ссылка — это ссылка в формуле на ячейку, которая содержит формулу (т. е. ссылка в ячейке на эту же ячейку). По умолчанию такая ситуация считается ошибочной, поскольку означает, что для вычисления значения ячейки по формуле с циклической ссылкой необходимо использовать значение этой ячейки. Круг замкнулся!

На заметку

Круг может замыкаться и не так очевидно. Циклическую ссылку можно получить, если использовать последовательность ссылок такую, что формула в ячейке ссылается на эту самую ячейку через цепочку других ссылок.

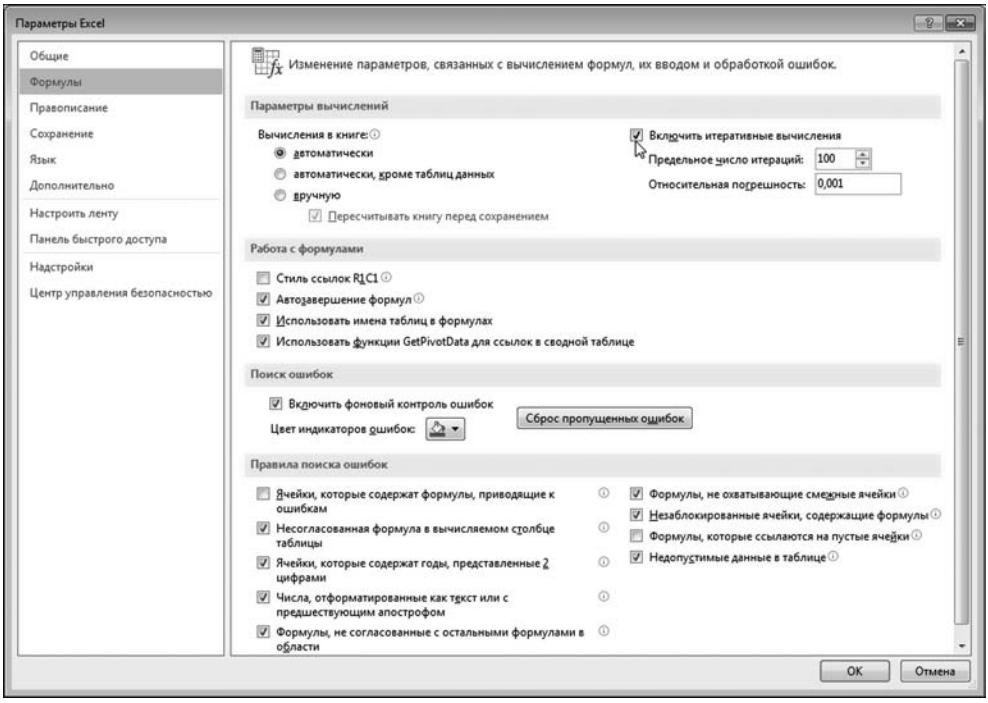


Рис. 5.26
Переход в режим использования циклических ссылок (итеративных вычислений)

Для использования циклических ссылок необходимо предварительно перейти в соответствующий режим. С этой целью в окне настроек приложения **Параметры Excel** в разделе **Формулы** устанавливаем флажок опции **Включить итеративные вычисления** (рис. 5.26).

В полях **Предельное число итераций** и **Относительная погрешность** указываются параметры, которые непосредственно влияют на процедуру вычисления циклических ссылок. В большинстве случаев достаточно оставить те значения, что предлагаются по умолчанию.

На заметку

Поскольку циклическая ссылка предполагает вычисление значения ячейки на основе значения этой ячейки, вычисления производятся методом итераций — отсюда, собственно, и название опции **Включить итеративные вычисления**. Итерационный процесс реализуется так. Начальным значением ячейки является нулевое. Это нулевое значение используется в циклической ссылке для вычисления значения ячейки для первой итерации. Значение ячейки для первой итерации используется для вычисления значения ячейки для второй итерации и т. д. Предельное количество итераций определяется значением поля **Предельное число итераций**. Вычисления также заканчиваются, если относительная погрешность вычислений становится меньше того значения, что указано в поле **Относительная погрешность**.

Один из «математических» примеров применения циклических ссылок — решение алгебраических уравнений вида $x = f(x)$ методом последовательных итераций. Этот метод подразумевает, что новое приближение для корня уравнения вычисляется на основе приближения, вычисленного на предыдущем шаге. Если на k-м шаге вычислено приближенное значение корня x_k, то следующее приближение x_{k+1} вычисляется по формуле $x_{k+1} = f(x_k)$. Для применения этого метода нужно знать некоторое начальное приближение x_0 для корня уравнения. Эта схема как нельзя лучше подходит для реализации вычислений по циклической ссылке.

На заметку

Чтобы метод сходился, функция $f(x)$ в правой части уравнения должна удовлетворять некоторым условиям. Нас они пока не интересуют.

В качестве иллюстрации применения циклической ссылки решим алгебраическое уравнение $x^2 - 6x + 5 = 0$. У этого уравнения два корня $x = 1$ и $x = 5$. Мы попытаемся найти оба эти корня, причем будем искать их методом последовательных итераций. Начнем с меньшего корня и для его поиска представим уравнение в виде $x = \frac{x^2 + 5}{6}$. Решим это уравнение средствами Excel (с помощью циклической ссылки). В документе Excel в ячейку A1 вводим формулу =(A1$2+5)/6. Результат показан на рисунке 5.27.

Рис. 5.27
Вычисления по циклической ссылке

В результате в приближенном виде найден первый из корней. Точность этого решения определяется настройками

полей **Предельное число итераций** и **Относительная погрешность**. Но и так видим, что решение уравнения вычислено с неплохой точностью.

На заметку

Уточнить результат вычислений можно, нажав клавишу <F9>. При этом итерационный процесс продолжается, начиная с текущего числового значения ячейки. Нечто похожее происходит при пересчете документа после изменения одной из ячеек документа.

Главная неприятность при работе с циклическими ссылками связана с тем, что итерационный процесс начинается с нулевого начального значения. Например, чтобы найти второй корень уравнения методом итераций, это уравнение представим в виде $x = \sqrt{6x - 5}$. Проблема в том, что для корня нулевое начальное приближение в данном случае неприменимо — под корнем получается отрицательное выражение. Поэтому если в ячейку A1 введем формулу =КОРЕНЬ(6*A1-5), получим вместо корня сообщение об ошибке в ячейке (рис. 5.28).

На заметку

Функция КОРЕНЬ() используется для вычисления квадратного корня.

Проблема не в самой циклической ссылке, а в том, что для вычислений используется именно нулевое начальное значение. Но эта проблема устранима. Устранять начнем прямо сейчас.

Решаем все то же уравнение $x = \sqrt{6x - 5}$, но теперь в качестве начального значения для корня используем не ноль, а некоторое значение a. Это начальное значение мы будем указывать в ячейке A2. Фактически, речь идет о решении некоторого уравнения $x = f(x)$ с начальным значением $x = a$. Мы сведем задачу к решению другого уравнения, но уже с нулевым начальным значением для аргумента. Идея состоит в том, чтобы ввести новую переменную $z = x - a$ или $x = a + z$. Делаем подстановку в исходное уравнение и получаем новое уравнение $z = f(z + a) - a$ и для переменной z начальное значение нулевое. Что касается нашего конкретного случая, то решаем уравнение $z = \sqrt{6(z + a) - 5} - a$, где значение a является начальным приближением для поиска корня.

Практическая фаза вычислений состоит в том, что в ячейку A2 вводится начальное приближение для корня, а в ячейку A1 вводится формула =КОРЕНЬ(6*(A1+A2)-5)-A2. Результат показан на рисунке 5.29.

Рис. 5.28
Вычисление по циклической ссылке приводит к ошибке

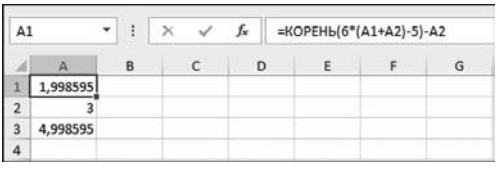


Рис. 5.29
Начальное значение для итерационных вычислений задается в явном виде

На заметку

Обратите внимание, что в результате решения уравнения $z = f(z + a) - a$ находим решение для переменной z. Чтобы найти значение переменной x, необходимо к значению z прибавить начальное приближение a, т. е. вычислить $x = z + a$.

В ячейке A1 вычисляется значение переменной z. Чтобы вычислить корень уравнения, необходимо к значению ячейки A1 (с циклической ссылкой) прибавить значение ячейки A2 (начальное приближение для корня). Искомое решение вычисляем в ячейке A3 по формуле =A1+A2. Числовое значение в этой ячейке очень близко к точному значению для второго (большего) корня уравнения.

На заметку

Выше мы проигнорировали условия, которые накладываются на функцию $f(x)$ для того, чтобы уравнение могло решаться методом последовательных итераций. Сейчас мы можем приоткрыть завесу тайны. Итак, функция $f(x)$ должна быть такой, что в области поиска корня ее производная по модулю была меньше единицы, т. е. должно выполняться условие $|f'(x)| < 1$. И уж, как минимум, это условие должно выполняться в точке начального приближения для корня.

ССЫЛКИ В РАЗНЫХ ЛИСТАХ И КНИГАХ

Сила искусства и талант режиссера
ведут нас на дно.

Из к/ф «Старый знакомый»

Формула может содержать ссылки на ячейки в том же рабочем листе, в другом листе той же книги или вообще в другой книге. Если мы ссылаемся на ячейку в другом рабочем листе (другом по отношению к рабочему листу, в котором содержится формула со ссылкой), то имя рабочего листа указывается перед адресом ячейки. Имя рабочего листа и непосредственно адрес ячейки разделяются восклицательным знаком. Например, формула со ссылкой на ячейку B3 в рабочем листе Лист2 выглядит как =Лист2!B3. Для надежности имя рабочего листа лучше заключить в одинарные кавычки. А если имя рабочего листа состоит из нескольких слов (содержит пробелы), то без одинарных кавычек не обойтись. Допустим, мы хотим создать формулу со ссылкой на ячейку B3 в рабочем листе с названием Второй лист. Формула будет выглядеть как ='Второй лист'!B3.

Если ячейка, на которую выполняется ссылка, находится в другой (открытой) книге, то имя книги указывается в квадратных скобках перед именем рабочего листа. Формула со ссылкой на ячейку B3 рабочего листа Лист2 книги Книга5.xlsx имеет вид =[Книга5.xlsx]Лист2!B3. Опять же, блок с именем книги и названием рабочего листа желательно заключить в одинарные кавычки. Вот пример такой ссылки: ='[Моя книга.xlsx]Второй лист'!B3. Это формула со ссылкой на ячейку B3 рабочего листа Второй лист рабочей книги

Моя книга.xlsx. Выше был сделан намек на то, что в таком формате выполняется ссылка на открытые рабочие книги или рабочие книги в одном каталоге. Если ситуация настолько сложна, что рабочие книги находятся где-то очень далеко, то это самое место указывается перед именем книги. Другими словами, в названии книги можно указывать полный путь к ней. Путь к книге размещается перед именем книги (вне квадратных скобок). Формула ='C:\Книги\Excel\[Моя книга.xlsx]Второй лист'!B3 является ссылкой на ячейку B3 в рабочем листе Второй лист в рабочей книге Моя книга.xlsx, которая находится в каталоге C:\Книги\Excel\.

На заметку

После ввода знака равенства приложение переходит в режим ссылки, при котором ссылку на ячейку можно ввести с помощью мыши. Для этого достаточно щелкнуть мышью на соответствующей ячейке. Этот же подход применим, если речь идет о ячейках в разных листах или книгах. Принцип неизменен: размещаем курсор ввода в том месте формулы, где следует ввести ссылку, после чего мышью выделяем нужную ячейку. Не беда, если для этого предварительно придется перейти (щелкнув левой кнопкой мыши) к нужному листу или книге. После завершения ввода формулы нажимаем клавишу <Enter>.

На заметку

Если в формуле содержится ссылка на ячейку в другой открытой книге в «коротком» формате (т. е. только имя книги, рабочий лист и адрес ячейки), то после закрытия этой книги в ссылку автоматически будет добавлен полный путь к файлу книги.

Когда открывается книга, содержащая ссылки на внешние рабочие книги, появляется диалоговое окно, в котором пользователю предлагается отреагировать на такую ситуацию. Связи нужно обновить или не обновлять — в зависимости от обстоятельств. Для каждого из этих действий предусмотрена специальная кнопка с недвусмысленным названием. На самый крайний случай в диалоговом окне имеется кнопка для вызова справки.

УТИЛИТА ПОДБОРА ПАРАМЕТРА

Вот это другое дело.
На такой скорости можно даже
успеть законспектировать.

Из к/ф «Старый знакомый»

Большинство расчетных возможностей Excel реализуется через систему встроенных функций. Вместе с тем в некоторых случаях удобнее прибегнуть к помощи специальных утилит или надстроек. В данном случае мы обсудим простую, элегантную и полезную утилиту *подбора параметра*. В общих чертах назначение утилиты состоит в том, чтобы подобрать значение в какой-то определенной ячейке такое, что значение в другой ячейке примет нужное значение.

Запускается утилита исключительно просто — на вкладке **Данные** ленты в группе **Работа с данными** в раскрывающемся списке пиктограммы **Анализ «что-если»** выбираем команду **Подбор параметра** (рис. 5.30).

В результате открывается диалоговое окно, представленное на рисунке 5.31.

Окно **Подбор параметра** содержит три поля, в которые вводится некоторая полезная информация. В поле **Установить в ячейке** указывается адрес целевой ячейки, т. е. той ячейки, значение которой нужно «подогнать». К какому значению выполняется «подгонка», указываем в поле **Значение**. В поле **Изменяя значение ячейки** указывается адрес той ячейки, значение которой следует изменять для «подгонки» значения целевой ячейки.

На заметку

Значения в полях **Установить в ячейке** и **Изменяя значение ячейки** — адреса ячеек. Эти адреса можно ввести с помощью набора на клавиатуре, но проще сделать это, выбрав нужные ячейки мышью в рабочем документе. При этом адреса ячеек вводятся как абсолютные. Также справа от указанных полей есть специальные пиктограммы. Щелчок на такой пиктограмме приводит к тому, что окно переходит в режим поля ввода, как показано на рисунке 5.32. Повторный щелчок на пиктограмме вернет окну **Подбор параметра** исходный полноценный вид. Эта же манипуляция с пиктограммами характерна для многих других окон приложения Excel.

Проиллюстрируем возможности утилиты подбора параметра на примере. Решим с помощью этой утилиты уже рассмотренное выше уравнение

Рис. 5.30
Запуск утилиты подбора параметра

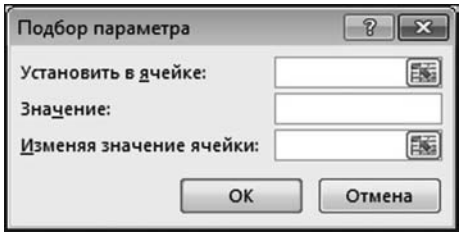


Рис. 5.31
Окно утилиты подбора параметра

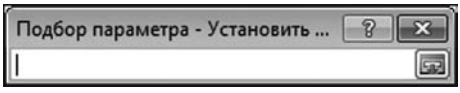


Рис. 5.32
Окно утилиты подбора параметра в режиме поля ввода

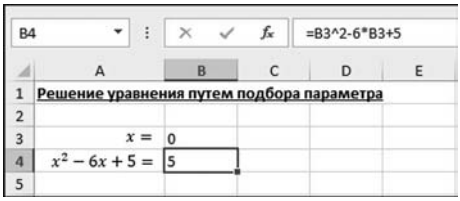


Рис. 5.33
Документ для решения уравнения $x^2 - 6x + 5 = 0$

$x^2 - 6x + 5 = 0$. Для этого воспользуемся документом, представленным на рисунке 5.33.

В ячейку B3 вводится начальное нулевое приближение для корня уравнения, а ячейка B4 содержит формулу =B3$2-6*B3+5.

На заметку

Документ содержит формулы — не те формулы, что вводятся в ячейки, а графические формулы для отображения математических выражений. Напомним, в версии Excel 2013 есть собственный довольно элегантный редактор формул. На вкладке **Вставка** ленты в группе **Символы** имеется пиктограмма-раскрывающийся список **Уравнение** для вставки графических формул (рис. 5.34).

Нужное выражение набирается с помощью редактора формул и вставляется в графическую подложку рабочего листа. Поэтому такие формулы являются, фактически, графическими объектами, которые можно перемещать по области рабочего листа, без привязки к ячейкам.

Таким образом, если в ячейке B3 указать корень уравнения $x^2 - 6x + 5 = 0$, то в ячейке B4 будет отображаться нулевое значение. Другими словами, в ячейке B3 нужно подобрать такое значение, чтобы в ячейке B4 значение стало нулевым. Для решения этой задачи и воспользуемся утилитой подбора параметра. Запускаем утилиту и заполняем поля окна утилиты **Подбор параметра** (табл. 5.2).

На рисунке 5.35 показано на фоне окна рабочего документа окно утилиты **Подбор параметра** с выполненными настройками.

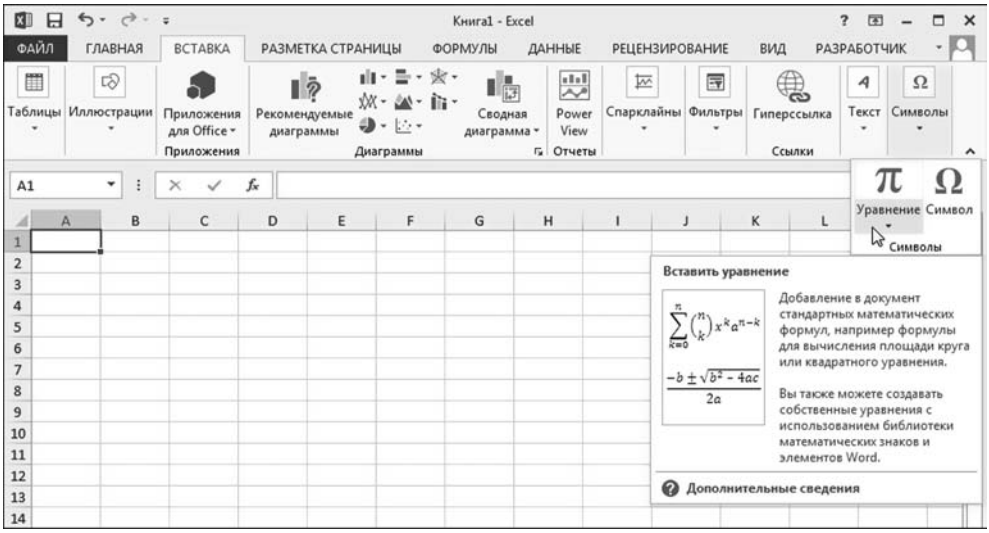


Рис. 5.34
Пиктограмма вставки формулы

Таблица 5.2

Заполнение полей окна утилиты Подбор параметра

Поле	Значение
Установить в ячейке	В этом поле указывается адрес ячейки B4
Значение	Здесь указываем нулевое значение, поскольку именно нулевым должно быть значение в ячейке B4, если в ячейке B3 указано значение для корня уравнения
Изменяя значение ячейки	В поле указывается адрес той ячейки, значение которой изменяется в процессе поиска корня уравнения, т. е. адрес ячейки B3

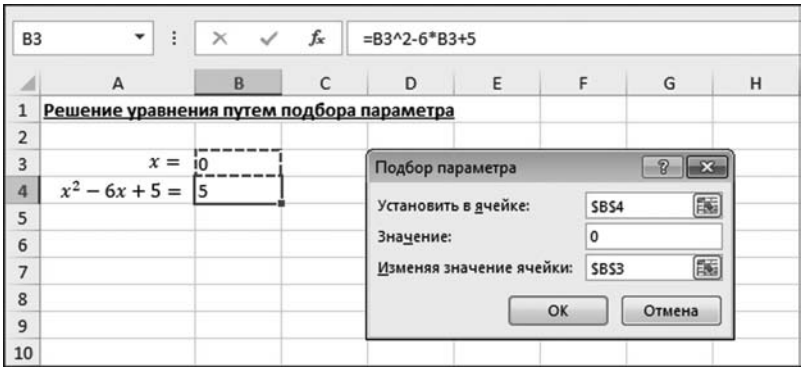


Рис. 5.35
Рабочий документ с выполненными настройками в окне утилиты **Подбор параметра**

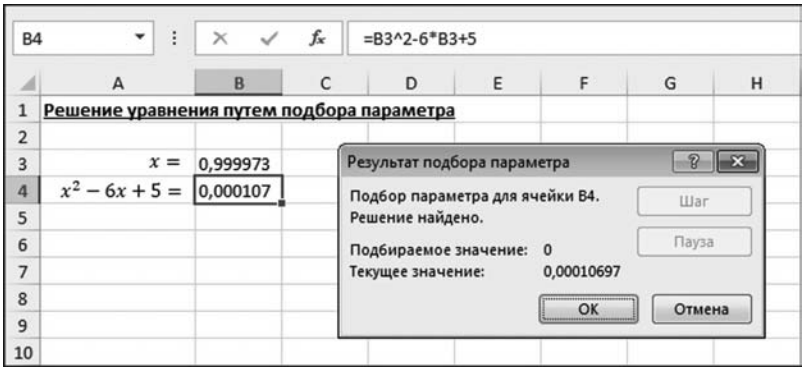


Рис. 5.36
Результат поиска корня уравнения с помощью утилиты **Подбор параметра**

После щелчка на кнопке **OK** в окне **Подбор параметра** получаем результат, представленный на рисунке 5.36.

С достаточно неплохой точностью найден корень $x = 1$. Если результат нас устраивает, щелкаем на кнопке **OK** в окне **Результаты подбора параметра**. Кнопка **Отмена** предназначена для случая, когда нужно вернуть все в исходное состояние — как это было до того, как мы начали процедуру подбора параметра.

На заметку

Найден один из двух корней уравнения. Второй корень $x = 5$ можно найти, если изменить начальное значение в ячейке B3. Например, если перед запуском утилиты **Подбор параметра** в ячейку B3 ввести значение 8 (рис. 5.37), то в результате найдем второй корень уравнения (рис. 5.38).

К сожалению, угадать, как начальное приближение влияет на результат подбора параметра, бывает крайне сложно. Приведенный пример слишком прост для того, чтобы на его основе делать какие-то глобальные выводы. Главный рецепт состоит в том, чтобы пробовать разные варианты.

Утилита подбора параметра достаточно простая и имеет ряд серьезных ограничений. Главное из них, пожалуй, связано с тем, что варьируемая ячейка всего одна. Другими словами, в процессе подбора параметра изменяется значение всего одной ячейки.

На заметку

Целевая ячейка тоже одна. Но с практической точки зрения это не так неудобно, как невозможность одновременного варьирования значений сразу нескольких ячеек. В этом отношении более эффективной является надстройка **Поиск решения**, о которой речь пойдет несколько позже. Утилиту подбора параметра обычно используют для решения несложных задач.

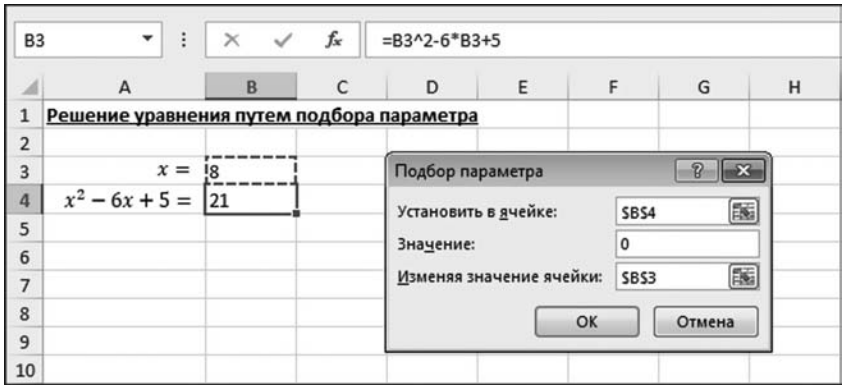


Рис. 5.37
В ячейку B3 введено значение 8

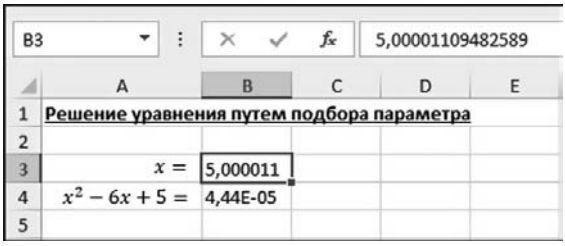


Рис. 5.38
Найден второй корень уравнения

ПОИСК И ФИЛЬТРОВАНИЕ ДАННЫХ

— Ох, тетушка, мы с Вами вроде и по-русски говорим,
да на разных языках. Я Вам про что толкую?
Про смысл бытия! Для чего живет человек на земле, скажите?
— Как же так, сразу-то? И потом, где живет?
Ежели у нас, в Смоленской губернии — это одно.
Ежели в Тамбовской губернии — это другое.
— Нет, сие невыносимо!

Из к/ф «Формула любви»

Есть несколько фундаментальных проблем, которые приходится решать при работе с приложением Excel. Во-первых, важно правильно и красиво заполнить документ и внести в него данные. Во-вторых, необходимо уметь данные в рабочем документе находить. Здесь остановимся на второй проблеме. В частности, предполагаем, что документ с данными у нас уже есть и теперь следует научиться находить те из них, которые соответствуют некоторому критерию.

На заметку

Разумеется, можно последовательно просмотреть все ячейки рабочего листа в поисках нужной информации. Потом, если надо, еще раз просмотреть, и еще... Но это не самый эффективный подход.

В группе **Редактирование** вкладки **Главная** есть пиктограмма **Найти и выделить**, которой и воспользуемся для поиска нужных данных в ячейках документа (рис. 5.39).

На заметку

Надо отметить, что исследуемый документ содержит несколько ячеек с числовыми значениями, формулами и текстом.

Нас интересуют ячейки содержащие формулы. Поэтому в раскрывающемся списке пиктограммы **Найти и выделить** выбираем позицию **Формулы** (рис. 5.39).

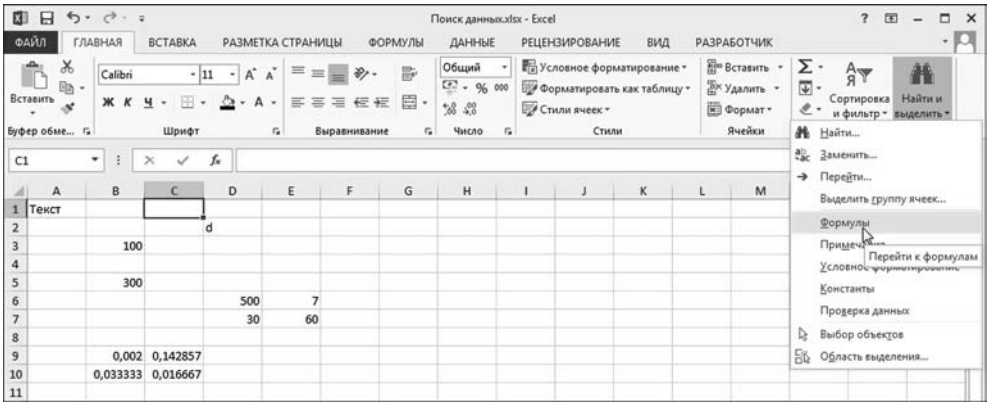


Рис. 5.39
Выделение в документе с данными ячеек, содержащих формулы

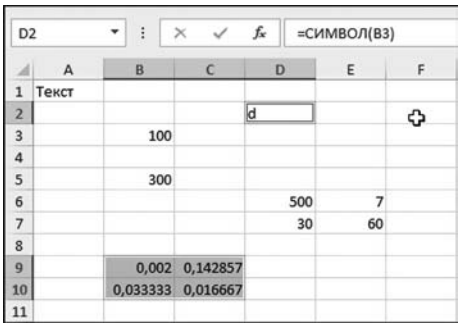


Рис. 5.40
В документе выделены ячейки с формулами

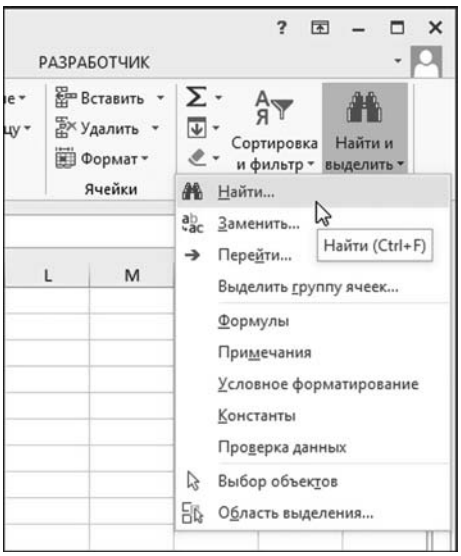


Рис. 5.41
Поиск текстовых значений в рабочем документе

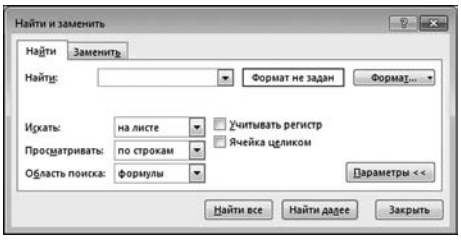


Рис. 5.42
Окно для поиска (и замены) текстовых значений в рабочем документе

Результат не заставит себя долго ждать — на рисунке 5.40 показан документ, в котором как следствие манипуляций с пиктограммой **Найти и выделить** выделены все ячейки рабочего листа, в которых содержатся формулы.

Аналогичным образом выделяются ячейки, содержащие константы, примечания или, например, условное форматирование. Есть в списке пиктограммы **Найти и выделить** и другие полезные команды. Например, с помощью команды **Найти** выполняется поиск (и, в случае необходимости, замена) текстовых значений в ячейках рабочего документа (рис. 5.41).

В результате открывается диалоговое окно **Найти и заменить**, которое показано на рисунке 5.42.

В этом окне задаются параметры для поиска и замены текстовых значений в ячейках рабочего документа (с учетом форматирования и места поиска).

Другая полезная и часто используемая команда предназначена для сортировки значений в ячейках рабочего листа. На рисунке 5.43 показан небольшой документ с импровизированным списком, который содержит несколько фамилий и числовые данные о возрасте. Допустим, нам необходимо отсортировать список так, чтобы он размещался в алфавитном порядке. Для этого выделяем диапазон ячеек с фактическими данными (имеются в виду ячейки с фамилиями и соответствующие им смежные ячейки с данными о возрасте) и в раскрывающемся списке пиктограммы **Сортировка и фильтр** выбираем команду **Сортировка от А до Я** (рис. 5.43).

В результате выполняется сортировка по содержимому первого столбца с текстовыми значениями. При этом соответствующим образом переставляются и смежные ячейки (рис. 5.44).

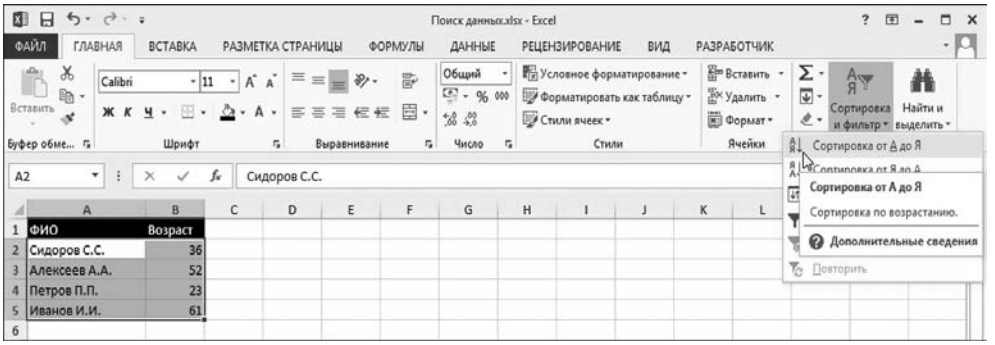


Рис. 5.43
Сортировка данных в выделенном диапазоне ячеек

Если такой способ сортировки по каким-то причинам не устраивает, то можно воспользоваться режимом настраиваемой сортировки, при котором способ обработки данных задается пользователем в полуавтоматическом режиме. Полезной в этом случае будет команда **Настраиваемая сортировка** в раскрывающемся списке команд пиктограммы **Сортировка и фильтр** (рис. 5.45).

Открывается диалоговое окно **Сортировка**, которое показано на рисунке 5.46.

В этом окне выполняются настройки параметров сортировки, такие, например, как позиция, по которой выполняется сортировка, параметр сортировки, а также алгоритм.

Еще одна полезная возможность связана с применением фильтра. В отношении рассматриваемого документа выполним следующие действия: выделяем диапазон ячеек A1:B1 (диапазон заголовков нашей импровизированной таблицы) и в списке команд пиктограммы **Сортировка и фильтр** щелкаем меню-переключатель **Фильтр** (рис. 5.47).

Результат этих действий представлен на рисунке 5.48.

Главное внешнее проявление состоит в том, что заголовки таблицы обзавелись пиктограммами раскрывающихся

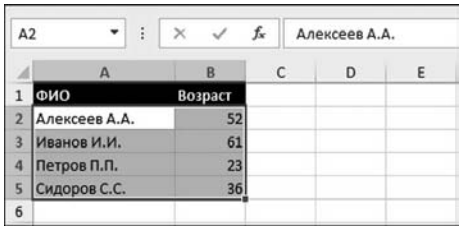


Рис. 5.44
Результат сортировки данных в выделенном диапазоне

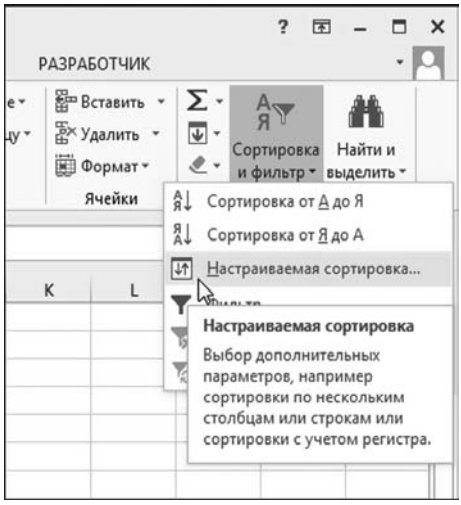


Рис. 5.45
Применение особых правил сортировки данных

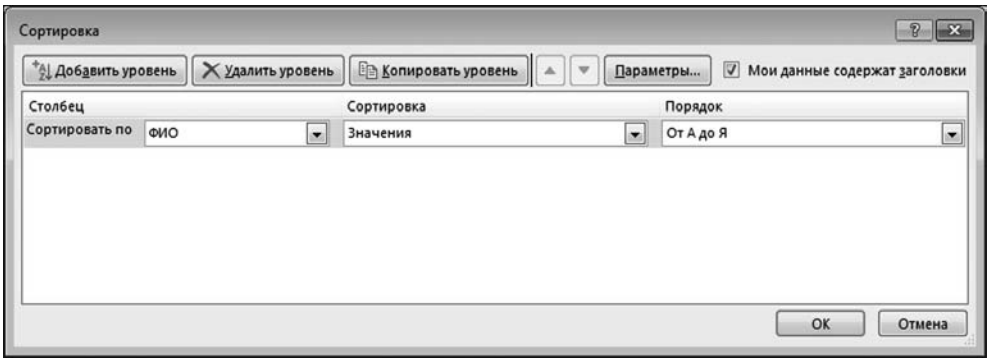


Рис. 5.46
Диалоговое окно **Сортировка** для настройки параметров сортировки данных

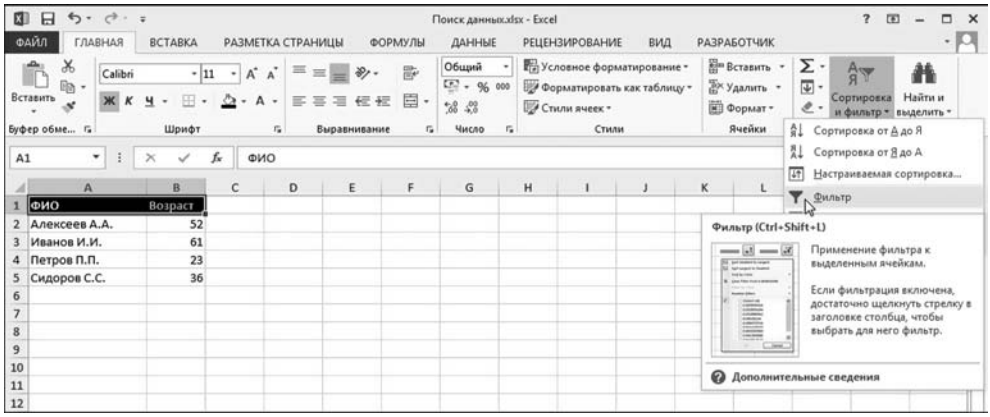


Рис. 5.47
Добавление фильтра в документ

списков. Этот эффект не иллюзорный — щелчок пиктограммы приводит к тому, что раскрывается список с довольно любопытным содержимым (рис. 5.49).

В данном случае мы раскрыли список для позиции ФИО, и этот список содержит, среди прочего, опционные позиции для фильтрации отображаемых данных — убрав флажок той или иной опции, скрываем соответствующие данные из области видимости. Например, на рисунке 5.50 показано, как будет выглядеть исходная таблица, если при фильтрации данных убран флажок опции для первой и последней позиций в списке фамилий.

На заметку

В данном случае строки таблицы скрыты, но скрыты особым образом. Вернуть все, как было, можно с помощью команд пиктограммы **Сортировка и фильтр** — в частности, командой **Очистить** можем отменить настройки фильтра или вообще отменить фильтрацию, повторно нажав пункт-переключатель **Фильтр**.

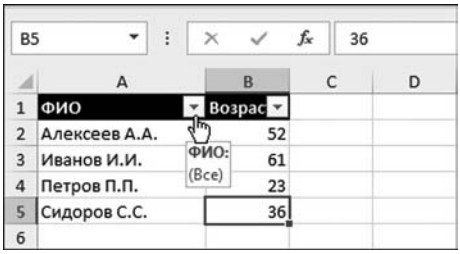


Рис. 5.48
Результат добавления фильтра

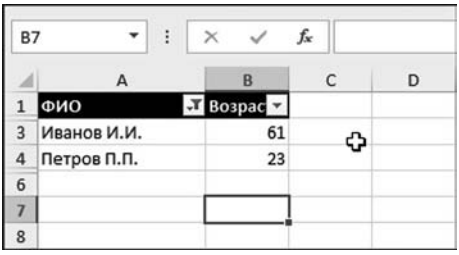


Рис. 5.50
В результате применения фильтра несколько строк скрыты

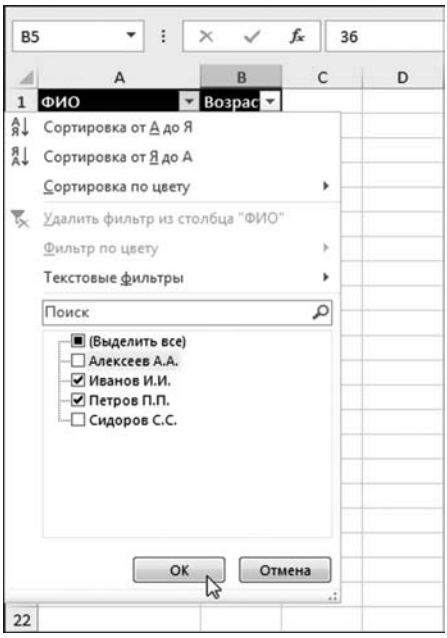


Рис. 5.49
Настройка параметров фильтра

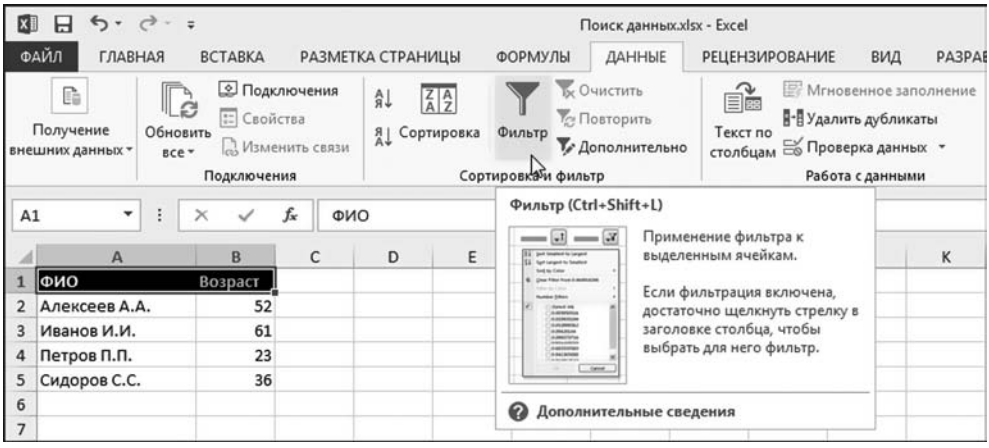


Рис. 5.51
Вкладка **Данные** содержит полезные утилиты для фильтрации и сортировки данных

Фильтровать и сортировать данные можно также с помощью пиктограмм вкладки Данные (рис. 5.51).

Например, с помощью пиктограммы **Фильтр** в группе **Сортировка и фильтр** можно создать фильтр, аналогичный тому, что описывался выше. Не должно у пользователя возникнуть затруднений и с другими пиктограммами этой вкладки.

ГЛАВА 6 ФУНКЦИИ

Довольно, сеньора!
Вы прибыли в Россию, извольте говорить по-русски!
Этого требует от нас Великий Магистр.

Из к/ф «Формула любви»

В этой главе речь пойдет о встроенных функциях Excel и основных приемах их использования для выполнения различных вычислений. Начнем с краткого обзора встроенных функций и методов использования таких функций в рабочем документе.

ДОБАВЛЕНИЕ ФУНКЦИИ В ФОРМУЛУ

Прямо картина из недалекого будущего!

Из к/ф «Безумный день инженера Баркасова»

В принципе, ничего сложного в использовании встроенных функций Excel нет. В нужном месте формулы указывается имя встроенной функции с соответствующими аргументами. Аргументы указываются в круглых скобках. Разделяются аргументы функции, если их несколько, точкой с запятой. Если у функции аргументов нет, круглые скобки все равно указываются.

На заметку

Функции, о которых здесь идет речь, обычно называются функциями Excel или функциями рабочего листа (документа) Excel, доступны по умолчанию и используются в формулах непосредственно в ячейках рабочего документа Excel. Кроме этих функций, есть еще функции языка программирования Visual Basic for Applications (сокращенно VBA). С последними мы познакомимся, когда будем осваивать методы создания макросов и функций пользователя.

Замечание о том, что аргументы функций разделяются точкой с запятой, относится к русифицированной версии Excel. В оригинальной, англоязычной версии приложения аргументы функций рабочего листа разделяются запятыми. Более того, в русифицированной версии Excel русифицированы также и названия встроенных функций, так что в основной своей массе они имеют кириллические названия.

Что касается функций VBA, то все они имеют англоязычные названия не зависимо от версии приложения. Аргументы функций VBA разделяются запятыми. Но этот вопрос для нас пока не актуален.

У описанного подхода есть один тонкий момент — при вводе имен функций с клавиатуры необходимо знать, как эти самые функции называются. С другой стороны, при вводе имени функции появляется оперативная контекстная справка — такой режим используется по умолчанию (рис. 6.1).

Если при вводе формулы мы начинаем набирать имя функции, выпадает контекстное меню со списком функций, названия которых соответствуют введенному фрагменту имени функции. По мере ввода имени функции этот список сокращается. Совсем плохо, если он станет пустым еще до того, как имя функции введено в формулу. Если в списке функций имеется нужное имя, достаточно выбрать это имя из списка (двойным щелчком мыши, например). После этого имя функции добавляется в формулу, причем для ввода аргументов выводится справка с кратким обозначением аргументов и подсветкой вводимого аргумента (рис. 6.2).

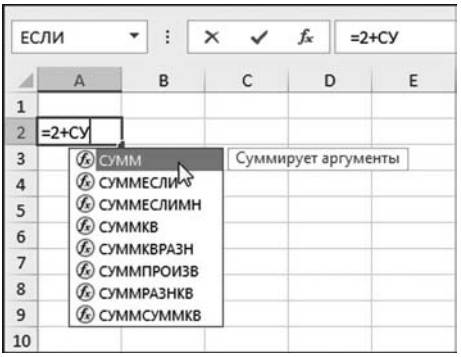


Рис. 6.1
При вводе имени функции появляется контекстная подсказка

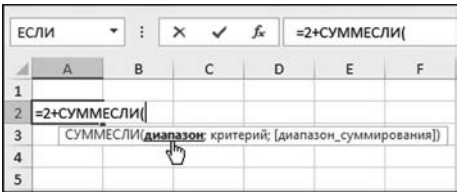


Рис. 6.2
Подсказка для ввода аргументов функции

На заметку

Появляющиеся подсказки — следствие соответствующих настроек приложения. Такие настройки используются по умолчанию. Поэтому теоретически при вводе имени функции подсказки могут и не появляться.

Формула может вводиться как непосредственно в ячейку (как в предыдущем случае), так и в строку формул (при выделенной ячейке ввода). При работе со строкой формул ситуация меняется не сильно. Как и раньше, при вводе имени функции в строке формул появляется список с именами функций, которые соответствуют шаблону вводимой функции (рис. 6.3).

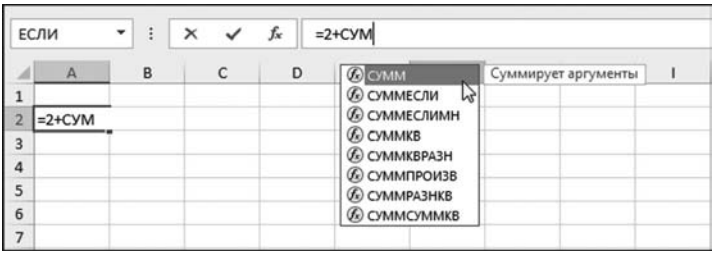


Рис. 6.3
Ввод формулы с функцией в строке формул

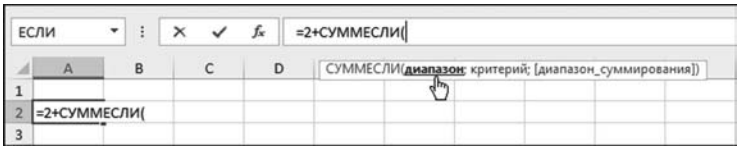


Рис. 6.4
Подсказка по аргументам функции при вводе формулы в строке формул

Рис. 6.5
Щелчок на специальной пиктограмме слева от строки формул нередко бывает полезным

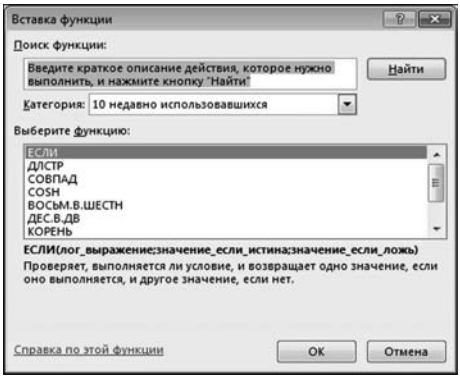


Рис. 6.6
Окно вставки функции **Вставка функции**

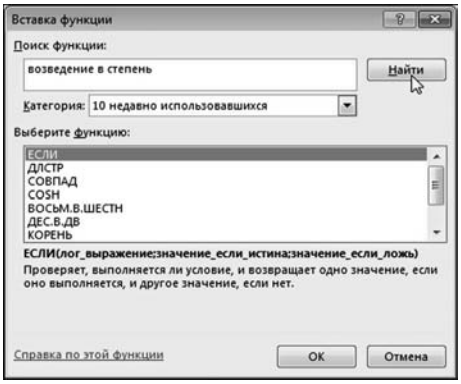


Рис. 6.7
Поиск функции по ключевой фразе

Точно так же, при вводе аргументов появляется простая, но эффектная подсказка (рис. 6.4).

Конечно, это мелочь, но мелочь очень полезная и приятная.

Слева от строки формул есть пиктограмма с изображением символа функции. Пиктограмма весьма полезная. Щелчок на ней изменяет процедуру ввода формулы до неузнаваемости (рис. 6.5).

В частности, отображается диалоговое окно **Вставка функции**, представленное на рисунке 6.6.

На заметку

При щелчке на пиктограмме вставки функции знак равенства, с которого начинаются все формулы, добавляется автоматически.

В общем и целом окно предназначено для поиска нужной функции из набора встроенных функций (и не только). Для успешной работы с окном желательно хотя бы примерно представлять, что и где там «спрятано».

В текстовой области **Поиск функции** вводится текстовая фраза, по которой выполняется поиск нужной функции. Например, вводим в это текстовое поле фразу возведение в степень (рис. 6.7).

Далее щелкаем кнопку **Найти** и получаем результат, представленный на рисунке 6.8.

Попытка оказалась неудачной, и нас просят повторить ее. Мы так и по-

ступаем, введя теперь в текстовую область **Поиск функции** всего одно слово степень (рис. 6.9).

На этот раз получаем не очень большой список функций, к которым так или иначе (на уровне описания функции) может относиться фраза степень (рис. 6.10).

Если в списке есть нужная функция, выбираем ее двойным щелчком. Если же просто выделить функцию в списке, в нижней части окна появится краткая справка по функции. Для получения более полной (системной) справки щелкаем на гиперссылке **Справка по этой функции** в левом нижнем углу окна **Вставка функции** (рис. 6.11).

Окно полноценной справки, выведенное в результате щелчка на гиперссылке, показано на рисунке 6.12.

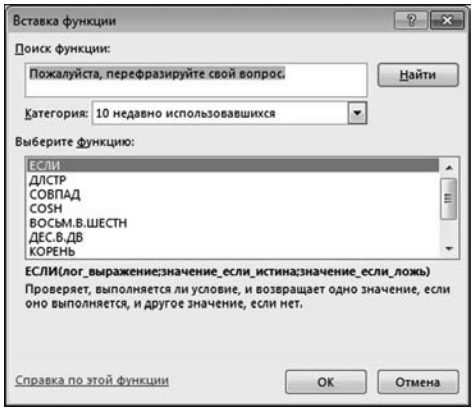


Рис. 6.8
Результат поиска функции не очень удачный

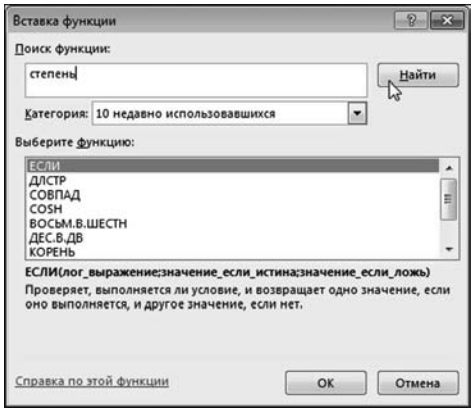


Рис. 6.9
Еще одна попытка найти функцию по ключевой фразе

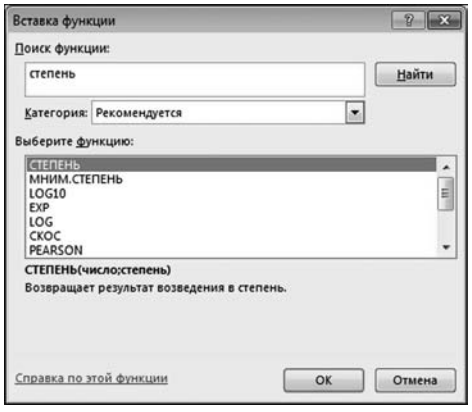


Рис. 6.10
Список функций, найденных по ключевой фразе

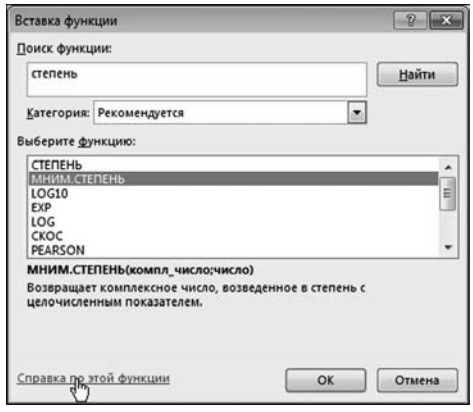


Рис. 6.11
При выборе функции в списке отображается краткая справка по функции

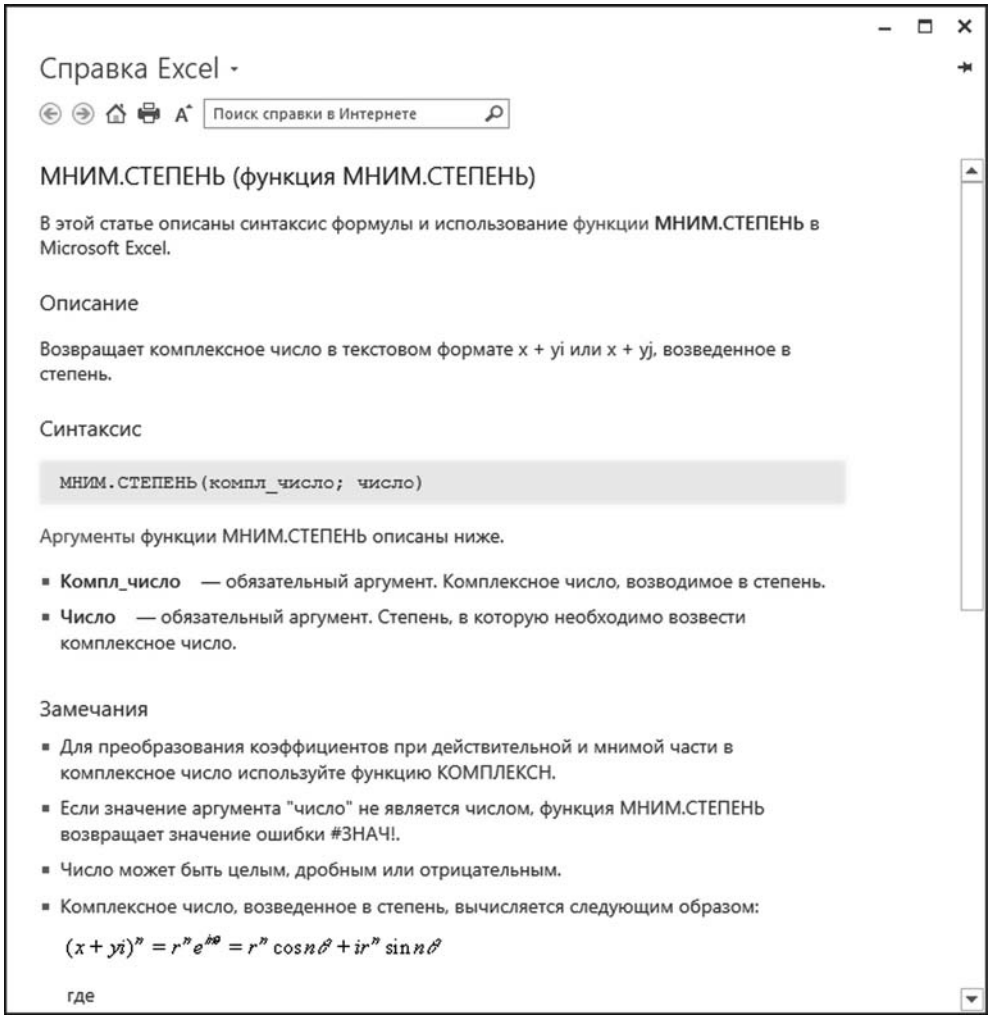


Рис. 6.12
Справка по функции

В этом окне обычно можно найти практически все, что нужно знать для успешного использования функции в рабочем документе.

Другой, более надежный способ поиска функции базируется на сужении круга поиска путем выбора категории функции в раскрывающемся списке **Категория**. Среди доступных категорий, например, есть **Финансовые** (для финансовых функций), **Математические** (математические функции), **Полный алфавитный перечень** (полный перечень встроенных функций в алфавитном порядке), **10 недавно использовавшихся** (последние использовавшиеся десять функций) и т. д. Выбрав ту или иную категорию, можно значительно сократить список в области **Выберите функцию**. На рисунке 6.13 показан вид окна **Вставка функции**, когда в списке **Категория** выбрана кате-

гория **Инженерные** (функции для работы с довольно специфическими математическими функциями, которые согласно классификации разработчиков Excel относятся к инженерным функциям).

В данном случае выбрана функция ДЕС.В.ДВ(), с помощью которой выполняется преобразование десятичного числа в двоичное. После выделения функции в списке и щелчка на кнопке **ОК** получаем следующее окно **Аргументы функции** для ввода аргументов функции (рис. 6.14).

В принципе, это окно можно проигнорировать и вводить аргументы функции уже после набора формулы. Однако окно на самом деле достаточно удобное. Для разных функций оно разное. Но идея одна и та же: окно содержит поля для ввода аргументов, отображает подсказку по аргументу активного поля, а также результат вызова функции с указанными аргументами. На рисунке 6.15 показано окно **Аргументы функции** с заполненными аргументами для функции ДЕС.В.ДВ().

Первый аргумент этой функции — это преобразуемое в двоичный код десятичное число (в данном случае **12**). Второй аргумент функции — количество позиций, используемых для отображения двоичного кода (в данном случае **10**). Результат вызова функции с такими аргументами отображается в левом нижнем углу после слова **Значение**.

На заметку

В поля ввода аргументов можно вводить адреса ячеек, причем в режиме выбора ячеек в рабочем документе с помощью мыши.

Результат выполненных манипуляций представлен в документе на рисунке 6.16 (для удобства восприятия увеличена ширина первого столбца).

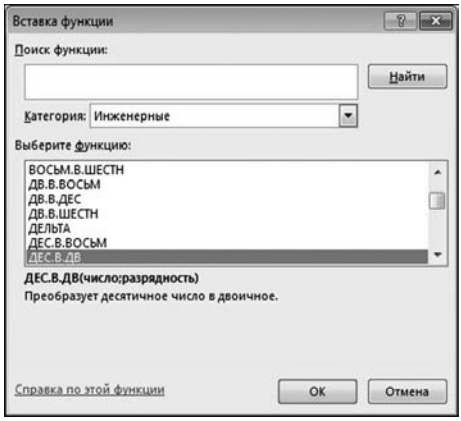


Рис. 6.13
Выбор инженерной функции

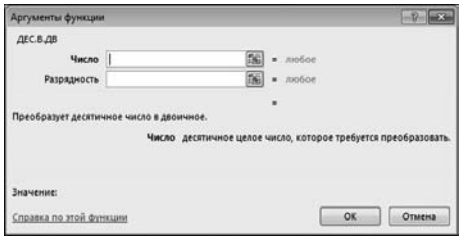


Рис. 6.14
Окно ввода аргументов функции

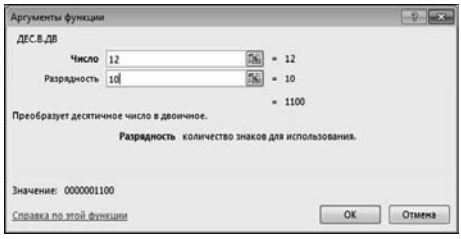


Рис. 6.15
При указанных аргументах в окне **Аргументы функции** отображается результат вызова функции

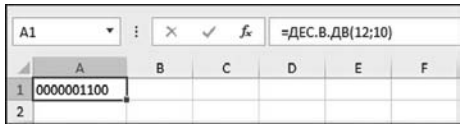


Рис. 6.16
Результат ввода функции

На заметку

При вводе формулы (после ввода знака равенства) в поле имен отображается имя последней использованной функции. Если щелкнуть на пиктограмме этого поля, откроется список недавно использовавшихся функций, как показано на рисунке 6.17.

Из этого списка можно выбрать нужную функцию, конечно, при условии, что она там есть. Если в списке выбрать позицию **Другие функции**, откроется окно вставки функций **Вставка функции**, которое описывалось ранее.

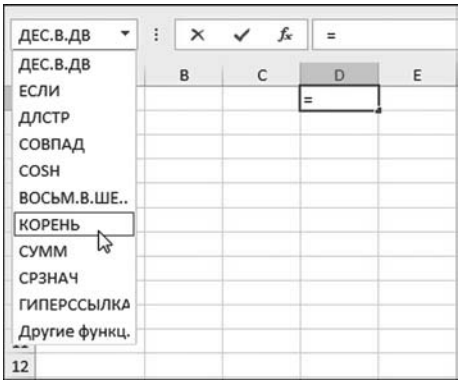


Рис. 6.17
Вставка функции в формулу с помощью списка поля имен

Для работы с функциями предназначена вкладка **Формулы** ленты. Краткое знакомство с этой вкладкой у нас уже состоялось. Правила вежливого тона обязывают нас пополнить багаж знаний об этой вкладке. Вкладка состоит из нескольких групп, каждая из которых содержит весьма полезные пиктограммы и прочие утилиты.

Самая большая пиктограмма в группе **Библиотека функций** внешне напоминает пиктограмму в строке формул и предназначена для тех же целей — отображения окна **Вставка функции** для вставки функции (рис. 6.18).

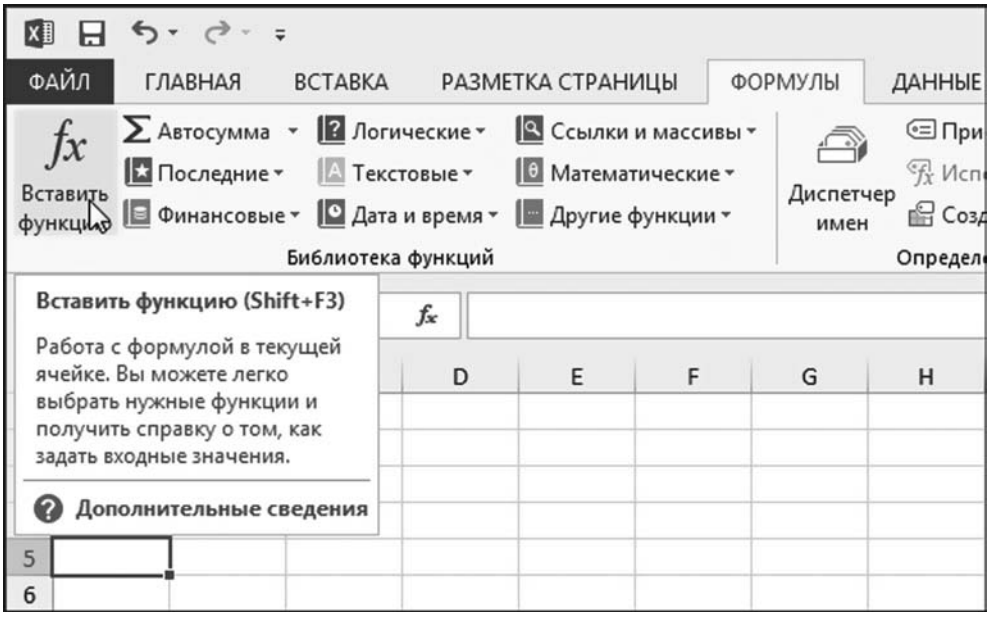


Рис. 6.18
Пиктограмма вставки функции в группе **Библиотека функций** на вкладке **Формулы**

Прочие пиктограммы соответствуют разным группам функций и представляют собой раскрывающиеся списки с функциями или подгруппами функций. На рисунке 6.19 показано содержимое списка, который раскрывается при щелчке на пиктограмме **Логические** (список логических функций).

Выбрав функцию в списке, получаем возможность указать ее аргументы в окне **Аргументы функции**, которое откроется само собой сразу после сделанного выбора.

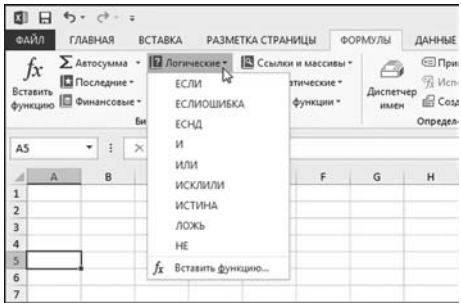


Рис. 6.19
Содержимое списка пиктограммы **Логические**

На заметку

Помимо имен функций, в списке представлена позиция **Вставить функцию**. Выбор этой позиции приводит к открытию окна **Вставка функции**.

ИСПОЛЬЗОВАНИЕ ИМЕН

— Господа! Вот это и есть Великий Магистр!
А это его спутница, Мария Ивановна.
— Мария Ивановна? Хм, а ты говорила
у итальянцев имена трудные!

Из к/ф «Формула любви»

Достаточно удобная и полезная характеристика Excel связана с возможностью присваивать уникальные имена ячейкам, диапазонам ячеек и формулам. Начнем с самых простых приемов — с именования ячеек и диапазонов. Разумеется, существует несколько способов добиться желаемого результата. Мы остановимся на основных, наиболее интересных.

Поскольку отдельная ячейка является частным случаем диапазона ячеек, то именно на диапазоне сосредоточим свое внимание. Если коротко, то мы можем присвоить диапазону имя. Поступаем так: выделяем диапазон ячеек A1:D4. При этом в поле имен отображается адрес левой верхней ячейки диапазона — в данном случае это ячейка A1. Выделяем двойным щелчком содержимое поля имен (рис. 6.20).

Рис. 6.20
Выделяем диапазон ячеек для присваивания имени

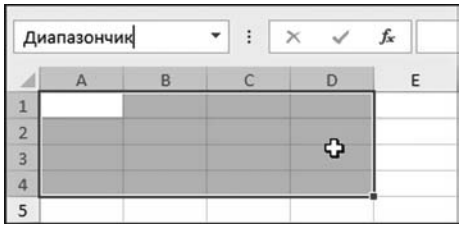


Рис. 6.21
Диапазону присвоено имя

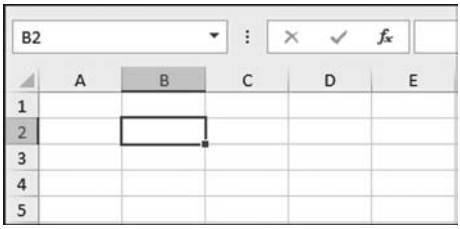


Рис. 6.22
Выделена внутренняя ячейка именованного диапазона

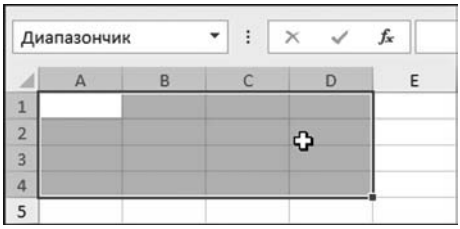


Рис. 6.23
Выделен именованный диапазон ячеек

Рис. 6.24
Выбор имени диапазона в списке имен

Затем вместо адреса A1 вводим имя диапазона — у нас он называется Диапазончик (см. рис. 6.21).

Имя диапазона вводится в поле имен, и затем нажимается клавиша <Enter>. После этого при выделении диапазона ячеек A1:D4 в поле имен будет отображаться имя диапазона. Причем должен быть выделен в точности именованный диапазон. Например, на рисунке 6.22 показана ситуация, когда в документе выделена ячейка B2, которая является внутренней ячейкой именованного диапазона.

Несложно убедиться, что в поле имен отображается адрес ячейки. Совсем иначе обстоят дела, когда выделен диапазон A1:D4 (рис. 6.23).

Работа с именованными диапазонами и ячейками имеет несколько преимуществ. Одно маленькое преимущество связано с возможностью выделения диапазона/ячейки путем выбора соответствующего имени в раскрывающемся списке поля имен. На рисунке 6.24 проиллюстрирован процесс выбора имени диапазона из раскрывающегося списка.

Второе большое преимущество применения именованных диапазонов/ячеек связано с возможностью создания простых и элегантных формул с именованными ссылками (для диапазонов и ячеек).

Продолжим наши эксперименты и заполним диапазон ячеек A1:D4 целыми числами — какими именно, не имеет значения (рис. 6.25).

В ячейке A5 вычисляем сумму чисел, записанных в диапазон ячеек A1:D4. Для этого используем функцию СУММ(), аргументом которой указывается диапазон ячеек, по которому вычисляется сумма. Вообще, если бы нас не интересовали именованные диапазоны, соответствующая формула имела бы вид =СУММ(A1:D4). Но поскольку диапазон A1:D4 имеет имя Диапазончик, мы можем использовать формулу =СУММ(Диапазончик). Вводить формулу тоже можно по-разному: например, при вводе аргумента щелкнуть на пиктограмме **Использовать в формуле** в группе **Определенные имена** на вкладке **Формулы** ленты (рис. 6.25). Можно просто выделить мышью диапазон ячеек, или, в крайнем случае, ввести имя диапазона с клавиатуры. В любом случае получаем результат, показанный на рисунке 6.26.

На заметку

Если воспользоваться командой **Вставить имена** из раскрывающегося списка пиктограммы **Использовать в формуле**, откроется диалоговое окно со списком имен. Имя можно выбрать в этом списке.

На рисунке 6.27 выделена ячейка с формулой, содержащей имя диапазона. В ячейке отображается результат вычислений по формуле, а в строке формул видим формулу со ссылкой на диапазон ячеек — имя этого диапазона.

Более широкие возможности по манипулированию со всевозможными именами реализуются через пиктограммы группы **Определенные имена** вкладки **Формулы**. Центральное место группы занимает пиктограмма **Диспетчер имен** (рис. 6.28).

Щелчок на этой пиктограмме открывает окно настроек **Диспетчер имен**, представленное на рисунке 6.29.

Это довольно занимательное окно и с его помощью решаются многие полезные задачи. В первую очередь сто́ит кратко остановиться на основных функциональных элементах окна. Оно содержит несколько кнопок, поле ввода (называется **Диапазон**) и большую белую область в центре окна со списком используемых имен (отображаются имена в открытой на данный момент рабочей книге). В конкретном случае используется всего одно имя Диапазончик.

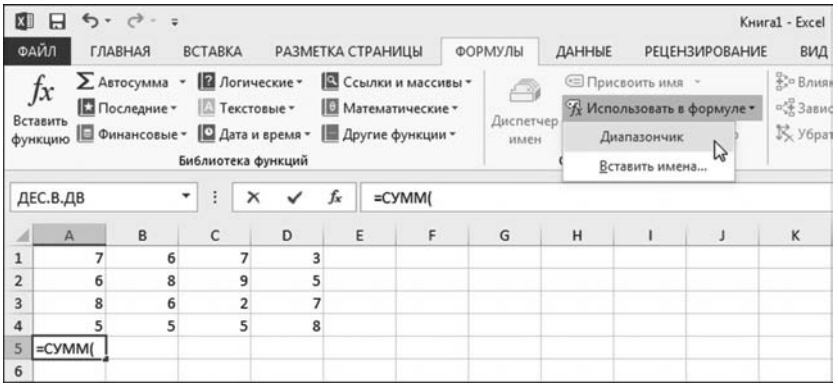


Рис. 6.25
Именованный диапазон заполнен числами, а в документ вводится функция, содержащая ссылку на именованный диапазон

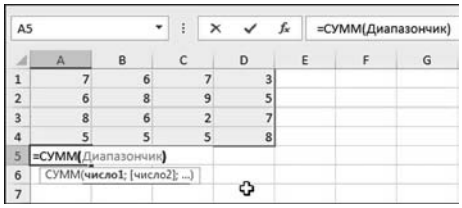


Рис. 6.26
Формула содержит ссылку на именованный диапазон

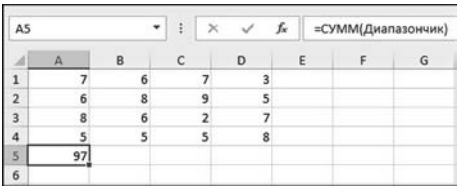


Рис. 6.27
Результат использования формулы со ссылкой на именованный диапазон

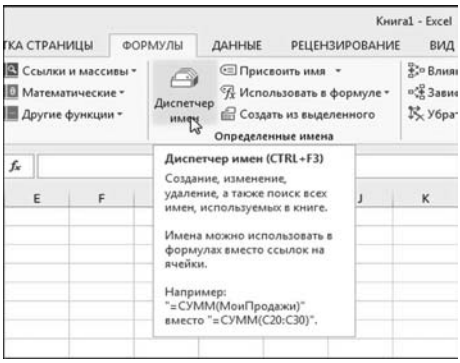


Рис. 6.28
Пиктограмма **Диспетчер имен** в группе **Определенные имена**

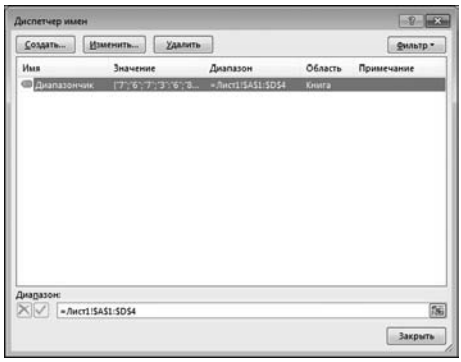


Рис. 6.29
Окно **Диспетчер имен**

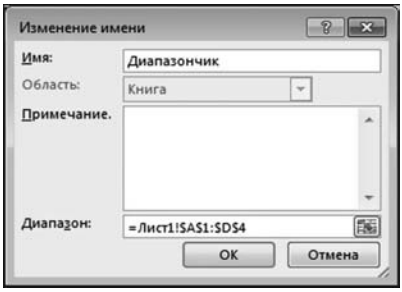


Рис. 6.30
В окне **Изменение имени** можно изменить параметры имени

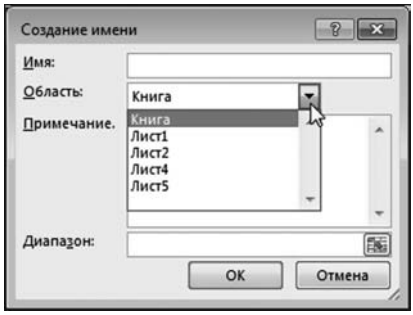


Рис. 6.31
Окно создания имени

На заметку

Информация в списке разбита на четыре группы/столбца. В столбце **Имя** отображается самая важная информация — имя, которое используется в рабочей книге. В столбце **Значение** отображается (нередко в сокращенном виде) значение именованного диапазона. Адрес именованного диапазона отображается в столбце **Диапазон**. Также есть столбцы **Область** и **Примечание**, в которых отображается соответственно уровень доступности имени (в рассматриваемом случае имя доступно в любом листе книги) и примечание к имени (если оно есть).

Есть несколько базовых операций, которые, так или иначе, приходится выполнять с именами. Это создание, удаление и редактирование имени. Для удаления или редактирования параметров имени необходимо выделить это имя в списке. Имя удаляется с помощью кнопки **Удалить**. Щелкнув на кнопке **Изменить**, можем поменять параметры имени. Правки вносятся с помощью диалогового окна **Изменение имени**, которое открывается после щелчка на кнопке **Изменить** (рис. 6.30).

Фактически, в этом окне можно изменить имя, именованный диапазон, добавить примечание и т. д.

На заметку

Окно **Изменить имя** также открывается двойным щелчком мыши на имени в списке имен окна **Диспетчер имен**. Внести изменение в адрес именованного диапазона можем непосредственно в поле **Диапазон** окна **Диспетчер имен**.

Чтобы создать новое имя в окне **Диспетчер имен** щелкаем кнопку **Создать**. Открывается окно **Создание имени**, похожее внешне на окно **Изменение имени** (рис. 6.31).

Окно содержит те же управляющие элементы, что и окно изменения параметров имени. Здесь сто́ит обратить внимание на раскрывающийся список **Область**, в котором задается область доступности создаваемого имени. Само имя указывается в поле **Имя**, а именуемый диапазон — в поле **Диапазон**. При этом диапазон можно выделить в рабочем документе с помощью мыши.

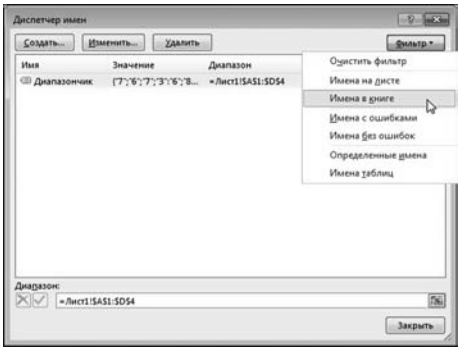


Рис. 6.32
Пиктограмма-список **Фильтр** в окне **Диспетчер имен** позволяет определить категорию имен для отображения

Пиктограмма с раскрывающимся списком **Фильтр** позволяет задать критерий (или фильтр) для отображения только тех имен, которые соответствуют некоторому шаблону. Что это за критерии, легко представить по содержимому списка на рисунке 6.32.

Прочие пиктограммы в группе **Определенные имена**, как несложно догадаться, также предназначены для работы с именованными диапазонами/ячейками (и формулами). Например, если в списке команд пиктограммы **Присвоить имя** щелкнуть на одноименной команде (рис. 6.33), откроется диалоговое окно **Создание имени** (рис. 6.31), в котором задаются параметры именуемого диапазона.

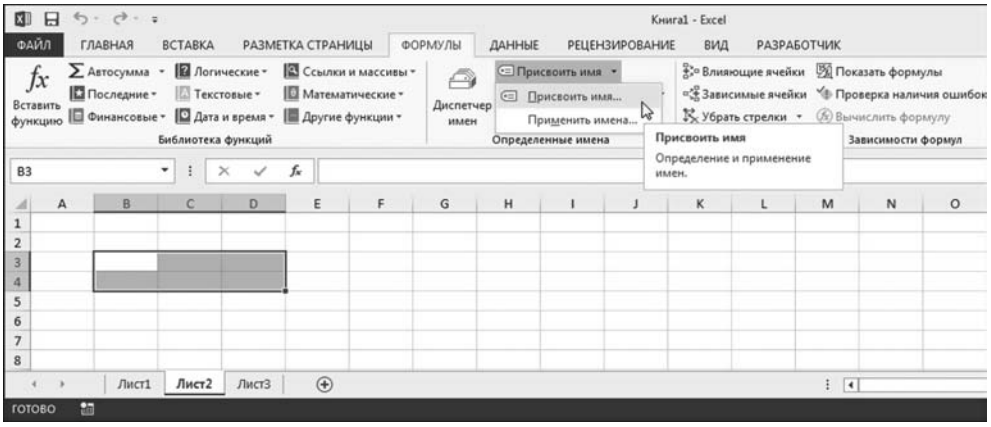


Рис. 6.33
Присваивание имени диапазону ячеек

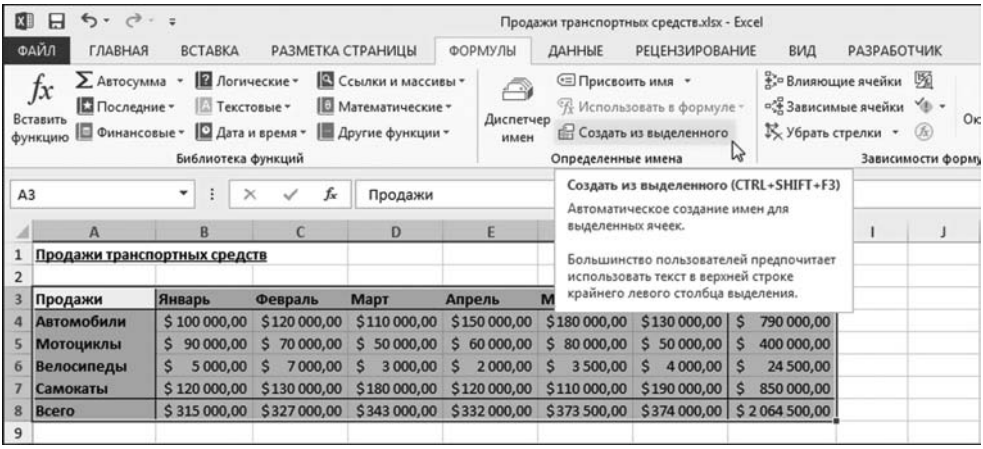


Рис. 6.34
Создание имен на основе данных таблицы

Еще один удобный способ создания именованных диапазонов/ячеек доступен при работе с таблицами.

На заметку

Здесь под таблицей подразумевается любой диапазон ячеек с данными, который имеет текстовую строку и/или столбец заголовков. Кроме того, в Excel существует специальная структура, которая тоже называется таблицей. Для ее создания необходимо выполнить некоторые дополнительные действия.

Нам нужен диапазон ячеек, окруженный (хотя бы с одной стороны) текстовыми заголовками. Эти заголовки можно использовать в качестве имен для ячеек. Для иллюстрации обратимся к документу на рисунке 6.34. В этом документе диапазон ячеек A3:H8 содержит некоторую условную информацию об уровне продаж транспортных средств (по месяцам за полгода). Кроме фактических данных, диапазон содержит и текстовые заголовки (ячейки A3:A8 и B3:H3). Мы выделяем диапазон ячеек A3:H8 и щелкаем пиктограмму **Создать из выделенного** в группе **Определенные имена** вкладки **Формулы** (рис. 6.34).

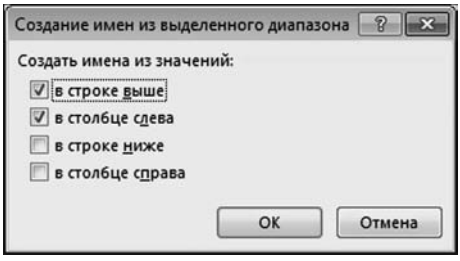


Рис. 6.35
Диалоговое окно для создания имен на основе данных выделенного диапазона

В результате открывается диалоговое окно **Создание имени из выделенного диапазона**, представленное на рисунке 6.35.

В окне имеется четыре опции, которые позволяют задать способ именования ячеек в выделенном диапазоне. Обычно Excel предлагает «свой выбор», который нередко является оптимальным. В данном случае установлены флажки опций **в строке выше** и **в столбце слева**. Это означает, что для имено-

вания ячеек будут использоваться текстовая информация в первой строке таблицы и первом ее столбце. А именно, названия столбцов именуют диапазоны с фактическими данными таблицы в соответствующих столбцах, а названия строк таблицы именуют диапазоны с фактическими данными в соответствующих строках. Например, текстовое значение Январь в ячейке B3 служит именем для диапазона ячеек B4:B8. Более того, при вводе формул соответствующие подсказки появляются в контекстном меню. На рисунке 6.36 показана ситуация, когда мы в ячейке B10 пытаемся вычислить определенную сумму и аргументом функции СУММ() вводится имя Январь.

Результат вычисления формулы =СУММ(Январь) показан в документе на рисунке 6.37.

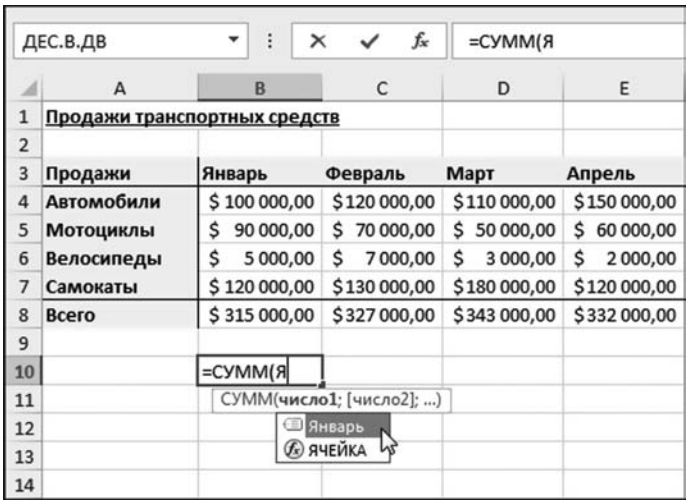


Рис. 6.36
В формуле можно использовать ссылку на столбец таблицы

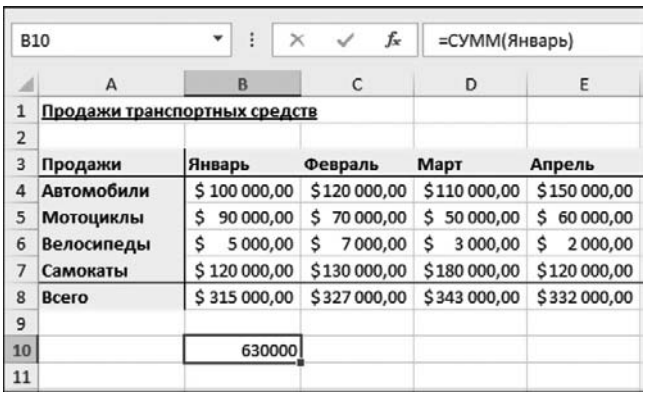


Рис. 6.37
Текстовое поле таблицы использовано в формуле

На заметку

Числовое значение в ячейке B10, вычисленное по формуле =СУММ(Январь) ровно в два раза больше значения в ячейке B8, которое вычисляется по формуле =СУММ(B4:B7). Объяснение простое и очевидное — диапазон ячеек Январь включает диапазон B4:B7 и ячейку B8, поэтому в результате вычисляется двойная сумма по диапазону ячеек B4:B7. В этом отношении было бы разумнее при именовании ячеек таблицы на начальном этапе выделить не диапазон ячеек A3:H8, а диапазон A3:G7. Но это, как говорится, дело вкуса.

Убедиться в том, что имя Январь является ссылкой на диапазон ячеек B4:B8 можно по-разному. Например, выделяем (делаем активной) ячейку B10 с формулой, содержащей имя Январь, и затем щелкаем мышью в строке формул. В рабочем документе при этом выделяются диапазоны ячеек, на которые в формуле есть ссылки. В данном случае такой диапазон один (рис. 6.38).

Абсолютно такая же ситуация со строками таблицы. Текстовые значения диапазона ячеек A4:A8 именуют диапазоны соответствующих строк таблицы. На рисунке 6.39 выделен диапазон ячеек B6:H6. При этом в поле имен отображается название этого диапазона Велосипеды. Именно это текстовое значение содержится в ячейке A6, которая находится слева от диапазона B6:H6.

На заметку

Далеко не каждое текстовое значение может использоваться в качестве имени. Например, имя должно состоять из одного слова. Если текст в поле таблицы состоит из нескольких слов, то при создании имени на основе такого текстового значения пробелы заменяются символами подчеркивания.

Ячейка A3 с (текстовым значением Продажи), находящаяся на пересечении строк и столбца названий полей таблицы, является особенной. Она (точнее, ее текстовое значение) служит именем для диапазона B4:H8, т. е. диапазона ячеек таблицы со всеми фактическими данными. На рисунке 6.40 выделен диапазон ячеек B4:H8, и при этом в поле имен отображается название диапазона Продажи.

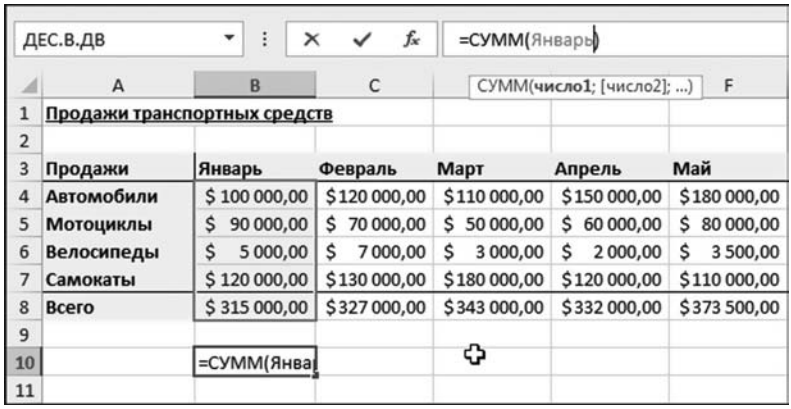
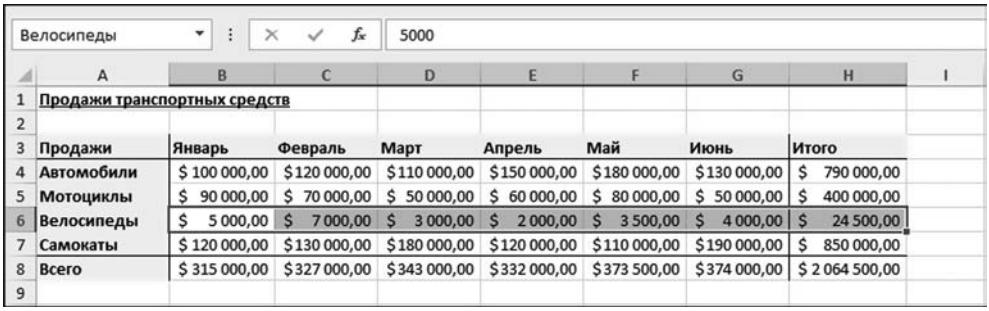


Рис. 6.38
Название столбца таблицы именует диапазон ячеек с данными в этом столбце

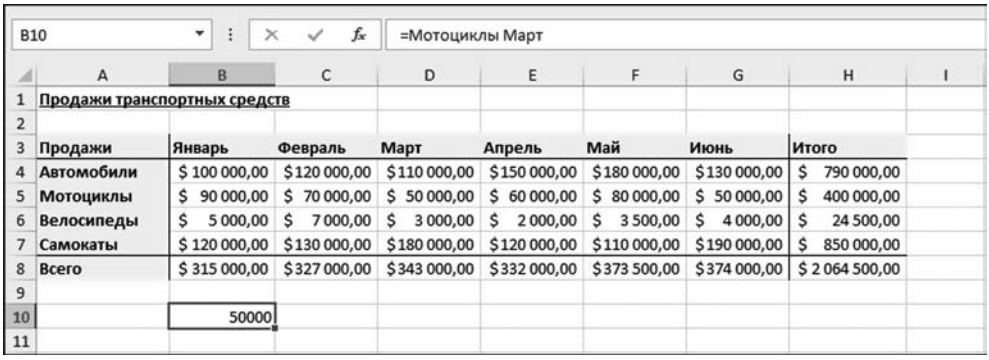


	A	B	C	D	E	F	G	H	I
1	Продажи транспортных средств								
2									
3	Продажи	Январь	Февраль	Март	Апрель	Май	Июнь	Итого	
4	Автомобили	$ 100 000,00	$120 000,00	$110 000,00	$150 000,00	$180 000,00	$130 000,00	$ 790 000,00	
5	Мотоциклы	$ 90 000,00	$ 70 000,00	$ 50 000,00	$ 60 000,00	$ 80 000,00	$ 50 000,00	$ 400 000,00	
6	Велосипеды	$ 5 000,00	$ 7 000,00	$ 3 000,00	$ 2 000,00	$ 3 500,00	$ 4 000,00	$ 24 500,00	
7	Самокаты	$ 120 000,00	$130 000,00	$180 000,00	$120 000,00	$110 000,00	$190 000,00	$ 850 000,00	
8	Всего	$ 315 000,00	$327 000,00	$343 000,00	$332 000,00	$373 500,00	$374 000,00	$ 2 064 500,00	
9									

Рис. 6.39
Имена строк таблицы задаются текстовыми ячейками первого столбца

	A	B	C	D	E	F	G	H	I
1	Продажи транспортных средств								
2									
3	Продажи	Январь	Февраль	Март	Апрель	Май	Июнь	Итого	
4	Автомобили	$ 100 000,00	$120 000,00	$110 000,00	$150 000,00	$180 000,00	$130 000,00	$ 790 000,00	
5	Мотоциклы	$ 90 000,00	$ 70 000,00	$ 50 000,00	$ 60 000,00	$ 80 000,00	$ 50 000,00	$ 400 000,00	
6	Велосипеды	$ 5 000,00	$ 7 000,00	$ 3 000,00	$ 2 000,00	$ 3 500,00	$ 4 000,00	$ 24 500,00	
7	Самокаты	$ 120 000,00	$130 000,00	$180 000,00	$120 000,00	$110 000,00	$190 000,00	$ 850 000,00	
8	Всего	$ 315 000,00	$327 000,00	$343 000,00	$332 000,00	$373 500,00	$374 000,00	$ 2 064 500,00	
9									

Рис. 6.40
Ячейка в левом верхнем углу таблицы именует диапазон данных таблицы



	A	B	C	D	E	F	G	H	I
1	Продажи транспортных средств								
2									
3	Продажи	Январь	Февраль	Март	Апрель	Май	Июнь	Итого	
4	Автомобили	$ 100 000,00	$120 000,00	$110 000,00	$150 000,00	$180 000,00	$130 000,00	$ 790 000,00	
5	Мотоциклы	$ 90 000,00	$ 70 000,00	$ 50 000,00	$ 60 000,00	$ 80 000,00	$ 50 000,00	$ 400 000,00	
6	Велосипеды	$ 5 000,00	$ 7 000,00	$ 3 000,00	$ 2 000,00	$ 3 500,00	$ 4 000,00	$ 24 500,00	
7	Самокаты	$ 120 000,00	$130 000,00	$180 000,00	$120 000,00	$110 000,00	$190 000,00	$ 850 000,00	
8	Всего	$ 315 000,00	$327 000,00	$343 000,00	$332 000,00	$373 500,00	$374 000,00	$ 2 064 500,00	
9									
10		50000							
11									

Рис. 6.41
Ссылка на ячейку содержит имена полей таблицы

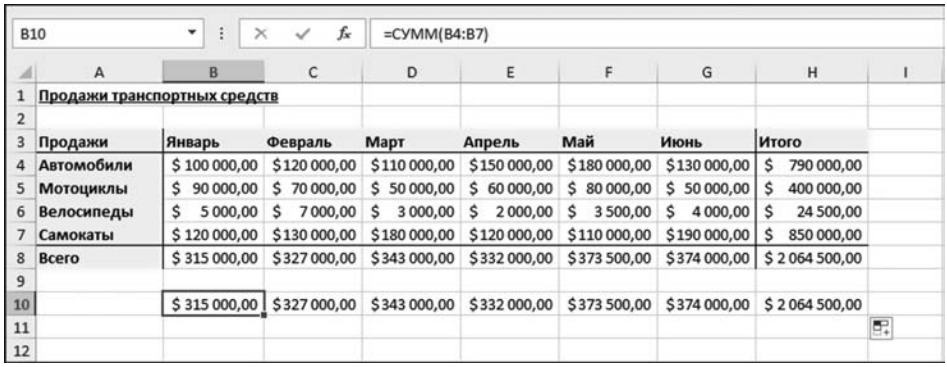
На заметку

Интересно то, что описанный выше способ именования ячеек таблицы может использоваться для ссылки на отдельные ячейки или группы ячеек внутри таблицы. Например, для того, чтобы получить значение ячейки, находящейся на пересечении строки Мотоциклы и столбца Март можем воспользоваться формулой =Мотоциклы Март (между словами пробел), как показано на рисунке 6.41.

С технической точки зрения в данном случае речь идет о пересечении диапазонов. Пробел как раз и является оператором пересечения. Результат синтаксической конструкции, состоящей из двух разделенных пробелом диапазонов, есть множество ячеек, которые входят в каждый из двух диапазонов.

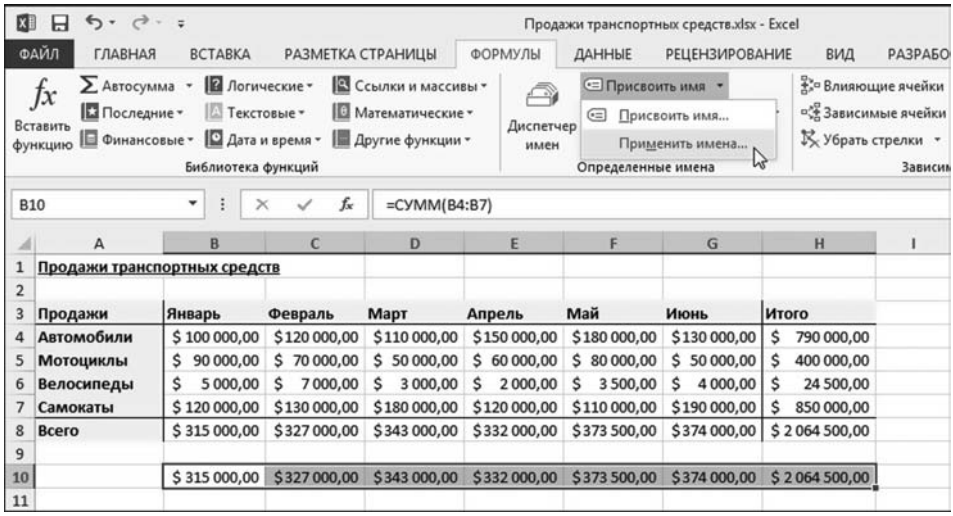
Взаимоотношения с именованными диапазонами не всегда такие безоблачные, как может показаться на первый взгляд. Проведем небольшое, но показательное исследование. В ячейку B10 вводим формулу =СУММ(B4:B7) и копируем эту формулу (например, методом автоматического заполнения) во все ячейки диапазона B10:H10 (рис. 6.42).

Ситуация банальная, за тем небольшим обстоятельством, что в формулах ячеек диапазона B10:H10 мы использовали обычные адреса, а не имена, как



	A	B	C	D	E	F	G	H	I
1	Продажи транспортных средств								
2									
3	Продажи	Январь	Февраль	Март	Апрель	Май	Июнь	Итого	
4	Автомобили	$ 100 000,00	$120 000,00	$110 000,00	$150 000,00	$180 000,00	$130 000,00	$ 790 000,00	
5	Мотоциклы	$ 90 000,00	$ 70 000,00	$ 50 000,00	$ 60 000,00	$ 80 000,00	$ 50 000,00	$ 400 000,00	
6	Велосипеды	$ 5 000,00	$ 7 000,00	$ 3 000,00	$ 2 000,00	$ 3 500,00	$ 4 000,00	$ 24 500,00	
7	Самокаты	$ 120 000,00	$130 000,00	$180 000,00	$120 000,00	$110 000,00	$190 000,00	$ 850 000,00	
8	Всего	$ 315 000,00	$327 000,00	$343 000,00	$332 000,00	$373 500,00	$374 000,00	$ 2 064 500,00	
9									
10		$ 315 000,00	$327 000,00	$343 000,00	$332 000,00	$373 500,00	$374 000,00	$ 2 064 500,00	
11									
12									

Рис. 6.42
Ссылки в формулах не содержат имен диапазонов (только адреса)



	A	B	C	D	E	F	G	H	I
1	Продажи транспортных средств								
2									
3	Продажи	Январь	Февраль	Март	Апрель	Май	Июнь	Итого	
4	Автомобили	$ 100 000,00	$120 000,00	$110 000,00	$150 000,00	$180 000,00	$130 000,00	$ 790 000,00	
5	Мотоциклы	$ 90 000,00	$ 70 000,00	$ 50 000,00	$ 60 000,00	$ 80 000,00	$ 50 000,00	$ 400 000,00	
6	Велосипеды	$ 5 000,00	$ 7 000,00	$ 3 000,00	$ 2 000,00	$ 3 500,00	$ 4 000,00	$ 24 500,00	
7	Самокаты	$ 120 000,00	$130 000,00	$180 000,00	$120 000,00	$110 000,00	$190 000,00	$ 850 000,00	
8	Всего	$ 315 000,00	$327 000,00	$343 000,00	$332 000,00	$373 500,00	$374 000,00	$ 2 064 500,00	
9									
10		$ 315 000,00	$327 000,00	$343 000,00	$332 000,00	$373 500,00	$374 000,00	$ 2 064 500,00	
11									

Рис. 6.43
Выделяем диапазон ячеек с формулами и в списке пиктограммы **Присвоить имя** щелкаем команду **Применить имена**

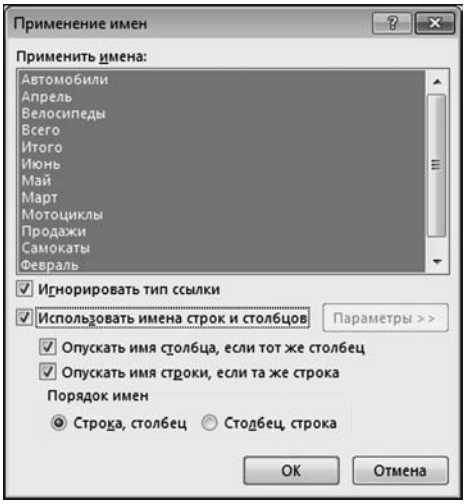


Рис. 6.44
Окно **Применение имен** содержит список имен,
на которые следует поменять адреса диапазонов в формулах

G10 =СУММ((Автомобили Июнь):(Самокаты Июнь))

	A	B	C	D	E	F	G	H	I
1	Продажи транспортных средств								
2									
3	Продажи	Январь	Февраль	Март	Апрель	Май	Июнь	Итого	
4	Автомобили	$ 100 000,00	$120 000,00	$110 000,00	$150 000,00	$180 000,00	$130 000,00	$ 790 000,00	
5	Мотоциклы	$ 90 000,00	$ 70 000,00	$ 50 000,00	$ 60 000,00	$ 80 000,00	$ 50 000,00	$ 400 000,00	
6	Велосипеды	$ 5 000,00	$ 7 000,00	$ 3 000,00	$ 2 000,00	$ 3 500,00	$ 4 000,00	$ 24 500,00	
7	Самокаты	$ 120 000,00	$130 000,00	$180 000,00	$120 000,00	$110 000,00	$190 000,00	$ 850 000,00	
8	Всего	$ 315 000,00	$327 000,00	$343 000,00	$332 000,00	$373 500,00	$374 000,00	$ 2 064 500,00	
9									
10		$ 315 000,00	$327 000,00	$343 000,00	$332 000,00	$373 500,00	$374 000,00	$ 2 064 500,00	
11									

Рис. 6.45
Результат замены адресов диапазонов на имена

можно было бы ожидать. Теперь мы хотим заменить адреса диапазонов именами диапазонов (разумеется, там, где это уместно). Альтернативой выполнению этой процедуры «в ручном режиме» является «автоматический» режим. А именно, мы выделяем тот диапазон ячеек B10:H10, в котором хотим поменять ссылки (адреса на имена), щелкаем пиктограмму **Присвоить имя** и выбираем команду **Применить имена** (рис. 6.43).

Открывается окно **Применение имен**, представленное на рисунке 6.44.

Окно содержит несколько элементов управления и список с именами диапазонов. Этот список позволяет выполнить многократное выделение. Выделенные имена будут использованы для замены адресов соответствующих диапазонов, если такие содержатся в ссылках в формулах. Мы не скупимся и используем весь арсенал имен. Результат показан на рисунке 6.45.

Для наглядности в этом документе выделена ячейка G10, которая до этого содержала формулу =СУММ(G4:G7). Теперь эта формула выглядит как =СУММ((Автомобили Июнь):(Самокаты Июнь)), в чем можно убедиться, взглянув на рисунок 6.45.

На заметку

Понять формулу =СУММ((Автомобили Июнь):(Самокаты Июнь)) несложно. Аргументом функции СУММ(), как и раньше, является диапазон ячеек G4:G7, только теперь ссылка на диапазон выглядит как (Автомобили Июнь):(Самокаты Июнь). Со ссылкой все тоже просто: инструкция Автомобили Июнь означает ячейку G4, а инструкция Самокаты Июнь означает ячейку G7. Между ними — двоеточие.

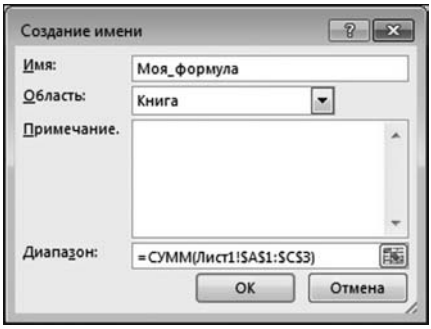


Рис. 6.46
Присваивание имени формуле

К сожалению, если мы в какой-то момент удалим из рабочей книги задействованные в формулах имена, обратное преобразование (имен диапазонов на адреса диапазонов) в формулах не выполняется. Но даже несмотря на это, работа с именованными диапазонами/ячейками обычно сложностей не вызывает. Вместе с тем, намного более эффектно выглядит именование формул. Общий подход базируется на том, что некоторой формуле можно присвоить имя. Затем это имя можно использовать в других формулах рабочей книги, со всеми вытекающими из этого последствиями.

Для создания именованной формулы на вкладке **Формулы** в группе **Определенные имена** в списке пиктограммы **Присвоить имя** щелкаем одноименную команду. Открывается уже знакомое нам окно **Создание имени**. Особенность ситуации в том, что в поле **Диапазон** вводится формула (рис. 6.46).

В данном случае мы в поле **Имя** указываем имя формулы Моя_формула. В поле **Диапазон** вводим формулу =СУММ(Лист1!$A$1:$C$3). Формула означает вычисление (с помощью встроенной функции СУММ()) суммы по диапазону ячеек A1:C3. В данном случае важно, что ссылка на диапазон ячеек абсолютная.

На заметку

Ввести диапазон ячеек лучше всего, выделив его мышью в рабочем листе. Ссылки при этом по умолчанию используются абсолютные. Также автоматически указывается рабочий лист.

После подтверждения именованная формула готова к использованию. На рисунке 6.47 показан фрагмент документа, в котором заполнен числовыми значениями диапазон ячеек A1:C3, а в ячейку D4 введена формула =Моя_формула.

В результате в ячейке D4 отображается сумма значений ячеек диапазона A1:C3.

Именованные формулы можно использовать и в более сложных выражениях — собственно, для этого именованные формулы обычно и создаются.

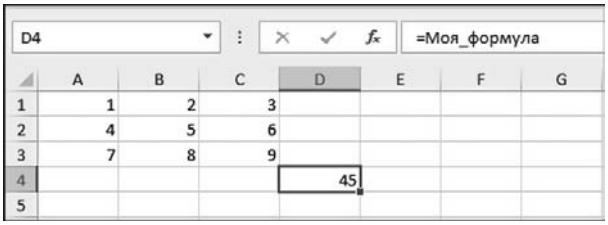


Рис. 6.47
Результат применения именованной формулы

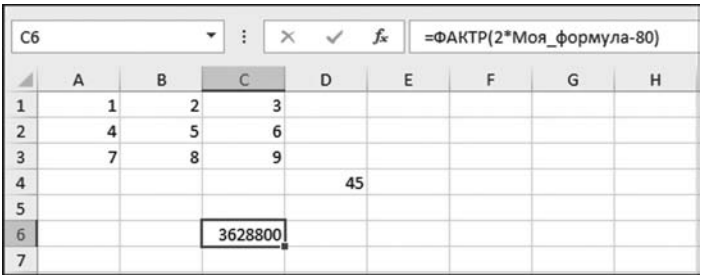


Рис. 6.48
Использование именованной формулы в выражении

Например, в документе на рисунке 6.48 значение ячейки C6 вычисляется по формуле =ФАКТР(2*Моя_формула-80), в которой, в свою очередь, использована именованная формула Моя_формула.

Независимо от того, где и в каком выражении использовано имя Моя_формула, значение для этого имени вычисляется как сумма значений ячеек диапазона A1:C3 (рабочего листа Лист1). Это следствие того, что в созданной нами именованной формуле ссылки абсолютные. Если бы там были относительные ссылки, ситуация была бы совершенно иной. Некоторые случаи мы рассмотрим далее.

На заметку

Многим, наверное, интересно, что это за формула =ФАКТР(2*Моя_формула-80)? Ее основу составляет функция ФАКТР() для вычисления факториала числа. Аргумент этой функции вычисляется как умноженная на два сумма ячеек диапазона A1:C3 (имя Моя_формула), из которой вычитается значение 80. В данном случае это 10! = 3 628 800.

На заметку

Учитывая неуклонно возрастающую популярность математики и других точных наук, нелишним будет напомнить, что факториалом числа по определению называется произведение натуральных чисел от 1 до этого числа (например, $10! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 \cdot 8 \cdot 9 \cdot 10 = 3\,628\,800$).

Если создаваемая именованная формула, как предполагается, будет содержать относительные ссылки, то далеко не последнюю роль при определении такой формулы играет то, какая ячейка была активной при вызове окна

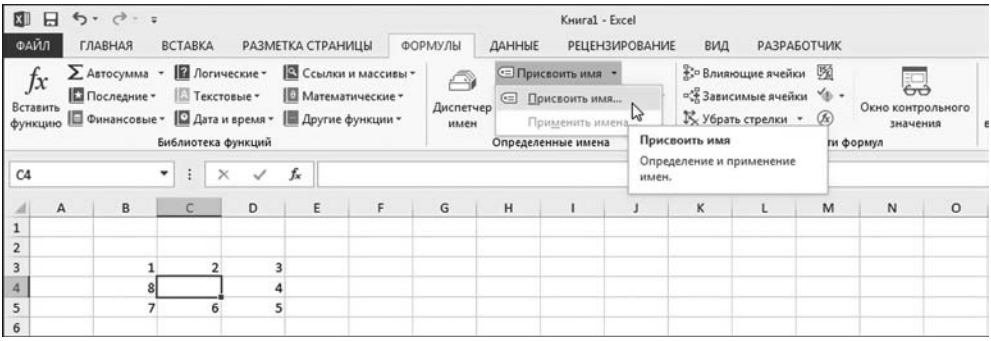


Рис. 6.49
Для определения именованной формулы выделена ячейка C4 внутри диапазона B3:D5

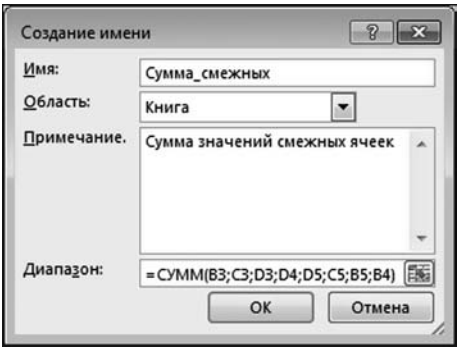


Рис. 6.50
Определение именованной формулы в окне **Создание имени**

создания имени. Обратимся к примеру. А именно, создадим именованную формулу, с помощью которой будет вычисляться сумма значений ячеек, смежных с той, куда введена эта формула. Для этого выполняем короткую последовательность простых действий. Для начала заполняем ячейки диапазона B3:D5 числами (кроме внутренней ячейки диапазона C4), выделяем ячейку C4 и щелкаем пиктограмму **Присвоить имя** (рис. 6.49).

В результате откроется окно **Создание имени**. В этом окне в поле **Имя** вводим название именованной формулы (формулу назовем Сумма_смежных). В поле **Диапазон** вводим формулу =СУММ(B3;C3;D3;D4;D5;C5;B5;B4), как показано на рисунке 6.50.

На заметку

Если мы хотим (а мы этого действительно хотим), чтобы при вводе имени формулы в рабочем документе отображалась подсказка, в поле **Примечание** добавляем текст подсказки.

Когда все готово, подтверждаем создание именованной формулы щелчком кнопки **ОК**. После этого именованную формулу применяем на деле: вводим в ячейку C4 выражение =Сумма_смежных (рис. 6.51).

При вводе формулы, благодаря заранее предпринятым нами действиям (имеется в виду подсказка), появляется элегантная подсказка о назначении именованной формулы. Результат наших усилий представлен в документе на рисунке 6.52.

Не будет сюрпризом, что все вычисления выполнены корректно — в ячейке C4 отображается значение суммы ячеек, окружающих ячейку C4. Можем

ввести формулу =Сумма_смежных в другую ячейку. На рисунке 6.53 эта формула введена в ячейку C2.

Теперь сумма вычисляется по ячейкам, окружающим ячейку C2. Чтобы по достоинству оценить результат, следует учесть, что по умолчанию пустые ячейки при вычислениях интерпретируются как такие, что содержат нули. Поэтому фактический вклад в сумму дают только ячейки с ненулевыми значениями (т. е. ячейки диапазона C3:D3).

Не будет большой проблемы, если ввести формулу =Сумма_смежных в угловую ячейку A1 (рис. 6.54). Особенность этой ячейки как раз в том, что она угловая, и у нее есть далеко не все соседи. Тем не менее, особых проблем здесь тоже нет (во всяком случае, внешне они не проявляются — а на самом деле ссылки на недостающие ячейки сверху и слева циклически «перебрасываются» соответственно на самую нижнюю ячейку столбца и самую правую ячейку строки).

На заметку

Несмотря на то, что при вводе именованной формулы в поле **Диапазон** окна **Создание имени** (рис. 6.50) мы имя рабочего листа в ссылках не указывали, Excel автоматически добавляет ссылку на рабочий лист в адресах ячеек. В последнем несложно убедиться, если открыть окно **Диспетчер имен** (на рисунке 6.55 обратите внимание на содержимое поля **Диапазон**).

Также следует быть предельно аккуратным при наборе формулы в поле **Диапазон** — нажатие клавиш со стрелками, например, приводит к выделению ячеек в рабочем листе и вставке соответствующей ссылки в формулу. Так что при работе с этим полем надежней все же пользоваться мышью.

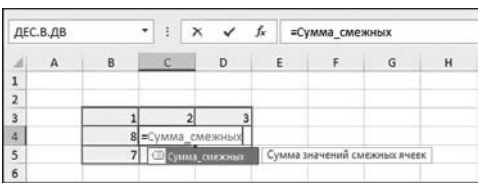


Рис. 6.51
Ввод в центральную ячейку диапазона вновь созданной именованной формулы

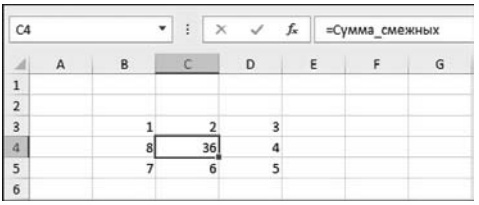


Рис. 6.52
Результат вычисления с помощью именованной формулы

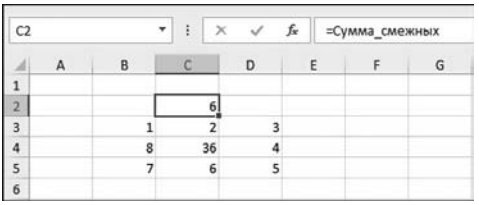


Рис. 6.53
Именованная формула введена в другую ячейку

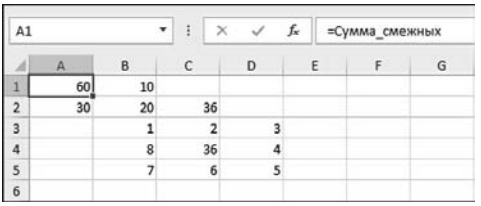


Рис. 6.54
Именованная формула в левом верхнем углу рабочего листа

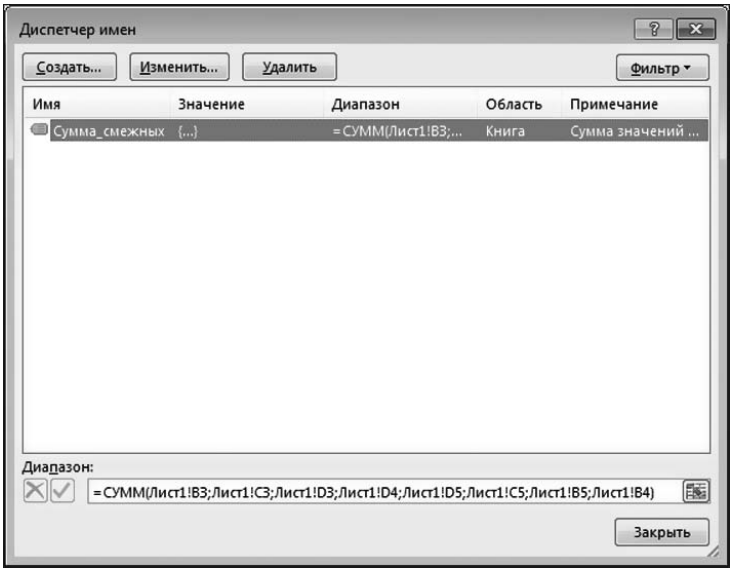


Рис. 6.55
В формулу автоматически добавляется ссылка на рабочий лист

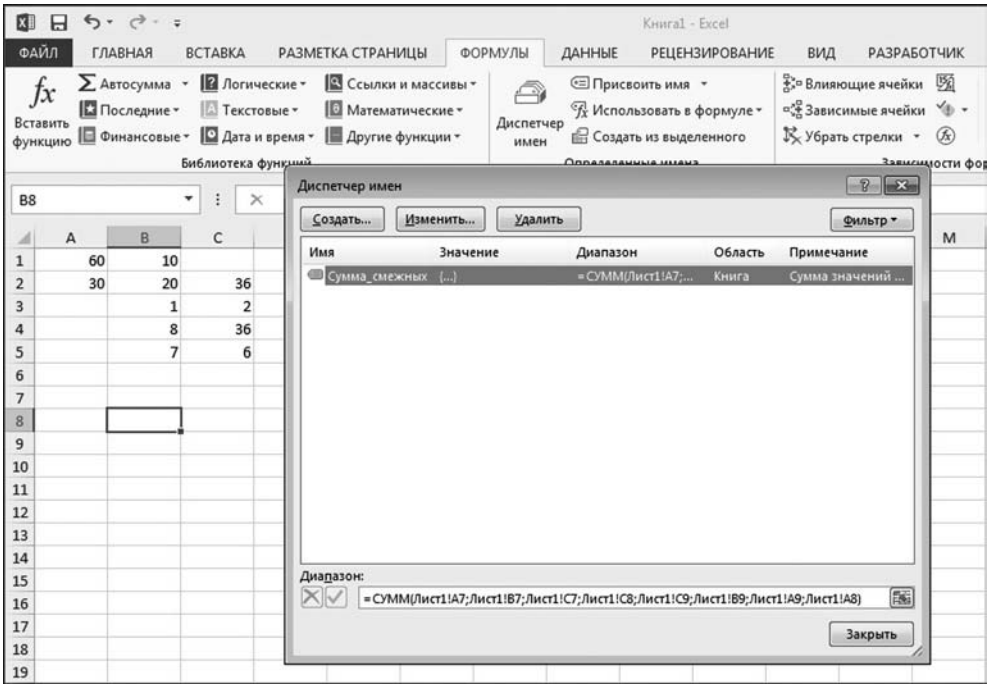


Рис. 6.56
Отображаемое в поле **Диапазон** выражение для именованной формулы с относительными ссылками зависит от активной ячейки

Вместе с тем, ситуация не такая однозначная, как может показаться на первый взгляд. Связана она с тем, что при использовании именованных формул с относительными ссылками действует правило их преобразования — только это происходит не так очевидно, как при копировании обычных ссылок. Хотя общий принцип тот же — ссылки изменяются так, чтобы относительное положение ячейки с формулой и ячеек, на которые есть ссылки, не изменилось. Практическим следствием данного фундаментального подхода является то, что в окне **Диспетчер имен** в поле **Диапазон** выражение для именованной формулы с относительными ссылками зависит от адреса активной (на момент открытия окна) ячейки в рабочем листе. На рисунке 6.56 показано, как будет выглядеть выражение для именованной формулы Сумма_смежных при активной ячейке B8.

На заметку

Что касается примера с угловой ячейкой A1, в которую вводилась формула =Сумма_смежных, то при преобразовании относительных ссылок «недостающие» адреса слева и сверху от ячейки заменяются циклически адресами ячеек соответственно справа и снизу от границ рабочего листа.

На заметку

Нередко процедуру создания имен используют для определения в рабочем документе констант. По большому счету, константа — это формула, значение которой вычисляется безотносительно к тому, в какую ячейку эта формула введена. Достаточно популярны текстовые и числовые константы. Например, для создания текстовой константы, выполняем настройки в окне **Создание имени**, как показано на рисунке 6.57.

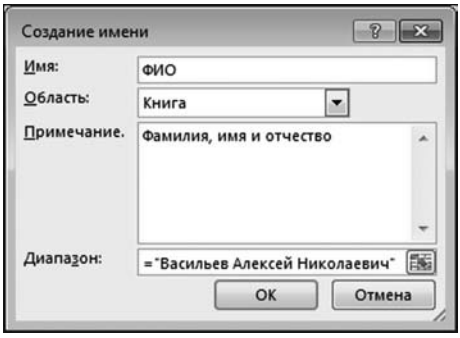


Рис. 6.57
Создание текстовой константы

В поле **Имя** указываем название именованной константы ФИО, в поле **Примечание** вводим поясняющий текст Фамилия, имя и отчество, а в поле **Диапазон** — самое главное — формулу ="Васильев Алексей Николаевич". Текстовое значение заключено в двойные кавычки. Константа готова.

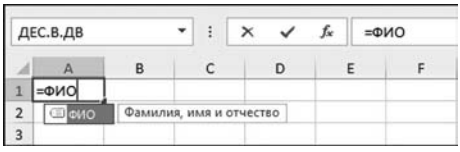


Рис. 6.58
В формуле использована текстовая именованная константа

Для проверки результатов, в ячейку A1 вводим формулу =ФИО. При этом появляется подсказка, что дает неплохие шансы на успех (рис. 6.58).

Окончательный результат представлен на рисунке 6.59.

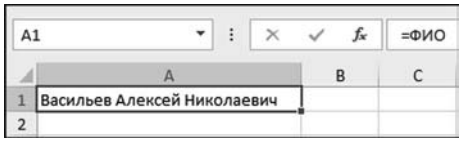


Рис. 6.59
Результат использования в формуле текстовой константы

Для удобства восприятия увеличена ширина ячеек первого столбца.

ОТСЛЕЖИВАНИЕ ОШИБОК

Виновные в этом безобразии будут наказаны.
Ферштейн?

Из к/ф «Старый знакомый»

Обычно знакомство с вычислениями в Excel начинается со знакомства с ошибками, которые при этом с неизбежностью возникают. Надо отдать должное разработчикам Excel — приложение в случае любых недоразумений пытается «намекнуть» пользователю, что возможно он где-то неправ. Для начала неплохо разобраться в этих самых «намеках». Они обычно появляются в виде диалоговых окон с достаточно подробным описанием сложившейся ситуации. Также сообщения могут появляться непосредственно в рабочем документе. В этом случае они не такие информативные, хотя полезную информацию можно почерпнуть и из них. Речь идет о сообщениях об ошибках, которые отображаются в ячейках с формулами в том случае, если вычисления некорректны. Сообщения, которые могут появляться при возникновении ошибок в ячейках документа Excel, представлены (и кратко описаны) в таблице 6.1.

Иногда по сообщению об ошибке в ячейке можно догадаться, в чем причина его появления. Но, откровенно говоря, одного сообщения об ошибке бывает маловато — особенно если таких сообщений много.

На заметку

Обычно в документе ошибок или нет вовсе, или их много. Здесь все как в жизни — стоит один раз ошибиться и начинается цепная реакция! Одна ошибка тянет за собой другую, и они множатся как кролики.

Таблица 6.1

Сообщения об ошибках

Сообщение об ошибке	Описание
#####	Такая конструкция отображается в ячейке, если ширина ячейки недостаточна для отображения фактического значения ячейки. Такая же «сетка» может появиться в ячейке, если в результате вычислений получено отрицательное значение для даты
#ЗНАЧ!	Это сообщение появляется в тех случаях, когда в вычислениях использован недопустимый операнд — например, неверный аргумент функции
#ДЕЛ/0!	Сообщение появляется в ячейке, если предпринята попытка деления на ноль. Стоит напомнить, что по умолчанию при вычислениях отсутствие значения в ячейке означает наличие в ней нулевого значения
#ИМЯ?	С помощью этого сообщения Excel дает понять пользователю, что в формуле есть имена, которые непонятно что обозначают
#Н/Д	Если значение вычислить невозможно (значение недоступно) появляется такое сообщение
#ССЫЛКА!	Сообщение появляется, если в выражении использована недопустимая ссылка
#ЧИСЛО!	Сообщение появляется в случае, если в вычислениях использовано неверное числовое значение
#ПУСТО!	Сообщение появляется, если результатом пересечения двух областей (диапазонов) является пустое множество (т. е. диапазоны не пересекаются)

Таблица 6.2

Данные ячеек рабочего документа

Ячейка	Содержимое ячейки	Описание
A1	=ПИ()	Вычисление значения числа π ≈ 3,14159265 с помощью встроенной функции ПИ()
C5	=SIN(A1/E1)	Вычисление синуса с помощью встроенной функции SIN(). Аргумент синуса вычисляется на основе значений ячеек A1 и E1
C6	=COS(ПИ()/E2)	Вычисление косинуса с помощью встроенной функции COS(). Аргумент функции вычисляется на основе встроенной функции ПИ() и значения ячейки E2
E1	2	Ячейка с числовым значением
E2	3	Ячейка с числовым значением
E3	=E1*E2	Значение ячейки вычисляется как произведение значений ячеек E1 и E2
F8	=(E2*C5+E1*C6)/E3	Значение ячейки вычисляется на основе значений ячеек E1:E3 и C5:C6

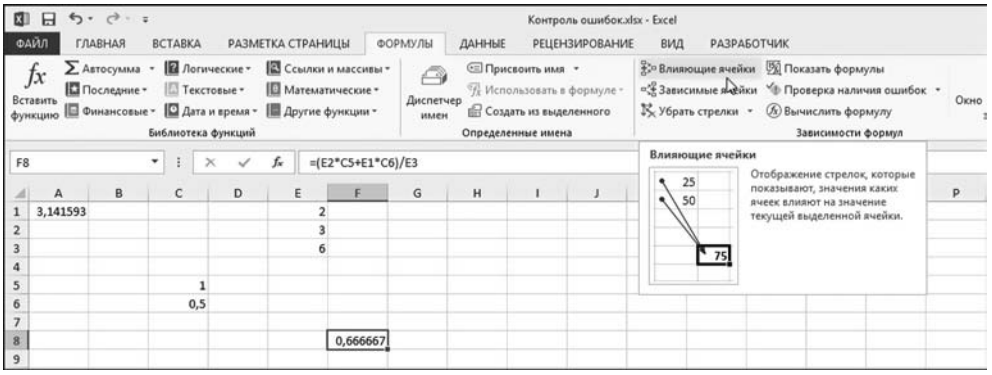


Рис. 6.60
Переходим в режим отображения влияющих ячеек

Устранить проблему помогает небольшое детективное исследование, базирующееся на выявлении связей (функциональных) между ячейками в рабочем документе. Неплохим подспорьем в этом станут пиктограммы группы **Зависимости формул** вкладки **Формулы**. Для большей конкретики рассмотрим иллюстративный документ с несколькими формулами. Содержимое ячеек этого документа описано в таблице 6.2.

Таким образом, ячеек не много, но они между собой связаны системой не очень сложных соотношений. Наша задача состоит в том, чтобы вывести эти соотношения, что называется, на чистую воду. Для этого выделяем анализируемую на наличие функциональных соотношений ячейку и щелкаем пиктограмму **Влияющие ячейки** на вкладе **Формулы** в группе **Зависимости формул**. В документе на рисунке 6.60 выделена ячейка F8, и именно для этой ячейки мы будем искать влияющие ячейки, т. е. ячейки, от которых зависит значение ячейки F8.

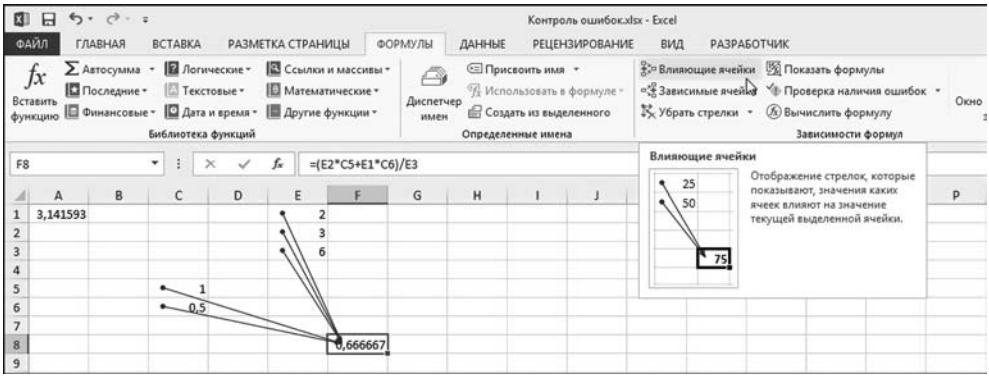


Рис. 6.61
Результат перехода в режим влияющих ячеек

После щелчка на пиктограмме **Влияющие ячейки** в документе отображаются стрелки (рис. 6.61) от влияющих ячеек к целевой ячейке (в данном случае, напомним, это ячейка F8).

Выделяются те ячейки, которые напрямую влияют на значение целевой ячейки. Для ячейки F8 такими являются ячейки E1:E3 и C5:C6 (табл. 6.2). Но ячейки E1:E3 и C5:C6, в свою очередь, также зависят от других ячеек. Чтобы увидеть всю цепочку взаимосвязей, еще раз щелкаем пиктограмму **Влияющие ячейки**. Результат показан на рисунке 6.62.

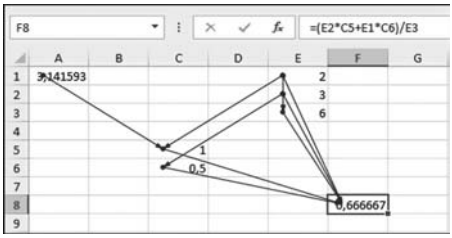


Рис. 6.62
Отображение влияющих ячеек второго уровня

Чтобы убрать стрелки, можно воспользоваться пиктограммой **Убрать ссылки** — команды этого списка позволяют убрать либо все стрелки (к влияющим ячейкам и к зависимым ячейкам), либо стрелки определенного типа (рис. 6.63).

Практически также выявляются зависимые ячейки — ячейки, значение которых зависит от значения текущей ячейки. На рисунке 6.64 выделена ячейка E1 и щелчком на пиктограмме **Зависимые ячейки** мы переходим в режим отображения связей между этой ячейкой и теми, что являются для нее зависимыми.

Результат такой процедуры представлен на рисунке 6.65.

Еще один щелчок на пиктограмме **Зависимые ячейки** позволяет отобразить зависимые ячейки для зависимых ячеек (рис. 6.66).

Имеется один достаточно полезный режим, который нередко используется для отслеживания ошибок и взаимосвязей между ячейками. Это режим, при котором в ячейках с формулами отображаются формулы, а не вычисляемые значения. Для перехода в режим отображения формул щелкаем пиктограмму **Показать формулы** на вкладке **Формулы** в группе **Зависимости формул**, как показано на рисунке 6.67.

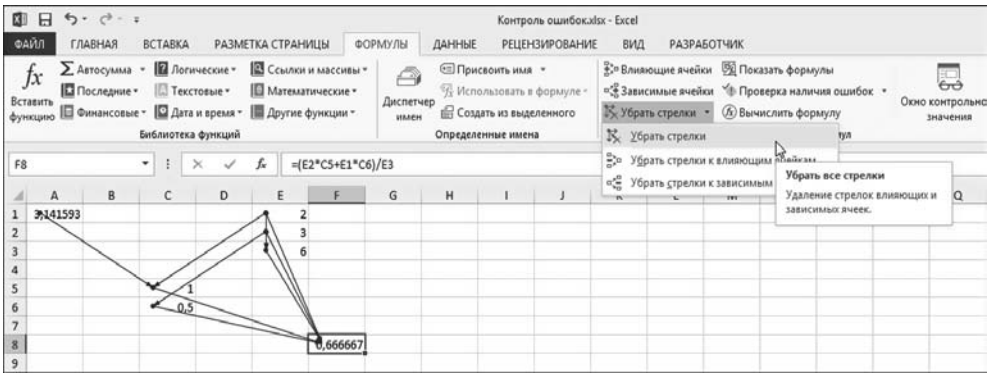


Рис. 6.63
Отмена режима отображения стрелок влияющих и зависимых ячеек

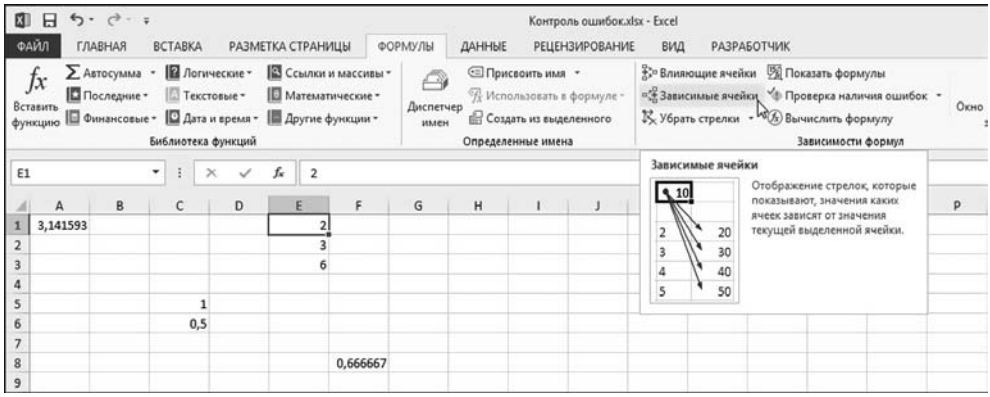


Рис. 6.64
Переход в режим отображения зависимых ячеек

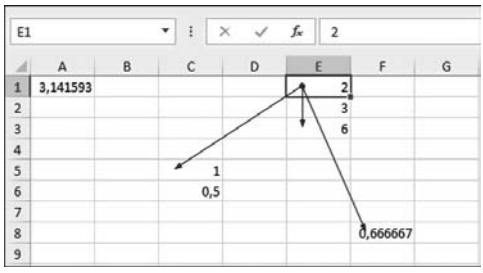


Рис. 6.65
Отображение зависимых ячеек

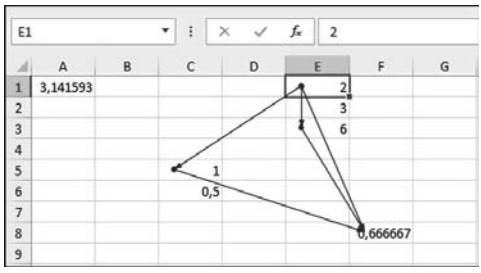


Рис. 6.66
Отображение зависимых ячеек второго уровня

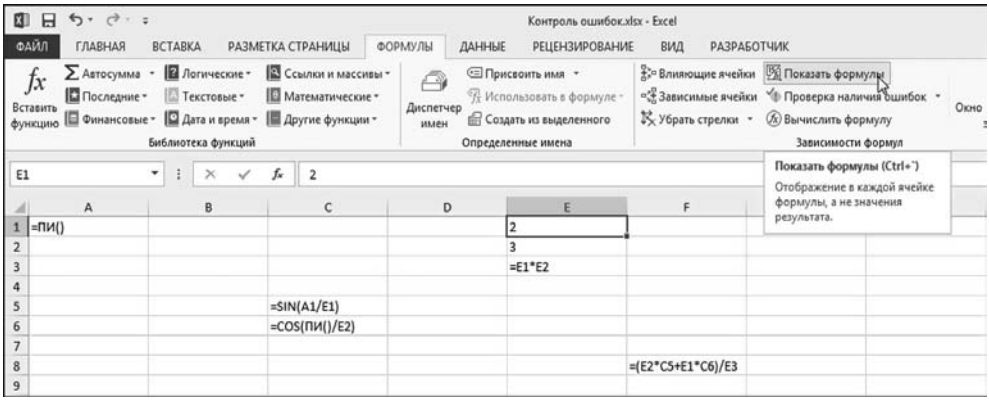


Рис. 6.67
Рабочий документ в режиме отображения формул

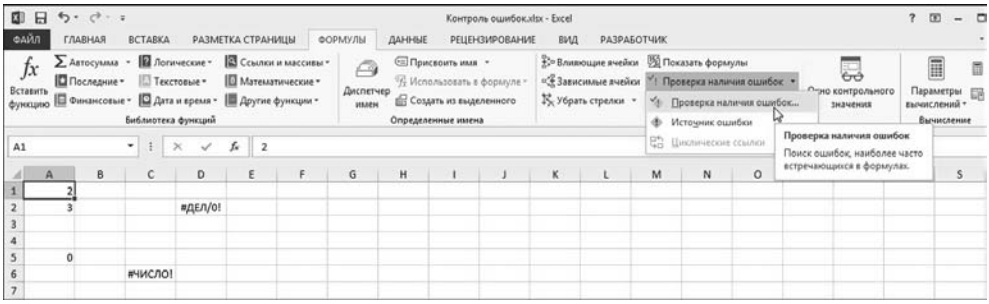


Рис. 6.68
Документ содержит ошибки

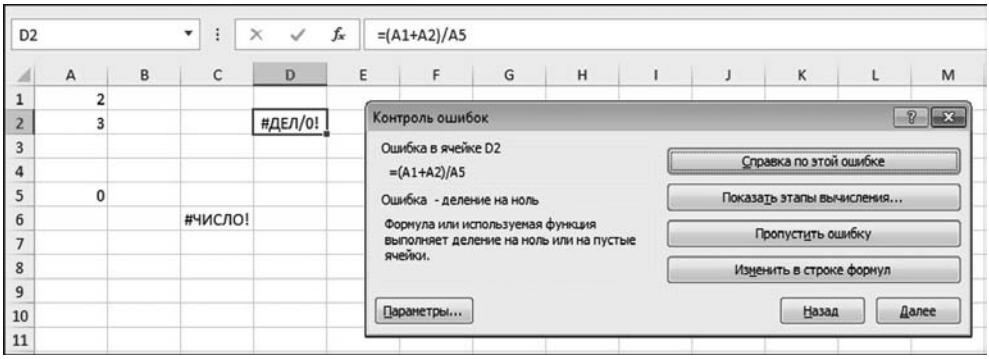


Рис. 6.69
Диалоговое окно **Контроль ошибок** на фоне рабочего документа

Чтобы выйти из данного режима, еще раз щелкаем пиктограмму **Показать формулы**.

Полезная для выявления и отслеживания ошибок утилита реализуется через пиктограмму **Проверка наличия ошибок** в группе **Зависимости формул** на вкладке **Формулы**. Для иллюстрации работы этой утилиты рассмотрим рабочий документ, представленный на рисунке 6.68.

Таблица 6.3

Содержимое ячеек документа с ошибкой

Ячейка	Содержимое
A1	2
A2	3
A5	=A2*2-3*A1
D2	=(A1+A2)/A5
C6	=КОРЕНЬ(A1-A2)

В этом документе искусственно сгенерированы две ошибки. Содержимое ячеек документа (с краткими комментариями) описано в таблице 6.3.

Ячейки A1 и A2 содержат числовые значения, значение в ячейке A5 вычисляется на основе этих значений (в результате в ячейке A5 вычисляется нулевое значение). В ячейке D2 выполняется деление на значение ячейки A5, поэтому в ячейке D2 возникает ошибка деления на ноль. Ошибка в ячейке C6 возникает, поскольку аргументом функции КОРЕНЬ() (извлечение квадратного корня) указано отрицательное число (аргумент вычисляется как разность значений ячеек A1 и A2).

После щелчка на команде **Проверка наличия ошибок** в окне начинается проверка ячеек на наличие ошибок. Найденная ячейка с ошибкой выделяется и для такой ячейки открывается диалоговое окно **Контроль ошибок**, в котором в достаточно удобном режиме можно попытаться исправить ситуацию (рис. 6.69).

Для выявления источника ошибки в формуле можем воспользоваться пиктограммой **Источник ошибки** в раскрывающемся списке команд пиктограммы **Проверка наличия ошибок** (рис. 6.70).

В этом случае легко отследить, какие ячейки стали источником проблем при вычислении значения целевой ячейки (см. рис. 6.71).

Убрать стрелки можно с помощью команд раскрывающегося списка пиктограммы **Убрать стрелки** (рис. 6.63).

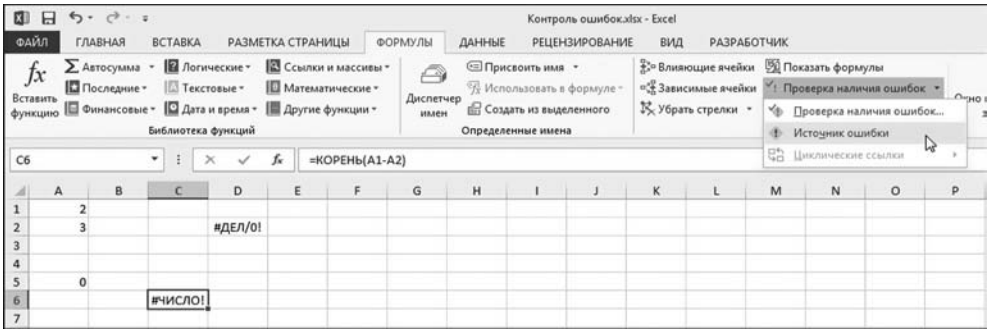


Рис. 6.70
Поиск источника ошибок

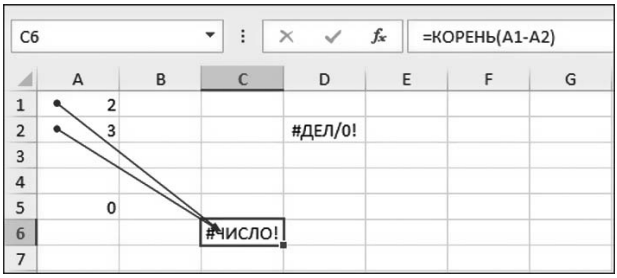


Рис. 6.71
Отслеживание ошибок путем выявления влияющих ячеек

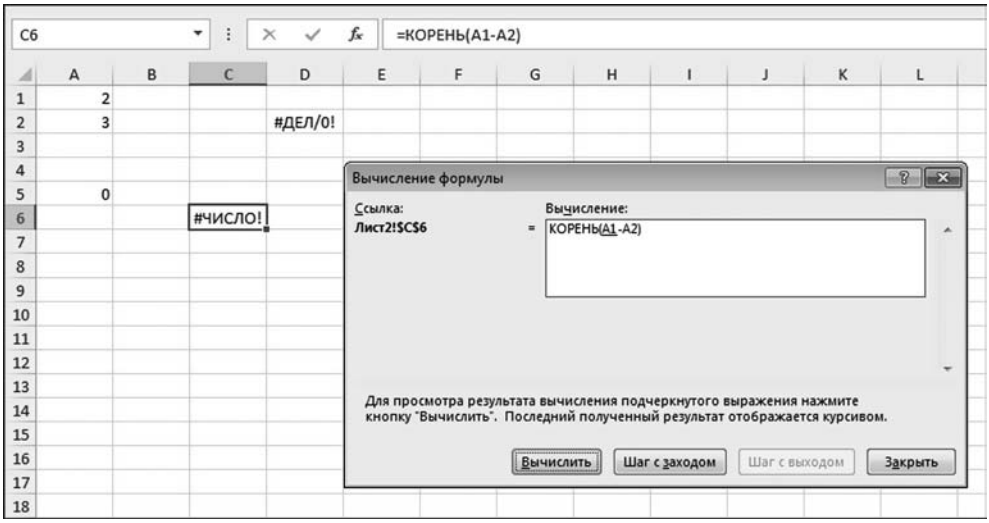


Рис. 6.72
Окно **Вычисление формулы** на фоне рабочего документа

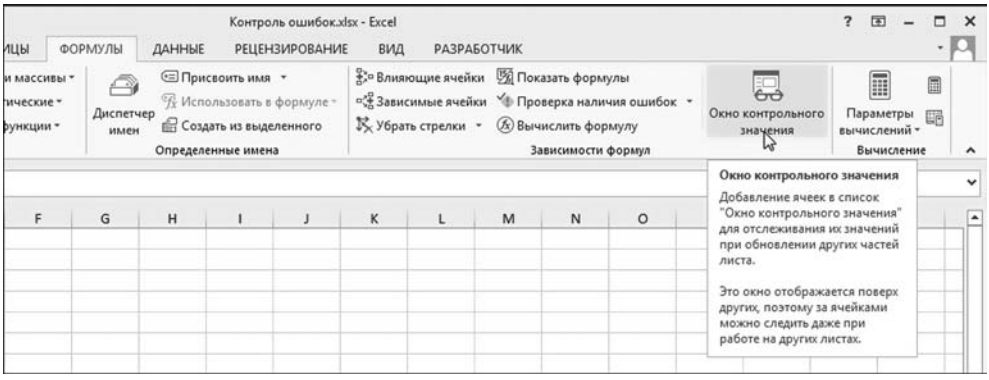


Рис. 6.73
Пиктограмма **Окно контрольного значения**

На заметку

Команда **Циклические ссылки** в списке команд пиктограммы **Проверка наличия ошибок** (рис. 6.70) предназначена для контроля циклических ссылок. Для отслеживания ошибок полезными будут и некоторые другие пиктограммы группы **Зависимости формул**. Например, щелкнув на пиктограмме **Вычислить формулу** (рис. 6.67), открываем диалоговое окно (рис. 6.72), в котором есть несколько полезных утилит, облегчающих поиск и анализ ошибок. Окно позволяет вычислять в пошаговом режиме формулу — иногда это помогает найти то место или тот этап вычислений, на котором происходит ошибка. Пиктограмма **Окно контрольного значения** (рис. 6.73) предназначена для отображения одноименного окна, которое показано на рисунке 6.74.

Окно предназначено для отображения и контроля в интерактивном режиме значений «избранных» ячеек. Сделать ячейку «избранной» достаточно легко — добавление контрольного значения в окно выполняется щелчком на пиктограмме **Добавить контрольное значение**. Если значение ячейки внесено в список контрольных, оно отображается в окне с учетом всех изменений, вносимых в рабочий документ.

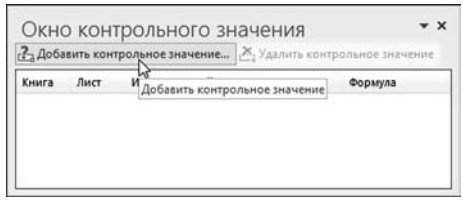


Рис. 6.74
Утилита **Окно контрольного значения**

И хотя система отслеживания ошибок у приложения Excel достаточно мощная и удобная, все же лучший способ борьбы с ошибками — стараться их не допускать. Кроме того, есть ошибки, против которых бессильна любая система контроля — логические ошибки. То есть такие ошибки, когда неверен сам алгоритм расчетов, притом, что с формальной точки зрения вычисления проводятся корректно. С такими ошибками бороться тяжелее всего.

ГЛАВА 7 ИЗБРАННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ ЗАДАЧИ

— Под звуки марша выезжает
двенадцать мотоциклистов.
— Максимум, четыре!
— Выезжает шесть мотоциклистов.
Они в скафандрах. Это водолазы.

Из к/ф «Старый знакомый»

В этой главе мы более детально остановимся на выполнении вычислений. Причем внимание будет сконцентрировано на нескольких «избранных» задачах. В этом смысле материал главы достаточно «субъективный» (вместе с тем это никак не означает, что он бесполезный, скорее наоборот). Также в этой главе мы познакомимся с надстройкой **Поиск решения**, которая незаменима при решении сложных вычислительных задач. Во всяком случае, наличие такой надстройки дает серьезные конкурентные преимущества приложению Excel по сравнению с другими электронными таблицами (хотя и здесь не все так просто). В конце главы будет рассмотрена еще одна надстройка, которая называется **Анализ данных**. С ее помощью мы научимся генерировать случайные числа (с заданными характеристиками их функции распределения) и применять статистические методы для решения некоторых задач — хотя это будет скорее иллюстрация возможностей приложения Excel. Кроме того, мы познакомимся с подходами и приемами, которые позволяют:

- решать алгебраические уравнения и системы;
- решать дифференциальные уравнения;
- вычислять интегралы.

На заметку

В последней, четвертой части книги все эти задачи рассматриваются более основательно. В этой главе мы лишь сделаем небольшой обзор возможностей Excel в области числовых расчетов.

НАДСТРОЙКА ПОИСК РЕШЕНИЯ

Ах, какая машина!
Это действительно новое слово в науке и технике!

Из к/ф «Иван Васильевич меняет профессию»

Надстройка в Excel — это, по большому счету, набор специальных утилит, которые становятся доступными после выполнения определенных действий, направленных на подключение надстройки. Надстройка **Поиск решения** входит в стандартный пакет Excel и предназначена для решения матема-

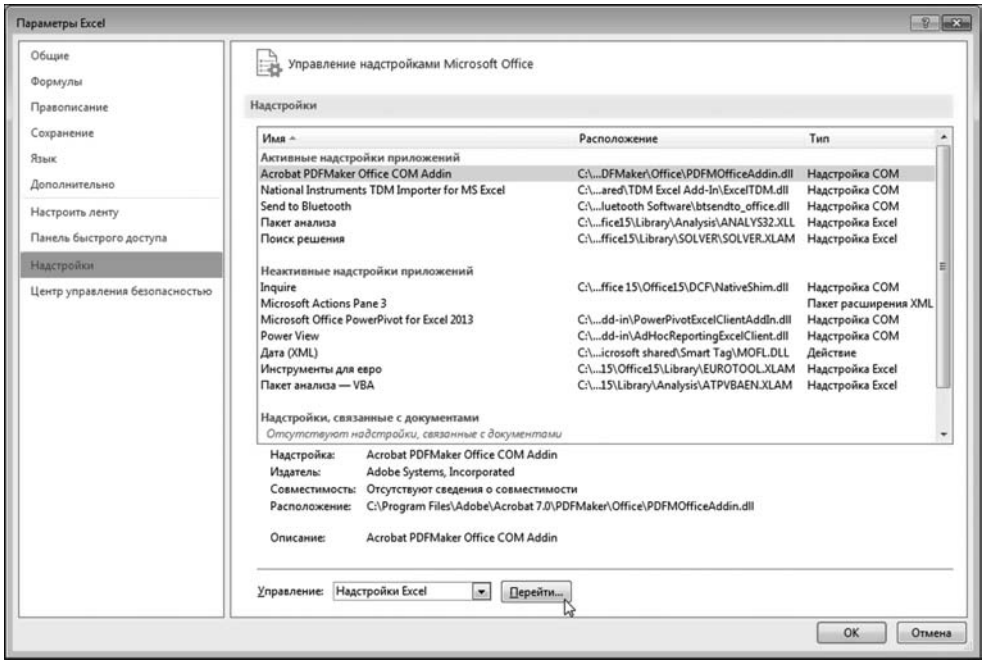


Рис. 7.1
Окно настройки приложения **Параметры Excel** открыто в разделе **Надстройки**

тических задач из области оптимизации и решения алгебраических уравнений и неравенств. Начнем с того, что опишем процедуру подключения надстройки **Поиск решения**. Сразу отметим, что подключение других надстроек осуществляется практически также.

Для подключения надстройки (любой, не только **Поиск решения**) открываем окно настройки приложения **Параметры Excel** в разделе **Надстройки** (рис. 7.1).

В раскрывающемся списке **Управление** выбираем пункт **Надстройки Excel** и щелкаем кнопку **Перейти**. В результате откроется окно **Надстройки**, представленное на рисунке 7.2.

В окне представлен список надстроек. Для подключения надстройки в соответствующей позиции следует выставить флажок. Нас интересует

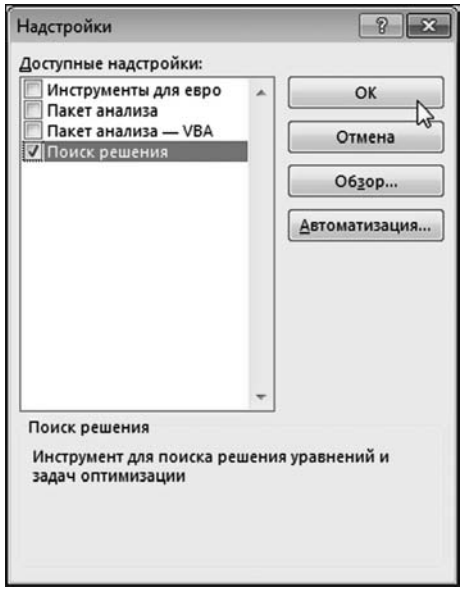


Рис. 7.2
В окне **Надстройки** необходимо выбрать подключаемую надстройку

надстройка **Поиск решения** (оригинальное англоязычное название надстройки **Solver Add in**). Таким образом, устанавливаем флажок опции **Поиск решения** и щелкаем кнопку **ОК**.

На заметку

Если в списке доступных надстроек окна **Надстройки** мы не встречаем нужного названия, можно попытаться найти файл надстройки вручную. В этом случае придется щелкнуть на кнопке **Обзор**.

Все, надстройка подключена! Убедиться в этом достаточно просто — на вкладке **Данные** должна появиться группа **Анализ** с пиктограммой надстройки **Поиск решения** (рис. 7.3).

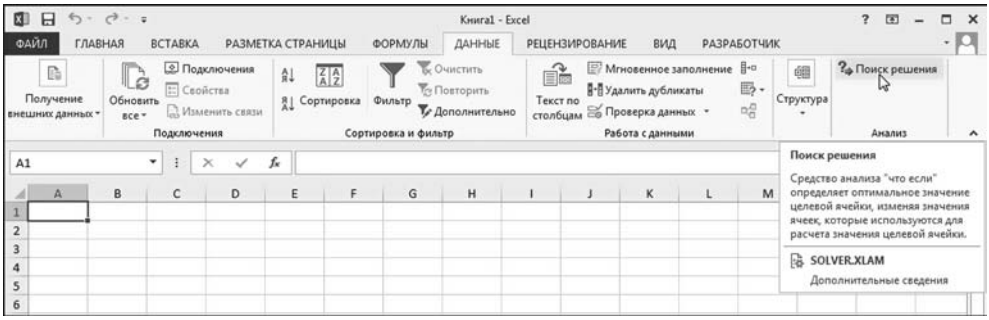


Рис. 7.3
На вкладке **Данные** появилась группа **Анализ** с пиктограммой **Поиск решения**

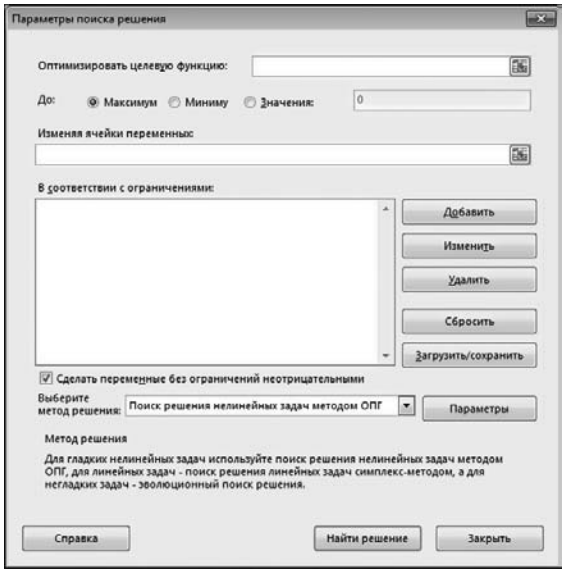


Рис. 7.4
Окно надстройки **Поиск решения**

Для запуска надстройки **Поиск решения** достаточно щелкнуть на одноименной пиктограмме.

На заметку

Многих, по-видимому, интересует, что же происходит при запуске надстройки. К сожалению, приходится признаться, что запуск надстройки — это не запуск большого адронного коллайдера. Ничего «такого» не происходит. Пожалуй, это даже к лучшему.

При запуске надстройки **Поиск решения** откроется диалоговое окно **Параметры поиска решения**, которое показано на рисунке 7.4.

Суть работы с надстройкой состоит в том, что в окне **Параметры поиска решения** выполняются настройки в соответствии с решаемой задачей. Затем щелкаем кнопку **Найти решение** и надеемся на то, что решение будет найдено (справедливости ради следует отметить, что случается это не всегда). Назначение функциональных элементов окна **Параметры поиска решения** легче пояснить на конкретных примерах, к ним и перейдем.

Для начала решим обычное алгебраическое уравнение. Пускай им будет уравнение $\sin(\pi x) + \cos(\pi x) = 0$. В принципе, уравнение тривиальное, но нас в данном случае интересует форма, а не содержание. Решений у уравнения бесконечное множество. Общее решение может быть записано как $x = -1/4 + n$, где через $n = 0, \pm1, \pm2, \ldots$ обозначено произвольное целое число. Таким образом, выбирая разные значения для числа n, получаем разные решения уравнения. Эти решения и будем искать с помощью надстройки **Поиск решения**. Перед началом «боевых действий», как учил великий Сунь-Цзы в своем «Трактате о военном искусстве», проведем предварительные расчеты. А именно, в ячейку B2 рабочего документа введем нулевое значение (хотя можно и какое-то другое — в данном случае это не принципиально), а в ячейку B3 вводим формулу =SIN(ПИ()*B2)+COS(ПИ()*B2), как это показано на рисунке 7.5.

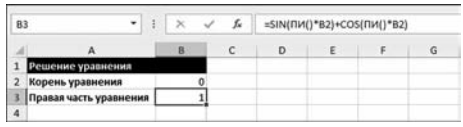


Рис. 7.5
Предварительная подготовка документа к вычислениям

На заметку

Здесь мы использовали встроенные функции SIN() и COS() для вычисления соответственно синуса и косинуса, а также функцию ПИ() для вычисления значения числа $\pi \approx 3{,}141592$.

Очевидно, что если в ячейке B2 указать значение для корня уравнения, то значение в ячейке B3 должно быть нулевым. Другими словами, нам необходимо подобрать такое значение в ячейке B2, чтобы в ячейке B3 появился 0. Как раз для решения этой задачи полезной будет надстройка **Поиск решения**, которую мы и запускаем (щелкаем пиктограмму **Поиск решения** в группе **Анализ** на вкладке **Данные**). Окно надстройки с выполненными настройками показано на рисунке 7.6.

Выполняем такие настройки. В поле **Оптимизировать целевую функцию** указываем ячейку B3. Вообще же в этом поле указывается ячейка, значение

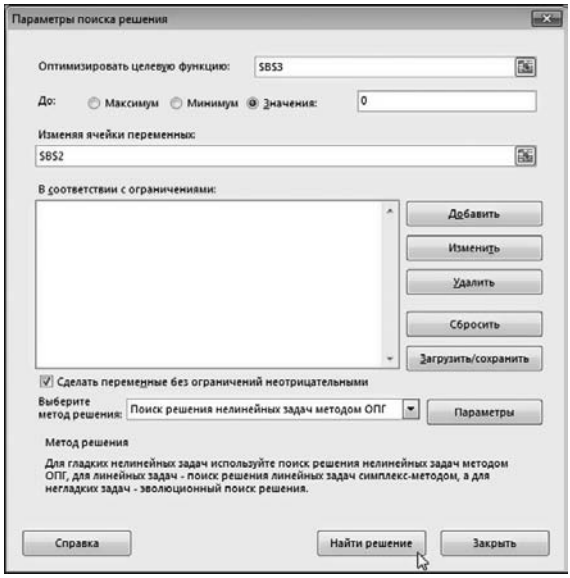


Рис. 7.6
Окно надстройки **Поиск решения** с выполненными настройками

которой контролируется в процессе поиска результата. Под «контролируется» подразумевается одна из трех процедур: поиск максимального значения, поиск минимального значения или приравнивание значения в ячейке к определенному числу. Какой способ «контроля» применяется, определяем с помощью переключателя **До**, у которого есть три позиции: **Максимум** (поиск максимального значения), **Минимум** (писк минимального значения) и **Значения** (приравнивание значения в ячейке к указанному числу). В последнем случае в поле справа необходимо указать число. Именно в этом положении находится переключатель и в рассматриваемом примере. В поле справа от переключателя указано нулевое значение.

В поле **Изменяя ячейки переменных** указывается ячейка B2. Это означает, что значение в ячейке B2 будет варьироваться так, чтобы в ячейке B3 значение стало равным 0. Кроме того, в диалоговом окне **Параметры поиска решения** установлен флажок опции **Сделать переменные без ограничений неотрицательными**. Название опции говорит само за себя. Эта опция установлена по умолчанию, и если против данного обстоятельства принципиальных возражений нет (а у нас их нет), то лучше оставить все как есть. Это же замечание относится и к выбору способа поиска решения в раскрывающемся списке **Выберите метод решения**.

На заметку

Поля **Оптимизировать целевую функцию** и **Изменяя ячейки переменных** работают в режиме выбора ячеек. Другими словами, вместо того, чтобы набирать адрес ячеек в этих полях, ячейки можно выделить мышью непосредственно в рабочем документе.

После того как настройки выполнены, щелкаем кнопку **Найти решение**. Открывается еще одно окно, которое называется **Результаты поиска решения**. Оно может содержать как приятную для нас информацию, так и не очень. На рисунке 7.7 показано окно как раз с «хорошим» сообщением.

На заметку

Окно **Результаты поиска решения** достаточно информативно и функционально. Предназначено оно для того, чтобы:

- во-первых, проинформировать пользователя о том, найдено решение или нет;
- во-вторых, дать возможность пользователю вернуть все в исходное состояние, как это было до начала поиска решения;
- в-третьих, изучить, так сказать, техническую сторону проделанных вычислений.

Нас все устраивает, поэтому мы принимаем результат и щелкаем на кнопке **ОК**. В результате значения в ячейках рабочего документа изменятся, как показано на рисунке 7.8.

Несложно заметить, что в данном случае действительно найдено решение уравнения (одно из решений), и это решение $x = 3/4 = 0{,}75$ (т. е. решение для значения $n = 1$).

Теперь несколько усложним процедуру. Для этого снова запускаем надстройку **Поиск решения**. Будем искать решение в определенном интервале значений. Окно надстройки **Поиск решения** с новыми настройками показано на рисунке 7.9.

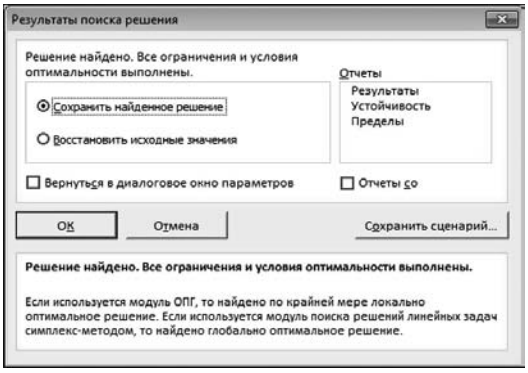


Рис. 7.7
Окно **Результаты поиска решения** появляется после попытки найти решение с помощью надстройки **Поиск решения**

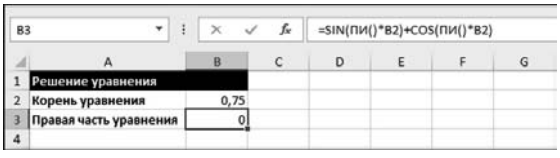


Рис. 7.8
Результат решения уравнения с помощью надстройки **Поиск решения**

По сравнению с предыдущим случаем изменений не много — в области **В соответствии с ограничениями** появилось два дополнительных условия B2<=10 и B2>=9. Таким образом, ищется не просто значение в ячейке B2, при котором в ячейке B3 значение будет нулевым: помимо этого, значение в ячейке B2 должно попадать в диапазон от 9 до 10.

Добавить ограничение для поиска решения достаточно просто — в окне **Параметры поиска решения** щелкаем кнопку **Добавить.** В результате открывается диалоговое окно **Добавление ограничения,** показанное на рисунке 7.10.

Окно содержит два поля, раскрывающийся список и три кнопки. В поле **Ссылка на ячейки** указывается ячейка (или диапазон ячеек), на которую накладывается ограничение. В раскрывающемся списке выбирается оператор отношения. В поле **Ограничение** указывается значение-константа или адрес ячейки, значение которой сравнивается со значением «ограничиваемой» ячейки. Например, для ввода первого ограничения в поле **Ссылка на ячейки** указываем адрес ячейки B2. В раскрывающемся списке выбираем операцию «больше или равно» (т. е. >=). В поле **Ограничение** указываем числовое значение 9 (рис. 7.11). После этого щелкаем кнопку **Добавить.**

На заметку

Поля **Ссылка на ячейки** и **Ограничение** работают в режиме выбора ячеек в рабочем документе — адреса ячеек можно выбирать выделением мышью в

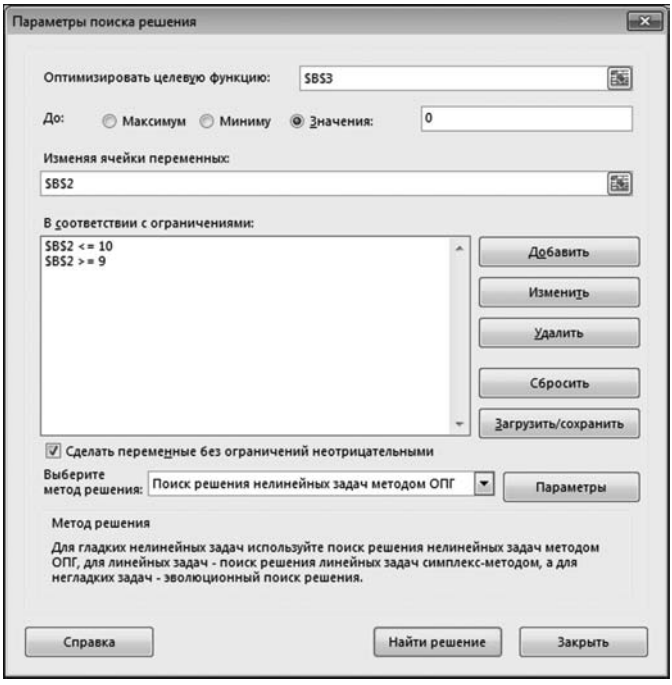


Рис. 7.9
Настройки утилиты **Поиск решения**
для поиска корня уравнения на указанном интервале

рабочей области документа. Разница между кнопками **ОК** и **Добавить** состоит в том, что после щелчка на кнопке **Добавить** ограничение принимается, но диалоговое окно **Добавление ограничения** не закрывается и в него можно ввести новое ограничение. Таким образом, кнопку **Добавить** щелкаем, если планируется вводить новые ограничения. Если с ограничениями вопрос решен, щелкаем кнопку **ОК**.

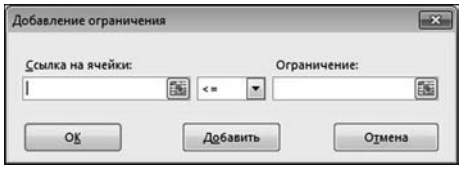


Рис. 7.10
Диалоговое окно
Добавление ограничения
для добавления дополнительных
условий (ограничений)

Процесс ввода второго ограничения иллюстрирует рисунок 7.12.

Здесь мы снова ограничиваем ячейку B2. Только на этот раз выбрана операция «меньше или равно» (т. е. <=) и в поле **Ограничение** указано числовое значение 10. Подтверждением корректности введенных значений служит щелчок на кнопке **ОК**. Результат вычислений показан на рисунке 7.13.

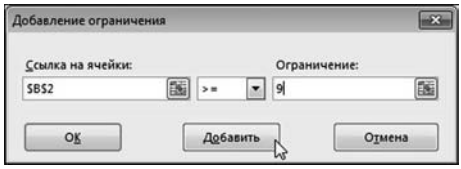


Рис. 7.11
Добавление
первого ограничения

Найдено решение $x = 9{,}75$. То, что в ячейке B3 отображается не совсем ноль — не страшно. Во-первых, там величина, отличная от нуля в девятой позиции (запись 3,44E-09 означает число $3{,}44 \cdot 10^{-9}$). Во-вторых, всегда есть ошибки округления при вычислениях, а у нас в формуле уравнения использовались синусы, косинусы и вычислялось число π — так что, как говорится, нет ничего удивительного. В-третьих, значение в ячейке B2 немного обманчиво. Дело в том, что при отображении значений в ячейках они округляются. Чтобы увидеть «реальное» значение, лучше обратиться к строке формул. На рисунке 7.14 выделена ячейка B2 (но мы смотрим на строку формул).

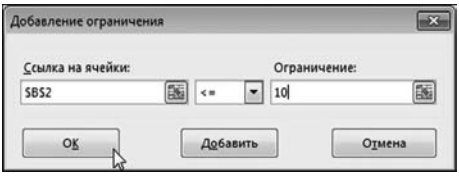


Рис. 7.12
Добавление
второго ограничения

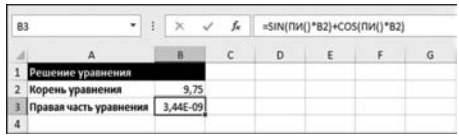


Рис. 7.13
Результат поиска корня
в указанном интервале

В строке формул четко видно, что решение найдено не совсем точное — хотя точность поиска решения в данном случае более чем приемлемая.

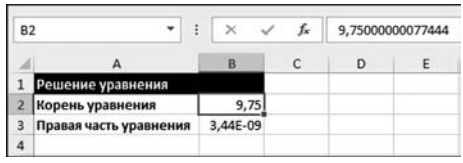


Рис. 7.14
В строке формул видим «реальное»
значения для найденного корня

Чтобы отшлифовать наши навыки работы с надстройкой **Поиск решения**, решим с ее помощью еще и *систему уравнений*. Систему рассмотрим «иде-

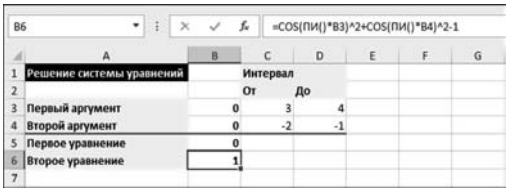


Рис. 7.15
Документ готов к решению системы уравнений

ологически близкую» к тому уравнению, что решалось выше. Она будет состоять из двух уравнений: $\sin(\pi x) + \sin(\pi y) = 0$ и $\cos^2(\pi x) + \cos^2(\pi y) - 1 = 0$. Решений у этой системы очень много, а именно: $x = \pm\frac{1}{4} + m$ и $y = \mp\frac{1}{4} + n$, где через $m, n = 0, \pm1, \pm2, \ldots$ обозначены произвольные целые числа. Именно эти решения будем искать. Документ, в котором планируется выполнять соответствующие вычисления, показан на рисунке 7.15.

Всего «в работе» находится восемь ячеек. Ячейки B3 и B4 содержат оценочные значения для решения системы уравнений. Подставим туда нули. В ячейки B5 и B6 вводятся формулы, которыми задаются уравнения системы. А именно, в ячейку B5 введена формула =SIN(ПИ()* B3)+ SIN(ПИ()*B4), а в ячейку B6 введена формула =COS(ПИ()*B3)^2+ COS(ПИ()*B4)^2-1. Задача состоит в том, чтобы подобрать такие значения в ячейках B3:B4, при которых в ячейках B5:B6 были бы нули. Но поскольку решений у системы много, мы хотим четко определить, какое именно решение ищем. Другими словами, задаем интервал поиска решения (по каждому из аргументов). Для первого аргумента (переменная x) границы интервала поиска решения задаются в ячейках C3:D3, а для второго аргумента (переменная y) границы интервала поиска решения задаются в ячейках C4:D4. Причем в ячейки C3:C4 вводятся числовые значения, а значения в ячейках D3 и D4 вычисляются соответственно по формулам =C3+1 и =C4+1. Поэтому, какие бы значения мы ни ввели в ячейки C3:C4, в смежных с ними ячейках D3:D4 будет всегда на единицу больше.

Запускаем надстройку **Поиск решения**. Окно надстройки с выполненными настройками показано на рисунке 7.16.

В качестве целевой указана ячейка B6. Ее значение приравнивается к нулю. Изменяемыми являются ячейки диапазона B3:B4. Но этого мало. Дело в том, что теперь решается не одно уравнение, а система уравнений. Одной целевой ячейки мало. Поэтому среди ограничений есть и такое: B5=0.

На заметку

Другими словами, из двух ячеек, значения которых необходимо приравнять к нулю, одну мы указываем как целевую, а другую «передаем» в качестве ограничения.

Чтобы решение искалось в определенном нами интервале, в окне **Параметры поиска решения** есть еще два ограничения: B3:B4<=D3:D4 и B3:B4>=C3:C4. Особенность этих условий в том, что в них использованы диапазоны. Соответствующие соотношения выполняются поэлементно, т. е. соотношение B3:B4<=D3:D4, например, означает, что должны выполняться условия B3<=D3 и B4<=D4. Использование в ограничениях диапазонов ячеек значительно упрощает процесс создания/добавления ограничений в окне настроек

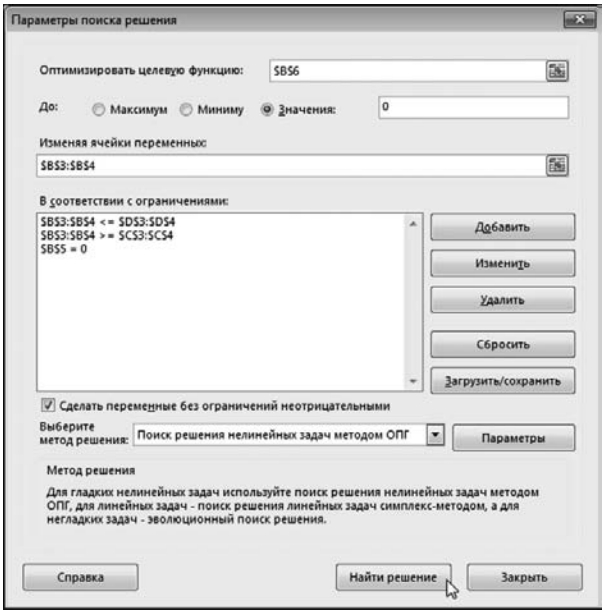


Рис. 7.16
Окно надстройки **Поиск решения** с настройками для поиска решения системы уравнений

Параметры поиска решения — благо, разработчиками такая возможность предусмотрена.

На заметку

Если вдруг ограничение введено неправильно, то его можно изменить или удалить. Для изменения ограничения в окне **Параметры поиска решения** щелкаем кнопку **Изменить** (предварительно выделив редактируемое ограничение). Для удаления ограничения из списка ограничений **В соответствии с ограничениями** выделяем ограничение и щелкаем кнопку **Удалить**. Кнопка **Сбросить** позволяет сбросить все настройки в окне **Параметры поиска решения**.

На рисунке 7.17 показан результат поиска решения системы уравнений для выставленных нами настроек утилиты **Поиск решения**.

В пределах точности вычислений решение найдено корректно. То есть все вроде в пределах приличия. Но есть одно немаловажное обстоятельство, на которое стоит обратить внимание. Если мы изменим интервал поиска решения (ячейки диапазона B3:B4), то решение автоматически пересчитано не будет. Для этого снова придется запускать надстройку **Поиск решения**. Хотя ничего страшного в этом нет, такая ситуация не вписывается в общую концепцию вычислений в рабочем документе,

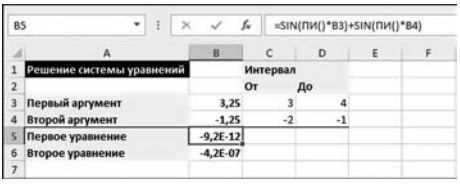


Рис. 7.17
Результат поиска решения системы уравнений

когда изменение хотя бы одной ячейки в документе приводит к пересчету всех зависимых ячеек.

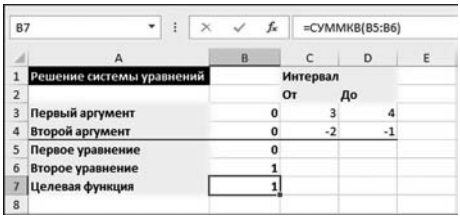
На заметку

Режим автоматического пересчета зависимых ячеек используется по умолчанию. Хотя из этого режима можно и выйти, однако ради любопытства делать этого не стоит.

Метод поиска экстремума с помощью надстройки **Поиск решения** проиллюстрируем на примере все той же системы уравнений. Для этого, правда, придется несколько изменить структуру рабочего документа. Новая версия документа показана на рисунке 7.18.

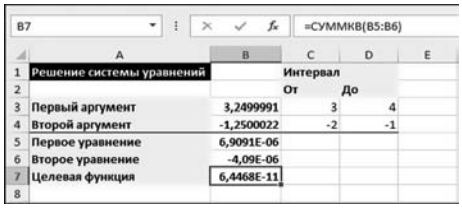
По сравнению с предыдущим случаем, мы изменили одну ячейку — точнее, задействовали одну дополнительную ячейку. А именно, в ячейку B7 вводим формулу =СУММКВ(B5:B6), в соответствии с которой вычисляется сумма квадратов ячеек диапазона B5:B6. Что нам это дает? А дает нам это многое.

Напомним, что необходимо решить систему уравнений $f_1(x, y) = 0$ и $f_2(x, y) = 0$, где мы обозначили $f_1(x, y) = \sin(\pi x) + \sin(\pi y)$ и $f_2(x, y) = \cos^2(\pi x) + \cos^2(\pi y) - 1$. В ячейке B7 вычисляется фактически выражение $f_1^2(x,y) + f_2^2(x,y)$. Это выражение по определению не может быть меньше нуля. Очевидно, что минимальным значением выражения является нулевое. И это минимальное значение выражение принимает в точке решения системы уравнений, т. е. в точке, для которой $f_1(x, y) = 0$ и $f_2(x, y) = 0$. Поэтому план действий такой: варьируем ячейки диапазона B3:B4 так, чтобы значение в ячейке B7 стало нулевым (или принимало минимальное значение).



	A	B	C	D	E
1	Решение системы уравнений	Интервал			
2		От	До		
3	Первый аргумент	0	3	4	
4	Второй аргумент	0	-2	-1	
5	Первое уравнение	0			
6	Второе уравнение	1			
7	Целевая функция	1			
8					

Рис. 7.18
Еще один способ решения системы уравнений



	A	B	C	D	E
1	Решение системы уравнений	Интервал			
2		От	До		
3	Первый аргумент	3,2499991	3	4	
4	Второй аргумент	-1,2500022	-2	-1	
5	Первое уравнение	6,9091E-06			
6	Второе уравнение	-4,09E-06			
7	Целевая функция	6,4468E-11			
8					

Рис. 7.20
Результат поиска минимума целевой функции: решение системы уравнений найдено

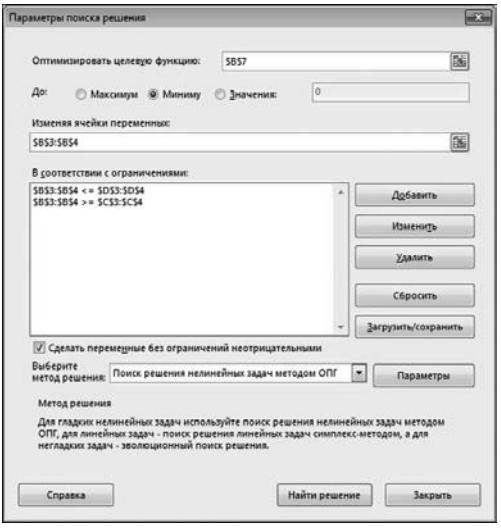


Рис. 7.19
Окно **Параметры поиска решения** с выполненными настройками для поиска минимума целевой функции

На заметку

Поиск экстремума (минимума) ячейки B7 и приравнивание значения этой ячейки к нулю не являются задачами эквивалентными. Строго говоря, нам нужно подогнать значение ячейки B7 к нулю. Поиск минимума дает надежду на успех, но не гарантирует того, что будет найдено решение системы уравнений. Тем не менее в целях наглядности будем искать именно минимум.

Окно **Параметры поиска решения** с выполненными настройками показано на рисунке 7.19.

Целевой является ячейка B7. Для этой ячейки, как и договаривались, ищем минимум. Ограничений стало меньше — точнее, всего два: B3:B4<=D3:D4 и B3:B4>=C3:C4. Результат поиска решения показан на рисунке 7.20.

Как видим, и в этом случае решение найдено с довольно неплохой точностью. Что касается общих рекомендаций относительно выбора метода решения систем уравнений, то их, собственно, нет. Дело в том, что определить, какой метод лучше, заранее крайне сложно. Поэтому обычно действуют, исходя из принципа, что лучший критерий — это практика. Другими словами, пробуем различные методы, пока не получим нужный результат.

РЕШЕНИЕ АЛГЕБРАИЧЕСКОГО УРАВНЕНИЯ

Стыдитесь!
Вот Маргадон, дикий человек, и то выучил!

Из к/ф «Формула любви»

Мы уже сталкивались с методом последовательных итераций, когда рассматривали циклические ссылки. Здесь при решении алгебраического уравнения попробуем обойтись без них и реализуем процесс в явном виде иными средствами в рабочем документе Excel.

На заметку

Еще раз отметим, что мы лишь иллюстрируем вычислительные возможности Excel. Решению алгебраических уравнений и систем всецело посвящена глава 12.

Рассматривать будем уже знакомое нам уравнение $x^2 - 6x + 5 = 0$, которое для применения метода последовательных итераций запишем в виде $x = \frac{x^2 + 5}{6}$. Рабочий документ, в котором предполагается решать это уравнение, представлен на рисунке 7.21.

На заметку

Напомним, что при решении уравнения вида $x = \varphi(x)$ методом последовательных итераций новое приближение x_{k+1} на основе уже известного приближения x_k вычисляется как $x_{k+1} = \varphi(x_k)$.

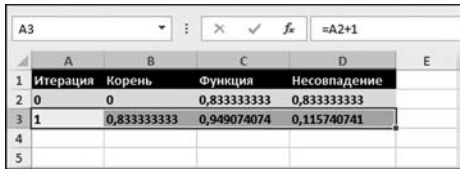


Рис. 7.21
Документ для решения уравнения методом последовательных итераций перед началом расчетов

Таблица 7.1

Значения ячеек диапазона A2:D2

Ячейка	Значение или формула	Комментарий
A2	0	Начальная итерация
A3	=A2+1	Номер итерации. Вычисляется увеличением номера предыдущей итерации на единицу
B2	0	Начальное приближение для корня уравнения
B3	=C2	Вычисление нового приближения для корня уравнения. Вычисляется как значение функции уравнения (функция $\varphi(x)$ в правой части соотношения $x = \varphi(x)$, которая определяет уравнение), рассчитанное на предыдущей итерации
C2	=(B2^2+5)/6	Вычисление функции уравнения
C3	=(B3^2+5)/6	Вычисление функции уравнения. Ячейка заполняется копированием содержимого ячейки C2
D2	=C2-B2	Разница между значением функции уравнения и текущим значением корня
D3	=C3-B3	Разница между значением функции уравнения и текущим значением корня. Значение ячейки получается копированием из ячейки D2

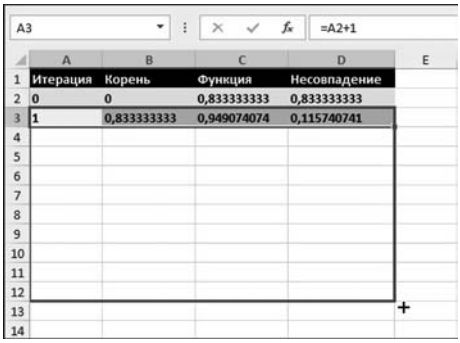


Рис. 7.22
Решение уравнения методом последовательных итераций — процесс копирования ячеек

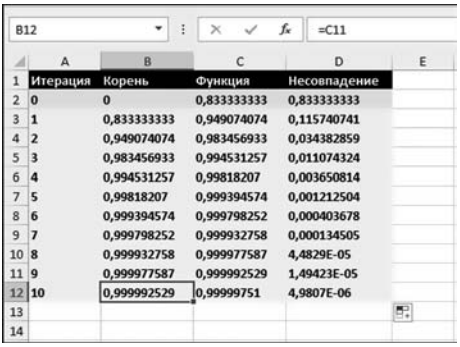


Рис. 7.23
Результат решения уравнения методом итераций

На начальном этапе заполняем диапазон ячеек A2:D2. Содержимое ячеек этого диапазона описано в таблице 7.1.

Данные разбиты по столбцам:

- в столбце A отображается номер итерации (начиная с нулевой, т. е. начального приближения);
- в столбце B отображается текущее значение для корня уравнения (левая часть выражения $x = \varphi(x)$);
- в столбце C вычисляется значение функции уравнения (функция в правой части уравнения) на основе значений в столбце B;
- в столбце D вычисляется разность функции уравнения и текущего вычисленного значения корня уравнения. Это значение тем меньше, чем точнее вычислен корень.

На заметку

Диапазон ячеек C3:D3 заполняется копированием диапазона C2:D2.

Процесс вычислений достаточно простой. Выделяем диапазон ячеек A3:D3 и выполняем автоматическое копирование (перетаскиванием маркера автоматического заполнения), как показано на рисунке 7.22.

Результат вычислений показан на рисунке 7.23.

Не считая начальной итерации, корень вычислялся на основе десяти приближений. Точность довольно неплохая. Кроме того, такой способ решения позволяет контролировать, как изменяется решение на каждой из итераций. Можно также проследить, как результат вычислений зависит от начального приближения (ячейка B2). Для этого достаточно изменить значение ячейки с начальным приближением.

На заметку

При копировании формул автоматически выполняется и копирование форматов. Мы, например, выделили исходные данные жирным шрифтом и выполнили заливку цветом. Этот формат применяется ко всем ячейками диапазона A3:D12.

РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНОГО УРАВНЕНИЯ

— Прежде всего, поздравляю тебя с открытием! От души!
— Что Вы, Галина Петровна!
В общем, это ведь чистая случайность.
— Случайность, отец, это непознанная закономерность.
Кто ищет — тот всегда найдет.

Из к/ф «Тридцать три»

Приложение Excel может использоваться для решения таких задач, которые, на первый взгляд, к Excel вообще никакого отношения не имеют. Например — решение дифференциальных уравнений. О дифференциальных уравнениях можно говорить и писать много. Мы пойдем иным путем — сразу рассмотрим небольшой пример.

На заметку

Дифференциальным уравнением называется соотношение вида $F(x, y, y', y'', \ldots, y^{(n)}) = 0$, связывающее аргумент x, неизвестную функцию $y(x)$ этого аргумента и ее производные $y'(x)$, $y''(x)$, …, $y^{(n)}(x)$. Порядок дифференциального уравнения определяется порядком старшей производной, которая входит в уравнение. Другими словами, задача состоит в том, чтобы на основании выражения, содержащего неизвестную функцию и ее производные, найти эту самую неизвестную функцию. Если уравнение первого порядка, то это означает, что в уравнение входит аргумент, неизвестная функция и ее первая производная.

Но все же немного теории нам понадобится, чтобы проиллюстрировать суть применяемого для расчетов метода. Итак, рассматриваем уравнение вида $y'(x) = f(x, y(x))$. Наша задача состоит в том, чтобы найти неизвестную

функцию $y(x)$. Понятно, что речь может идти лишь о решении уравнения в числовом виде. Для этого необходимо указать начальное условие — значение функции $y(x)$ в точке x_0, т. е. указать числовое значение y_0 такое, что $y(x_0) = y_0$.

На заметку

Дифференциальное уравнение и начальные условия — это задача Коши.

Существует несколько способов решения дифференциальных уравнений. Мы воспользуемся не очень точным, но очень простым методом, который называется *методом Эйлера*. Внешне он очень похож на метод последовательных итераций, который рассматривался выше.

Чтобы найти значение функции $y(x)$ для указанного значения аргумента x (если известно, что $y'(x) = f(x, y)$ и $y(x_0) = y_0$), в рамках метода Эйлера интервал значений аргумента от x_0 до x разбивается на N одинаковых частей так, что $x_k = x_0 + k\Delta x$, где мы обозначили $\Delta x = \dfrac{x - x_0}{N}$, а индекс $k = 0, 1, 2, \ldots, N$. По определению, таким образом, $x_N \equiv x$. Для вычисления значения $y(x) \equiv y(x_N)$ последовательно вычисляем значения $y_k \equiv y(x_k)$ в узловых точках x_k ($k = 1, 2, \ldots, N$). При этом используется рекуррентное соотношение $y_{k+1} = y_k + \Delta x \cdot f(x_k, y_k)$ с начальным приближением $y_0 = y(x_0)$. Именно эту итерационную процедуру реализуем в рабочем документе. Реализуем мы ее для уравнения $y'(x) = y(x) \cdot \sin x$ с начальным условием $y(\pi/2) = 10$.

На заметку

Эта задача имеет «точное» аналитическое решение $y(x) = 10\exp(-\cos x)$.

Документ (до начала вычислений) представлен на рисунке 7.24.

В документе заполнены начальные ячейки. Распределение данных по столбцам такое:

- столбец A — номер узла;
- столбец B — значение аргумента в узле;
- столбец C — значение искомой функции в узловой точке;
- столбец D — точное решение уравнения в узловой точке;
- столбец E — разность точного и приближенного решений в узловой точке.

В ячейке F2 указано значение для шага приращения по аргументу. Что касается заполнения ячеек, то более подробная информация приведена в таблице 7.2.

	A	B	C	D	E	F	G
1	Номер узла	Аргумент x	Функция y(x)	Точное решение	Разность	Шаг dx	
2	0	1,570796327	10	10	0	0,001	
3	1	1,571796327	10,01	10,010005	5E-06		
4							

Рис. 7.24
Документ для решения дифференциального уравнения

Таблица 7.2

Заполнение ячеек для решения дифференциального уравнения

Ячейка	Значение или формула	Комментарий
A2	0	Начальный номер для узловой точки
A3	=A2+1	Вычисление номера узла: предыдущее значение плюс один
B2	=ПИ()/2	Значение начальной узловой точки (точка, в которой задано начальное условие)
B3	=$B$2+A3*$F$2	Вычисление значения следующего узла (в соответствии с выражением $x_k = x_0 + k\Delta x$)
C2	10	Значение искомой функции в начальной точке
C3	=C2+$F$2*C2*SIN(B2)	Вычисление искомой функции в узловой точке (в соответствии с выражением $y_{k+1} = y_k + \Delta x \cdot f(x_k, y_k)$)
D2	=10*EXP(-COS(B2))	Точное значение для решения дифференциального уравнения в начальной точке
D3	=10*EXP(-COS(B3))	Точное значение для решения дифференциального уравнения в узловой точке. Формула копируется из ячейки D2
E2	=D2-C2	Разность точного и приближенного решений в начальной точке
E3	=D3-C3	Разность точного и приближенного решений в узловой точке. Формула копируется из ячейки E2
F2	0,001	Значение для шага приращения по аргументу

C62 | =C61+$F$2*C61*SIN(B61)

	A	B	C	D	E	F	G
1	Номер узла	Аргумент x	Функция y(x)	Точное решение	Разность	Шаг dx	
2	0	1,570796327	10	10	0	0,001	
3	1	1,571796327	10,01	10,010005	5E-06		
4	2	1,572796327	10,02000999	10,02002	1E-05		
5	3	1,573796327	10,03002998	10,030045	1,5E-05		
10	8	1,578796327	10,08027986	10,08031999	4,01E-05		
11	9	1,579796327	10,09035981	10,09040499	4,52E-05		
12	10	1,580796327	10,10044976	10,10049999	5,02E-05		
13	11	1,581796327	10,11054971	10,11060498	5,53E-05		
14	12	1,582796327	10,12065965	10,12071997	6,03E-05		
15	13	1,583796327	10,13077958	10,13084496	6,54E-05		
30	28	1,598796327	10,28377737	10,28391922	0,000142		
31	29	1,599796327	10,29405712	10,2942041	0,000147		
32	30	1,600796327	10,30434684	10,30449897	0,000152		
33	31	1,601796327	10,31464655	10,31480383	0,000157		
34	32	1,602796327	10,32495625	10,32511867	0,000162		
35	33	1,603796327	10,33527592	10,33544349	0,000168		
57	55	1,625796327	10,56483111	10,56511323	0,000282		
58	56	1,626796327	10,57537996	10,57566734	0,000287		
59	57	1,627796327	10,58593876	10,5862314	0,000293		
60	58	1,628796327	10,59650751	10,59680542	0,000298		
61	59	1,629796327	10,6070862	10,60738938	0,000303		
62	60	1,630796327	10,61767483	10,61798328	0,000308		
63							

Рис. 7.25
Результат решения в числовом виде дифференциального уравнения

На заметку

В приведенных в таблице 7.2 формулах ссылки на ячейки B2 (начальное значение аргумента) и F2 (шаг приращения по аргументу) абсолютные. При копировании эти ссылки не должны меняться.

Для решения дифференциального уравнения выделяем диапазон ячеек A3:E3 и применяем процедуру автоматического заполнения ячеек. На рисунке 7.25 показан результат таких вычислений.

На заметку

Поскольку шаг приращения выбран достаточно маленький, для получения решения на значительном интервале по аргументу приходится заполнить очень приличное количество ячеек. Для удобства восприятия часть строк с фактическими данными о результатах вычислений скрыта.

В результате мы получили табулированную функцию, которая дает четкое представление о решении дифференциального уравнения. Сравнение с точным решением показывает, что приближенное решение найдено с очень неплохой точностью (до трех знаков после запятой). Тем не менее еще раз отметим, что использованный нами метод Эйлера не самый лучший (в плане точности). При необходимости читатель может обратиться к более надежным схемам — соответствующие подходы описываются в главе 15, посвященной решению дифференциальных уравнений.

ВЫЧИСЛЕНИЕ ИНТЕГРАЛОВ

— За сколько сделаешь?
— За день сделаю.
— А за два?
— Ну, сделаем и за два.
— А за пять дней?
— Ну, ежели постараться, можно и за пять.
— А за десять?
— Ну, барин, ты задачи ставишь!
За десять дён одному не справиться.
Тут помощник нужен — хомо сапиенс.
— Бери помощников, но чтобы не раньше!

Из к/ф «Формула любви»

Несложно догадаться, что если в Excel можно решать дифференциальные уравнения, то вычисление интегралов не составит большой проблемы. Остановимся на задаче о вычислении интеграла вида $y(x)=\int_{x_0}^{x} f(z)dz$. Здесь мы не случайно рассматриваем определенный интеграл как функцию от верхней границы интегрирования — так намного удобнее, особенно в части реализации итерационной схемы в Excel.

На заметку

По большому счету, вычисление интеграла вида $y(x)=\int_{x_0}^{x} f(z)dz$ есть не что иное, как решение дифференциального уравнения специального вида $y'(x)=f(x)$, в

котором правая часть (т. е. функция $f(x)$) не зависит от неизвестной функции $y(x)$, а зависит только от аргумента x. В этом случае решение дифференциального уравнения может быть записано в виде $y(x) = y_0 + \int_{x_0}^{x} f(z)dz$, где обозначено $y_0 = y(x_0)$ (начальное условие). А как решать дифференциальные уравнения в Excel мы уже некоторое представление имеем.

Для интегрирования будем использовать метод трапеций. Суть его в следующем. Интервал интегрирования разбиваем на N частей и выбираем узловые точки $x_k = x_0 + k\Delta x$, где обозначено $\Delta x = (x - x_0)/N$, а индекс $k = 0, 1,$

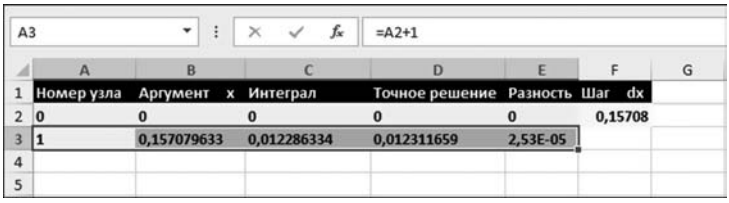


Рис. 7.26
Документ предназначен для вычисления интеграла

Таблица 7.3

Значения ячеек документа для вычисления интеграла

Ячейка	Значение или формула	Комментарий
A2	0	Номер начального узла
A3	=A2+1	Номер узла: к предыдущему значению прибавляется единица
B2	0	Левая граница интервала интегрирования
B3	=B2+$F$2	Значение узловой точки: к предыдущему значению прибавляется шаг дискретности (абсолютная ссылка на ячейку F2)
C2	0	Начальное значение для интеграла
C3	=C2+$F$2/2*(SIN(B2)+SIN(B3))	Вычисление интеграла: к текущему значению (ячейка C2) прибавляется площадь трапеции — в соответствии с выражением $\frac{\Delta x(f(x_k) + f(x_{k+1}))}{2}$, где $f(x) = \sin x$
D2	=1-COS(B2)	Точное значение для интеграла
D3	=1-COS(B3)	Точное значение для интеграла (формула копируется из ячейки B2)
E2	=D2-C2	Разница между точным решением и приближенным
E3	=D3-C3	Разница между точным решением и приближенным. Формула копируется из ячейки E2
F2	=ПИ()/20	Шаг приращения по аргументу. В данном случае с учетом периодичности подынтегральной функции шаг выбран в пропорции к числу π

2, …, N. При вычислении интеграла воспользуемся тем обстоятельством, что числовое значение интеграла равно площади под графиком подынтегральной функции на интервале интегрирования. Для вычисления площади под кривой для подынтегральной функции разбиваем область на трапеции, с основаниями в узловых точках. Ребро трапеции в точке x_k равняется $f(x_k)$. Тогда площадь области под кривой равняется (примерно) сумме площадей таких трапеций. А площадь одной трапеции равна $\frac{\Delta x(f(x_k)+f(x_{k+1}))}{2}$ (индекс $k = 0, 1, 2, \ldots, N-1$). Таким образом, в качестве оценки для значения интеграла получаем выражение $\frac{\Delta x}{2}\sum_{k=0}^{N-1}(f(x_k)+f(x_{k+1}))$. Именно этой формулой воспользуемся для вычисления с помощью Excel интеграла $\int_0^x \sin(z)dz = 1-\cos(x)$. Рабочий документ с начальными данными представлен на рисунке 7.26.

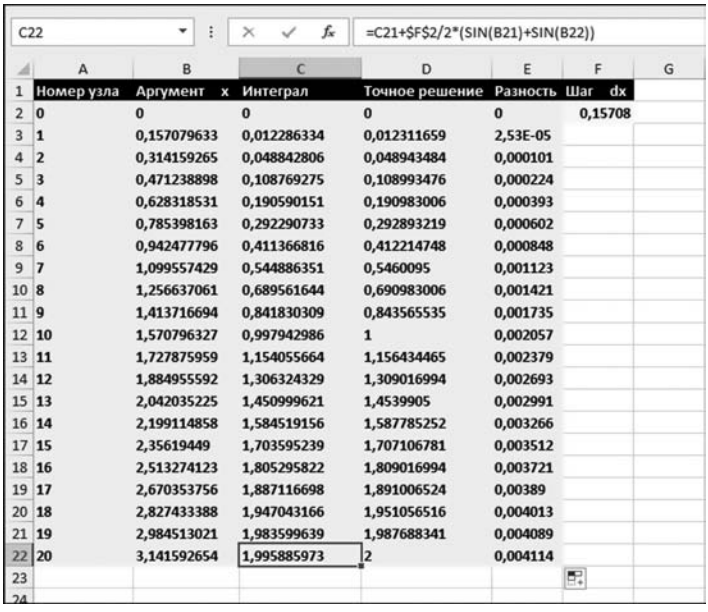
Документ по своей структуре очень сильно напоминает тот, что использовался при решении дифференциального уравнения. Но это, в общем-то, и неудивительно. Способы заполнения ячеек этого документа описаны в таблице 7.3.

Процедура вычисления такая же, как и ранее. Результат показан на рисунке 7.27.

В принципе, ничего особенного — все как обычно.

На заметку

Методы вычисления интегралов, в том числе и несобственных, описываются в главе 14.



	A	B	C	D	E	F	G
1	Номер узла	Аргумент x	Интеграл	Точное решение	Разность	Шаг dx	
2	0	0	0	0	0	0,15708	
3	1	0,157079633	0,012286334	0,012311659	2,53E-05		
4	2	0,314159265	0,048842806	0,048943484	0,000101		
5	3	0,471238898	0,108769275	0,108993476	0,000224		
6	4	0,628318531	0,190590151	0,190983006	0,000393		
7	5	0,785398163	0,292290733	0,292893219	0,000602		
8	6	0,942477796	0,411366816	0,412214748	0,000848		
9	7	1,099557429	0,544886351	0,5460095	0,001123		
10	8	1,256637061	0,689561644	0,690983006	0,001421		
11	9	1,413716694	0,841830309	0,843565535	0,001735		
12	10	1,570796327	0,997942986	1	0,002057		
13	11	1,727875959	1,154055664	1,156434465	0,002379		
14	12	1,884955592	1,306324329	1,309016994	0,002693		
15	13	2,042035225	1,450999621	1,4539905	0,002991		
16	14	2,199114858	1,584519156	1,587785252	0,003266		
17	15	2,35619449	1,703595239	1,707106781	0,003512		
18	16	2,513274123	1,805295822	1,809016994	0,003721		
19	17	2,670353756	1,887116698	1,891006524	0,00389		
20	18	2,827433388	1,947043166	1,951056516	0,004013		
21	19	2,984513021	1,983599639	1,987688341	0,004089		
22	20	3,141592654	1,995885973	2	0,004114		
23							
24							

Рис. 7.27
Результат вычисления интеграла

НАДСТРОЙКА АНАЛИЗ ДАННЫХ

— Скажите, Вы у нас случайно на четвертом курсе статистическую физику не читали?
— Читал. Не мешайте!

Из к/ф «Старый знакомый»

Еще одной полезной надстройке, которая называется **Анализ данных**, мы уделим внимание в силу той простой причины, что она содержит ряд полезных утилит, которые позволяют решать задачи с намеком на статистические методы. Предполагается, что надстройка уже подключена.

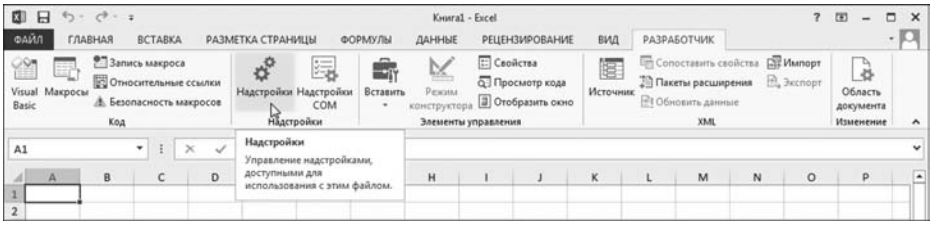


Рис. 7.28
Для подключения надстройки щелкаем пиктограмму **Надстройки** в одноименной группе вкладки **Разработчик**

На заметку

На всякий случай напомним последовательность действий при подключении надстройки. Для этого открываем (команда **Файл ▷ Параметры**) окно настроек приложения **Параметры Excel** в разделе **Надстройки** и щелкаем кнопку **Перейти**. Но, откровенно говоря, есть и более простой способ: на вкладке **Разработчик** в группе **Надстройки** можно щелкнуть на одноименной пиктограмме, как показано на рисунке 7.28.

В результате открывается диалоговое окно **Надстройки**, в котором представлен список доступных надстроек. Нам в данном конкретном случае следует установить флажок у опции **Пакет анализа** (англоязычное название надстройки **Analysis ToolPak**), как показано на рисунке 7.29.

После этого на вкладке **Данные** в группе **Анализ** появится пиктограмма для запуска утилиты **Анализ данных**.

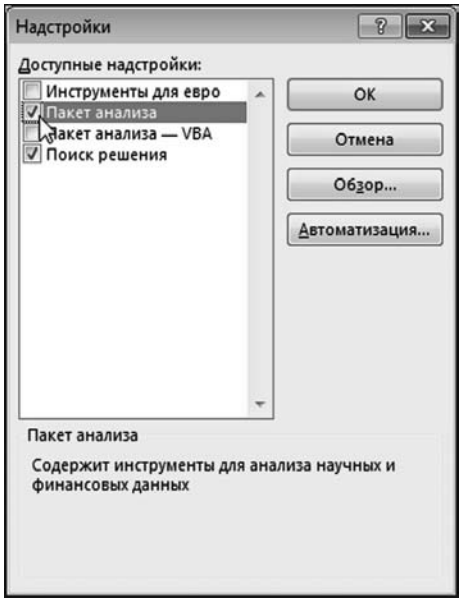


Рис. 7.29
Диалоговое окно **Надстройки** со списком надстроек для подключения

Для запуска надстройки **Анализ данных** на вкладке **Данные** в группе

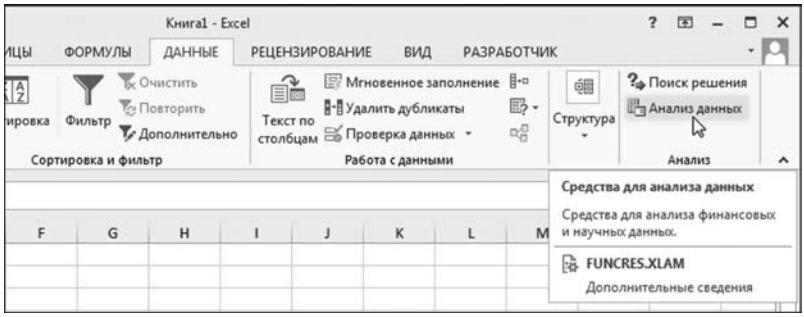


Рис. 7.30
Запуск надстройки **Анализ данных**

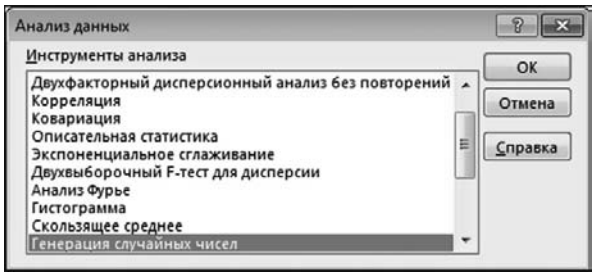


Рис. 7.31
В окне **Анализ данных** выбираем подходящий инструмент для анализа данных

Анализ щелкаем пиктограмму **Анализ данных**, как показано на рисунке 7.30.

После запуска надстройки открывается диалоговое окно **Анализ данных**, которое показано на рисунке 7.31.

Окно достаточно простое и интригующее. Все окно — это список из всевозможных утилит с очень специфическими названиями. Здесь представлены средства для решения различных по сложности задач, так или иначе относящихся к математической статистике (или теории вероятностей). Среди списка доступных пользователю инструментов есть утилиты для проведения дисперсионного анализа, проверки гипотез, создания гистограмм, генерирования случайных чисел, вычисления корреляционных показателей, основных статистических характеристик, и многое другое. В этом разделе рассматривается несколько не очень сложных с математической точки зрения задач.

На заметку

В данном случае математическая «несложность» задачи есть показатель относительный, поскольку определяется на фоне сложности прочих задач, послуживших основой для арсенала инструментов надстройки **Анализ данных**.

Начнем с простой задачи о генерировании случайных чисел. Относиться к этой задаче как к очень специфичной было бы неправильно. Такого рода

(или очень близкие) задачи возникают достаточно часто, поэтому собственно данному вопросу и уделяется значительное внимание не только в любом математическом пакете, но и в таком эффективном офисном приложении, как Excel.

На заметку

При работе со случайными величинами важное место занимает такая характеристика, как функция распределения или закон распределения. Если речь идет о дискретной случайной величине, то закон распределения определяет, какие значения и с какой вероятностью может принимать случайная величина. Для непрерывных случайных величин речь может идти о функции распределения. Если известно, что $F(x)$ есть функция распределения некоторой случайной величины, то значение $F(x)$ определяет вероятность того, что случайная величина принимает значение, не превышающее x. Плотностью распределения случайной величины называется функция $p(x)=\frac{dF(x)}{dx}$. Вероятность того, что случайная величина принимает значение в интервале от a до b дается интегралом $\int_a^b p(x)dx$.

При генерировании случайных чисел важно знать, какому закону распределения подчиняются случайные числа. Законы могут быть самые разные, но есть наиболее популярные и часто используемые. Все (или почти все) эти варианты предусмотрены в надстройке **Анализ данных**.

Чтобы от теории перейти к практике, в окне **Анализ данных** выбираем утилиту **Генерация случайных чисел** и щелкаем кнопку **ОК** (рис. 7.31). Откроется окно **Генерация случайных чисел**, представленное на рисунке 7.32.

В известном смысле определяющим является раскрывающийся список **Распределение**, в котором выбирается закон распределения для случайных чисел. На рисунке 7.33 проиллюстрирован процесс выбора закона распределения.

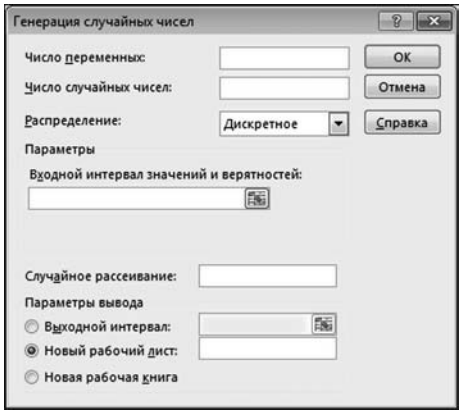


Рис. 7.32
Диалоговое окно
Генерация случайных чисел

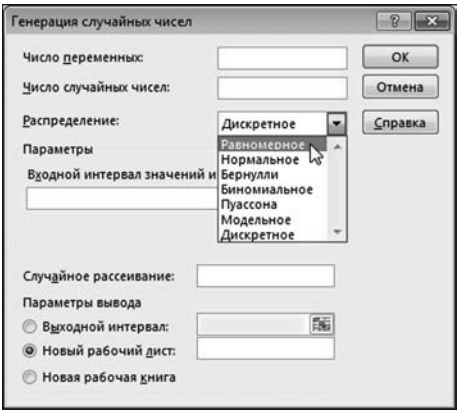


Рис. 7.33
Выбор типа распределения случайных чисел в списке **Распределение**

Вариантов не очень много, но они «перекрывают» все основные типы распределений. Нас в данном случае интересует равномерное распределение, для чего выбираем в раскрывающемся списке позицию **Равномерное**.

На заметку

От того, какой тип распределения выбран, зависит вид окна **Генерация случайных чисел**. Причина в том, что для разных распределений необходимо задавать различные наборы параметров. Так, для равномерного распределения задаются границы интервала a и b, на котором распределены случайные числа. Плотность распределения случайной величины, равномерно распределенной на интервале от a до b задается соотношением $p(x) = 1/(b - a)$, если $a \le x \le b$ или $p(x) = 0$ в противном случае.

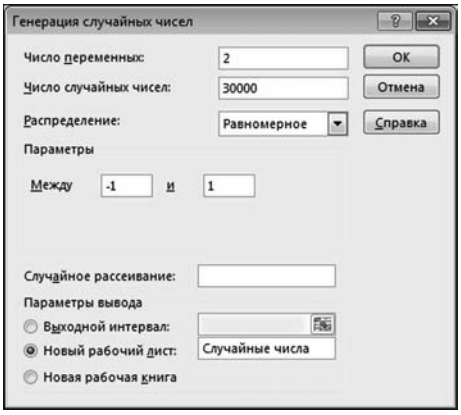


Рис. 7.34
Настройки в диалоговом окне **Генерация случайных чисел** для генерирования случайных равномерно распределенных чисел

На рисунке 7.34 показаны настройки, выполняемые для генерирования равномерно распределенных случайных чисел.

Параметры распределения (границы интервала распределения случайных чисел) задаются в полях **Между**. Мы в данном случае будем генерировать случайные числа в диапазоне от -1 до 1. В поле **Число переменных** мы указали значение 2, а в поле **Число случайных чисел** мы указали значение 30000. Это означает, что будет сгенерировано два столбика случайных чисел по 30 000 чисел в каждом столбике (итого, 60 000 случайных чисел). Переключатель **Параметры вывода** установлен в положение **Новый рабочий лист** для вывода результата (столбиков со случайными числами) в новый рабочий лист. Для нового рабочего листа задано название Случайные числа.

На заметку

В поле **Случайное рассеивание** можно указать число, которое будет задействовано для инициализации генератора случайных чисел. В этом случае каждый раз будет генерироваться одна и та же последовательность случайных чисел.

Результат генерирования столь значительного количества случайных чисел представлен в документе на рисунке 7.35.

Автоматически создан новый лист с именем **Случайные числа**, в котором ячейки A1:B30000 заполнены случайными числами в диапазоне от −1 до 1. На этом, собственно, чудеса заканчиваются. Возникает резонный вопрос: а зачем нам столько случайных чисел? Попробуем ими воспользоваться для вычисления (приближенного) значения числа $\pi \approx 3{,}141592$. Прибегнем к помощи метода Монте-Карло, для чего проведем такой мысленный эксперимент. Представим, что у нас есть квадрат со сторонами 2×2 и внутри этого квадрата вписан

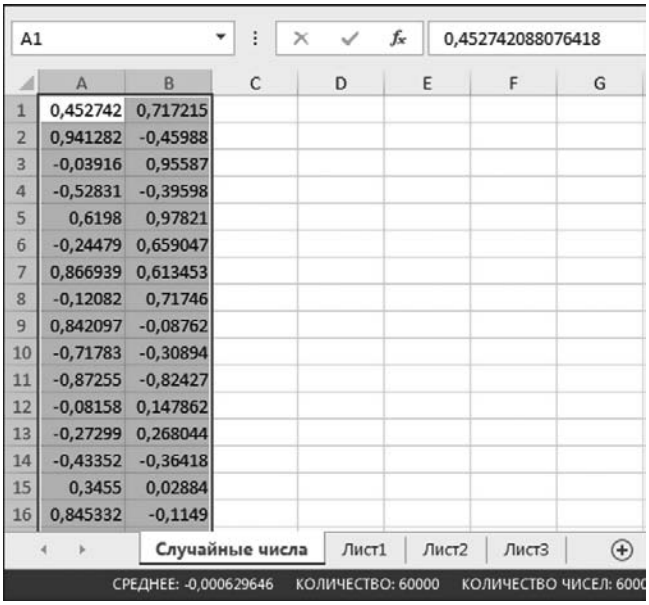


Рис. 7.35
В новом рабочем листе сгенерированы случайные числа

круг единичного радиуса. После этого в пределах квадрата случайным образом выбираем точки, равномерно распределенные по площади квадрата. Все они внутри квадрата, а некоторые попадают и внутрь круга. Отношение количества точек, попавших внутрь круга, к общему количеству точек примерно равняется отношению площади круга к площади квадрата. Площадь квадрата равна 4, а площадь круга единичного радиуса равна π. Если взять достаточно много точек, то относительное количество тех из них, что попали внутрь круга, стремится к величине $\pi/4$. Если точка имеет координаты $-1 \le x \le 1$ и $-1 \le y \le 1$, то условие того, что точка попадает внутрь круга, запишется как $x^2 + y^2 \le 1$. В качестве координат случайных точек используем случайные числа в ячейках A1:B30000 (в ячейках A1:A30000 будет, например, первая координата точки, а в ячейках B1:B30000 будет вторая координата точки). Для удобства в ячейках C1:C30000 вычислим сумму квадратов значений смежных ячеек. Ячейки диапазона C1:C30000 затем используем для проверки того, попадает ли точка «внутрь круга» или нет (значение в соответствующей ячейке не должно превышать единицы). В принципе, здесь все просто. Проблема однако в том, что такой большой диапазон ячеек не так просто заполнить. Поэтому применим небольшую хитрость. Сначала выделим диапазон ячеек C1:C30000 для последующего их заполнения. Для этого в поле имен вводим адрес диапазона C1:C30000, как показано на рисунке 7.36.

После того как диапазон выделен, в строку формул вводим формулу =A1:A30000^2+B1:B30000^2, причем вводим ее как *формулу массива*, нажатием комбинации клавиш <Ctrl>+ <Shift>+<Enter>. Процесс ввода формулы проиллюстрирован на рисунке 7.37.

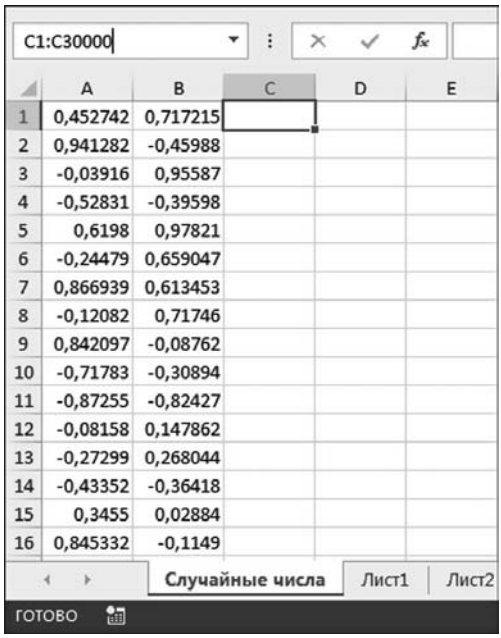


Рис. 7.36
Выделение большого диапазона ячеек:
адрес диапазона вводится в поле имен

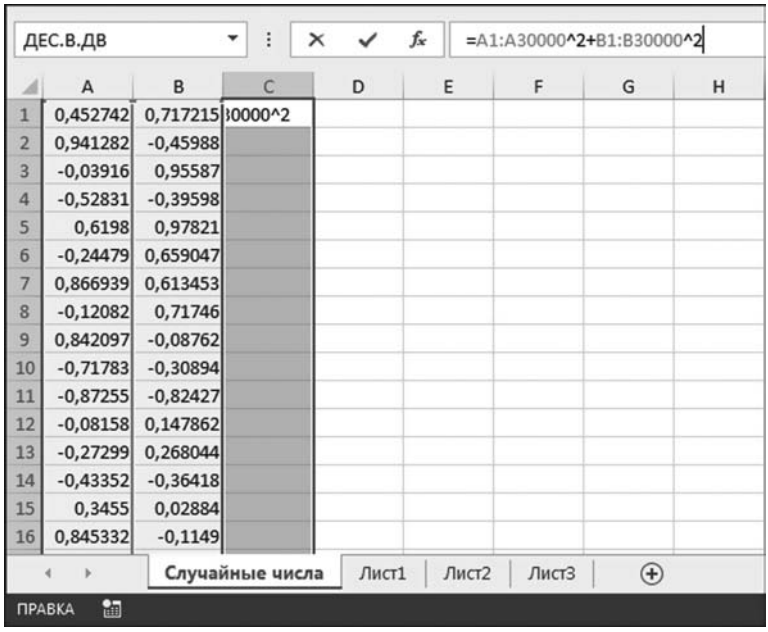
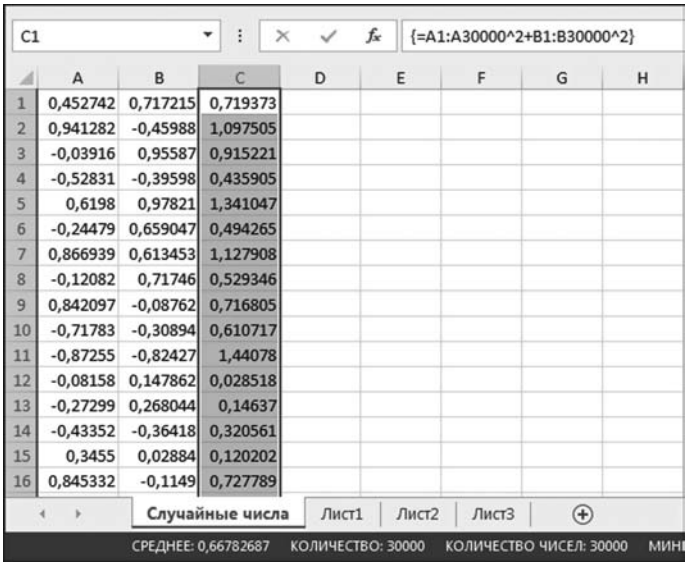


Рис. 7.37
Ввод формулы для вычисления значений ячеек в очень большом диапазоне



	A	B	C
1	0,452742	0,717215	0,719373
2	0,941282	-0,45988	1,097505
3	-0,03916	0,95587	0,915221
4	-0,52831	-0,39598	0,435905
5	0,6198	0,97821	1,341047
6	-0,24479	0,659047	0,494265
7	0,866939	0,613453	1,127908
8	-0,12082	0,71746	0,529346
9	0,842097	-0,08762	0,716805
10	-0,71783	-0,30894	0,610717
11	-0,87255	-0,82427	1,44078
12	-0,08158	0,147862	0,028518
13	-0,27299	0,268044	0,14637
14	-0,43352	-0,36418	0,320561
15	0,3455	0,02884	0,120202
16	0,845332	-0,1149	0,727789

Рис. 7.38
Для вычисления значения очень большого диапазона использована формула массива

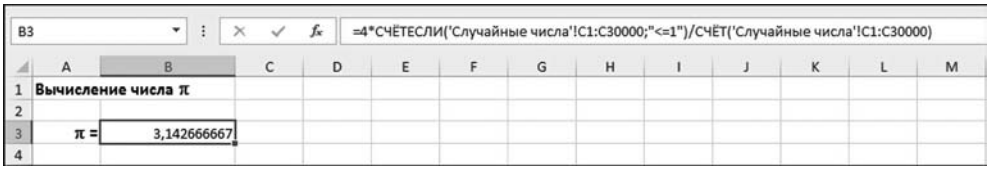


Рис. 7.39
Вычисление значения числа π

На рисунке 7.38 показан уже готовый результат. Фигурные скобки вокруг формулы в строке формул являются признаком формулы массива и добавляются автоматически.

На заметку

Фигурные скобки вводить не нужно. Они добавляются автоматически, и это первый и главный признак формулы массива. Особенность этих формул связана со специфическими правилами их вычисления. Операндами в формулах массива могут выступать диапазоны ячеек. При этом соответствующие операции выполняются поэлементно. Результатом формулы массива является, как правило, массив, поэтому вводится такая формула сразу в диапазон ячеек — получается одна формула на все ячейки диапазона. Например, результатом инструкции A1:A30000^2 является массив значений, которые получаются возведением в квадрат значений ячеек диапазона A1:A30000. Аналогично обстоят дела с инструкцией B1:B30000^2. Поэтому в результате формулой =A1:A30000^2+B1:B30000^2 вычисляется массив значений. Каждое значение равно сумме квадратов соответствующих ячеек в диапазонах A1:A30000 и B1:B30000. Значения этого массива-результата поэлементно заносятся в ячейки диапазона C1:C30000.

Теперь наступил черед вычислить число π. Для этого в чистом рабочем листе в ячейку B3 вводим формулу =4*СЧЁТЕСЛИ('Случайные числа'!C1:C30000; "<=1")/СЧЁТ('Случайные числа'!C1:C30000). Результат показан на рисунке 7.39.

На заметку

Формула немного длинновата из-за явной ссылки на рабочий лист, в котором содержатся ячейки для вычислений. Функцией СЧЁТ() в качестве результата возвращается количество ячеек с числовыми значениями в диапазоне, указанном аргументом функции. Функция СЧЁТЕСЛИ() позволяет подсчитать ячейки, значения в которых удовлетворяют некоторому критерию. Диапазон для подсчета ячеек указывается первым аргументом функции, а второй аргумент функции — это проверяемое условие, заключенное в двойные кавычки.

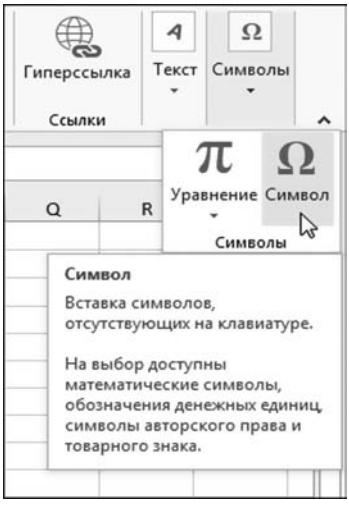


Рис. 7.40
Для ввода символов используем пиктограмму **Символ** в группе **Символы** вкладки **Вставка**

Как видим, число π вычислено с точностью до нескольких знаков после запятой (от сеанса к сеансу значение может меняться). Назвать такую точность очень хорошей вряд ли можно. Для увеличения точности необходимо увеличить (причем значительно) количество генерируемых случайных точек.

На заметку

Выше в рабочем документе Excel в текстовых значениях есть греческая буква π. Для ее ввода в текст можно воспользоваться пиктограммой **Символ** в группе **Символы** вкладки **Вставка** (рис. 7.40).
После щелчка на пиктограмме откроется диалоговое окно **Символ**, в котором собственно и выбирается нужный символ для вставки в текст (рис. 7.41).
Окно содержит две вкладки. На вкладке **Символы** в списке **Шрифт** рекомендуется выбрать шрифт **Symbol**, после чего в палитре символов появится множество интересных значков.

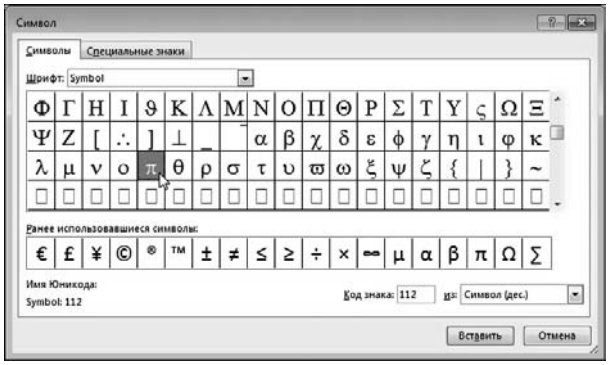


Рис. 7.41
Диалоговое окно для ввода символов

ГЛАВА 8 ДИАГРАММЫ

Карусель наших достижений!..
Прошу, обратите внимание.
Вокруг этого барабана на уровне глаз
крутящихся трудящихся размещены данные,
рисующие достижения нашего района.

Из к/ф «Старый знакомый»

До этого мы обходили вопрос о создании в Excel графиков, диаграмм, схем и прочих интересных и полезных конструкций. Эта глава всецело посвящена методам работы с диаграммами, а также некоторым другим графическим утилитам.

СОЗДАНИЕ ДИАГРАММЫ И ВЫПОЛНЕНИЕ ОСНОВНЫХ НАСТРОЕК

Если б Вы знали, Иван Васильевич,
над каким полезнейшим изобретением я сейчас работаю,
какие опыты ставлю, Вы бы так не говорили.

Из к/ф «Иван Васильевич меняет профессию»

В принципе, как создать простую диаграмму мы уже знаем. Здесь процесс создания диаграмм рассмотрим более подробно и под несколько иным ракурсом. Однако каким бы способом мы ни создавали диаграмму, необходимо иметь набор данных, на основании которых эта диаграмма создается. Обратимся к рисунку 8.1, на котором представлен фрагмент документа, содержащий исходные данные для создания диаграммы.

В данном случае документ содержит условные данные о среднемесячной температуре по каждому месяцу за два года. Именно эти данные предполагается отобразить на диаграмме. Алгоритм действий совсем несложный — все начинается с выделения диапазона ячеек, на основе которого создается диаграмма.

На заметку

Приложение Excel достаточно эвристически подходит к процессу создания диаграмм. Например, если перед созданием диаграммы выделить всего одну ячейку в таблице данных или рядом с ней, то, скорее всего, таблица, на основе которой создается диаграмма, будет определена автоматически.

Но поскольку мы вознамерились «идти в обход», то в обход и пойдем — выделяем ячейку подальше от таблицы данных и в раскрывающемся списке пиктограммы **Гистограмма** (группа **Диаграммы** вкладки **Вставка**) выбира-

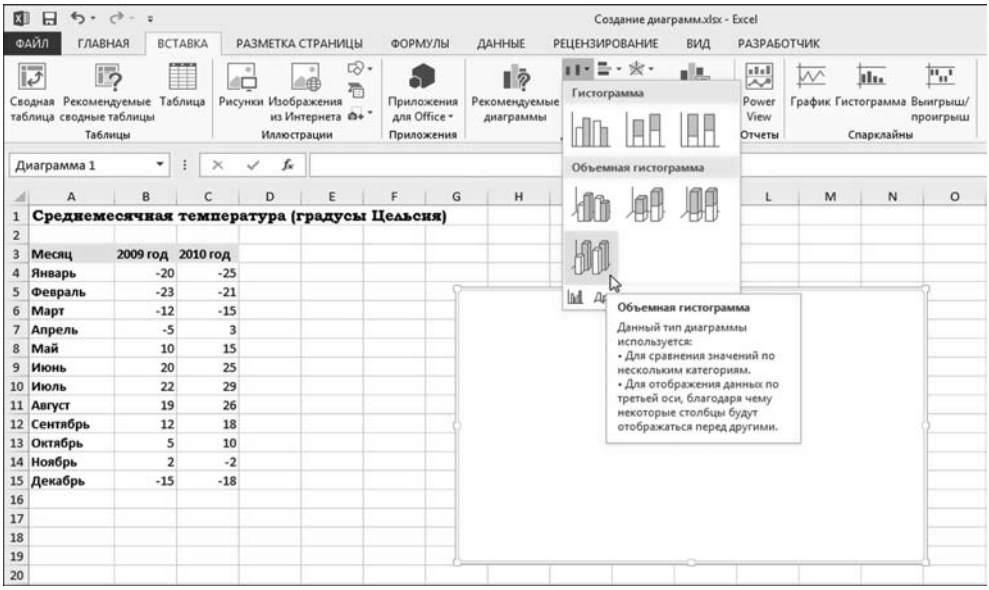


Рис. 8.1
Документ с данными для создания диаграммы и выбор типа создаваемой диаграммы

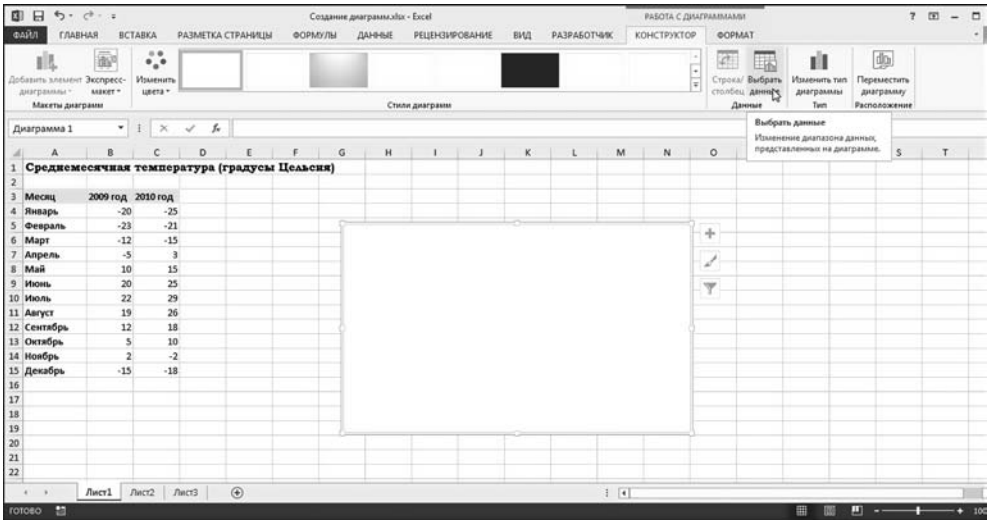


Рис. 8.2
Создана пустая диаграмма

ем тип создаваемой диаграммы (например, создаем объемную гистограмму), как показано на рисунке 8.1. Поскольку при этом в рабочем документе выделена отдаленная пустая ячейка, то нет ничего удивительного в том, что в результате создается пустая диаграмма — фактически в рабочий документ

добавляется только область диаграммы и при этом там ничего интересного или полезного не отображается (рис. 8.2).

На заметку

В Excel 2013 диаграммы получили новые элементы настройки и управления — речь идет о трех пиктограммах, отображаемых справа вверху от области диаграммы. Щелчок на каждой из этих пиктограмм приводит к раскрытию списка с командами или иными элементами управления, которые делают работу с диаграммой простой, удобной и понятной.

Теперь наша задача состоит в том, чтобы добавить в диаграмму те компоненты, которые собственно и делают диаграмму диаграммой. Поэтому мы выделяем диаграмму (если она до этого не была выделена). При выделенной диаграмме в ленте появляется дополнительная вкладка **Работа с диаграммами**. У этой вкладки есть две внутренние вкладки (в версии Excel 2010 таких внутренних вкладок не две, а три). Нас в данном случае интересует внутренняя вкладка **Конструктор** — в группе **Данные** необходимо щелкнуть на пиктограмме **Выбрать данные** (рис. 8.2).

На заметку

Здесь мы разделили непосредственно процесс создания диаграммы и определения данных, на основе которых диаграмма создается. Абсолютно так же выполняется редактирование уже созданной диаграммы, когда необходимо изменить структуру данных для диаграммы.

В результате наших действий открывается диалоговое окно **Выбор источника данных**, представленное на фоне рабочего документа на рисунке 8.3.

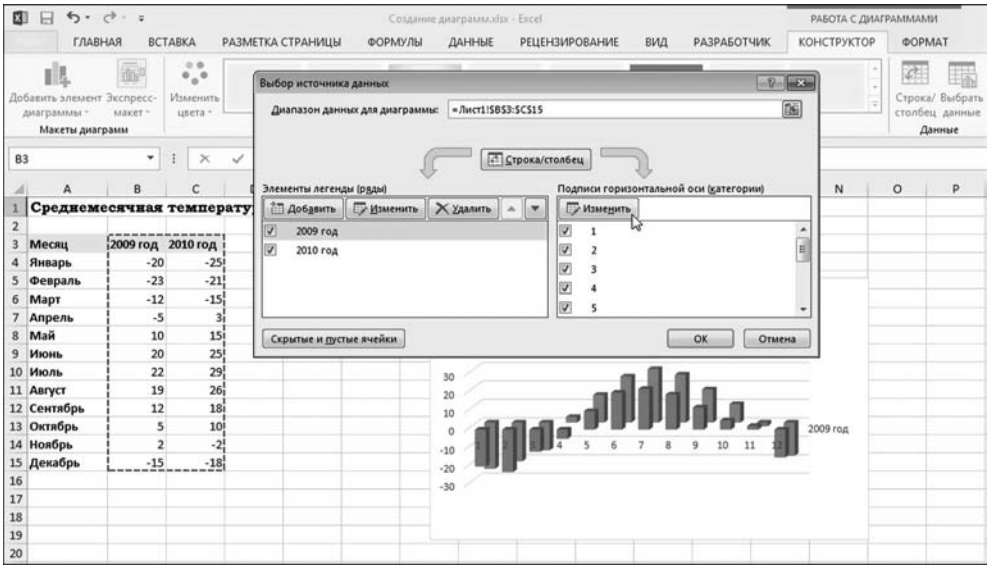


Рис. 8.3
Выбор источника данных для диаграммы

В окне **Выбор источника данных** в поле **Диапазон данных для диаграммы** необходимо указать диапазон ячеек, на основе которого создается диаграмма. В данном случае это диапазон ячеек B3:C15.

На заметку

Как только мы укажем в поле **Диапазон данных для диаграммы** диапазон ячеек, в области диаграммы автоматически начнется отображение результата. Другими словами, вносимые в настройки диалогового окна **Диапазон данных для диаграммы** изменения в интерактивном режиме применяются к диаграмме, что весьма удобно. Кроме того, поле **Диапазон данных для диаграммы** функционирует в режиме указателя (или режиме поля ввода) — вместо того, чтобы вводить адрес ячеек вручную, соответствующий диапазон можно выделить прямо в рабочем документе.

Также желательно добавить адекватные подписи у координатных осей (по умолчанию это ряд натуральных чисел — в данном случае от 1 до 12). Для этого в области **Подписи горизонтальной оси (категории)** щелкаем кнопку **Изменить** (рис. 8.3). В результате откроется диалоговое окно **Подписи оси**, в поле **Диапазон подписей оси** которого указываем диапазон ячеек A4:A15 (рис. 8.4).

На заметку

Поле ввода **Диапазон подписей оси** также функционирует в режиме указателя (поля ввода).

На рисунке 8.5 показано диалоговое окно **Выбор источника данных** со всеми выполненными настройками.

На заметку

Помимо уже знакомых нам функциональных элементов, диалоговое окно **Выбор источника данных** содержит ряд полезных кнопок. В первую очередь

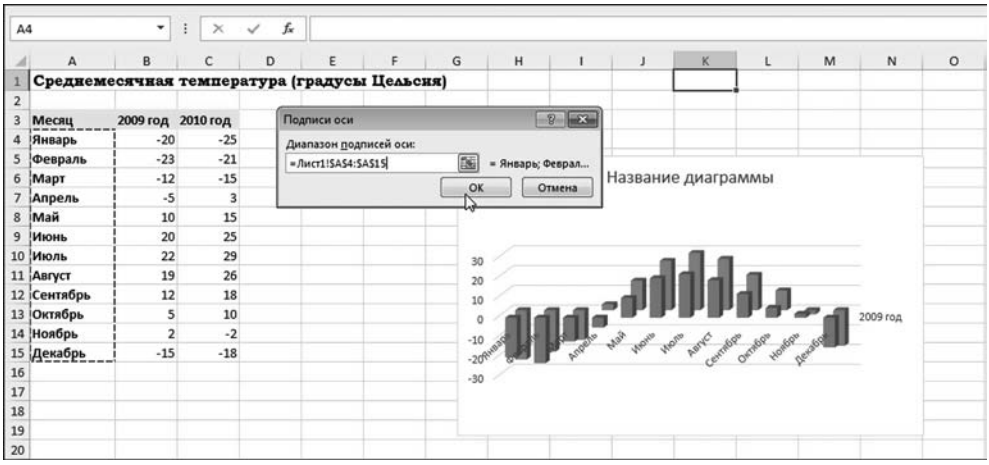


Рис. 8.4
Выбор диапазона ячеек для отображения подписей по осям диаграммы

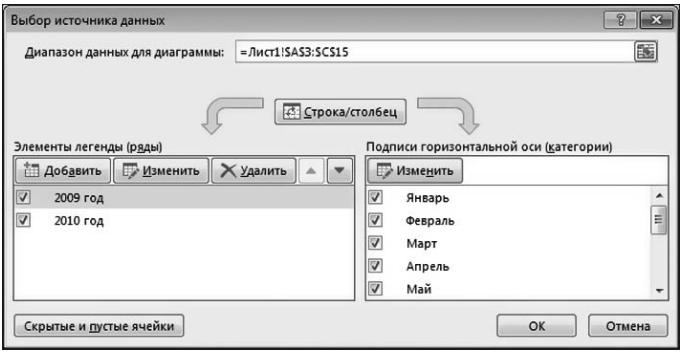


Рис. 8.5
Окно **Выбор источника данных** с выполненными настройками

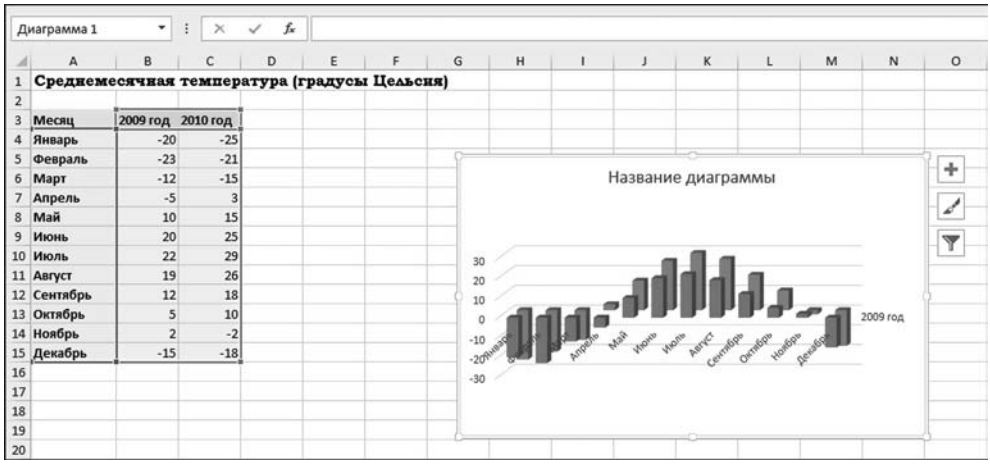


Рис. 8.6
В рабочем документе создана такая диаграмма

стоит обратить внимание на область **Элементы легенды (ряды)**, которая содержит список рядов данных, отображаемых в диаграмме. Ряды данных можно редактировать, удалять или добавлять новые. Для добавления ряда данных используют кнопку **Добавить**. Если выделить ряд данных и щелкнуть на кнопке **Удалить**, соответствующие данные в диаграмме отображаться не будут. Наконец, редактировать данные можно, воспользовавшись кнопкой **Изменить** (предварительно следует выделить ряд данных для изменения). Пиктограммы с изображением мини-стрелок позволяют менять порядок отображения (последовательность) рядов данных.

Кнопка **Скрытые и пустые ячейки** позволяет открыть окно для настройки режима отображения в диаграмме скрытых и пустых ячеек. Кнопка **Строка/столбец** используется в том случае, если нужно поменять местами ряды и категории данных.

После всех выполненных настроек наша диаграмма будет выглядеть так, как показано на рисунке 8.6.

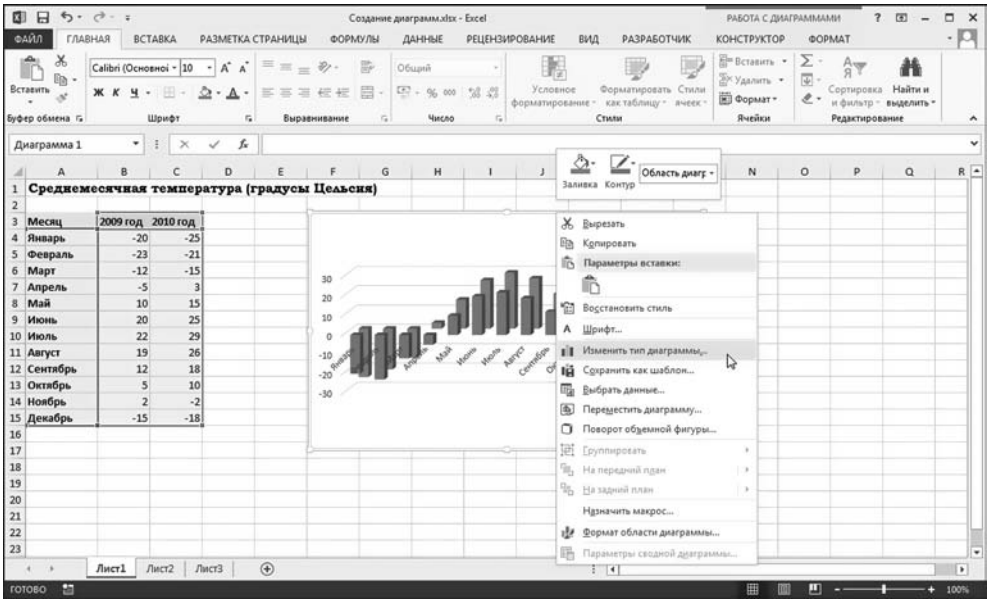


Рис. 8.7
В контекстном меню выбираем команду **Изменить тип диаграммы**

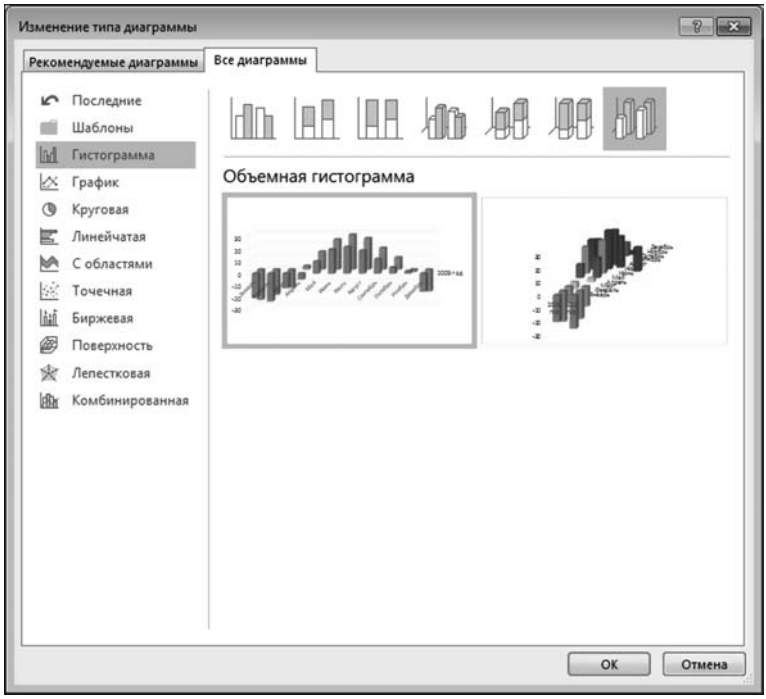


Рис. 8.8
Изменение типа существующей диаграммы в окне **Изменение типа диаграммы**

Тем не менее даже в этом случае диаграмма далеко не идеальна. Благо, Excel предоставляет пользователю широчайшие возможности по настройке вида диаграмм. Например, если уже после создания диаграммы мы захотим изменить ее тип — не беда, ситуацию можно исправить. Скажем, выделим диаграмму, раскроем контекстное меню (щелчок правой кнопкой мыши) и в раскрывшемся списке выберем команду **Изменить тип диаграммы** (рис. 8.7).

Откроется диалоговое окно **Изменение типа диаграммы**, показанное на рисунке 8.8.

Окно достаточно информативное, содержит несколько пиктограмм, но в основном — пиктограммы для выбора типа диаграммы. Для удобства пиктограммы разбиты на группы (тип группы выбирается в списке в левой части окна).

На заметку

Большинство действий по настройке/редактированию диаграмм могут выполняться как с помощью пиктограмм на вкладке приложения, так и с помощью команд контекстного меню. Не является исключением и процедура изменения типа диаграммы. Помимо команды контекстного меню **Изменить тип диаграммы** того же результата можно добиться с помощью пиктограммы **Изменить тип диаграммы** в группе **Тип** дополнительной вкладки **Работа с диаграммами ▷ Конструктор**.

Редко бывает так, что в диаграмме абсолютно все устраивает. Поэтому обычно после создания диаграммы некоторые усилия придется направить на «отшлифовку» результата. «Шлифовать» приходится по-разному и в разных местах. Скажем, в нашей диаграмме совершенно нежелательным образом появляется надпись (название для координатной оси), которую следует удалить. Для этого выделяем соответствующий элемент и в контекстном меню элемента выбираем лаконичную и понятную команду **Удалить** (рис. 8.9).

До этого мы аналогичным образом поступили с названием диаграммы.

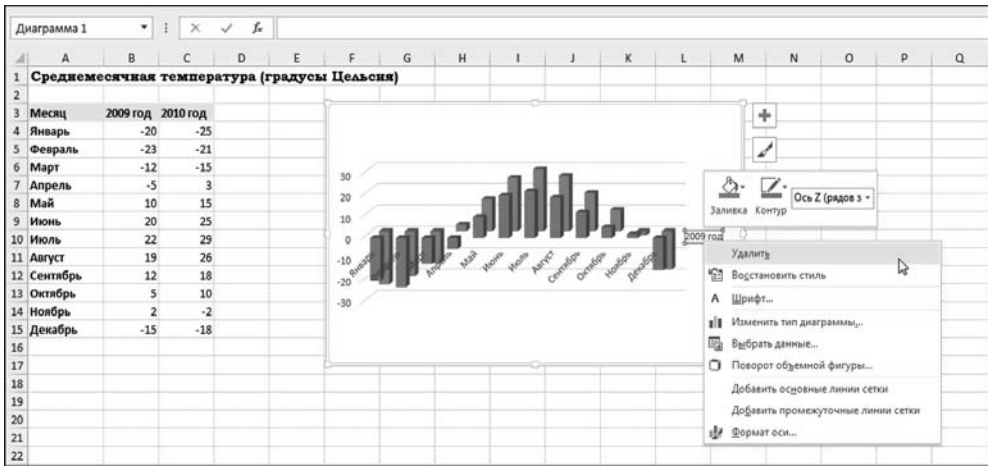


Рис. 8.9
Удаление текстовой надписи

На заметку

Если элемент в области диаграммы в принципе можно редактировать, то его, скорее всего, можно и выделить. Если его можно выделить, то, скорее всего, для этого элемента щелчком правой кнопки мыши можно открыть контекстное меню. Если удалось открыть контекстное меню, то оно, скорее всего, содержит те команды, которые могут применяться по отношению к этому элементу — так сказать, весь список.

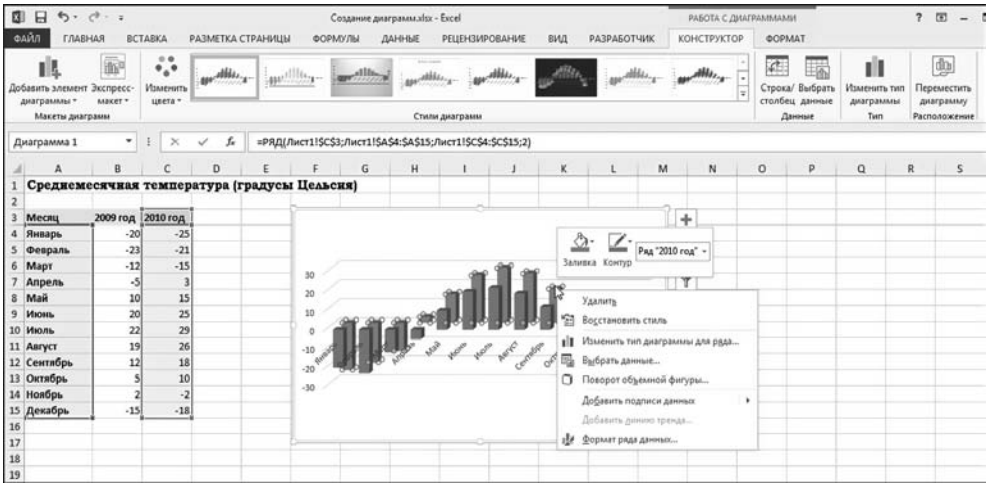


Рис. 8.10
Редактирование отдельного ряда данных для диаграммы

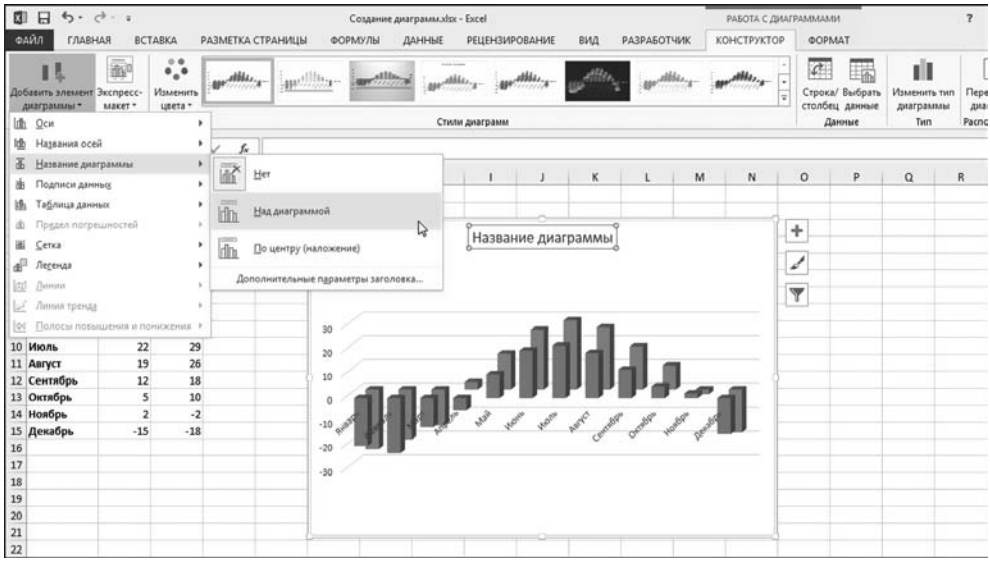


Рис. 8.11
Отображение названия для диаграммы

Здесь мы рассматриваем случай, когда в диаграмме отображается два ряда данных. По умолчанию предлагается определенный способ отображения столбцов диаграммы (например, цвет заливки). Мы не обязаны соглашаться с выбором по умолчанию. Другими словами, способ отображения каждого ряда данных можно редактировать в отдельности. Технически это сводится к возможности выделить отдельный ряд столбиков диаграммы, как показано на рисунке 8.10.

На заметку

На рисунке 8.10 диаграмма несколько увеличилась в размерах. Хочется верить, что читатель без труда разберется с тем, как диаграмма перемещается по рабочему документу или как изменить ее размеры. Подсказка: полезной в этом случае будет мышь. Компьютерная.

Как и в предыдущих случаях, после выделения ряда данных можно воспользоваться контекстным меню. Вместе с тем, выделение элементов диаграммы непосредственно в области диаграммы не всегда удобно. Те же (или похожие) операции могут быть выполнены с помощью пиктограмм дополнительной вкладки **Работа с диаграммами**.

На заметку

Не будет лишним напомнить, что вкладка **Работа с диаграммами** появляется, только если в рабочем окне выделена диаграмма.

Например, мы замечаем, что у нашей диаграммы нет названия (заголовка) — мы его уже успели удалить. Теперь пытаемся исправить эту досадную неприятность и на вкладке **Работа с диаграммами** ▷ **Конструктор** в группе **Макеты диаграмм** выбираем пиктограмму-меню **Название диаграммы**, как показано на рисунке 8.11 (в версии Excel 2010 это была бы пиктограмма **Название диаграммы** в группе **Подписи** на дополнительной вкладке **Макет** вкладки **Работа с диаграммами**).

Пиктограмма **Название диаграммы** скрывает за собой раскрывающийся список, в котором следует сделать выбор — в данном случае выбираем позицию **Над диаграммой** для размещения названия над диаграммой (хотя это совершенно не принципиально). В результате в области диаграммы появится специфическое текстовое поле, текст которого (и параметры формата) можно редактировать (рис. 8.12).

Скорее всего, предлагаемый по умолчанию текст Название диаграммы пользователь захочет заменить на нечто более информативное — например, Среднемесячная температура (см. рис. 8.13).

Здесь же показано, как в диаграмму добавляется легенда: среди команд пиктограммы-меню **Добавить элемент диаграммы** выбираем команду **Легенда** ▷ **Справа**, в результате чего в правой части отображается легенда.

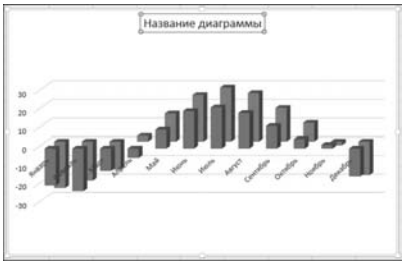


Рис. 8.12
В диаграмму добавлено название, которое предстоит отредактировать

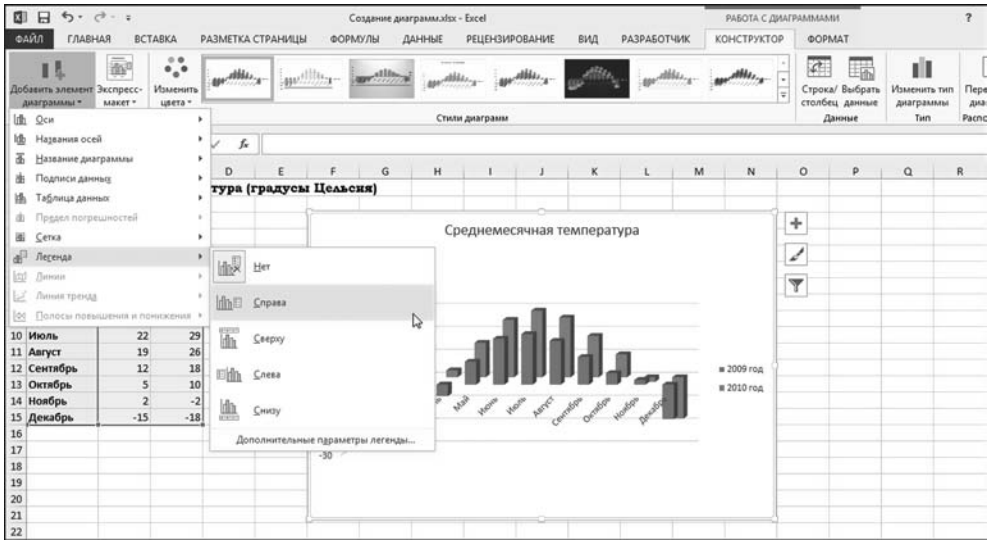


Рис. 8.13
Добавление легенды в диаграмму после добавления и редактирования названия

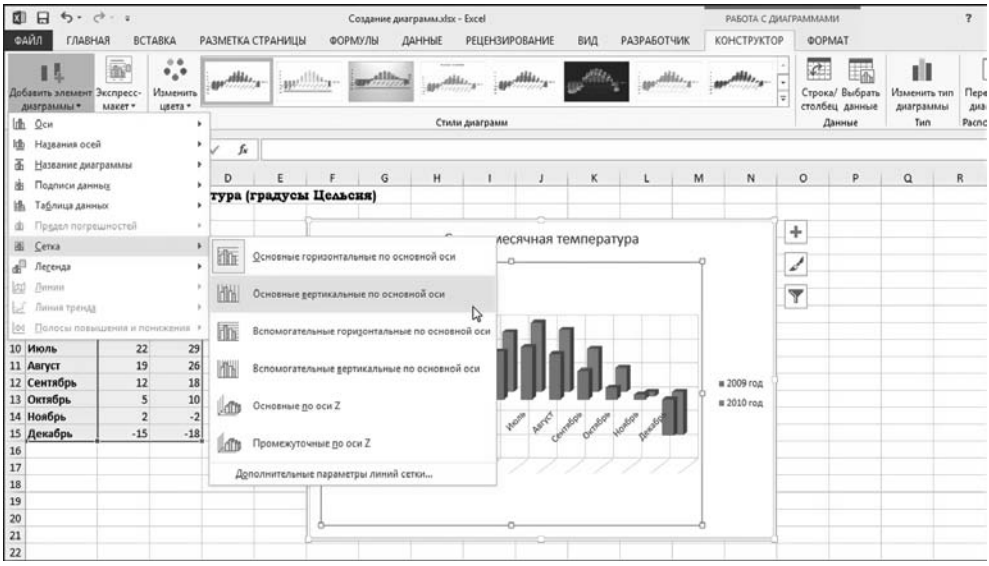


Рис. 8.14
Добавление основных линий вертикальной сетки

На заметку

Редактирование текста названия диаграммы выполняется прямо в соответствующем текстовом окне. Также отметим, что в предыдущей версии Excel 2010 рассматриваемые здесь утилиты распределены не по двум внутренним вкладкам (**Конструктор** и **Формат**), а по трем (**Конструктор**, **Макет** и

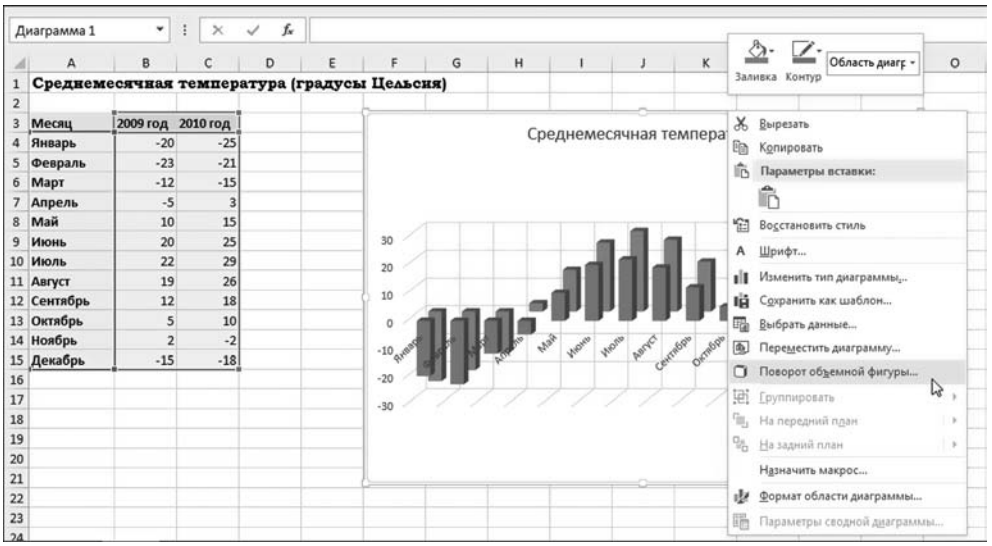


Рис. 8.15
В контекстном меню диаграммы выбрана команда **Поворот объемной фигуры**

Формат). Вместе с тем, несмотря на разное размещение пиктограмм, методы их использования практически неизменны.

На рисунке 8.14 проиллюстрирован процесс добавления в диаграмму вертикальной координатной сетки. Для этого на вкладке **Работа с диаграммами ▷ Конструктор** в раскрывающемся списке пиктограммы-меню **Добавить элемент диаграммы** выбираем команду **Сетка ▷ Основные вертикальные по основной оси**.

Появятся линии вертикальной сетки. Существуют и более эффектные настройки. Например, можем выполнить пространственный поворот диаграммы. Для этого в контекстном меню выбираем команду **Поворот объемной фигуры**, как это показано на рисунке 8.15.

В результате откроется многофункциональное диалоговое окно **Формат области диаграммы**. В окне выполняются самые разные настройки. Нас интересует поворот диаграммы, поэтому в разделе **Поворот объемной фигуры** задаем углы поворота для диаграммы (рис. 8.16).

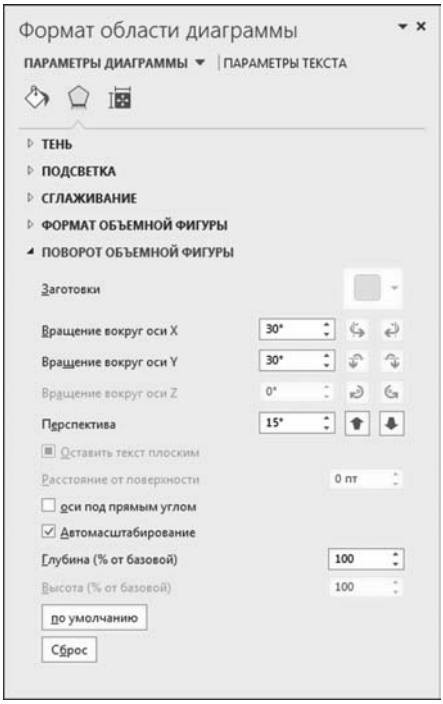


Рис. 8.16
Диалоговое окно **Формат области диаграммы** раскрыто в разделе **Поворот объемной фигуры**

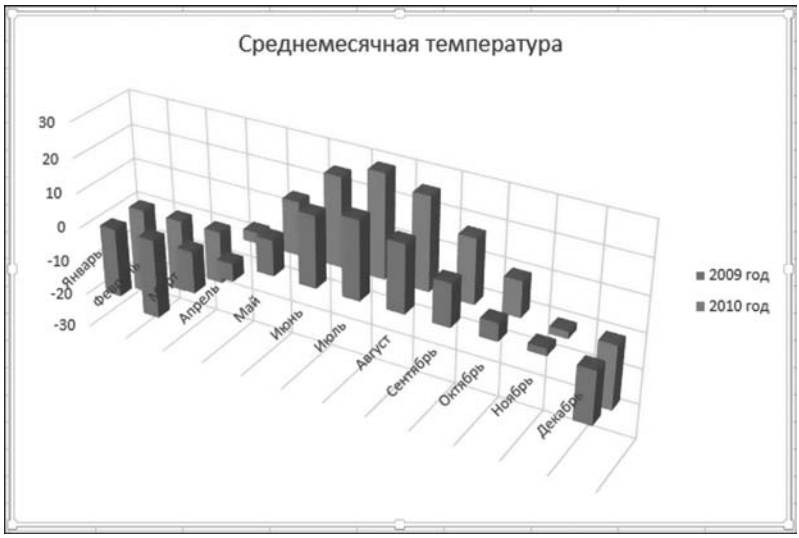


Рис. 8.17
Окончательный вид созданной диаграммы

На рисунке 8.17 показано, как может выглядеть диаграмма после применения таких настроек.

На заметку

Если на практике результат получился несколько иной — не страшно. Здесь важно уловить основной принцип.

СОЗДАНИЕ ГРАФИКА ФУНКЦИИ

Если показания совпадут с теми,
которые я рассчитал теоретически,
Ваше марсианское происхождение будет доказано.

Из к/ф «Тридцать три»

Мы продолжаем рассматривать процесс создания диаграмм. Причем во многом повторим проделанные выше процедуры — фактически, создадим еще одну диаграмму, но только несколько иного типа. А именно, построим график функции. В принципе, ситуация довольно простая. Рассматриваем ее мы исключительно потому, что подобная задача возникает достаточно часто в самых разных контекстах. В отличие от предыдущего случая, здесь начнем не непосредственно с построения диаграммы, а с создания набора данных, по которым затем будем строить график функции (диаграмму).

На заметку

Мы не только будем строить график функции, но еще по ходу процесса проиллюстрируем основные приемы, применяемые при работе с элементами

диаграммы. Поэтому некоторые настройки будут выполняться по нескольку раз различными методами, а элементы будут добавляться, удаляться и снова добавляться, но уже по-другому.

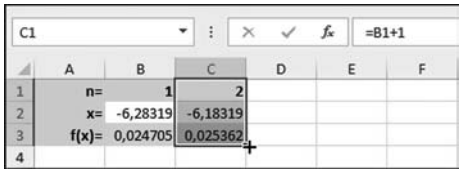


Рис. 8.18
Документ перед началом вычислений и созданием графика

Итак, построим график функции $f(x)=\frac{\cos(x)}{1+x^2}$. Любой график строится по узловым точкам. Чем больше узловых точек, тем точнее будет график. Для создания графика средствами Excel нам необходимо, во-первых, задать значения аргумента в узловых точках и, во-вторых, задать (вычислить) значения функции в узловых точках. Работу начинаем в документе, представленном на рисунке 8.18.

Ячейки A1:A3 содержат текст — названия для массивов данных. В первой строке нумеруются узловые точки. Вторая строка содержит значения аргумента в узловых точках. Третья строка содержит значения функции в узловых точках. Все эти значения вычисляются на основе значений в ячейках B1:B3. На рисунке 8.18 числовыми значениями и формулами заполнены ячейки B1:C3. Значения, вводимые в эти ячейки, перечислены в таблице 8.1.

После того как начальные ячейки заполнены, приступаем к следующему этапу: выделяем ячейки C1:C3 и путем перетаскивания маркера заполнения захватываем область справа от ячеек. На рисунке 8.19 показан документ, в котором таким образом произведены вычисления для 130 узловых точек.

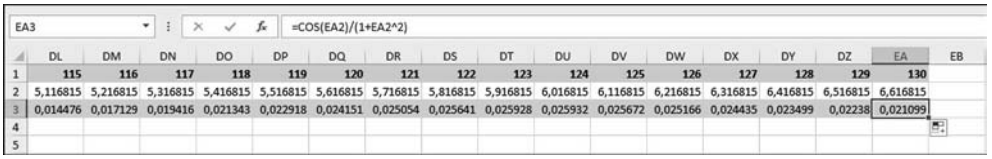


Рис. 8.19
Автоматическое заполнение ячеек документа

Таблица 8.1

Значения в ячейках B1:C3

Ячейка	Значение или формула	Описание
B1	1	Начальный номер узла
C1	=B1+1	Следующий номер узла на единицу больше предыдущего
B2	=-2*ПИ()	Начальное значение аргумента для первого узла
C2	=B2+0,1	Значение аргумента в следующей узловой точке на 0,1 больше значения аргумента в предыдущей узловой точке
B3	=COS(B2)/(1+B2^2)	Значение функции в узловой точке
C3	=COS(C2)/(1+C2^2)	Значение функции в узловой точке — получается копированием формулы из ячейки B3

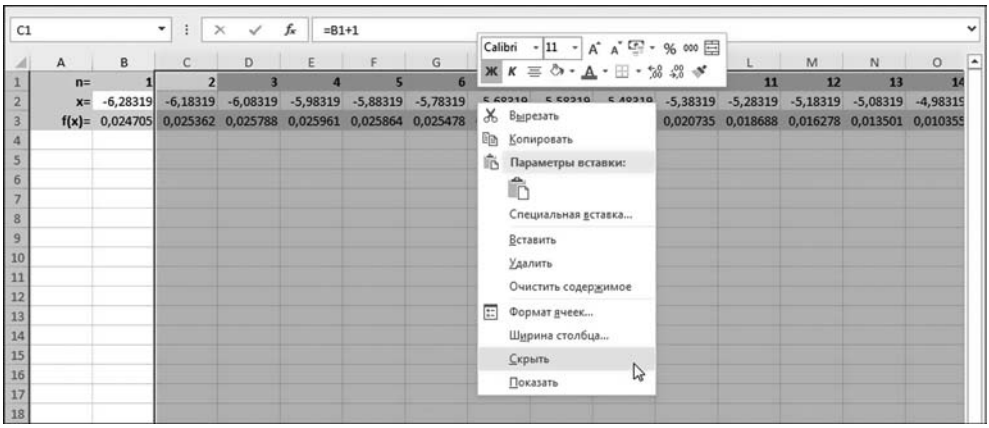


Рис. 8.20
Скрытие столбцов с данными для создания графика

С таким документом реально работать очень сложно. Поэтому скроем столбцы с узловыми точками от 2 до 129 включительно. Процесс сокрытия соответствующих ячеек проиллюстрирован на рисунке 8.20.

На заметку

Выделяем скрываемые ячейки и в контекстном меню щелкаем команду **Скрыть**.

Результат представлен на рисунке 8.21.

Документ выглядит достаточно компактно. Приступаем непосредственно к созданию графика функции. Для этого (при выделенной ячейке EA3) в группе **Диаграммы** вкладки **Вставка** раскрываем список пиктограммы **График** и находим там пиктограмму для создания графика с маркерами (т. е. помимо основной кривой на графике специальными значками-маркерами отображаются узловые точки), как показано на рисунке 8.22.

Результат может показаться странным (рис. 8.23).

Странный вид графика связан с тем, что по умолчанию отображаются (точнее, не отображаются) скрытые ячейки. Нам нужно перейти в режим, при котором эти ячейки в диаграмме будут отображаться. Выделяем диаграмму и щелкаем пиктограмму **Выбрать данные** на вкладке **Работа с диаграммами ▷ Конструктор** (или в контекстном меню выбираем одноименную команду). В результате откроется окно **Выбор источника данных**, в котором необходимо выполнить настройки. Для начала удаляем два лишних ряда данных, которые оказались в списке **Элементы легенды (ряды)**. На рисунке 8.24 показан процесс удаления из списка ряда данных с номерами узлов.

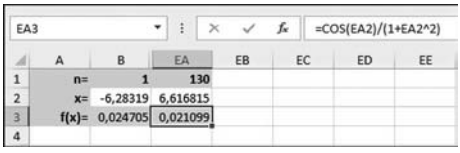


Рис. 8.21
Документ со скрытыми столбцами

Точно так же удаляем ряд данных со значениями аргумента в узловых точках (см. рис. 8.25).

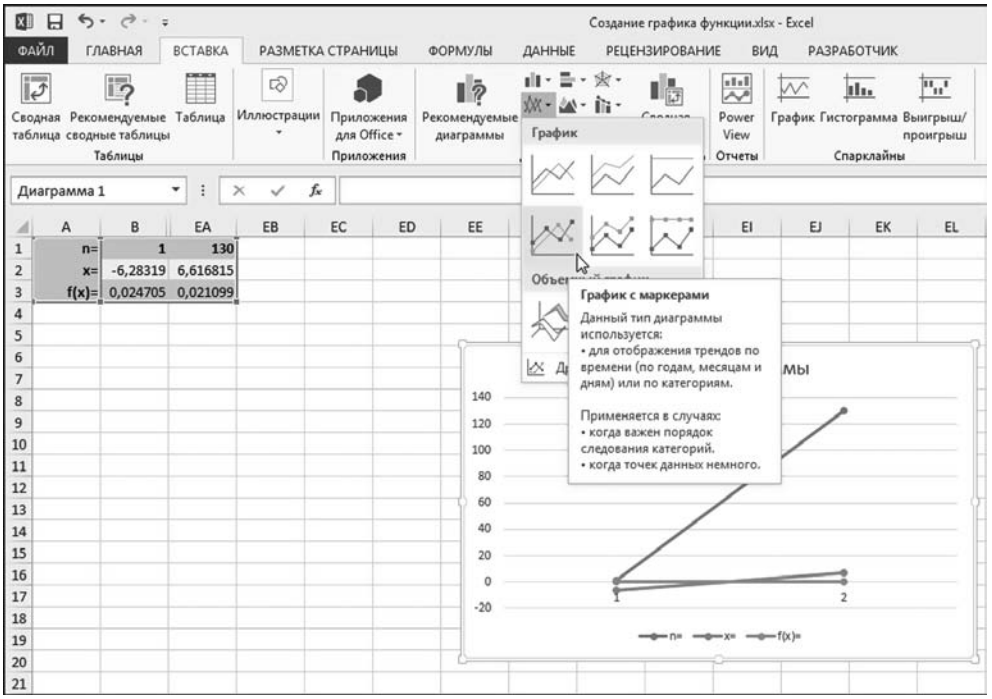


Рис. 8.22
Выбор типа диаграммы (графика) в раскрывающемся списке пиктограммы **График** на вкладке **Диаграммы**

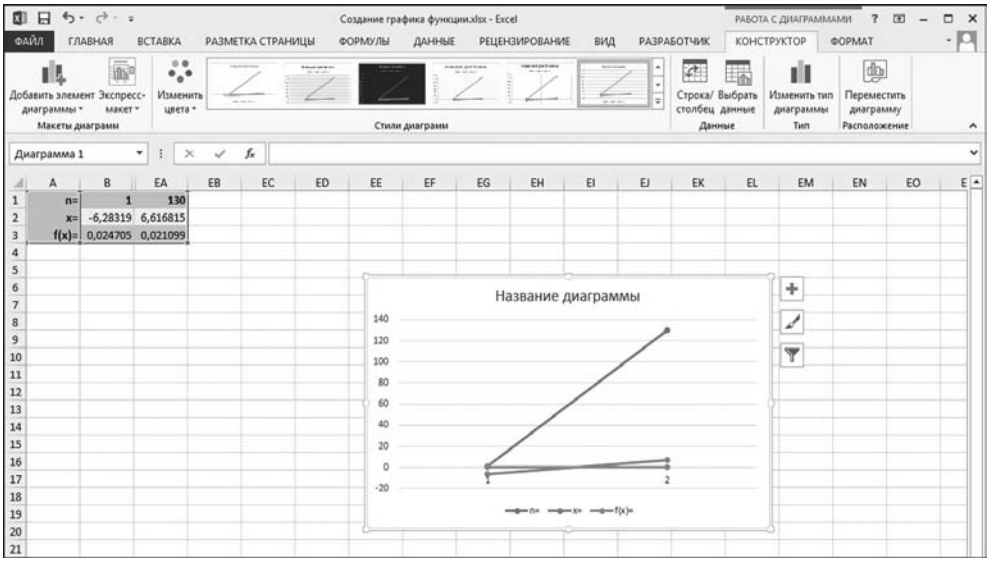


Рис. 8.23
Промежуточный результат создания графика

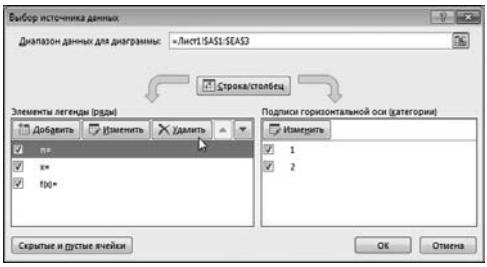


Рис. 8.24
Удаление лишнего ряда данных из графика

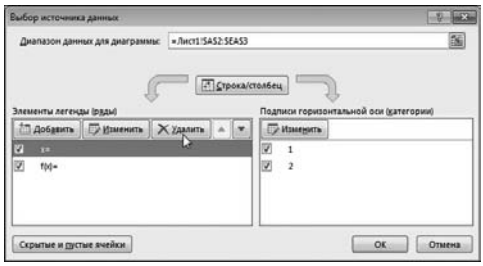


Рис. 8.25
Еще один лишний ряд данных удаляется из графика

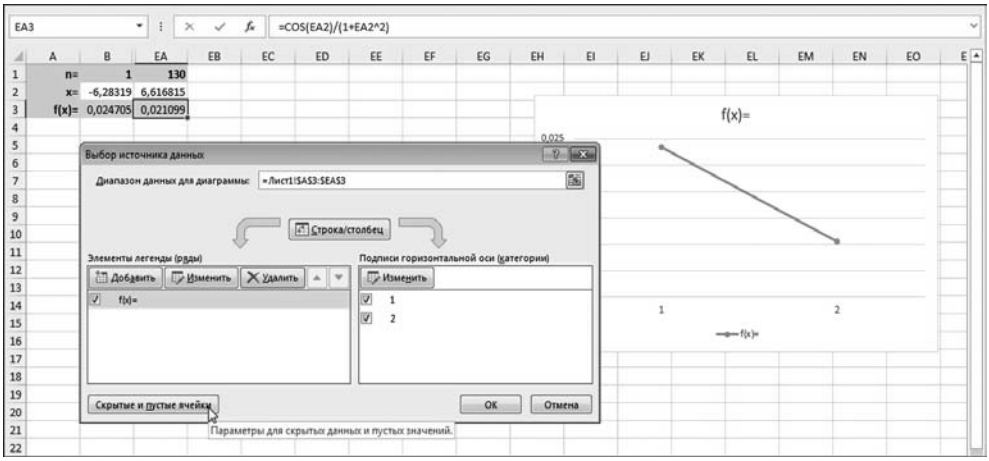


Рис. 8.26
Нажимаем на кнопку **Скрытые и пустые ячейки**

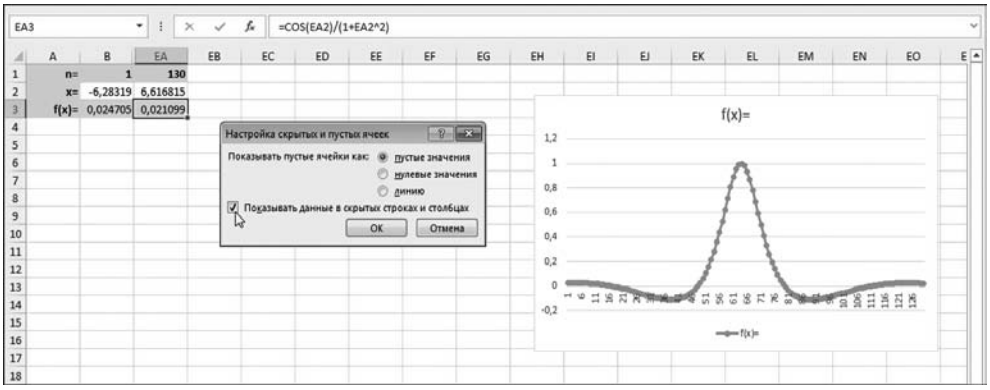


Рис. 8.27
Переход в режим отображения данных в скрытых ячейках

Но это не все. Как отмечалось, по умолчанию скрытые ячейки на графике не отображаются. Чтобы учесть при построении графика и скрытые ячейки, в диалоговом окне **Выбор источника данных** нажимаем на кнопку **Скрытые и пустые ячейки** (рис. 8.26).

Откроется окно **Настройка пустых и скрытых ячеек**, в котором следует установить флажок опции **Показывать данные в скрытых ячейках и столбцах** (рис. 8.27).

В результате график моментально преобразится. Однако настройку его внешнего вида мы еще не закончили. Например, оставленный нами для отображения на графике единственный ряд данных имеет довольно неинформативное название (это автоматически предложенное значение f(x)=). Изменим название для ряда данных. В списке **Элементы легенды (ряды)** выделяем ряд и щелкаем кнопку **Изменить** (рис. 8.28).

Окно **Изменение ряда** содержит, кроме прочего, поле **Имя ряда** для ввода имени ряда. В принципе, там можно указать адрес ячейки, текстовое значение которой будет задавать имя ряда данных. Мы поступим иначе и зададим статическое название, т. е. название, которое существует само по себе, безотносительно к содержимому ячеек в документе. Такое название вводим в виде текстовой формулы ="График функции" (рис. 8.29).

На заметку

В поле **Имя ряда** можно просто ввести текст График функции. В текстовую формулу текст будет преобразован автоматически (после закрытия диалогового окна **Изменение ряда**). Обратите также внимание, что по умолчанию название ряда используется в качестве названия графика.

Теперь наступил черед задать подписи у горизонтальной координатной оси. В принципе в данном случае набор натуральных чисел в качестве подписей не так уж плох, поскольку совпадает с номерами узловых точек. Но лучше все же для подписей использовать значения аргумента в узловых точках. Поэтому в окне **Выбор источника данных** в области **Подписи горизонтальной оси (категорий)** щелкаем кнопку **Изменить** (см. рис. 8.30).

В открывшемся диалоговом окне **Подписи оси** необходимо указать диапазон ячеек со значениями, которые будут использоваться в качестве подписей для горизонтальной координатной оси. Мы указываем диапазон ячеек B2:EA2 (см. рис. 8.31).

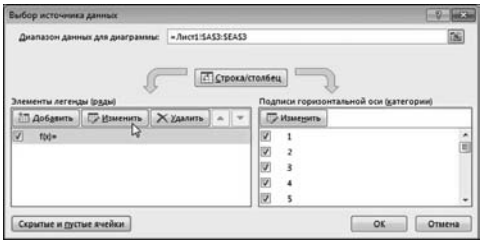


Рис. 8.28
Редактирование ряда данных

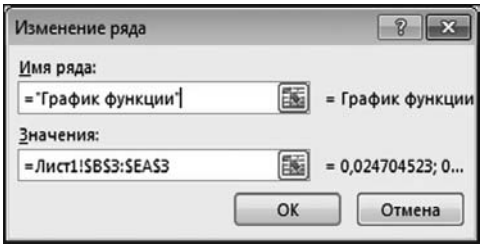


Рис. 8.29
Текстовая формула задает имя ряда

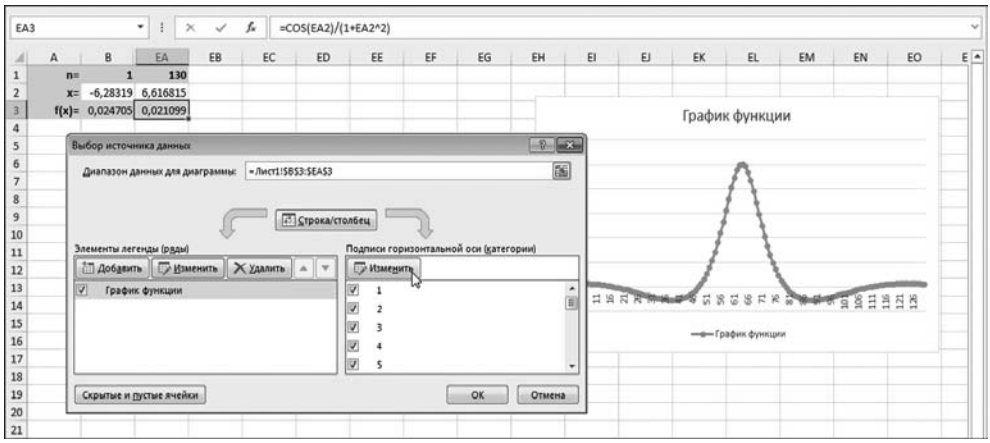


Рис. 8.30
Изменение подписей для оси категорий

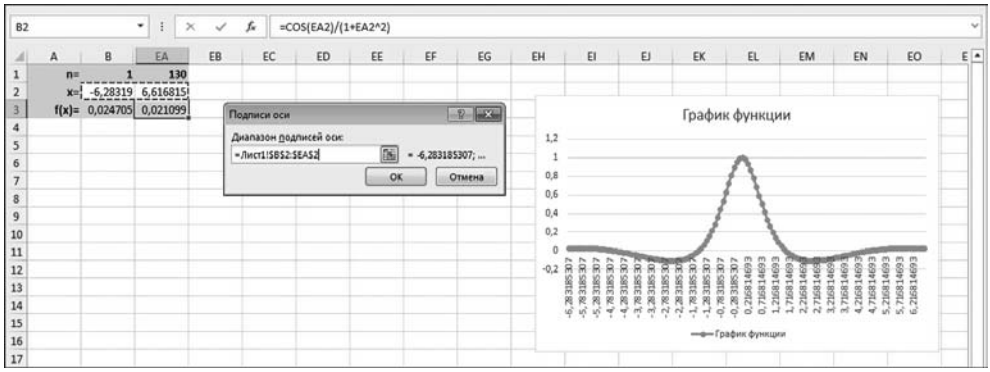


Рис. 8.31
В поле **Диапазон подписей оси** диалогового окна **Подписи оси** указывается диапазон ячеек (со скрытыми ячейками)

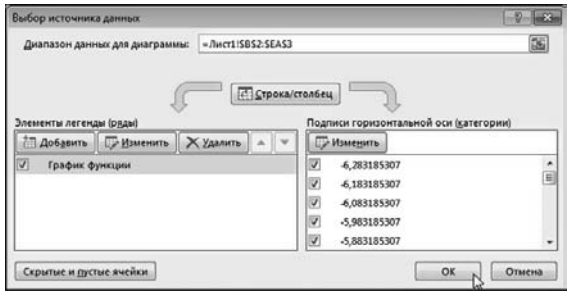


Рис. 8.32
Окончательные настройки в окне **Выбор источника данных**

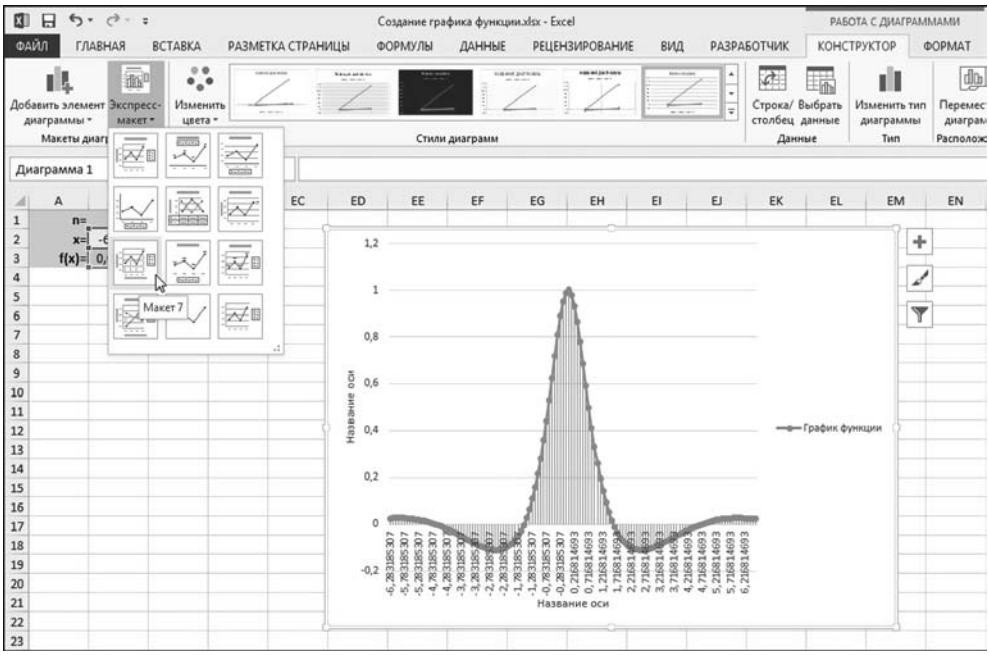


Рис. 8.33
Применение макета к созданному графику

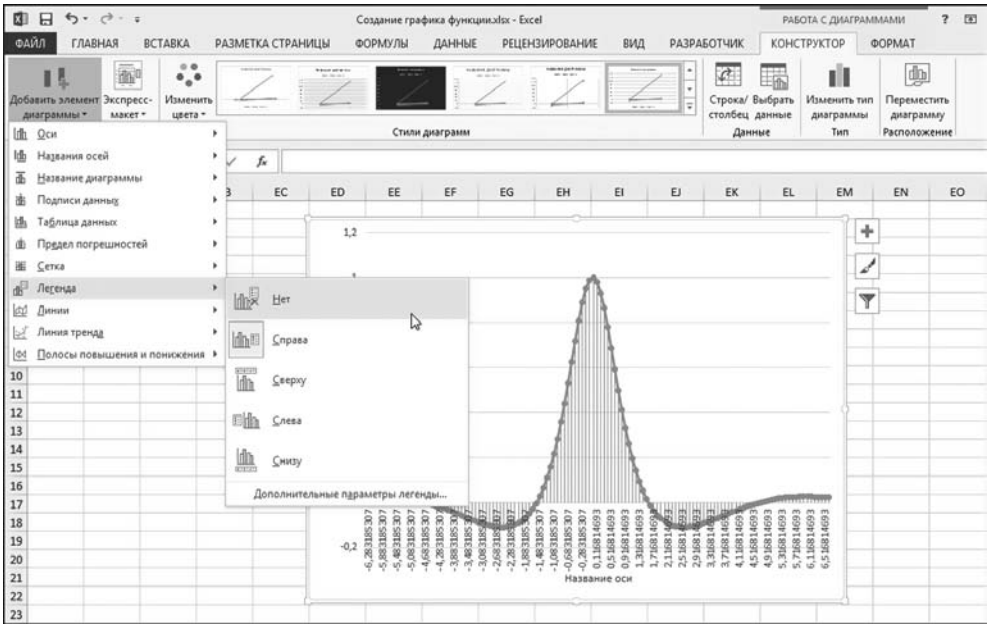


Рис. 8.34
Легенда больше не отображается

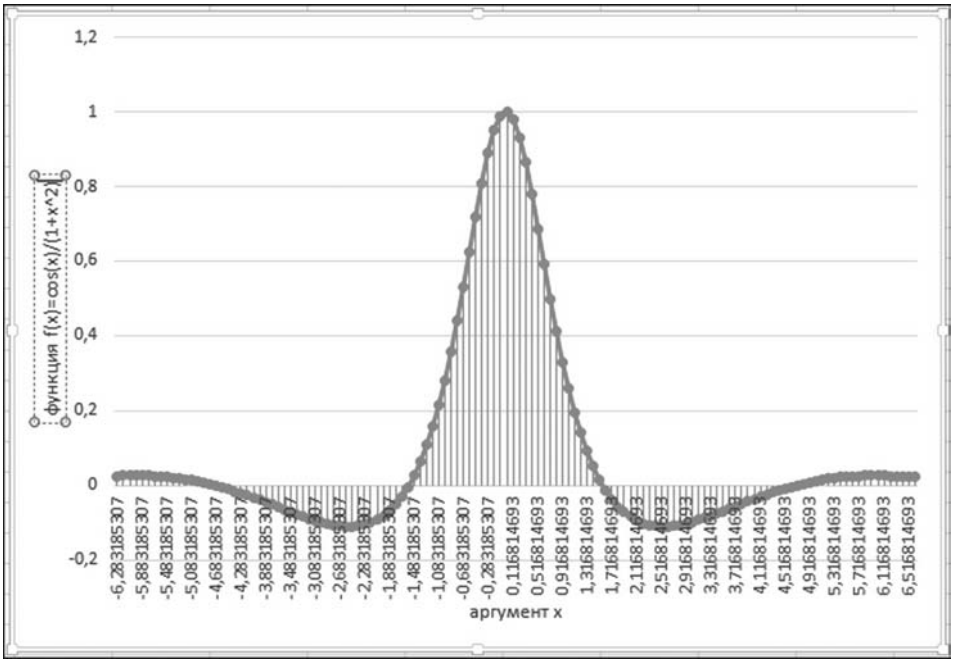


Рис. 8.35
Изменение подписей координатных осей

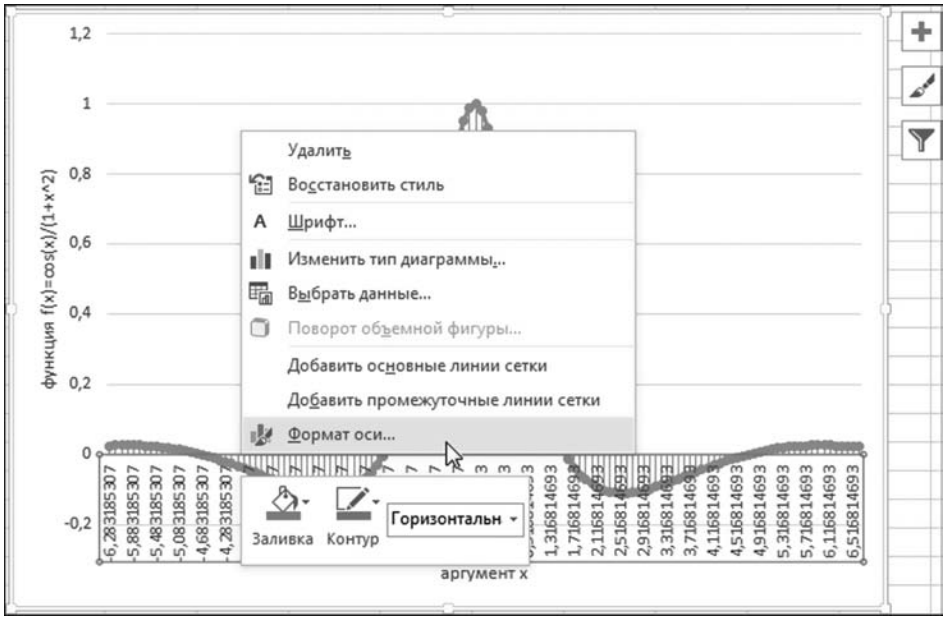


Рис. 8.36
Переход в режим форматирования горизонтальной оси

Полный набор настроек, выполненных в результате в диалоговом окне **Выбор источника данных**, показан на рисунке 8.32.

Результат применения этих (а также некоторых других) настроек к графику можно оценить по рисунку 8.33.

Кроме всего прочего, мы применили к графику макет. Для этого на дополнительной вкладке **Работа с диаграммами ▷ Конструктор** в группе **Макеты диаграмм** находим и раскрываем список пиктограммы **Экспресс-макет**. Выбираем одну из пиктограмм для применения макета. В нашем случае это **Макет 7**. Он подразумевает отображение подписей координатных осей, легенды справа, отсутствие заголовка, а также отображение вертикальных линий от узловых точек на графике до координатной оси. После применения макета мы решаем убрать легенду. С этой целью на дополнительной вкладке **Работа с диаграммами ▷ Конструктор** в группе **Макеты диаграмм** раскрываем список пиктограммы **Добавить элемент диаграммы** и выбираем позицию **Легенда ▷ Нет** (см. рис. 8.34).

Затем редактируем подписи координатных осей. На рисунке 8.35 для горизонтальной оси подпись уже изменена, а для вертикальной как раз показан процесс выделения соответствующей области.

Для форматирования горизонтальной координатной оси в ее контекстном меню выбираем команду **Формат оси** (рис. 8.36).

На заметку

Для редактирования элемента диаграммы можно также выполнить двойной щелчок мышью на этом элементе.

В окне **Формат оси** необходимо выполнить некоторые настройки, как показано на рисунке 8.37.

На заметку

Параметры оси задаются в разделах **Параметры оси**, **Деления** и **Подписи**. В частности, мы задали в поле **Интервал между делениями** значение **5** (количество узловых точек между засечками на координатной оси) и в поле **Число интервалов** переключателя **Интервал между метками** значение **10** (количество позиций между подписями). Таким образом, подписанной будет каждая вторая засечка на координатной оси. Для списка **Положение метки** выбрано положение **внизу** (рис. 8.37). Это означает отображение меток в нижней части области диаграммы.

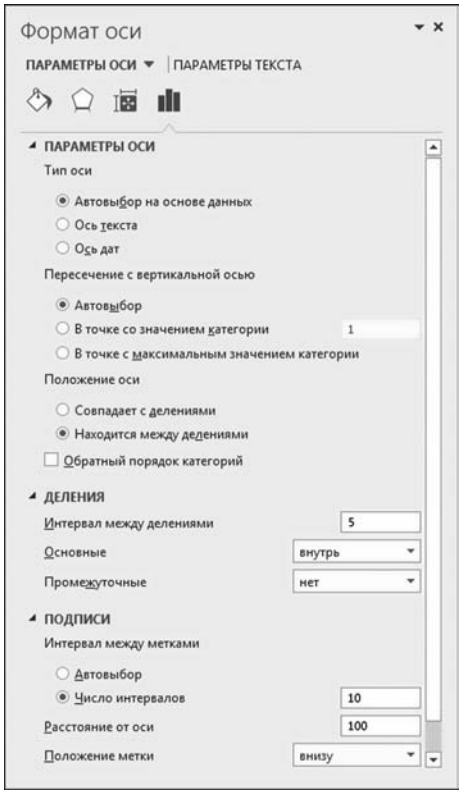


Рис. 8.37
Окно **Формат оси** с настройками для горизонтальной координатной оси

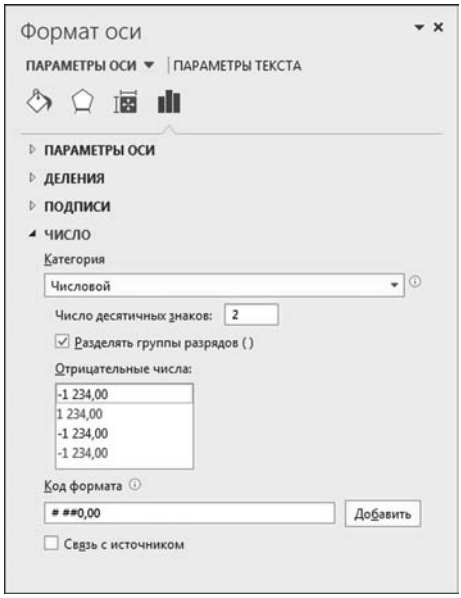


Рис. 8.38
Окно **Формат оси**
с настройками в разделе **Число**

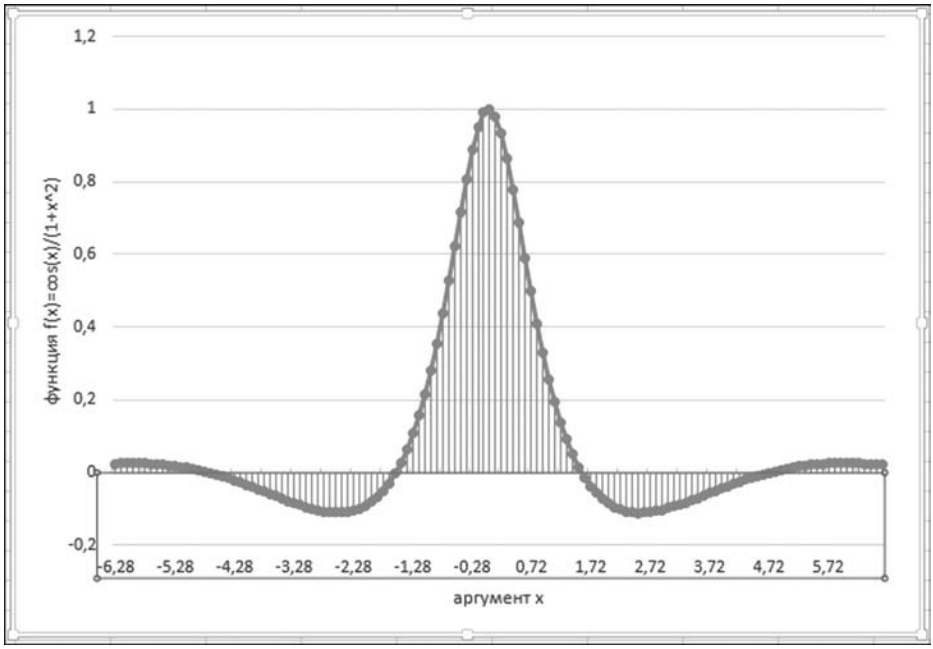


Рис. 8.39
Результат применения настроек для горизонтальной оси

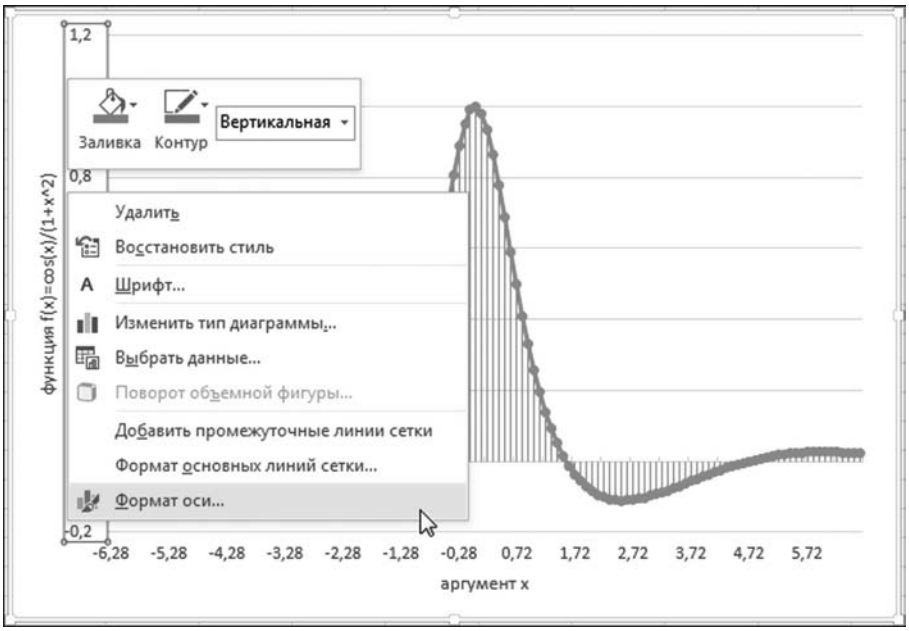


Рис. 8.40
Переход в режим выполнения настроек для вертикальной координатной оси

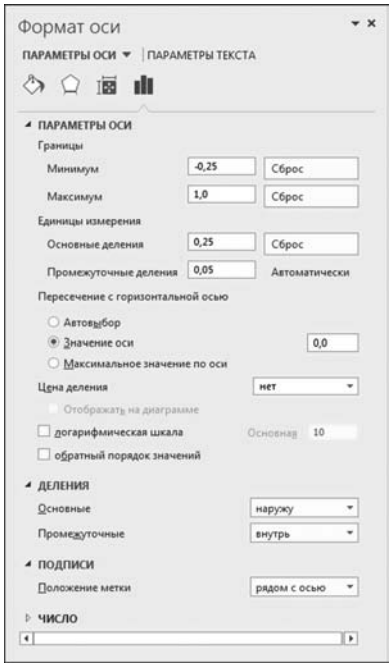


Рис. 8.41
Окно **Формат оси** с настройками в разделах **Параметры оси**, **Деления** и **Подписи**

Несложно заметить, что числовые данные, обозначающие аргумент функции в узловых точках, отображаются не самым лучшим образом: во-первых, их слишком много и, во-вторых, после десятичной запятой отображается слишком много цифр — такая точность нам ни к чему. Формат чисел задается в разделе **Число**. Чтобы числовые подписи отображались с меньшим количеством десятичных знаков после запятой (а именно, с двумя знаками), в разделе **Число** диалогового окна **Формат оси** выбираем формат **Числовой** (рис. 8.38).

После применения описанных выше настроек диаграмма будет выглядеть так, как показано на рисунке 8.39.

Далее настраиваем вертикальную ось. На рисунке 8.40 показано, как в контекстном меню вертикальной координатной оси выбирается уже знакомая нам команда **Формат оси**.

Окно, которое открывается, как и при настройке горизонтальной оси называется **Формат оси**, но имеет несколько иной вид. На рисунке 8.41 показано это окно с раскрытыми разделами **Параметры оси**, **Деления** и **Подписи**.

Мы отказываемся от автоматического определения ряда параметров оси и указываем эти значения явно: **минимальное значение** (установлено -0,25), **максимальное значение** (установлено 1,0), **цена основных делений** (установлено 0,25) и **цена промежуточных делений** (установлено 0,05). Также устанавливаем режим пересечения горизонтальной оси в нулевой точке. Остальные настройки не меняем. На рисунке 8.42 показан окончательный результат — так будет выглядеть диаграмма (график функции).

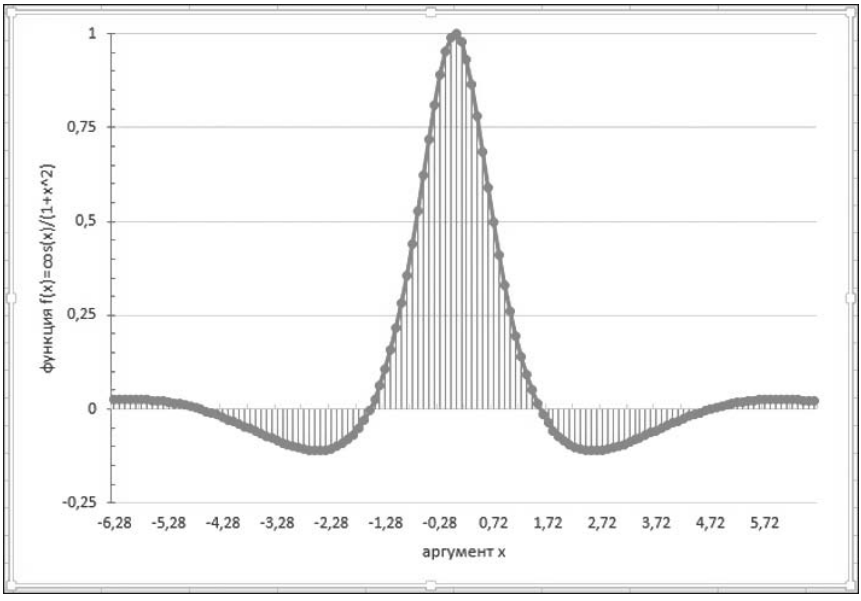


Рис. 8.42
Результат создания графика функции

На заметку

При создании графика функции мы использовали диаграмму из группы **График** (рис. 8.22). На самом деле это не лучший тип диаграммы для создания графика. Проблема в том, что такого типа диаграмма строится по набору равномерно распределенных узловых точек. У нас с этим проблем не возникло, поскольку узловые точки в рассмотренном примере распределены равномерно. Тем не менее на практике подобная ситуация имеет место далеко не всегда. Если нужно строить график функциональной зависимости с неравномерно распределенными узловыми точками (например, при построении графика функции, заданной в параметрическом виде), лучше использовать диаграммы группы **Точечная**.

Еще одна проблема — нединамическое название диаграммы. Другими словами, мы не можем просто так ввести имя диаграммы в ячейку рабочего листа и изменять затем содержимое этой ячейки, меняя тем самым название диаграммы. Для реализации такой задачи придется писать специальный программный код.

НАЗВАНИЕ ДИАГРАММЫ

Придется войти в контакт
с прямым обладателем.

Из к/ф «Тридцать три»

Нам нужно, чтобы название диаграммы можно было менять прямо в рабочем документе, меняя значение в некоторой ячейке. Общее решение состоит в написании программного кода. Необходимо лишь определиться с подробностями. Стратегически возможны два варианта: создание макроса и создание функции пользователя.

На заметку

В этом разделе, немного забегая вперед, мы познакомимся с азами составления программных кодов VBA. Проблеме программирования в VBA полностью посвящена третья часть книги. Поэтому при возникновении принципиальных трудностей читатель может обратиться к главам 9–11. Некоторая непоследовательность изложения материала, связанная с «преждевременным» использованием программного кода, обусловлена тем прикладным значением, которое рассматриваемая здесь задача имеет для работы с диаграммами.

На будущее заметим, что с помощью встроенного редактора программных кодов VBA можно создавать функции (наподобие встроенных функций, используемых нами в рабочем документе) и процедуры. Процедуры принято называть макросами. Функции используют в формулах, процедуры (макросы) запускают на выполнение — например, с помощью специальной кнопки или через стандартные команды графического интерфейса.

Идею с макросом отбрасываем, поскольку макрос нужно запускать, а нам хотелось бы, чтобы изменения вступали в силу автоматически. Следовательно, остается вариант с функцией пользователя. Разумно рассмотреть вариант, когда функция пользователя в качестве результата возвращает текст — название диаграммы.

На заметку

По идее, название диаграммы должно передаваться аргументом функции (равно как и имя диаграммы, чтобы ее можно было идентифицировать). Если функция в качестве результата возвращает название диаграммы, то получается, что функция в качестве результата возвращает свой аргумент. Хотя выглядит это не очень «эстетично», с технической точки зрения проблемы здесь нет.

Программный код нужной нам функции приведен в листинге 8.1.

Листинг 8.1. Функция для определения названия диаграммы

```
Function SetChartTitle(txt As String, obj As String) As String
 With ActiveSheet.ChartObjects(obj).Chart
   .HasTitle = True
   .ChartTitle.Text = txt
 End With
 SetChartTitle = txt
End Function
```

У функции два текстовых аргумента. Первый текстовый аргумент txt задает непосредственно название диаграммы. Второй текстовый аргумент obj функции определяет имя диаграммы в рабочем документе.

На заметку

Здесь имеется в виду имя объекта. То есть диаграмма является объектом, и у этого объекта есть имя. Это имя не следует путать с названием (заголовком) диаграммы, которое (если оно есть) отображается в области диаграммы. По умолчанию вставляемые в рабочий документ диаграммы имеют имена **Диаграмма 1**, **Диаграмма 2** и т. д.

В качестве результата функции возвращается первый текстовый аргумент (последняя команда SetChartTitle=txt в теле функции) — в этом отношении, конечно, интрига практически отсутствует. Тем не менее еще есть что обсуждать. А именно, инструкцией ActiveSheet.ChartObjects(obj).Chart в активном рабочем листе (объект ActiveSheet) в коллекции объектов-диаграмм выбирается диаграмма с именем, которое задано аргументом obj функции (инструкция ChartObjects(obj).Chart).

Свойству HasTitle диаграммы присваивается значение True. Это означает, что названию диаграммы быть! Какому именно названию быть, определяется командой присваивания значения свойству ChartTitle.Text. В данном случае значение определяется переменной txt — так обозначен первый текстовый аргумент функции SetChartTitle. Вот, собственно, и все. Осталось только проверить, как созданная нами функция ведет себя в «полевых условиях».

На заметку

Мало знать, каким должен быть программный код. Не менее важно записать его в «правильное место». Для записи приведенного выше программного кода выполняем такую последовательность действий:

- на вкладке **Разработчик** щелкаем по пиктограмме **Visual Basic** (или нажимаем комбинацию клавиш <Alt>+<F11>);

- в открывшемся окне редактора VBA выбираем команду **Insert ▷ Module**;
- в созданном модуле (по умолчанию название Module1) вводим программный код функции.

Если все же с записью программного кода возникнут проблемы, читатель может обратиться за помощью к следующей части книги.

Обратимся к документу, представленному на рисунке 8.43.

Примечателен он тем, что в активном рабочем листе имеется две диаграммы (имена соответствующих объектов Диаграмма 1 и Диаграмма 2). Диаграммы мы создали заранее, как именно — не очень важно. Важно то, что у обеих диаграмм нет названия (заголовка).

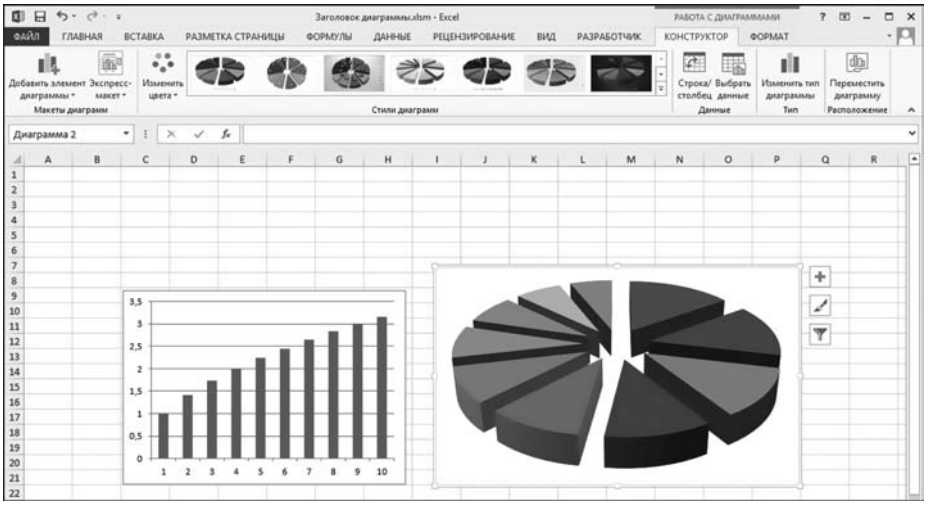


Рис. 8.43
Две диаграммы (объекты с именами **Диаграмма 1** и **Диаграмма 2**) в рабочем листе

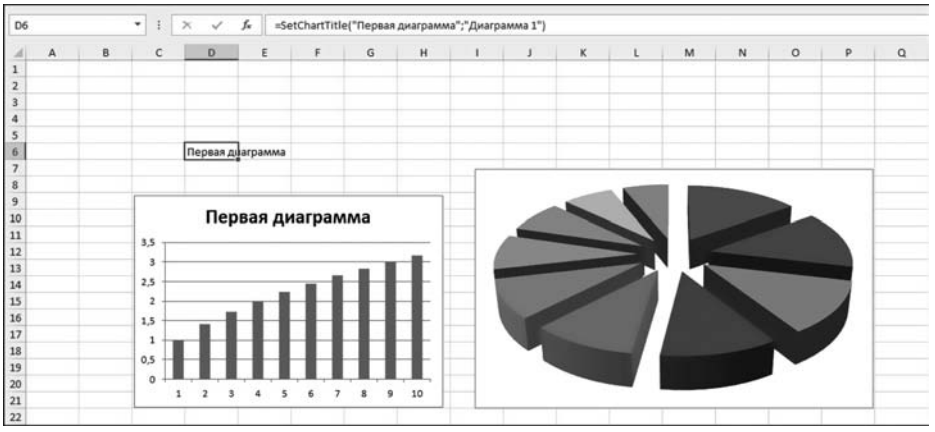


Рис. 8.44
С помощью функции пользователя первой диаграмме добавлено название

На заметку

Имя объекта можно увидеть в поле имен, если выделить объект в рабочем документе. На рисунке 8.43 выделена вторая диаграмма, а в поле имен отображается имя соответствующего объекта — Диаграмма 2.

Мы же, ничтоже сумняшеся, в ячейку D6 вводим формулу =SetChart Title("Первая диаграмма";"Диаграмма 1"). Результат представлен на рисунке 8.44.

Заголовок (название) у диаграммы действительно появился, и именно такой, как мы и ожидали. Теперь вводим в ячейку J5 формулу =SetChart Title("Вторая диаграмма";"Диаграмма 2"), как это показано на рисунке 8.45.

Название появляется и у второй диаграммы. Желающие могут убедиться, что если изменить первый текстовый аргумент функции SetChartTitle() в

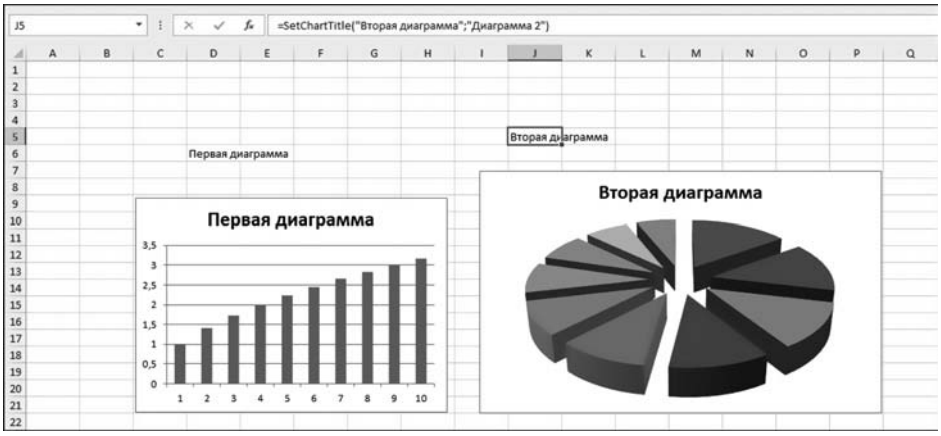


Рис. 8.45
С помощью функции пользователя второй диаграмме добавлено название

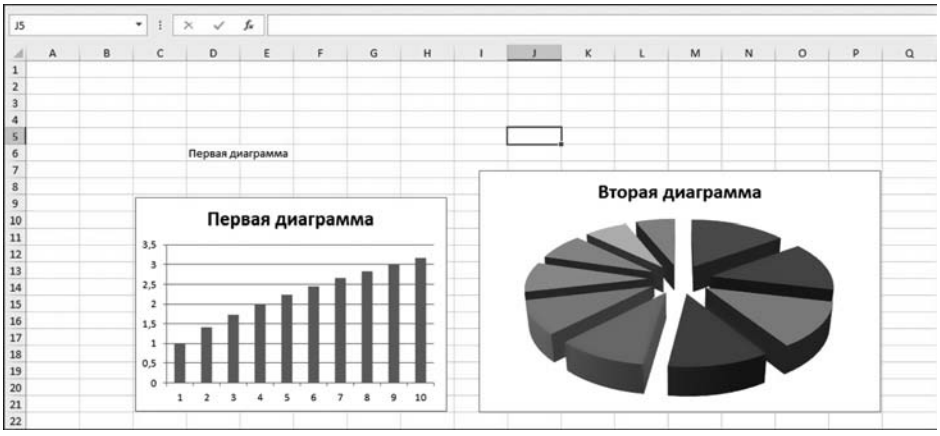


Рис. 8.46
После удаления формулы из ячейки J5 название диаграммы не меняется

любой из введенных в рабочий документ формул, соответствующим образом изменится и имя диаграммы. Более того, если формулу вовсе удалить, название (заголовок) у диаграммы не исчезнет. На рисунке 8.46 представлен результат такого смелого эксперимента.

В данном случае мы удалили формулу из ячейки J5. Эта формула определяла название для второй диаграммы. Несмотря на это, название у диаграммы осталось. Так и должно быть.

ПАРАМЕТРИЧЕСКАЯ КРИВАЯ

— Вы уже испытывали аппаратуру?
— Оф кос, на самом себе!

Из к/ф «Тридцать три»

Здесь мы рассмотрим один пример: покажем, как в Excel можно построить *параметрическую кривую*.

На заметку

Под параметрической кривой мы будем подразумевать кривую на плоскости, уравнение которой задано в параметрическом виде. Параметрическая форма определения функциональных зависимостей на практике используется достаточно часто. Идея определения в параметрическом виде зависимости y от x состоит в том, что задается зависимость каждой из этих переменных от некоторого параметра t. Другими словами, если заданы зависимости $x = f_1(t)$ и $y = f_2(t)$, то система из этих двух уравнений определяет (неявно) зависимость $y(x)$ (или $x(y)$). Такая зависимость называется параметрической. Если в системе уравнений $x = f_1(t)$ и $y = f_2(t)$ путем алгебраических преобразований удается исключить параметр t, получают явное соотношение между переменными x и y.
Если для построения графика для зависимости $y(x)$ нужно предварительно вычислить точки (x_k, y_k) ($k = 0, 1, 2, \ldots, n$), на основе которых строится график, то поступают обычно следующим образом: рассматривают набор значений параметра t_k и затем узловые точки для графика вычисляют в соответствии с соотношениями $x_k = x(t_k) = f_1(t_k)$ и $y_k = y(t_k) = f_2(t_k)$.

С одной стороны, это довольно специфическая задача, которая относится скорее к математическим. Тем не менее мы ее рассмотрим, во-первых, потому, что она позволяет продемонстрировать возможности по созданию в Excel диаграмм специальных типов и, во-вторых, поскольку в основе этой задачи лежит достаточно универсальная идея, имеющая отношение к огромному количеству других прикладных задач.

Мы построим график для функции, которая в полярных координатах задана соотношением $r = \sin(5\varphi)$. Здесь через r и φ заданы полярные координаты — радиус и полярный угол соответственно. Забегая вперед отметим, что должен получиться «цветок» из пяти лепестков.

На заметку

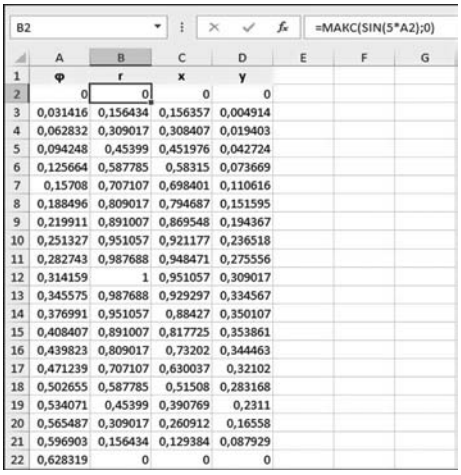
В полярной системе координат положение точки на плоскости задается двумя координатами — расстоянием от начала координат до точки (радиальная координата r) и углом, который задает направление на эту точку (полярная

координата φ). Полярный угол φ может принимать значения в диапазоне от 0 до 2π, а радиальная координата r должна быть неотрицательной. Полярные координаты связаны с декартовыми координатами x и y соотношениями $x = r\cos(\varphi)$ и $y = r\sin(\varphi)$. Эти соотношения нам понадобятся в процессе вычислений.
В роли параметра в данном случае выступает координата φ. Действительно, декартовы координаты точек на кривой определяются соотношениями $x(\varphi) = r(\varphi)\cos(\varphi)$ и $y(\varphi) = r(\varphi)\sin(\varphi)$, где $r(\varphi) = \sin(5\varphi)$.

Для построения кривой табулируем значения полярной координаты φ в диапазоне от 0 до 2π. На основе этих соотношений по формуле $r = \sin(5\varphi)$ вычисляем значения радиальной координаты r.

На заметку

Поскольку радиальная координата r по определению неотрицательна, то вычисления на самом деле производятся так: для фиксированного значения φ значение координаты r вычисляем по формуле $r = \sin(5\varphi)$. При этом, если окажется, что вычисленное значение меньше нуля, то следует полагать $r = 0$.



	A	B	C	D	E	F	G
1	φ	r	x	y			
2	0	0	0	0			
3	0,031416	0,156434	0,156357	0,004914			
4	0,062832	0,309017	0,308407	0,019403			
5	0,094248	0,45399	0,451976	0,042724			
6	0,125664	0,587785	0,58315	0,073669			
7	0,15708	0,707107	0,698401	0,110616			
8	0,188496	0,809017	0,794687	0,151595			
9	0,219911	0,891007	0,869548	0,194367			
10	0,251327	0,951057	0,921177	0,236518			
11	0,282743	0,987688	0,948471	0,275556			
12	0,314159	1	0,951057	0,309017			
13	0,345575	0,987688	0,929297	0,334567			
14	0,376991	0,951057	0,88427	0,350107			
15	0,408407	0,891007	0,817725	0,353861			
16	0,439823	0,809017	0,73202	0,344463			
17	0,471239	0,707107	0,630037	0,32102			
18	0,502655	0,587785	0,51508	0,283168			
19	0,534071	0,45399	0,390769	0,2311			
20	0,565487	0,309017	0,260912	0,16558			
21	0,596903	0,156434	0,129384	0,087929			
22	0,628319	0	0	0			

Рис. 8.47
Документ с числовыми данными для создания параметрической кривой

После того как мы получим табличные значения для координат φ и r, по формулам $x = r\cos(\varphi)$ и $y = r\sin(\varphi)$ вычисляем декартовы координаты x и y. На рисунке 8.47 показан документ (точнее, фрагмент документа) с заполненными числовыми данными в ячейках диапазона A1:D202 (на рисунке видна только часть документа).

Какие значения в какие ячейки нужно вводить, описано в таблице 8.2.

Выделяем диапазон ячеек C2:D202, на вкладке **Вставка** в группе **Диаграммы** раскрываем пиктограмму **Точечная** и выбираем тип диаграммы **Точечная с гладкими кривыми** (рис. 8.48).

В результате будет создана диаграмма, как на рисунке 8.49.

Вид диаграммы можно улучшить. Выполняем такие действия:

- удаляем легенду;
- по горизонтальной и вертикальной координатных осям устанавливаем одинаковые параметры (максимальное значение 1,0, минимальное значение -1,0, цена основного деления 0,25, цена промежуточного деления 0,05), а также устанавливаем режим отображения промежуточных делений (основные по умолчанию и так отображаются);
- устанавливаем одинаковые размеры для диаграммы по высоте и ширине (при выделенной диаграмме можно воспользоваться полями настройки размера в группе **Размер** дополнительной вкладки **Работа с диаграммами ▷ Формат**) — в данном случае ширина и высота установлены 15 см;

Таблица 8.2

Заполнение документа для создания параметрической кривой

Ячейка	Значение	Описание
A1-D1	текст	Диапазон ячеек содержит текстовые обозначения для координат точек на кривой (пара полярных координат в столбцах A и B, а также пара декартовых координат в столбцах C и D)
A2	0	Начальное значение для полярного угла
A3	=A2+2*ПИ()/200	Следующее значение для полярного угла
A4:A202	формулы	Значения полярного угла в узловых точках. Диапазон заполняется копированием формулы из ячейки A3
B2	=МАКС(SIN(5*A2);0)	Вычисление радиальной координаты. В данном случае для вычисления наибольшего значения из двух возможных использована встроенная функция МАКС()
C2	=B2*COS(A2)	Вычисление первой декартовой координаты
D2	=B2*SIN(A2)	Вычисление второй декартовой координаты
B3:D202	формулы	Диапазон ячеек содержит значения в узловых точках радиальной полярной координаты (столбец B), а также декартовых координат (столбцы C и D) и заполняется на основе значений ячеек диапазона B2:D2

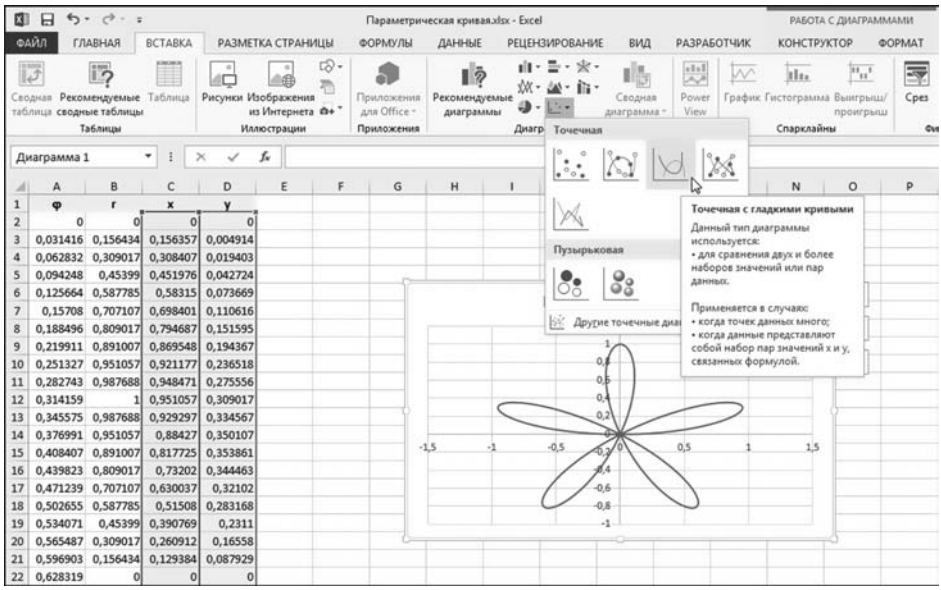


Рис. 8.48
Создание точечной диаграммы

- по горизонтальной оси отображаем основную сетку (по вертикальной оси основная сетка отображается по умолчанию);
- выполняем заливку области диаграммы бледно-голубым цветом.

В результате наша диаграмма примет вид, как показано на рисунке 8.50.

Разумеется, и это далеко не предел совершенства, но, в принципе, все равно неплохо.

Рис. 8.49
Результат создания диаграммы — параметрическая кривая

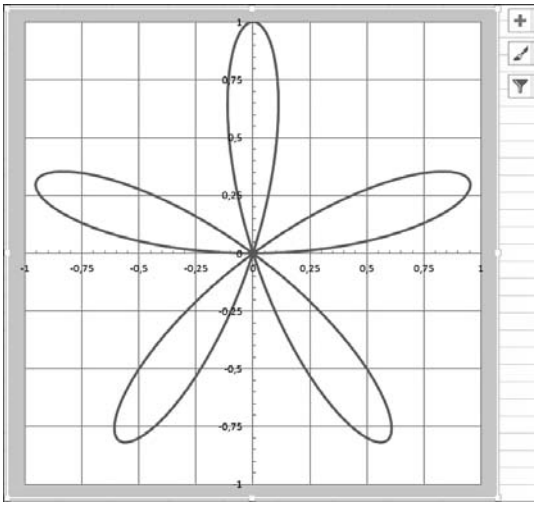


Рис. 8.50
Диаграмма после выполненных настроек

СТАТИЧЕСКИЕ ДИАГРАММЫ

— Да откуда ж ты взялся в палате царской?
Ведь не было тебя.
Батюшка-царь, кто ж это такой?
— А, это приятель Антон Семеныча Шпака.

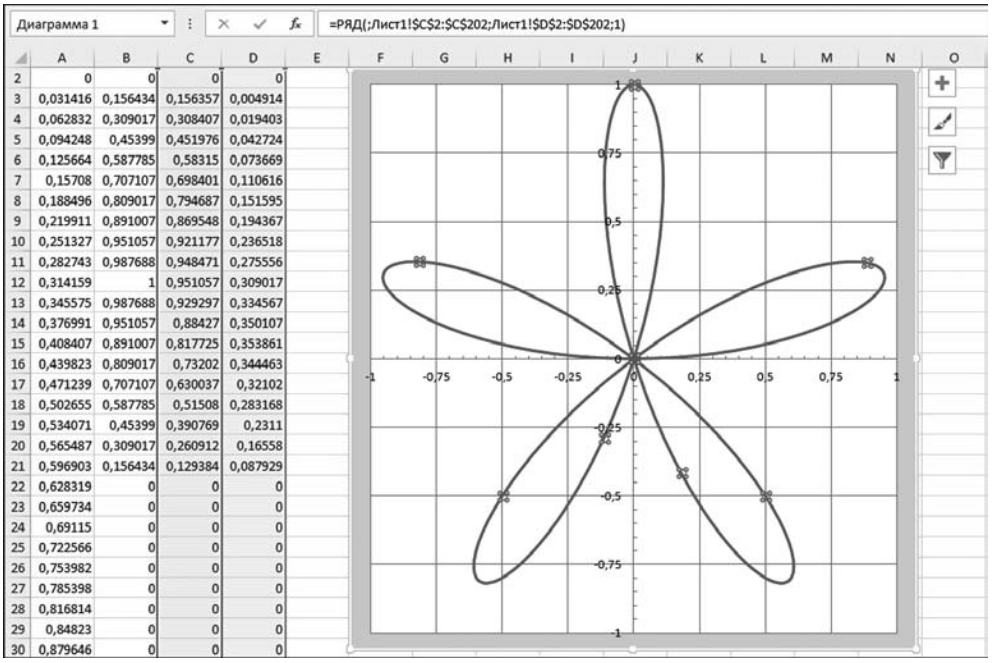
Из к/ф «Иван Васильевич меняет профессию»

Откровенно говоря, основное предназначение диаграмм состоит не просто в том, чтобы отображать в графическом виде информацию, а отображать ее динамически. Другими словами, вся прелесть использования диаграмм связана, в первую очередь, с тем, что изменение исходных данных, на основе которых построена диаграмма, приводит к автоматическому изменению в диаграмме (в соответствии с тем, как изменились данные). Здесь мы погово-

рим о статических диаграммах, которые абсолютно (или почти абсолютно) не чувствительны к изменению данных в рабочем документе. Зачем такие статические диаграммы нужны — вопрос отдельный. Ситуации могут быть самыми разными. Например, нам нужно предохранить диаграмму от возможных изменений. Или мы не хотим хранить в рабочем документе «в открытом виде» большой массив данных. В этом смысле задача по созданию графика функции очень четко попадает в контекст вышесказанного — если нас интересует функциональная зависимость как таковая, то очевидно, что числовые данные, на основе которых строится диаграмма, имеют вспомогательный характер и после того, как кривая (диаграмма) построена, они больше не нужны. Но просто так их удалить нельзя — это незамедлительно скажется на диаграмме, причем изменится она не в лучшую сторону. Придется применить «военную хитрость».

В качестве иллюстрации сделаем из динамической диаграммы, создание которой описано в предыдущем разделе, статическую диаграмму. Для этого в рабочем документе с диаграммой выделяем (в области диаграммы) кривую (или ряд данных, в общем случае), как это показано на рисунке 8.51.

При этом в рабочей области выделяется диапазон ячеек, на основе которых строилась диаграмма, а в строке формул должна появиться формула с функцией РЯД(). У этой функции несколько аргументов и среди них несложно заметить и ссылки на ячейки, содержащие данные для построения



	A	B	C	D
2	0	0	0	0
3	0,031416	0,156434	0,156357	0,004914
4	0,062832	0,309017	0,308407	0,019403
5	0,094248	0,45399	0,451976	0,042724
6	0,125664	0,587785	0,58315	0,073669
7	0,15708	0,707107	0,698401	0,110616
8	0,188496	0,809017	0,794687	0,151595
9	0,219911	0,891007	0,869548	0,194367
10	0,251327	0,951057	0,921177	0,236518
11	0,282743	0,987688	0,948471	0,275556
12	0,314159	1	0,951057	0,309017
13	0,345575	0,987688	0,929297	0,334567
14	0,376991	0,951057	0,88427	0,350107
15	0,408407	0,891007	0,817725	0,353861
16	0,439823	0,809017	0,73202	0,344463
17	0,471239	0,707107	0,630037	0,32102
18	0,502655	0,587785	0,51508	0,283168
19	0,534071	0,45399	0,390769	0,2311
20	0,565487	0,309017	0,260912	0,16558
21	0,596903	0,156434	0,129384	0,087929
22	0,628319	0	0	0
23	0,659734	0	0	0
24	0,69115	0	0	0
25	0,722566	0	0	0
26	0,753982	0	0	0
27	0,785398	0	0	0
28	0,816814	0	0	0
29	0,84823	0	0	0
30	0,879646	0	0	0

Рис. 8.51
При выделении кривой для ряда данных в строке формул отображается формула с системной функцией РЯД()

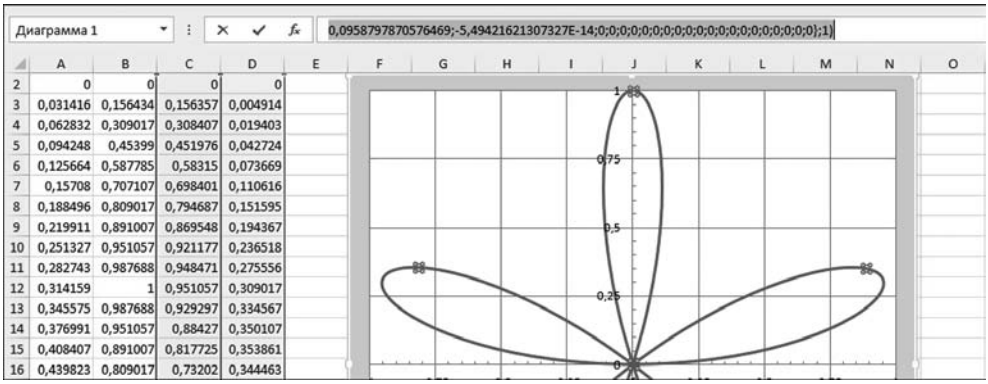


Рис. 8.52
Превращение динамической диаграммы в статическую

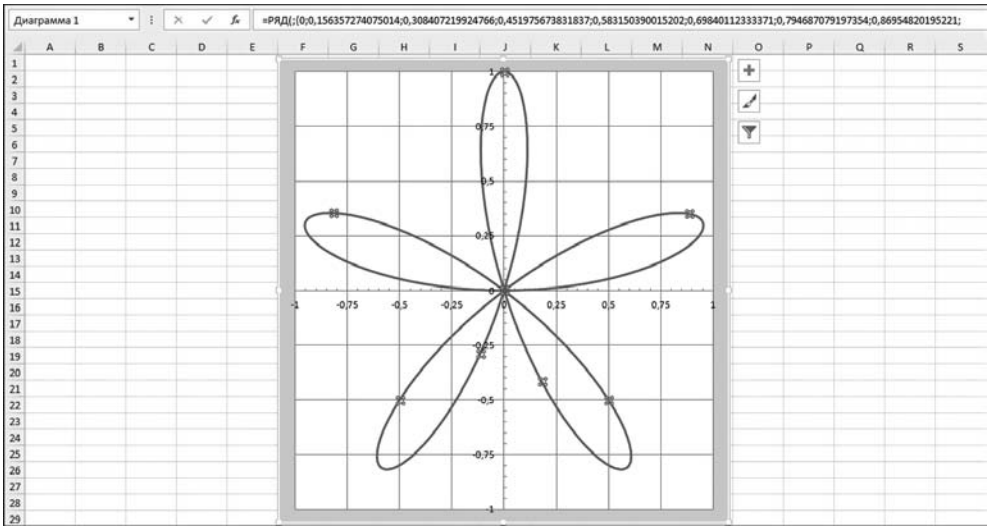


Рис. 8.53
Документ со статической диаграммой

диаграммы. Нас это обстоятельство не останавливает — мы нажимаем клавишу <F2> и переходим в режим редактирования формулы (курсор окажется в строке формул). Затем нажимаем клавишу <F9>. Как следствие, ссылки на ячейки будут заменены фактическими числовыми значениями. Такая ситуация показана на рисунке 8.52.

Нажимаем клавишу <Enter> — статическая диаграмма готова. Внешне она не изменит вид, но теперь диаграмма «нечувствительна» к изменению исходных данных в ячейках документа. На рисунке 8.53 показана ситуация, когда из рабочего документа вообще удалены все значения, на основе которых создавалась исходная динамическая диаграмма. Несложно заметить, что статическая диаграмма совершенно не изменилась.

На заметку

Функция РЯД() является системной и не может использоваться «напрямую». Другими словами, эту функцию нельзя использовать как обычную встроенную функцию Excel, например, для создания диаграммы. Тем не менее, если диаграмма уже создана, путем редактирования формулы с функцией РЯД() можно изменить вид диаграммы. Аргументы функции РЯД() передаются поразному в зависимости от типа диаграммы. В рассматриваемом случае назначение аргументов функции такое:

- имя ряда;
- значения по горизонтальной оси;
- значения по вертикальной оси;
- номер ряда данных (здесь он всего один).

Первый аргумент, определяющий имя ряда, не является обязательным.

КОМБИНИРОВАННЫЕ ДИАГРАММЫ

Известно, что новое содержание требует для себя новой формы. Но, с другой стороны, и новая форма требует для себя нового содержания. Ибо нельзя новое содержание втиснуть в прокрустово ложе старой формы. Так же как и старое содержание нельзя втиснуть в рамки новой формы. Поэтому, как я говорил уже выше, форма не должна быть в отрыве от содержания, т. е. она должна соответствовать ему.

Из к/ф «Тридцать три»

Мы уже знаем, что в Excel можно создавать диаграммы самых разных типов. Мы также в принципе догадываемся, что на одной диаграмме может отображаться несколько рядов данных. Но все намного интереснее: для каждого ряда данных можно (в разумных пределах) задавать свой «диаграммный» тип. Обратимся к примеру. Для иллюстрации этого положения создаем диаграмму с двумя рядами данных. На рисунке 8.54 показан документ с исходными данными и созданной диаграммой.

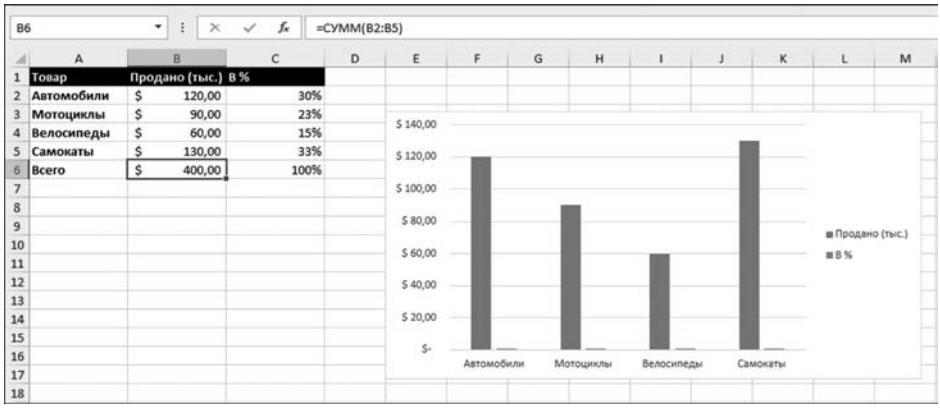


Рис. 8.54
Диаграмма с двумя рядами данных

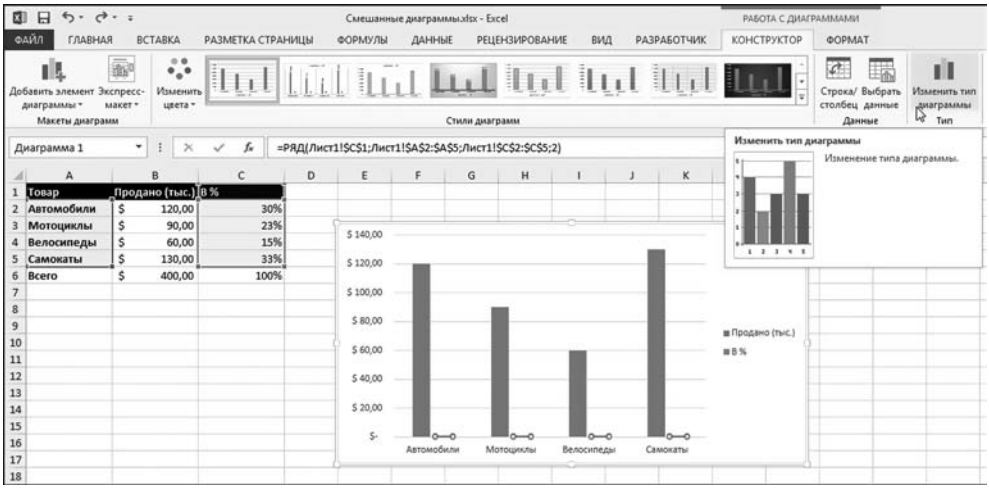


Рис. 8.55
Изменяем способ отображения ряда

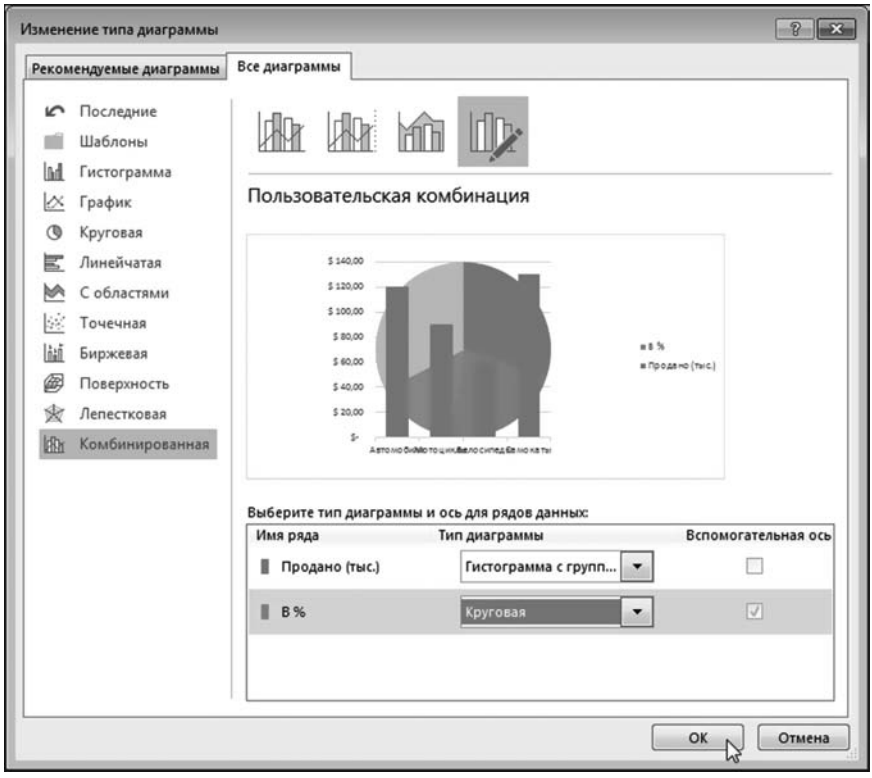


Рис. 8.56
В окне **Изменение типа диаграммы**
настраиваем параметры комбинированной диаграммы

В качестве типа диаграммы мы использовали гистограмму — каждый из двух рядов данных отображается столбиками. Причем специфика одного из рядов такова, что столбики этого ряда на фоне столбиков первого ряда практически не заметны. Один из возможных методов решения проблемы связан с изменением способа отображения второго ряда.

Выделяем диаграмму (при этом «редактируемый» ряд можно не выделять) и щелкаем пиктограмму **Измение типа диаграммы**, как показано на рисунке 8.55.

Откроется окно выбора типа диаграммы **Изменение типа диаграммы**, в котором на вкладке **Все диаграммы** нужно перейти к разделу **Комбинированная** (рис. 8.56).

В правой нижней части вкладки **Все диаграммы** диалогового окна **Изменение типа диаграммы** имеется группа утилит под общим названием **Выберите тип диаграммы и ось для рядов данных**. Чтобы второй ряд данных отображался в виде круговой диаграммы, в раскрывающемся списке напротив этого ряда выбираем пункт **Круговая** (рис. 8.56). В результате получим на одной картинке фактически две диаграммы (рис. 8.57).

На заметку

В более ранней версии Excel (например, в Excel 2010) эта задача решается так: выделяем в диаграмме соответствующий ряд данных и щелкаем (при выделенном ряде) пиктограмму **Изменить тип диаграммы** в группе **Тип** вспомогательной вкладки **Работа с диаграммами** ▷ **Конструктор**. В результате открывается диалоговое окно выбора типа диаграммы, в котором выбирается тип, но только не для всей диаграммы, а для конкретного ряда данных. В Excel 2013 все несколько иначе.

В данном случае созданная круговая диаграмма малоинформативна, поэтому выполним некоторые дополнительные настройки — в контекстном меню для второго ряда (того, что отображается круговой диаграммой) выбираем

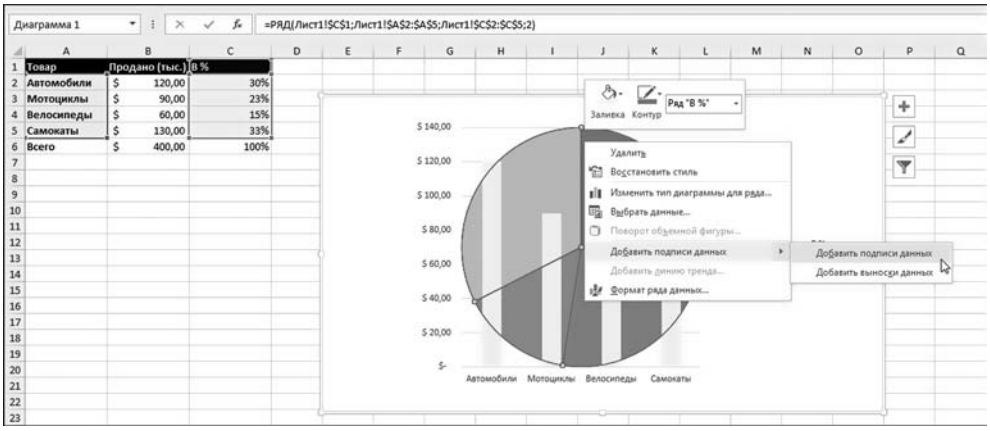


Рис. 8.57
Добавление подписей для ряда, отображаемого круговой диаграммой

команду **Добавить подписи данных** (рис. 8.57). В результате на «дольках» круговой диаграммы отображаются числовые значения ряда данных. Мы хотим добавить туда еще и подписи категорий (названия товаров), поэтому в контекстном меню круговой диаграммы выбираем команду **Формат подписей данных** (рис. 8.58).

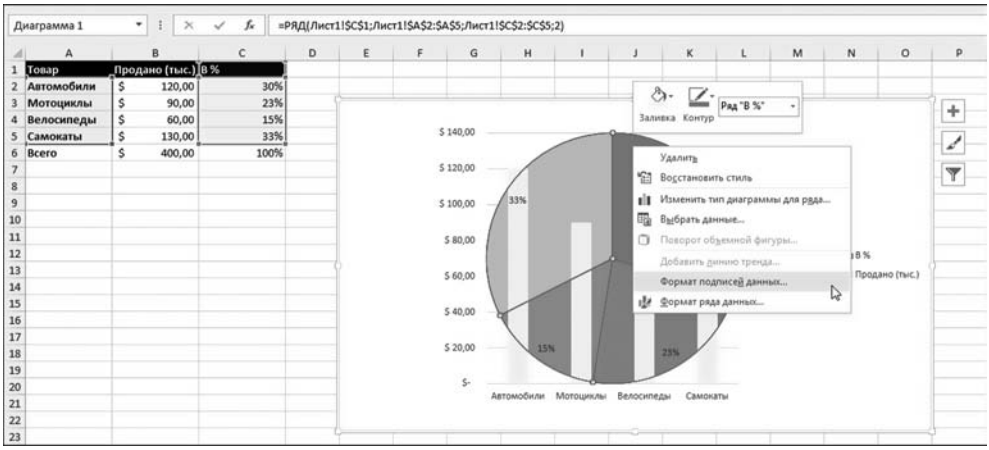


Рис. 8.58
Добавление для круговой диаграммы подписей категорий

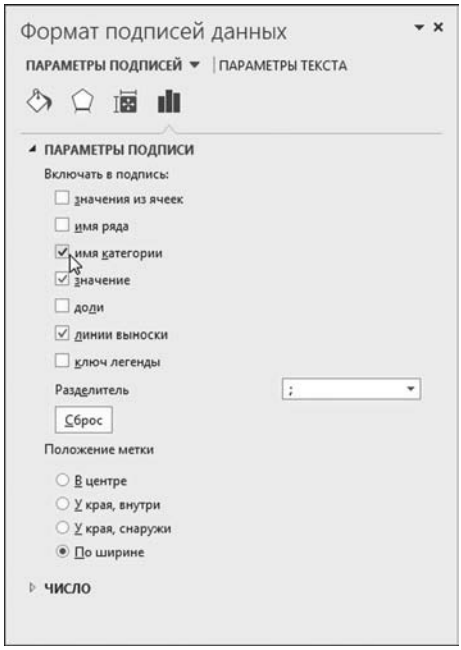


Рис. 8.59
В окне **Формат подписи данных** устанавливаем флажок опции **имена категорий**

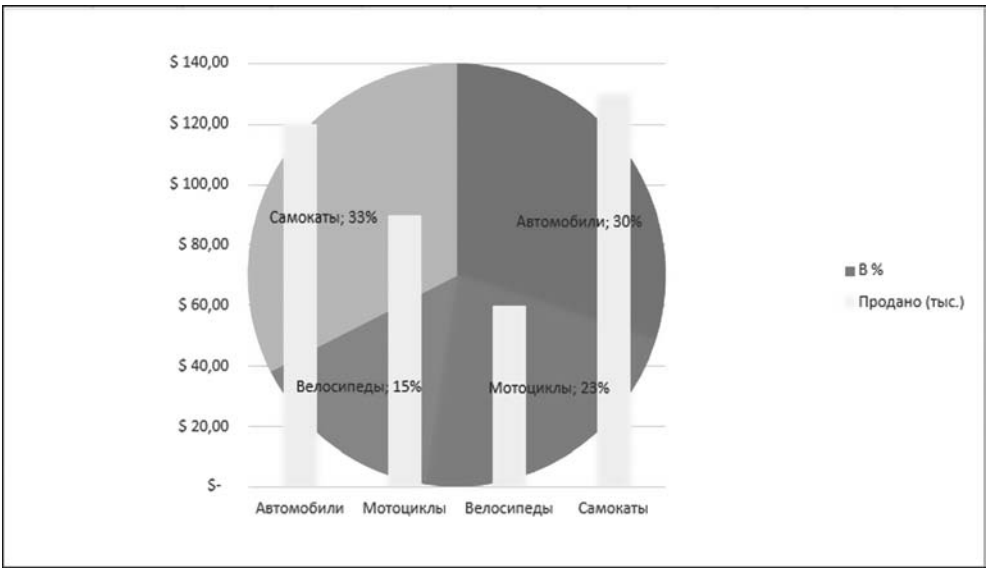


Рис. 8.60
Диаграмма комбинированного типа создана

В окне **Формат подписей данных** устанавливаем флажок опции **имена категорий** в разделе **Параметры подписи**, как это показано на рисунке 8.59.

Разумеется, при желании выполняются и другие настройки. Что касается наших построений, то ожидаемый результат показан на рисунке 8.60.

Таким образом, мы создали диаграмму комбинированного типа — разные ряды данных отображаются по-разному.

На заметку

Далеко не любые типы диаграмм можно «смешать» вместе. Такие типы должны быть совместимы. К сожалению, не всегда можно заранее угадать, какие типы диаграмм совместимы, а какие — нет.

СОЗДАНИЕ ШАБЛОНА ДИАГРАММЫ

— Знаю я вас, панычей!
— А я не паныч — я простой мещанин.
Это только сверху на мне образование!

Из к/ф «За двумя зайцами»

Часто возникает необходимость создавать однотипные (в плане настроек и форматирования) диаграммы, т. е. диаграммы, основанные на одном шаблоне. В этом случае имеет смысл выбрать одну диаграмму как базовую и создать на ее основе шаблон. Затем этот шаблон мы сможем использовать для создания новых диаграмм, благодаря чему отпадет необходимость каждый раз выполнять одни и те же настройки. Как пояснение к сказанному создадим шаблон на основе комбинированной диаграммы, которая рассматривалась в

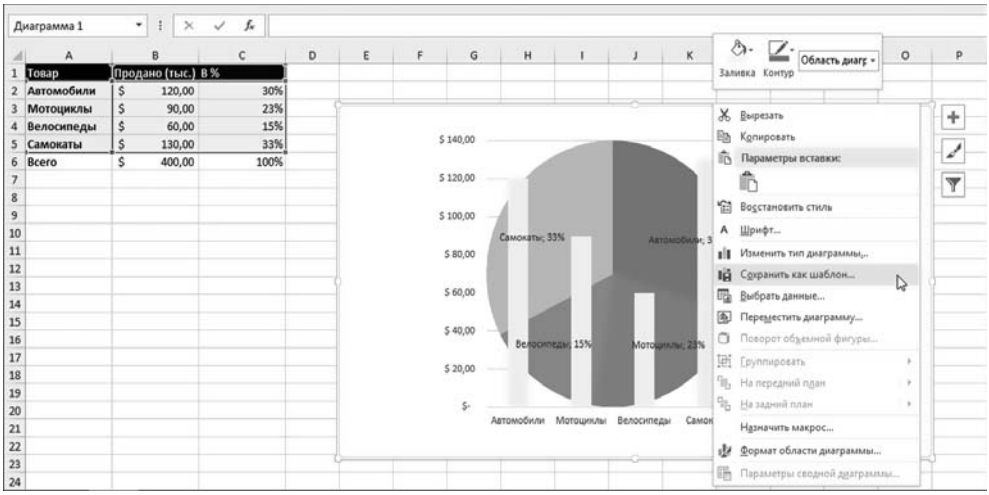


Рис. 8.61
Сохранение шаблона диаграммы

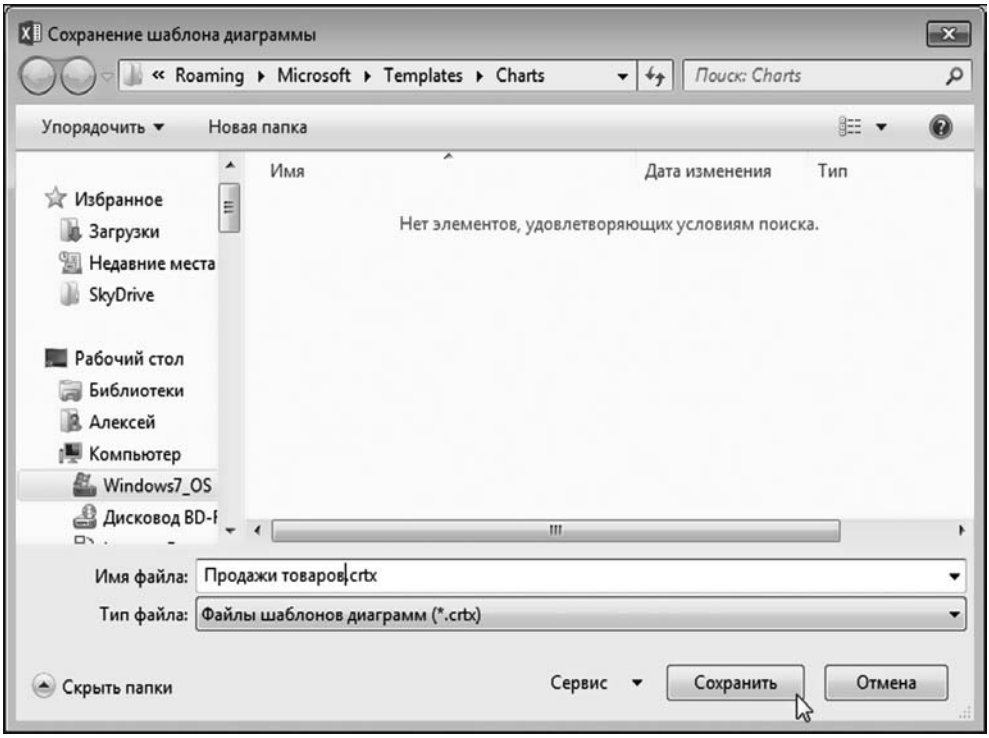


Рис. 8.62
В поле **Имя файла** указываем имя для шаблона диаграммы

предыдущем разделе. Процедура создания шаблона исключительно простая: выделяем диаграмму, щелкаем правой кнопкой мыши и в открывшемся контекстном меню выбираем команду **Сохранить как шаблон** (рис. 8.61).

В открывшемся диалоговом окне **Сохранение шаблона диаграммы** необходимо указать название файла, которое затем будет использоваться в качестве названия соответствующего пользовательского шаблона диаграммы (рис. 8.62).

После того как шаблон диаграммы сохранен, его можно использовать при создании новых диаграмм. Для этого при создании диаграммы в окне выбора типа диаграммы **Вставка диаграммы** необходимо щелкнуть на пиктограмме **Шаблоны** и выбрать нужный шаблон, как, например, показано на рисунке 8.63.

Диаграмма, которая будет создана в результате, получит тот же «типаж», что и диаграмма, использованная для создания шаблона.

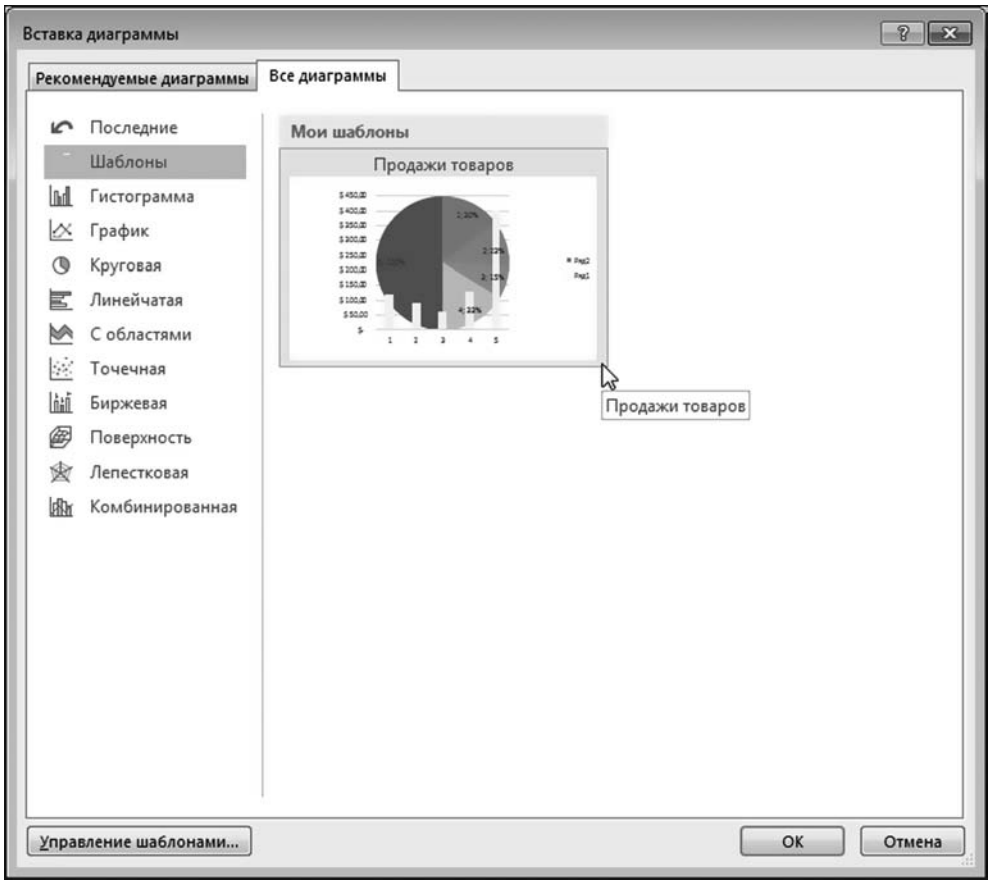


Рис. 8.63
В разделе **Шаблоны** представлен пользовательский шаблон диаграммы, который можно взять за основу при создании новой диаграммы

ГРАФИЧЕСКИЕ ОБЪЕКТЫ В ДИАГРАММАХ

Только для этого предприятия надо ж денёг,
шоб сменить фасон. Новая мода!

Из к/ф «За двумя зайцами»

Диаграммы в Excel красивые сами по себе. Но их можно сделать еще красивее. Для этого, например, используются всевозможные графические утилиты. Рассмотрим небольшой пример. На рисунке 8.64 показан документ с симпатичной, но все же стереотипной диаграммой.

Начнем с того, что добавим в качестве фона области построения диаграммы картинку (из файла). Для этого выделяем область построения диаграммы и двойным щелчком переходим в режим ее редактирования. В диалоговом окне **Формат области построения** в разделе **Заливка** щелкаем кнопку **Файл** для выбора файла, из которого будет вставлено изображение (рис. 8.65).

Откроется диалоговое окно, которое называется **Вставка рисунка** и в котором следует выбрать файл с изображением, как показано на рисунке 8.66.

На рисунке 8.67 показана диаграмма, у которой в качестве фона области построения отображается картинка. Для удобства восприятия мы изменили цвет заливки столбиков ряда данных.

На следующем этапе добавляем в диаграмму текстовую область. Для этого диаграмму выделяем (это важно!) и, затем на вкладке **Вставка** в раскрывающемся списке пиктограммы **Текст** выбираем команду с пиктограммой **Надпись** (рис. 8.68).

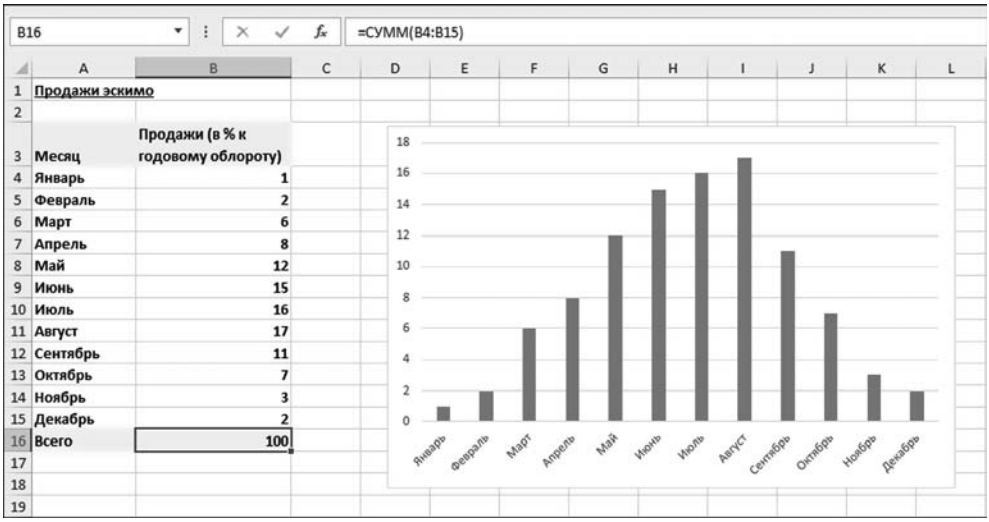


Рис. 8.64
Обычная диаграмма

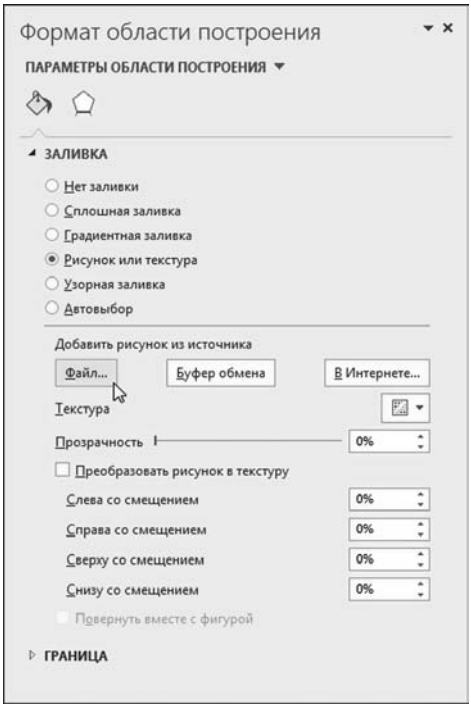


Рис. 8.65
Для фона области построения диаграммы используем картинку из файла

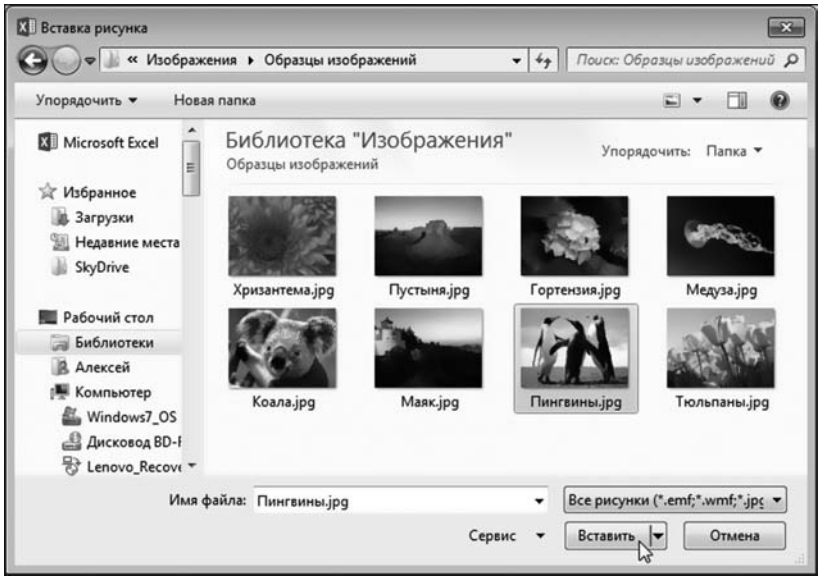


Рис. 8.66
Выбор картинки для отображения в качестве фона области построения диаграммы

Размещаем в области диаграммы текстовую область, заливаем ее цветом и вводим текст (рис. 8.69).

На заметку

Если перед вставкой текстовой области диаграмму не выделить, то текстовая область будет вставлена в документ, а не в диаграмму — при перемещении диаграммы текстовая область вместе с ней перемещаться не будет. Поэтому прежде, чем вставлять текстовую область в диаграмму, важно убедиться, что диаграмма выделена.

Описанная процедура вставки текстовой области в Excel 2010 выглядит несколько иначе. А именно, в этом случае нужно выделить диаграмму и на дополнительной вкладке **Работа с диаграммами** ▷ **Макет** щелкнуть в группе **Вставка** на пиктограмме **Надпись**.

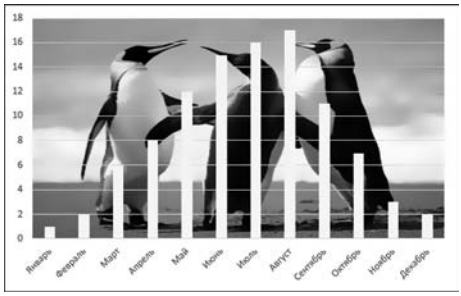


Рис. 8.67
Картинка использована как фон для области построения диаграммы

После ввода текста в текстовую область, применяем к нему жирный шрифт и подправляем размеры текстовой области. Затем при выделенной диаграмме на вкладке **Вставка** в группе **Иллюстрации** раскрываем список пиктограммы **Фигуры** и выбираем там объект с изображением ленты (рис. 8.70).

Размещаем ленточку в области диаграммы и добавляем в ленточку текст. Результат показан на рисунке 8.71.

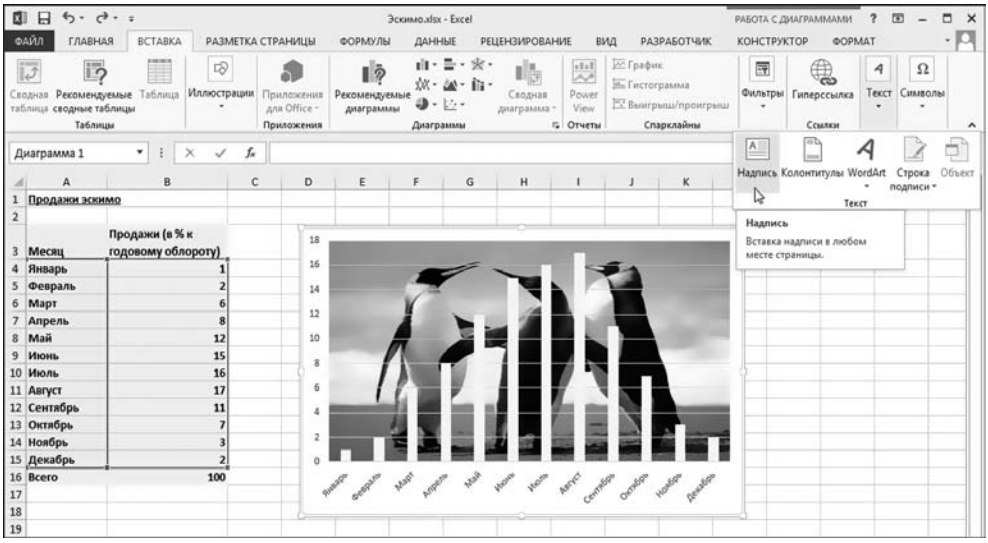


Рис. 8.68
Добавление в область диаграммы надписи

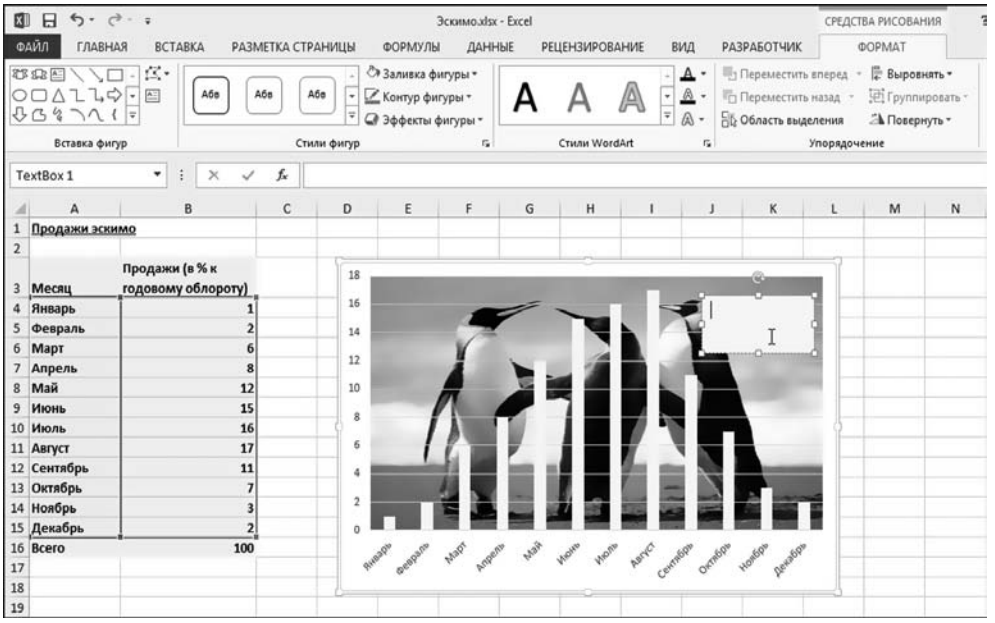


Рис. 8.69
Ввод текста в текстовую область, предварительно залитую желтым цветом

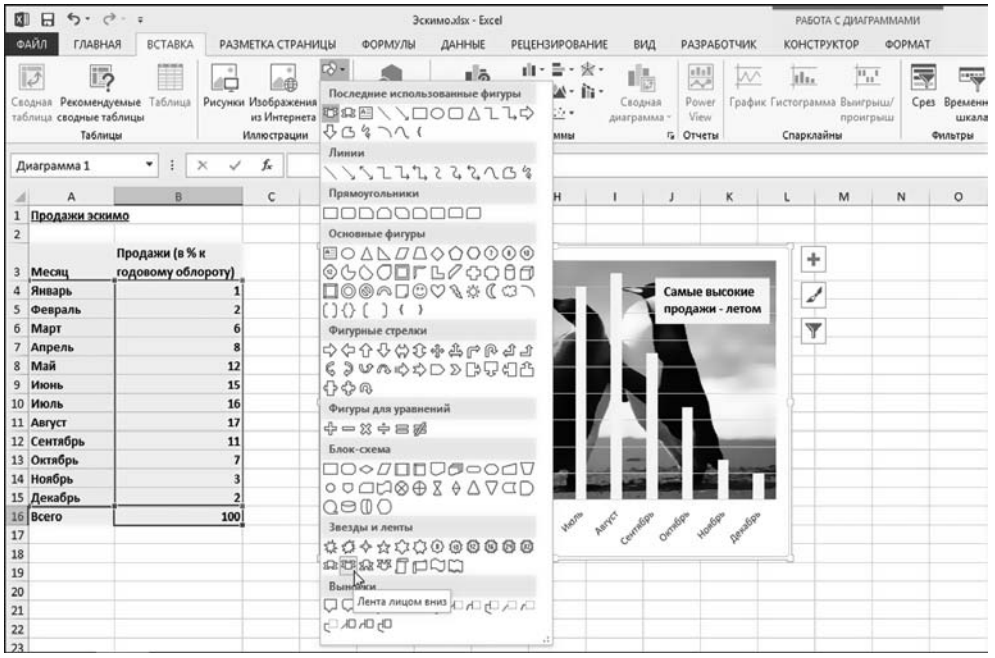


Рис. 8.70
Добавление в область диаграммы графической формы

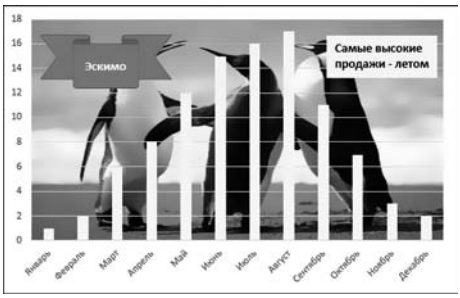


Рис. 8.71
Диаграмма с картинкой в качестве фона, текстовой областью и графической формой

На заметку

В Excel 2010 для вставки в диаграмму ленточки нужно воспользоваться пиктограммой-раскрывающимся списком **Фигуры** в группе **Вставка** дополнительной вкладки **Работа с диаграммами ▷ Макет**.

Понятно, что графических объектов, подходящих для добавления в диаграмму, очень много. Однако принципы и методы работы с ними достаточно универсальны и подобны тем, что описывались выше.

СПАРКЛАЙНЫ

Что мы имеем на данном отрезке времени с нашими показателями? Являются ли данные показатели показательными на данном отрезке времени? Не есть ли эти показатели малопоказательными сами по себе?

Из к/ф «Безумный день инженера Баркасова»

Начиная с версии Excel 2010, у приложения появилась одна интересная утилита, которую к диаграммам, откровенно говоря, отнести довольно сложно. Вместе с тем, если смотреть не на форму, а на идею, то вполне логично обсудить *спарклайны* (а именно о них идет речь) в этом разделе. Если кратко, то спарклайн — это изображение, наподобие диаграммы, но только размером с ячейку. Нечто вроде мини-диаграммы ячеечного масштаба. Штука довольно удобная, хотя немного и необычная.

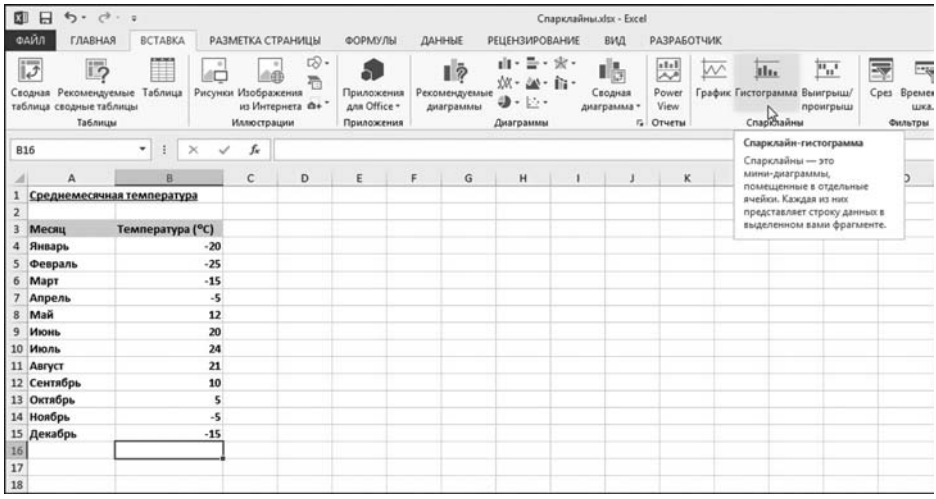


Рис. 8.72
Данные для создания спарклайна

Добавить в рабочий документ спарклайн достаточно просто. Для этого необходим набор данных, на основе которых создается спарклайн. На рисунке 8.72 представлен фрагмент документа с данными, которые затем используем для создания спарклайна.

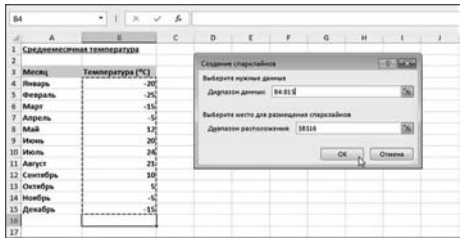
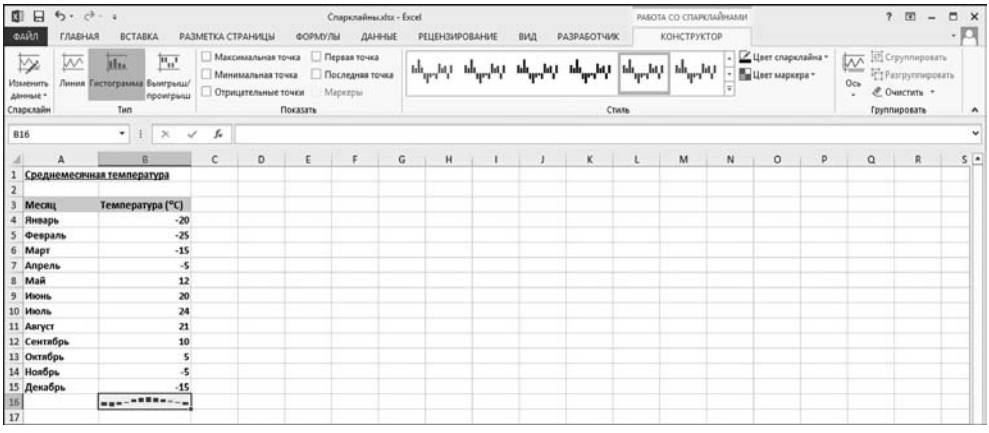


Рис. 8.73
Выбор данных
для создания спарклайна

Рабочий документ содержит импровизированную информацию о среднемесячной температуре по месяцам. Для создания на этой основе спарклайна выполняем следующую последовательность операций. Во-первых, в группе **Спарклайны** на вкладке **Вставка** щелкаем пиктограмму **Гистограмма**, как показано на рисунке 8.72. Пиктограмма используется для создания спарклайна в виде миниатюрной гистограммы (когда данные отображаются столбиками). Но прежде откроется диалоговое окно **Создание спарклайнов**, в котором указывается диапазон вывода и диапазон данных, на основе которых создается спарклайн. Окно **Создание спарклайнов** на фоне рабочего документа представлено на рисунке 8.73.

Результат создания спарклайна «гистограммного» типа представлен на рисунке 8.74.

Стоит обратить внимание, что при выделении ячейки со спарклайном автоматически отображается дополнительная вкладка **Работа со спарклайнами ▷ Конструктор**, содержащая внушительное количество утилит для работы и редактирования спарклайнов. Например, в группе пиктограмм **Тип** представлены пиктограммы для выбора типа спарклайна. Опции в группе **Показать** позволяют определить, какие дополнительные параметры следует отображать в спарклайне. Полезные инструкции (команды) содержит и пиктограмма-раскрывающийся список **Ось**. На рисунке 8.75 показано, как



	A	B
1	Среднемесячная температура	
2		
3	Месяц	Температура (°C)
4	Январь	-20
5	Февраль	-25
6	Март	-15
7	Апрель	-5
8	Май	12
9	Июнь	20
10	Июль	24
11	Август	21
12	Сентябрь	10
13	Октябрь	5
14	Ноябрь	-5
15	Декабрь	-15
16		
17		

Рис. 8.74
Спарклайн отображен столбиками — как гистограмма

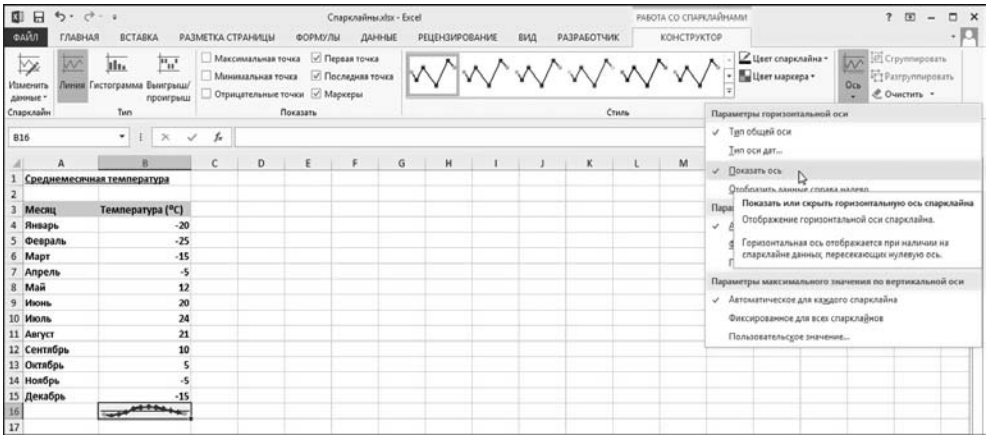


Рис. 8.75
Спарклайн отображен в виде линии

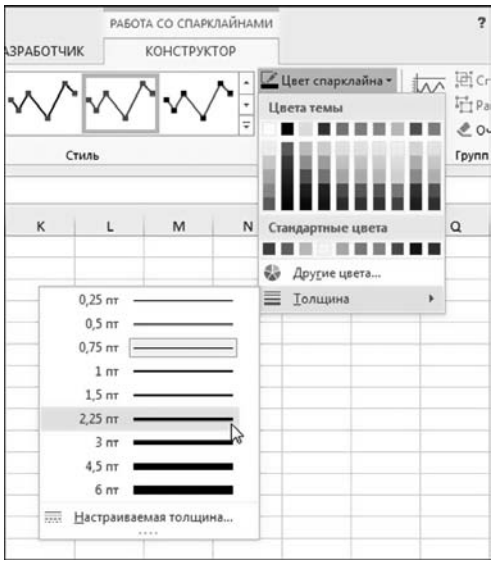


Рис. 8.76
Выбор типа линии для спарклайна

будет выглядеть спарклайн после применения некоторых настроек: спарклайн отображается линией, на нем показаны маркеры, первая и последняя точка, выделена ось и увеличена толщина линии спарклайна.

На заметку

Для изменения толщины линии спарклайна используем команду-список **Толщина** в раскрывающемся списке **Цвет спарклайна** (рис. 8.76).
Здесь же можно, например, изменить и цвет линии спарклайна.

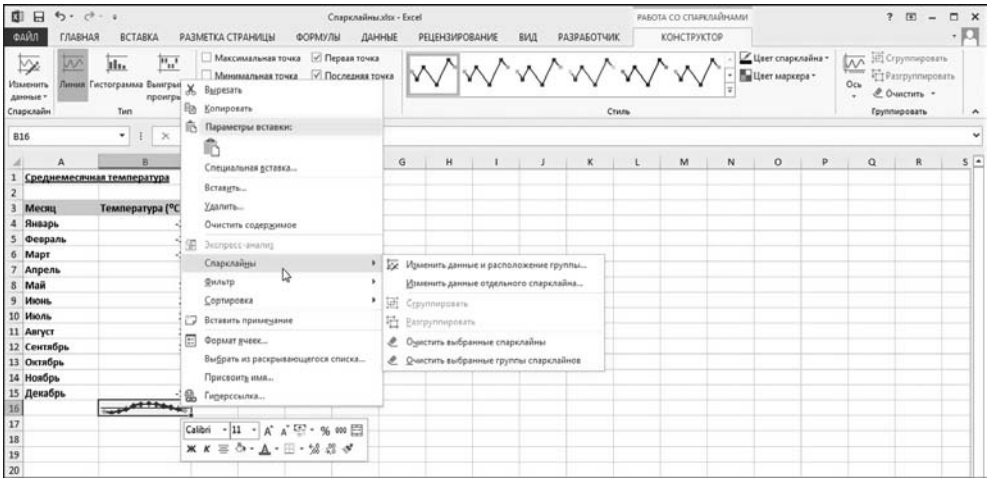


Рис. 8.77
Контекстное меню спарклайна

Удалить спарклайн из рабочего документа просто так не получится. Для удаления спарклайна можно воспользоваться, например, командой контекстного меню **Очистить содержимое** (рис. 8.77).

Стоит также обратить внимание на команды подменю **Спарклайны** контекстного меню. В работе они могут оказаться полезными.

ЧАСТЬ ТРЕТЬЯ

ПРОГРАММИРОВАНИЕ В EXCEL

ГЛАВА 9 ЗАПИСЬ МАКРОСОВ

— Я могу писать только ночью.
— Товарищ Пушкин, между прочим,
и другие ответственные поэты,
писали при свечах — и ничего, получалось!
— А я могу писать только ночью!

Из к/ф «Безумный день инженера Баркасова»

Макросы — очень полезная штука. Если бы не было макросов, то не исключено, что популярность приложения Excel была бы на несколько порядков ниже. С формальной точки зрения макрос — это процедура, которая выполняется под управлением приложения Excel. В этой главе мы узнаем, как макросы создаются, как они запускаются на выполнение, а также освоим азы анализа программных кодов макросов.

На заметку

Вообще с макросами и прочими программными кодами ситуация следующая. Есть язык программирования, который называется Visual Basic for Applications — в сокращенном варианте VBA. Язык предназначен для «обслуживания» приложений пакета Microsoft Office и, среди прочего, приложения Excel. Именно применение VBA с Excel нас и будет интересовать в дальнейшем. С помощью VBA можно создавать программные коды, которые, так или иначе, выполняются средствами приложения Excel. Программные коды обычно создаются с помощью специального встроенного редактора кодов VBA, который входит в комплект поставки Excel и запускается непосредственно из приложения. В первую очередь речь идет о создании функций пользователя и макросов. Макрос — это набор инструкций. Создать макрос можно или непосредственно в редакторе кодов, или воспользоваться специальной процедурой записи макросов. Сначала мы познакомимся с тем, как макросы записываются в автоматическом режиме, а затем научимся писать программные коды самостоятельно.

ЗНАКОМСТВО С МАКРОСАМИ

Послушай! Я тебя полюбил — я тебя научу.

Из к/ф «Кин-дза-дза»

Макрос — это набор программных инструкций, или команд, которые можно выполнять многократно. Другими словами, если нам при работе с приложением Excel часто приходится выполнять одни и те же действия или операции, то, скорее всего, хорошо было бы реализовать такую последователь-

ность действий в виде макроса и затем запускать этот макрос в случае необходимости. Есть два базовых подхода к созданию макроса. Первый и наиболее профессиональный состоит в том, чтобы создать программный код макроса, что называется, своими руками — обычно с помощью редактора кодов. Но для реализации такого подхода необходимо иметь хотя бы минимальные навыки работы с VBA. Другой способ подразумевает запись макроса. Происходит это в общих чертах так:

- переходим в специальный режим, который называется режимом записи макроса;
- выполняем действия, которые должны будут выполняться в результате работы макроса;
- выходим из режима записи макроса (с одновременным сохранением макроса).

В принципе это все. Главное преимущество такого подхода связано с тем, что, как правило, с программным кодом вообще не придется иметь дела. Дальше рассмотрим пример.

На заметку

Может показаться, что запись макроса — панацея от всех проблем. Конечно, все не так просто. Есть две достаточно серьезные проблемы, которые не позволяют при работе с макросами полагаться исключительно на процедуру их записи. Первая проблема связана с тем, что далеко не каждое действие можно записать. Другими словами, есть такие команды, которые намного легче реализовать самому через инструкции VBA, чем выполнять соответствующие действия в окне приложения Excel. Вторая проблема менее значительна, но, тем не менее, актуальна. Дело в том, что при записи макроса автоматически создается программный код для этого макроса. При повторном вызове макроса выполняется именно этот код. К сожалению, созданный автоматически код обычно является неоптимальным и содержит много лишних инструкций.

На вкладке **Разработчик** в группе **Код** есть специальная пиктограмма-переключатель (т. е. может находиться в нажатом или не нажатом состоянии) **Запись макроса** для перехода в режим записи макроса (рис. 9.1).

Выделяем ячейку A1 и щелкаем пиктограмму **Запись макроса**. Появляется диалоговое окно **Запись макроса**, представленное на рисунке 9.2.

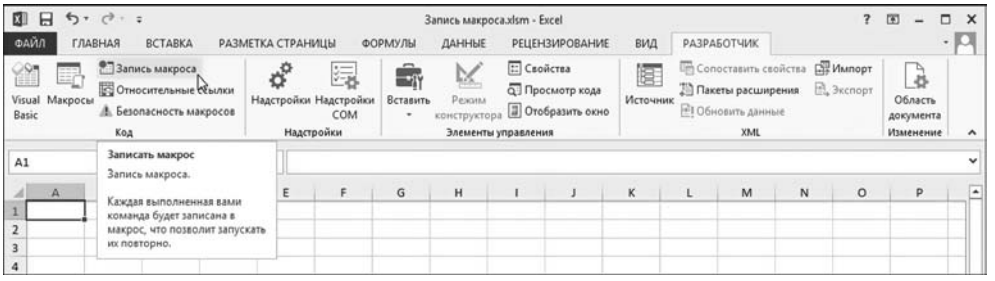


Рис. 9.1
Для перехода в режим записи макроса необходимо щелкнуть на пиктограмме-переключателе **Запись макроса**

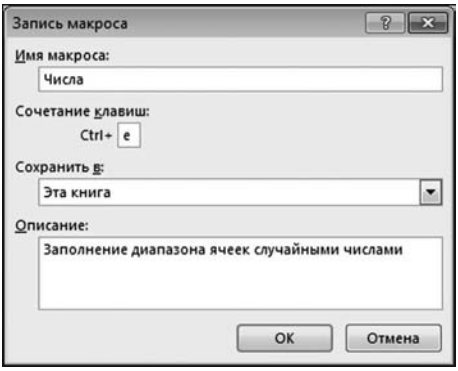


Рис. 9.2
Диалоговое окно **Запись макроса** с выполненными настройками

В поле **Имя макроса** указывается имя создаваемого макроса (мы называем макрос Числа). В области **Описание** вводится текст, который затем будет использоваться в качестве подсказки по макросу. Мы вводим текст Заполнение диапазона ячеек случайными числами. В раскрывающемся списке **Сохранить в** выбираем (точнее, оставляем выбор по умолчанию) **Эта книга**.

На заметку

Сохранить макрос можно в текущей рабочей книге (позиция **Эта книга**), в новой книге (позиция **Новая книга**) или в личной книге макросов (позиция **Личная книга макросов**). В личной книге макросов сохраняются макросы, область доступности которых выходит за пределы рабочей книги (текущей или новой). Такие макросы доступны всегда, в любом документе. Кроме того, следует иметь в виду, что если книга содержит макросы, то сохранять ее следует как книгу с макросами (расширение xlsm).

Для макроса назначаем сочетание клавиш <Ctrl>+<E>. Эта комбинация определяется в поле **Сочетание клавиш**.

На заметку

Сочетание или комбинация клавиш для макроса — это та комбинация, нажатие которой приводит к запуску макроса на выполнение. В эту комбинацию входит клавиша <Ctrl>, о чем свидетельствует соответствующая надпись в окне **Запись макроса**. Клавиша, которая нажимается в комбинации с клавишей <Ctrl>, указывается в поле **Сочетание клавиш**. При этом следует помнить, что вводимое в поле значение (клавиша) чувствительно к регистру. Например, если ввести литеру **e** (строчная), это будет означать, что макрос запускается комбинацией клавиш <Ctrl>+<E>. Если ввести в поле **Сочетание клавиш** литеру **E** (прописная), то макрос будет запускаться комбинацией клавиш <Ctrl>+<Shift>+<E>. В Excel различные комбинации клавиш используются для выполнения тех или иных действий. Если макросу назначить комбинацию клавиш, которая уже зарезервирована для команды Excel, то приоритет остается за макросом. То есть если мы определили комбинацию для макроса, несмотря ни на что, за макросом эта комбинация и останется закрепленной. Но все же лучше использовать свободные (незарезервированные) комбинации. С клавишей <Ctrl> **можно** использовать буквы E, J, M и Q. В комбинации с клавишами <Ctrl>+<Shift> **не рекомендуется** использовать клавиши <F>, <L>, <N>, <O>, <P> и <W> (они уже зарезервированы).

После выполнения всех настроек переходим непосредственно к записи макроса. Выполняем такие действия:

- выделяем диапазон ячеек A1:C5;
- в строке формул вводим выражение =СЛУЧМЕЖДУ(0;10);
- формулу вводим комбинацией клавиш <Ctrl>+<Enter> — для того, чтобы формула скопировалась во все ячейки выделенного диапазона.

На заметку

Формулой =СЛУЧМЕЖДУ(0;10) вычисляется случайное число в диапазоне значений от 0 до 10 включительно.

На рисунке 9.3 проиллюстрирован процесс заполнения диапазона ячеек A1:C5 случайными числами в режиме записи макроса.

После нажатия комбинации клавиш <Ctrl>+<Enter> диапазон ячеек A1:C5 заполняется случайными числами, как это показано на рисунке 9.4.

При этом диапазон ячеек A1:C5 выделен. Мы вместо этого выделяем ячейку C5 и останавливаем запись макроса с помощью пиктограммы **Остановить запись** (рис. 9.5).

На заметку

После нажатия пиктограммы **Запись макроса** она становится пиктограммой **Остановить запись**. И наоборот.

Запись макроса закончена, и макрос готов к использованию. Запускать макрос можно по-разному. Один способ мы уже знаем — с помощью комби-

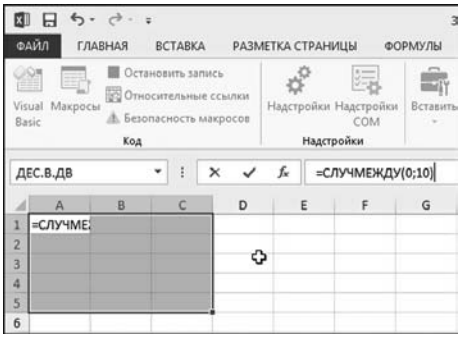


Рис. 9.3
В выделенный диапазон ячеек A1:C5 вводится формула для вычисления случайного числа

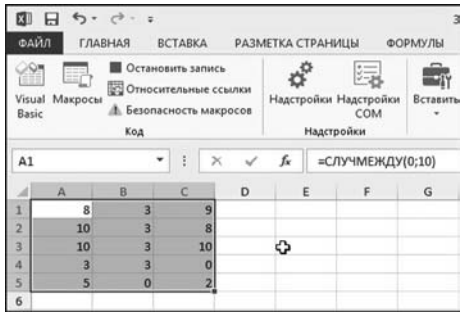


Рис. 9.4
Диапазон ячеек заполнен случайными числами

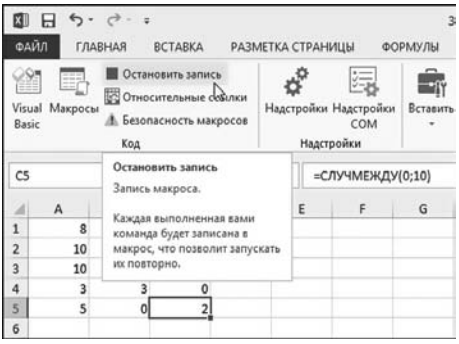


Рис. 9.5
Остановка записи макроса

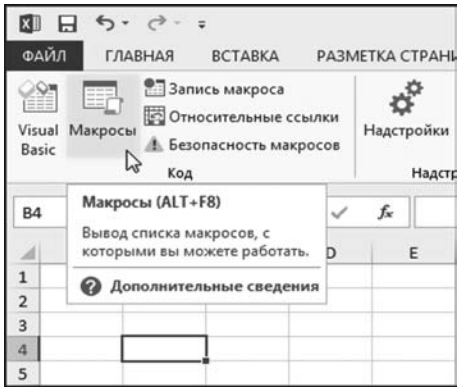


Рис. 9.6
Документ перед выполнением макроса

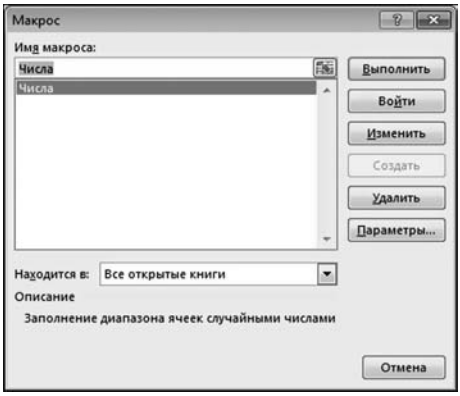


Рис. 9.7
Диалоговое окно запуска макроса на выполнение

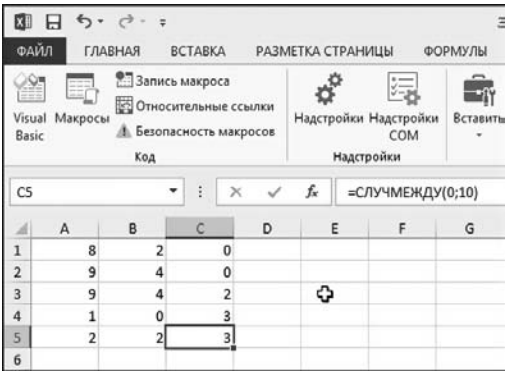


Рис. 9.8
Результат выполнения макроса

нации клавиш <Ctrl>+<E>. Мы, как всегда, пойдем другим путем. Запускать макрос будем с помощью пиктограммы **Макросы** в группе **Код** на вкладке **Разработчик**. Для запуска макроса мы используем новый чистый рабочий лист. Перед запуском макроса выделяем ячейку B4 (можно и другую — главное, чтобы не A1 и не C5). Запускаем макрос (см. рис. 9.6).

Эффект наступит не сразу (в отличие от случая, когда для запуска макроса используется комбинация клавиш). Сначала откроется окно **Макрос**, представленное на рисунке 9.7.

В этом диалоговом окне необходимо выбрать макрос для запуска и щелкнуть на кнопке **Выполнить**. Наша задача простая — макрос в списке всего один.

На заметку

В разделе **Описание** отображается подсказка по макросу. Это именно та подсказка, которую мы ввели при создании макроса. Для редактирования программного кода макроса щелкаем кнопку **Изменить**.

Результат выполнения макроса **Числа** представлен на рисунке 9.8.

Здесь стоит обратить внимание на два обстоятельства. Во-первых, несмотря на то, что на момент запуска макроса активной была ячейка B4, заполняется диапазон ячеек A1:C5.

На заметку

Иному читателю такое «поведение» макроса может показаться вполне ожидаемым и единственно возможным. Ведь при записи макроса заполнялся именно диапазон ячеек A1:C5, а не какой-то другой. Контраргумент состоит в том, что при записи макроса была выделена ячейка A1, а уже после этого выделен соответствующий диапазон и заполнен случайными числами. Поэтому не менее логичной была бы ситуация, когда заполняемый диапазон определялся бы активной на момент запуска макроса ячейкой. Но в данном конкретном случае победила, как видим, первая точка зрения. Причина в том, что при записи макросов все ссылки на ячейки полагаются абсолютными.

Во-вторых, после выполнения макроса активна ячейка C5, т. е. все, как и при записи макроса, вплоть до выделения ячеек.

На заметку

Хочется обратить внимание на одну особенность создаваемых нами документов со случайными числами. Как уже отмечалось, случайные числа генерируются с помощью формул =СЛУЧМЕЖДУ(0;10). Каждый раз, когда вычисляется эта формула, генерируется новое случайное число. В Excel (в используемом по умолчанию режиме) формулы пересчитываются при внесении изменений в документ. Как следствие, время от времени содержимое ячеек со случайными числами будет меняться случайным образом.

Макрос, который мы создали, оперирует абсолютными ссылками. Мы несколько видоизменим макрос (точнее, запишем новый) так, чтобы случайными числами заполнялся диапазон ячеек, начиная с активной ячейки на момент запуска макроса.

Как и в предыдущем случае, в чистом листе рабочего документа выделяем ячейку A1 и начинаем запись макроса. На рисунке 9.9 показано окно **Запись макроса** с выполненными настройками на фоне рабочего документа.

Новый макрос называется Числа_2 и запускается комбинацией клавиш <Ctrl>+<Shift>+<E>. Сразу после начала записи макроса переходим в режим использования относительных ссылок, для чего щелкаем пиктограмму-переключатель **Относительные ссылки** в группе **Код** вкладки **Разработчик** (см. рис. 9.10).

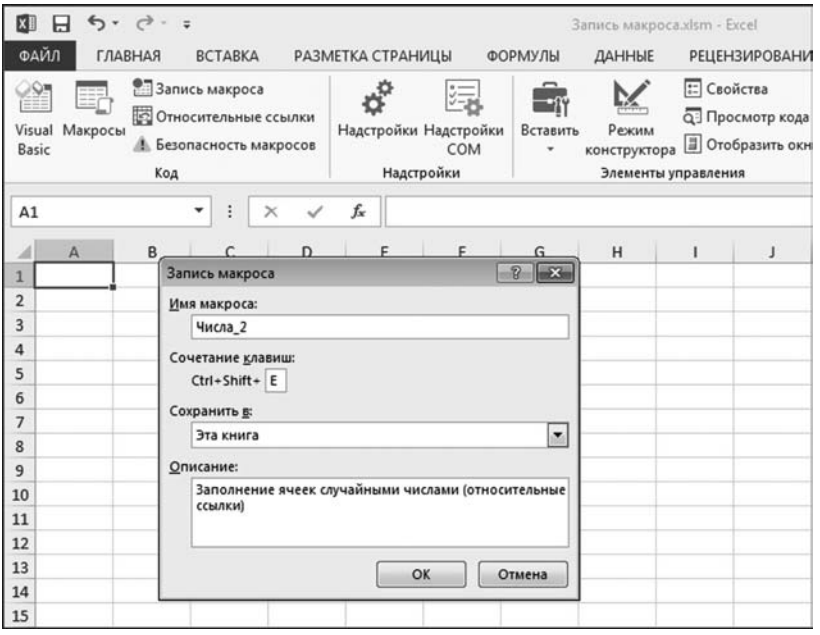


Рис. 9.9
Документ перед началом записи макроса с относительными ссылками

Затем выделяем диапазон ячеек A1:C5 и заполняем его случайными числами (формула =СЛУЧМЕЖДУ(0;10) вводится нажатием комбинации клавиш <Ctrl>+<Enter>), как показано на рисунке 9.11.

После того как диапазон ячеек A1:C5 заполнен случайными числами, выходим из режима использования относительных ссылок (щелкаем пиктограмму-переключатель **Относительные ссылки**, переводя пиктограмму в отжатое состояние) — этот этап записи макроса проиллюстрирован на рисунке 9.12.

Наконец, выйдя из режима относительных ссылок, выделяем ячейку A1 и останавливаем запись макроса щелчком на пиктограмме **Остановить запись** (рис. 9.13).

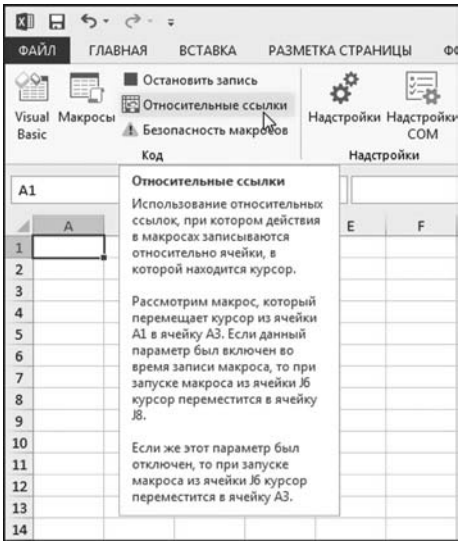


Рис. 9.10
Переход в режим относительных ссылок

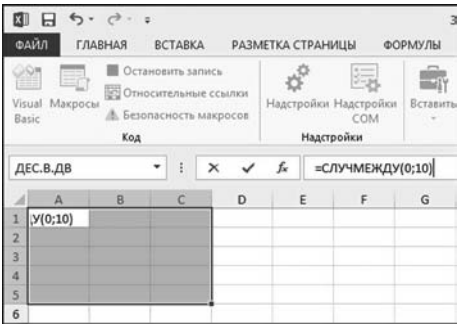


Рис. 9.11
Выделение диапазона и заполнение его случайными числами

Макрос записан. Теперь мы проверим, как он справляется с возложенной на него задачей. Для этого в чистом рабочем листе выделяем ячейку B4, открываем щелчком на пиктограмме **Макросы** диалоговое окно **Макрос** и запускаем из него макрос Числа_2 (рис. 9.14).

В результате выполнения макроса случайными числами заполняется диапазон ячеек B4:D8, после чего выделяется ячейка A1 (рис. 9.15).

В принципе, все ожидаемо и корректно. Заполняется случайными числами диапазон ячеек в 3 столбца и 5 строк с левой верхней ячейкой, которая была активна на момент запуска

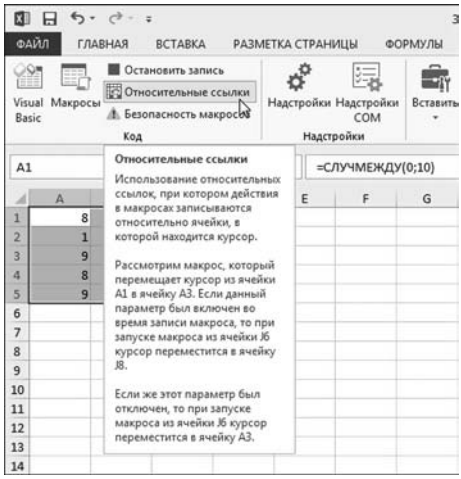


Рис. 9.12
Выход из режима относительных ссылок

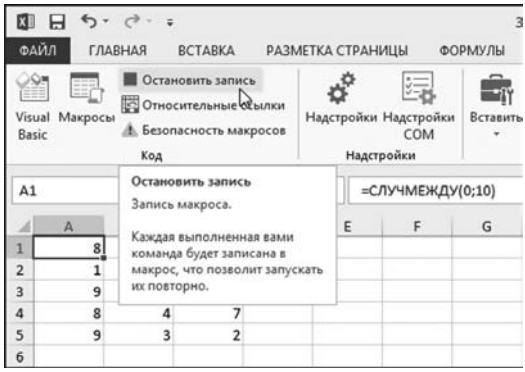


Рис. 9.13
Остановка записи макроса

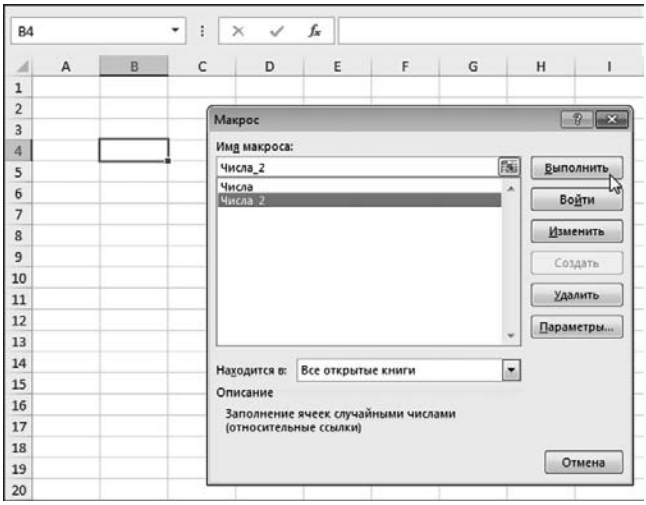


Рис. 9.14
Запуск макроса Числа_2

макроса на выполнение. Но не зависимо от того, какая ячейка была активна, после завершения макроса активной становится ячейка A1. Объяснение достаточно простое. Связано оно с тем, что при записи макроса, когда мы выделяли диапазон для заполнения случайными числами, использовался режим относительных ссылок. При выделении на финальном этапе ячейки A1 использовался режим абсолютных ссылок.

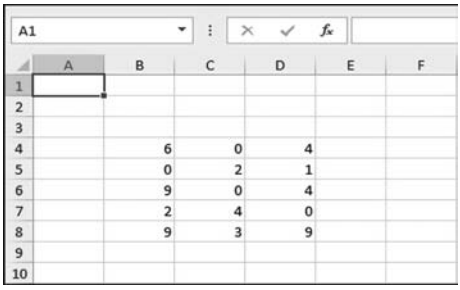


Рис. 9.15
Результат выполнения макроса с относительными ссылками

КНОПКИ ЗАПУСКА МАКРОСОВ

Мадам говорит, что хорошо помогает
толченая кора мангового дерева.

Из к/ф «Тридцать три»

Мы уже знаем, что макросы можно запускать через окно **Макрос**. Также мы знаем, что для макросов можно задавать «горячие» комбинации клавиш. Например, два созданных выше макроса запускаются соответственно с помощью комбинаций клавиш <Ctrl>+<E> и <Ctrl>+<Shift>+<E>. Но это еще не все. Для запуска макроса можно создать специальную пиктограмму или кнопку. Этим и займемся на следующих страницах. Начнем с задачи по добавлению пиктограммы запуска макроса Числа на панель быстрого доступа. Программа-минимум состоит в том, чтобы открыть окно настройки **Параметры Excel**. Проявим некоторую оригинальность и откроем это окно посредством команды раскрывающегося списка панели быстрого доступа **Другие команды** (рис. 9.16).

В результате открывается диалоговое окно **Параметры Excel**, в котором нас интересует раздел **Панель быстрого доступа** (рис. 9.17).

В этом разделе в раскрывающемся списке **Выбрать команды из** выбираем позицию **Макросы**. В расположенном снизу внушительном списке макросов выбираем знакомое название **Числа**. В раскрывающемся списке **Настройка панели быстрого доступа** выбираем рабочую книгу, в которой сохранен

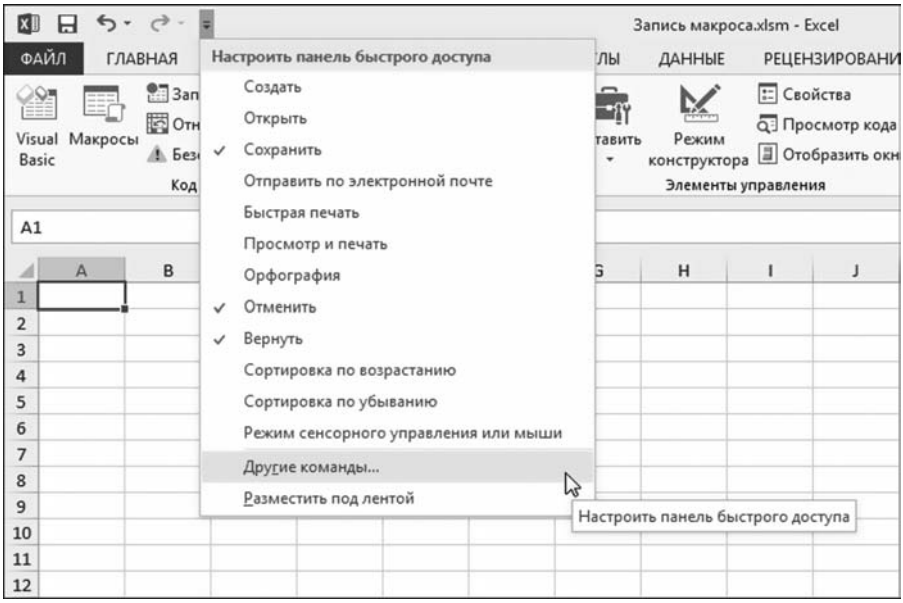


Рис. 9.16
В раскрывающемся списке команд настройки панели быстрого доступа выбираем команду **Другие команды**

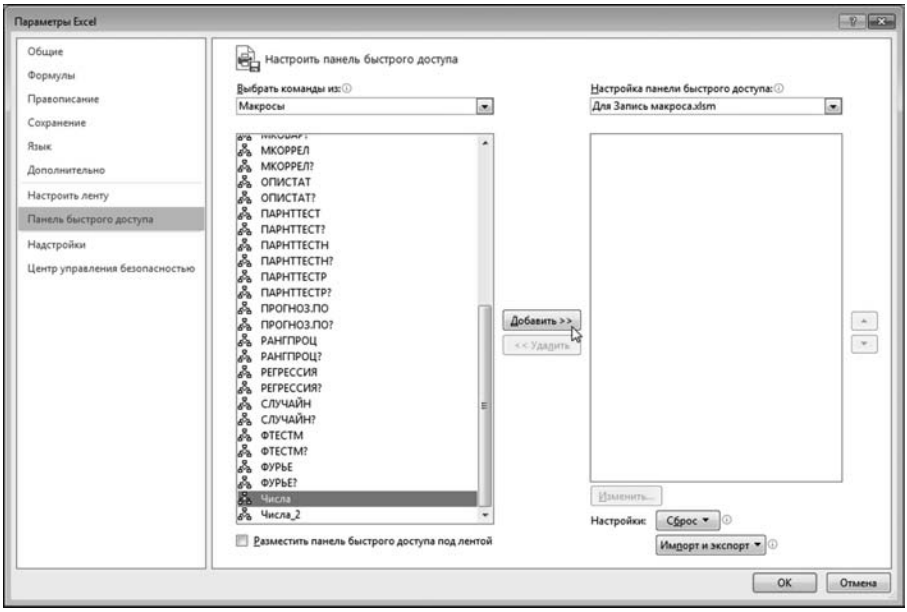


Рис. 9.17
В окне **Параметры Excel** добавляем на панель быстрого доступа пиктограмму макроса **Числа**

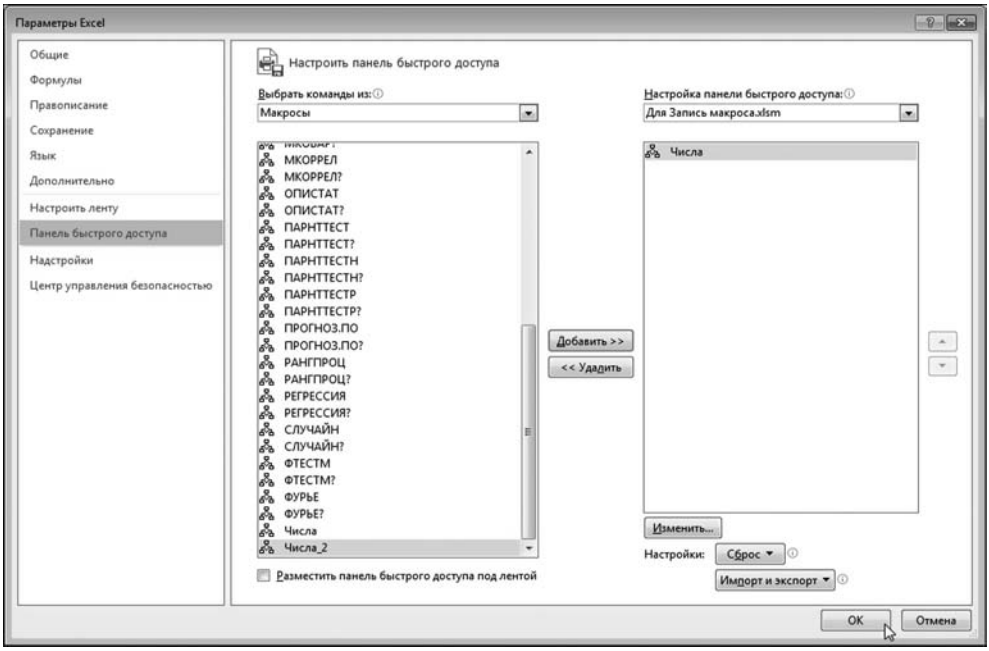


Рис. 9.18
Пиктограмма макроса **Числа** добавлена на панель быстрого доступа

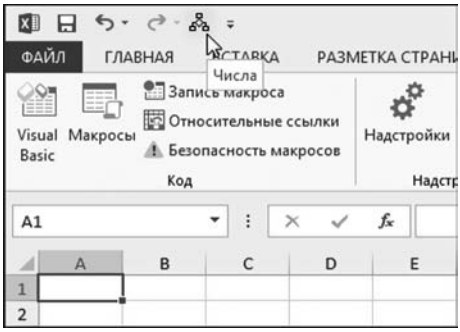


Рис. 9.19
Пиктограмма макроса **Числа** на панели быстрого доступа

макрос. В этом случае пиктограмма запуска макроса на панели быстрого доступа будет отображаться только для того документа, в котором сохранен макрос. Для добавления пиктограммы на панель необходимо щелкнуть на кнопке **Добавить**. В результате пиктограмма с названием макроса **Числа** появится и в правом списке. На рисунке 9.18 показана именно такая ситуация.

После щелчка на кнопке **OK** на панели быстрого доступа появляется пиктограмма запуска макроса **Числа**, как это показано на рисунке 9.19.

Собственно, на этом можно ставить жирную точку. Пиктограмма на панели быстрого доступа есть, и щелчок на ней приводит к выполнению нужного нам макроса. Но еще мы изменим пиктограмму запуска макроса, которая по умолчанию имеет довольно серый и неприглядный вид. Для этого снова открываем окно настройки **Параметры Excel** в том же самом разделе. В списке пиктограмм выделяем пункт **Числа** и щелкаем кнопку **Изменить**, как это показано на рисунке 9.20.

В результате открывается окно настройки изображения пиктограммы **Изменение кнопки**, показанное на рисунке 9.21.

В поле **Отображаемое имя** вводим текст подсказки, которая появляется в контекстной справке при наведении курсора на пиктограмму. Область **Символ**

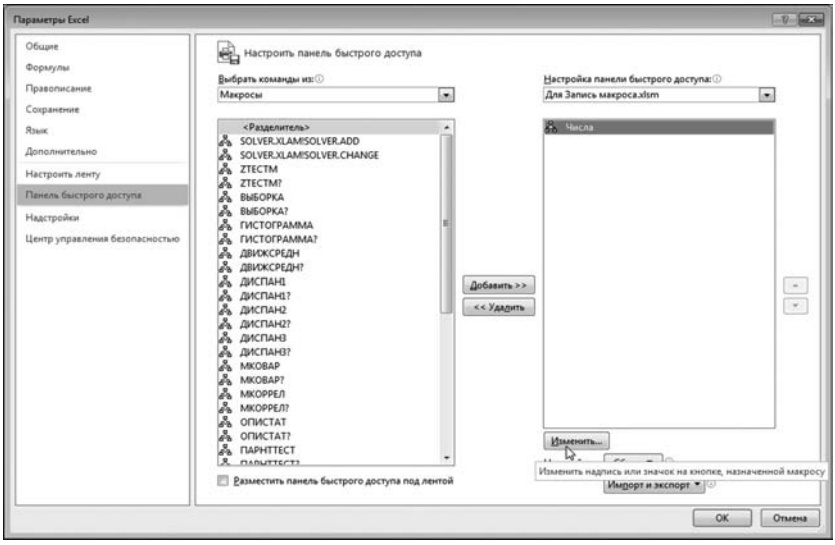


Рис. 9.20
Для редактирования свойств пиктограммы макроса в окне **Параметры Excel** щелкаем кнопку **Изменить**

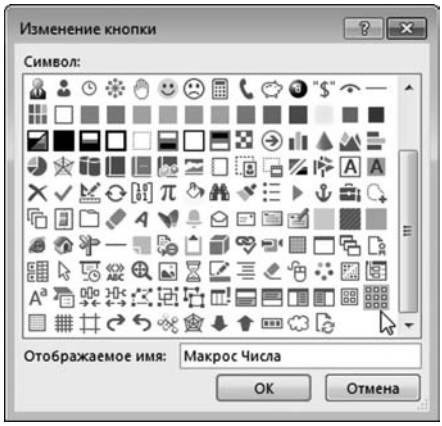


Рис. 9.21
Окно редактирования свойств пиктограммы

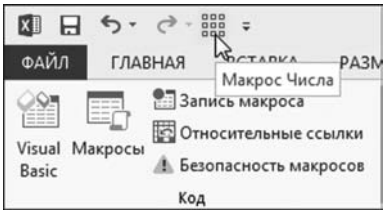


Рис. 9.22
Пиктограмма запуска макроса после внесения изменений

Рис. 9.23
Окно **Параметры Excel** открыто в разделе **Настройка ленты** для создания новой вкладки

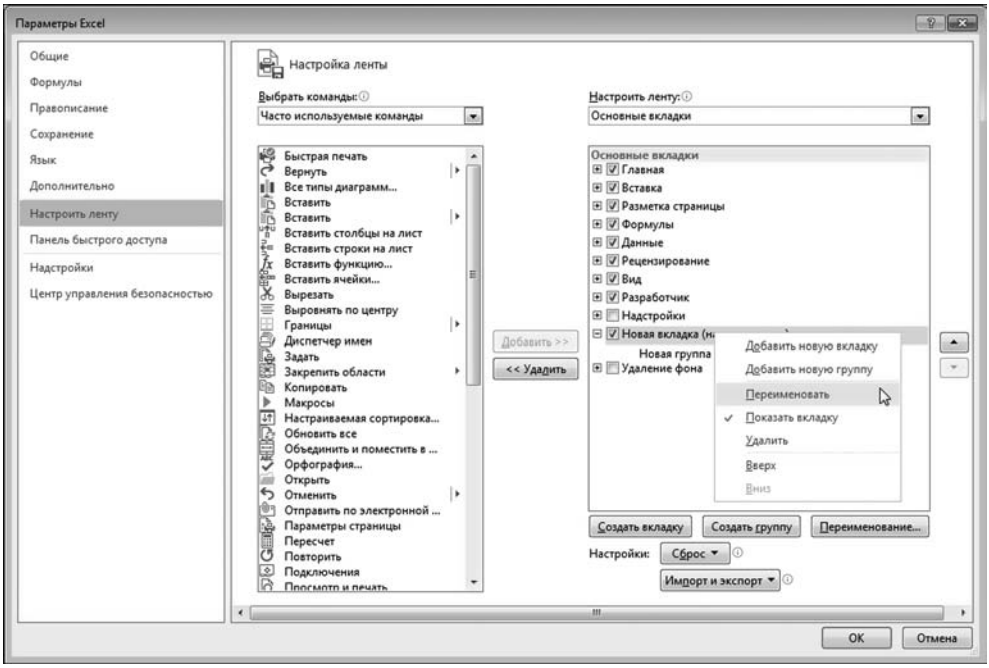


Рис. 9.24
Изменение имени вновь созданной вкладки

содержит список изображений, которые можно использовать для пиктограммы. После сделанного выбора пиктограмма изменяет свой вид и может выглядеть примерно так, как на рисунке 9.22.

Понятно, что точно таким же образом на панель быстрого доступа можно добавить и пиктограмму запуска второго макроса Числа_2. Данный подход (с размещением пиктограммы на панели быстрого доступа) достаточно удобный, однако не единственный. Есть и другие варианты. Один из этих вариантов — создать вкладку пользователя с пиктограммами запуска макросов. Так и поступим.

Открываем окно **Параметры Excel** в разделе **Настройка ленты** и с помощью **Создать вкладку** создаем новую вкладку, как это показано на рисунке 9.23.

После того как вкладка создана, нам предстоит изменить ее имя: текущее имя вкладки необходимо выделить и в списке команд контекстного меню выбрать команду **Переименовать** (рис. 9.24).

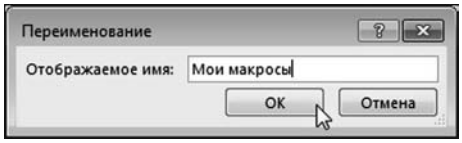


Рис. 9.25
В окне **Переименование** указывается новое имя созданной вкладки

Также можно воспользоваться кнопкой **Переименование**. В результате открывается диалоговое окно **Переименование**, в котором указывается имя вкладки (рис. 9.25).

Мы используем в качестве имени вкладки фразу Мои макросы. Кроме этого, изменяем имя группы, которая

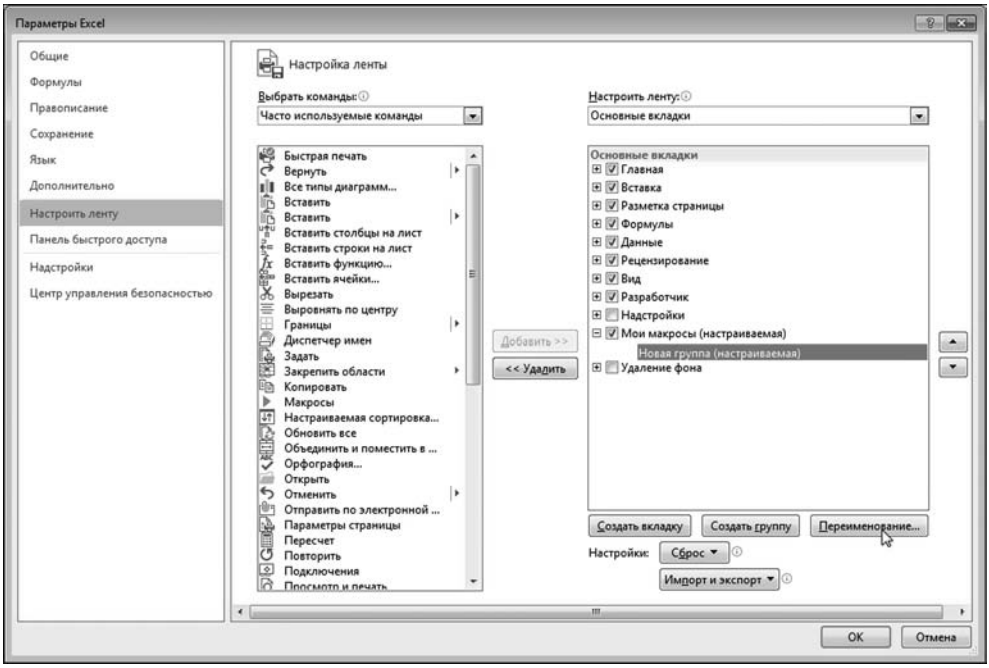


Рис. 9.26
Переименование имени группы на вкладке пользователя

автоматически добавляется во вкладку при ее создании. Выделяем текущее имя группы и щелкаем кнопку **Переименование** (рис. 9.26).

На заметку

Переименование здесь и далее можно выполнять как с помощью кнопки **Переименование**, так и с помощью команды **Переименовать** контекстного меню.

На рисунке 9.27 показано окно **Переименование**, в котором мы вводим имя для группы пиктограмм (имя Заполнение ячеек).

Теперь почти финальный аккорд — добавление в группу вкладки пиктограмм для запуска макросов. В раскрывающемся списке **Выбрать команды** выбираем позицию **Макросы**. В списке находим имя макроса **Числа**. В правом списке выбираем группу во вновь созданной вкладке. Щелкаем кнопку **Добавить** (см. рис. 9.28).

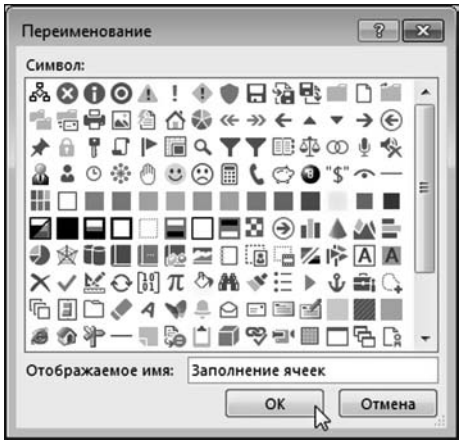


Рис. 9.27
В окне **Переименование** указывается имя для группы во вкладке пользователя

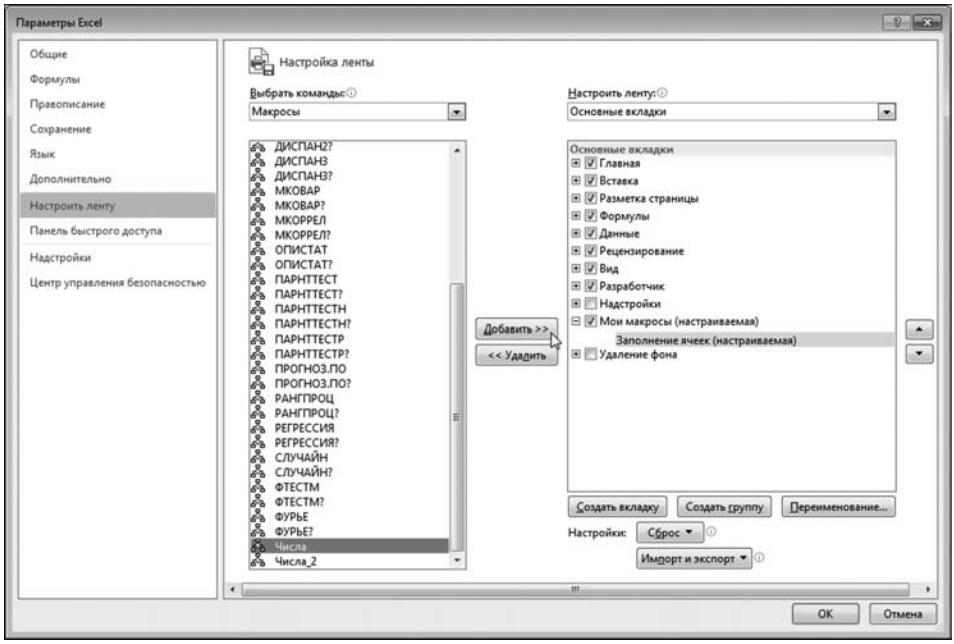


Рис. 9.28
Добавление в группу вкладки пользователя пиктограммы макроса **Числа**

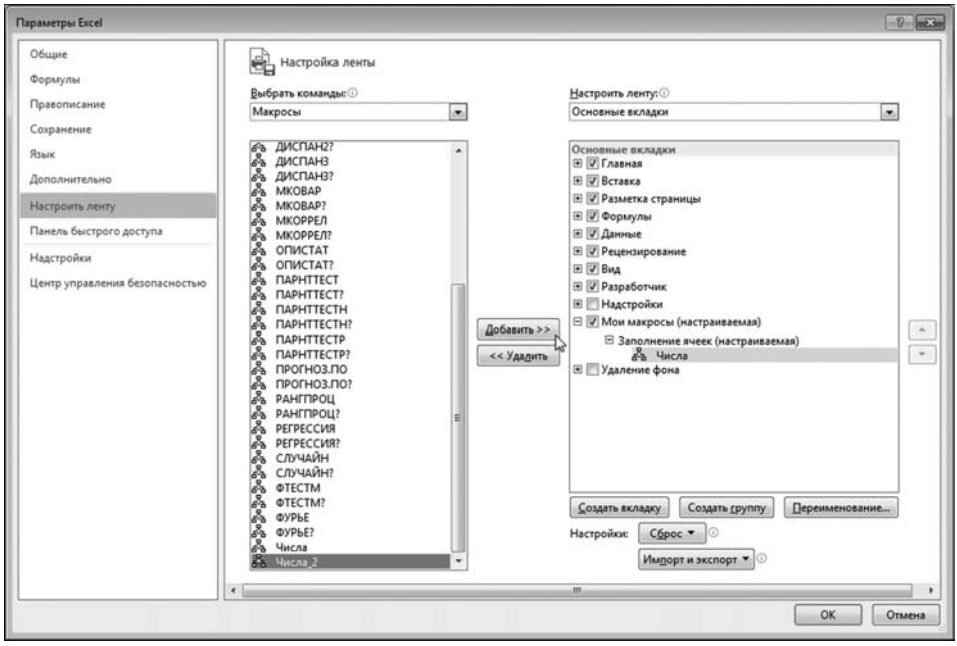


Рис. 9.29
Добавление в группу вкладки пользователя пиктограммы макроса **Числа_2**

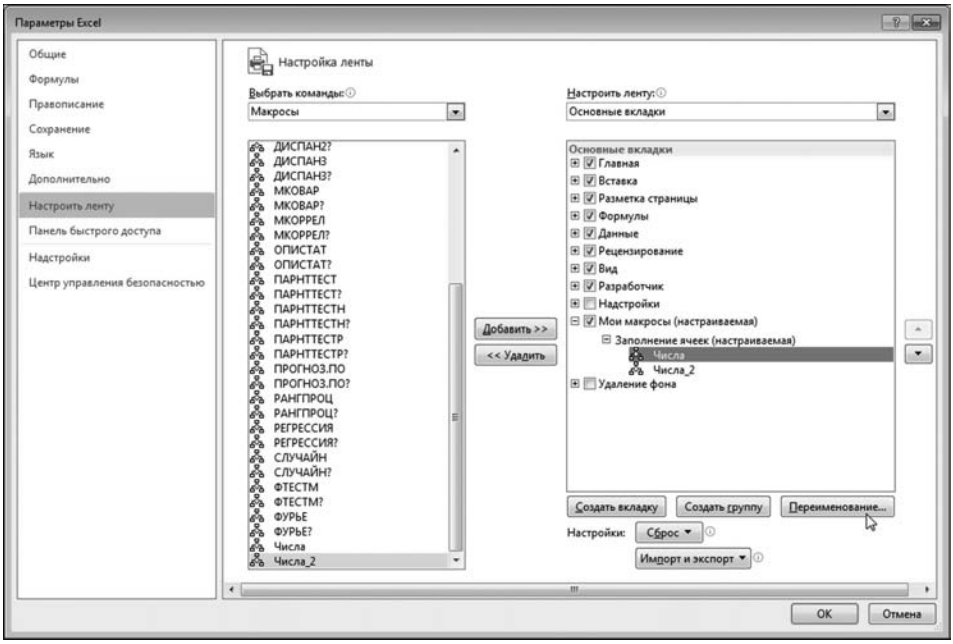


Рис. 9.30
Изменение пиктограммы макроса Числа

Пиктограмма макроса Числа появляется в списке пиктограмм группы **Заполнение ячеек** (рис. 9.29).

Точно так же поступаем с пиктограммой для макроса Числа_2. После этого обе пиктограммы помещаются в группу **Заполнение ячеек**. Собственно, можно открывать рабочее окно и наблюдать дополнительную вкладку **Мои макросы** с группой **Заполнение ячеек** из двух пиктограмм. Однако в этом случае, как и при размещении пиктограммы на панели быстрого доступа, изображения пиктограмм будут одинаковые и не очень красивые (хотя последнее — дело вкуса). В любом случае имеет смысл для пиктограмм подбирать уникальные идентификаторы (в том числе и изображения). Именно так мы и сделаем — изменим внешний вид пиктограмм запуска макросов. Для этого в списке пиктограмм группы **Заполнение ячеек** выделяем нужную позицию и щелкаем кнопку **Переименование** (рис. 9.30).

Рис. 9.31
В окне **Переименование** указывается название и изображение для пиктограммы макроса **Числа**

Открывается знакомое нам окно **Переименование**, в котором задаем текст пиктограммы и изображение.

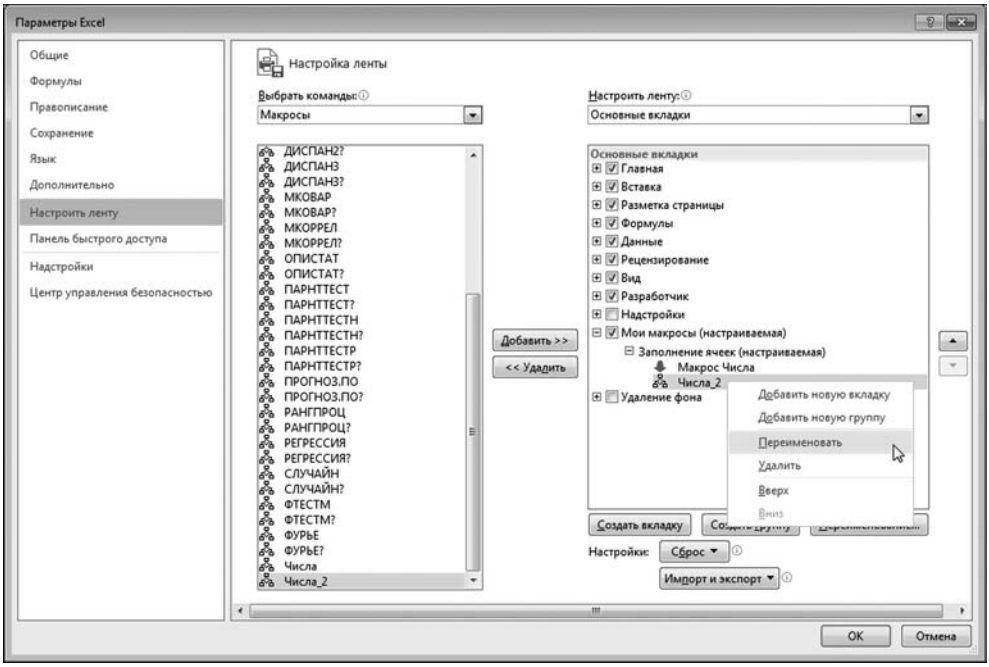


Рис. 9.32
Изменение пиктограммы макроса **Числа_2**

На рисунке 9.31 показан процесс редактирования параметров пиктограммы для макроса Числа.

Внесенные изменения отображаются в окне **Параметры Excel**. После того как изменены свойства первой пиктограммы (для макроса **Числа**), приступаем к изменению свойств пиктограммы для макроса Числа_2, для чего можем, например, воспользоваться командой контекстного меню **Переименовать** (рис. 9.32).

Рис. 9.33
В окне **Переименование** указывается название и изображение для пиктограммы макроса Числа_2

Процесс изменения свойств пиктограммы для макроса Числа_2 показан на рисунке 9.33.

После внесения изменений в свойства пиктограмм наша новая вкладка готова к работе. Для подтверждения внесенных правок в окне **Параметры Excel** стоит лишь щелкнуть на кнопке **OK** (рис. 9.34).

На заметку

Чтобы выбрать для созданной нами вкладки **Мои макросы** место на ленте можем

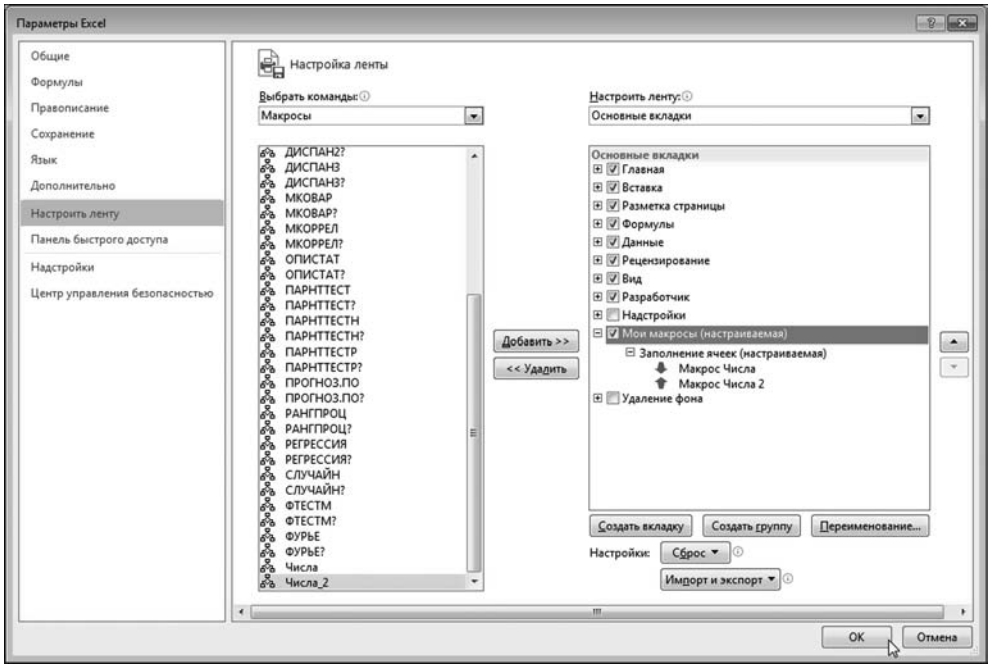


Рис. 9.34
Вкладка пользователя создана

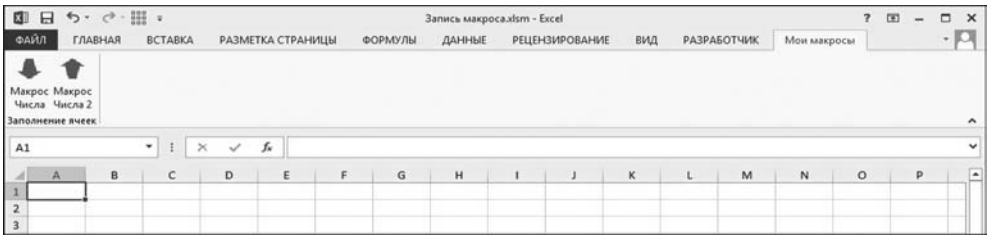


Рис. 9.35
Вкладка пользователя с пиктограммами в рабочем окне

воспользоваться кнопками перемещения вкладки в окне **Параметры Excel**. Кнопки с изображениями небольших стрелок находятся справа в окне, за списком вкладок. С их помощью выделенная вкладка перемещается вверх/вниз по списку вкладок. Место в списке определяет порядок отображения вкладок на ленте.

На рисунке 9.35 показано окно приложения с лентой, открытой на вкладке **Мои макросы**.

Как и ожидалось, вкладка **Мои макросы** состоит из одной группы (но при желании групп может быть и больше) **Заполнение ячеек**. В группе две пиктограммы — **Макрос Числа** и **Макрос Числа 2**. Щелчок на пиктограмме приводит к запуску соответствующего макроса.

Это еще не все. Для запуска макросов научимся создавать специальные кнопки, которые размещаются в рабочем документе. Обратимся к вкладке **Разработчик** в той ее части, что относится к группе **Элементы управления** (рис. 9.36).

В частности, в группе **Элементы управления** в раскрывающемся списке пиктограммы **Вставить** выбираем левую верхнюю пиктограмму, которая предназначена для вставки кнопки. После этого курсор мыши принимает вид небольшого креста. В этом режиме необходимо выбрать место в рабочем документе для размещения кнопки. Делается это захватом области с помощью мыши (рис. 9.37).

Как только область для размещения кнопки выделена, появляется окно **Назначить макрос объекту**, где следует выбрать макрос, который будет выполняться при щелчке на кнопке. Мы выбираем макрос **Числа** (рис. 9.38).

Кнопка размещается в области документа, но по умолчанию на кнопке отображается не очень информативный текст. Его желательно изменить. Поэтому выделяем кнопку и раскрываем контекстное меню, в котором выбираем команду **Изменить текст** (рис. 9.39).

На заметку

Выделить кнопку можно щелчком правой кнопкой мыши. Если щелкнуть левой кнопкой мыши, запустится на выполнение макрос.

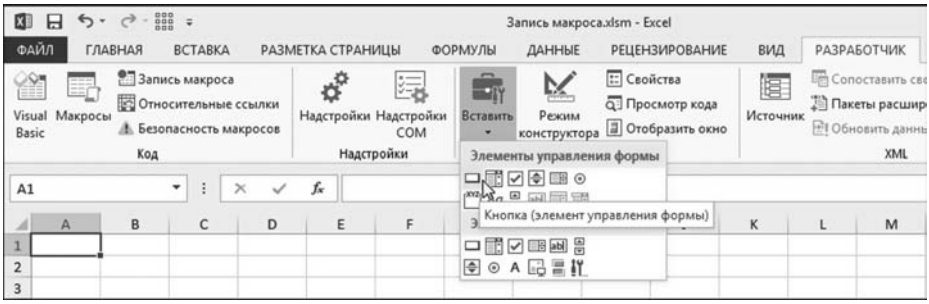


Рис. 9.36
В списке пиктограммы **Вставить** выбираем пиктограмму вставки кнопки

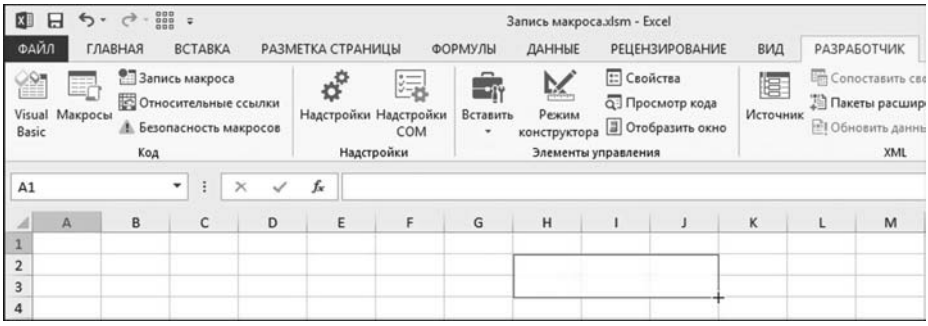


Рис. 9.37
Процесс добавления кнопки в рабочий документ

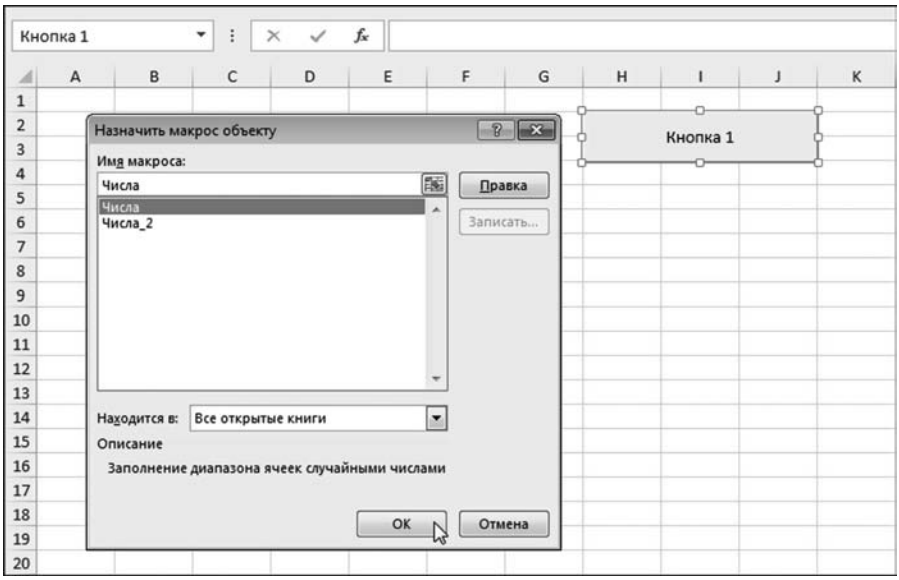


Рис. 9.38
Назначение кнопке макроса

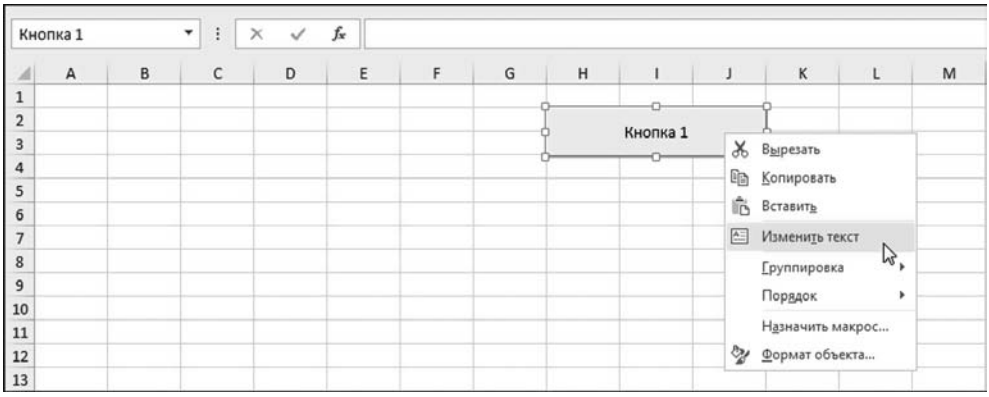


Рис. 9.39
Изменение текста кнопки

В качестве названия кнопки вводим текст Макрос “Числа”. При этом на вкладке **Главная** можно, например, задать стиль и размер шрифта, используемого при отображении названия кнопки (рис. 9.40).

На рисунке 9.41 показано окно рабочего документа с кнопкой для запуска макроса Числа.

Нажав на кнопку, получаем результат, показанный на рисунке 9.42.

В принципе, получилось достаточно элегантно — в то же время не всегда удобно, поскольку кнопка своим присутствием закрывает часть рабочей области документа.

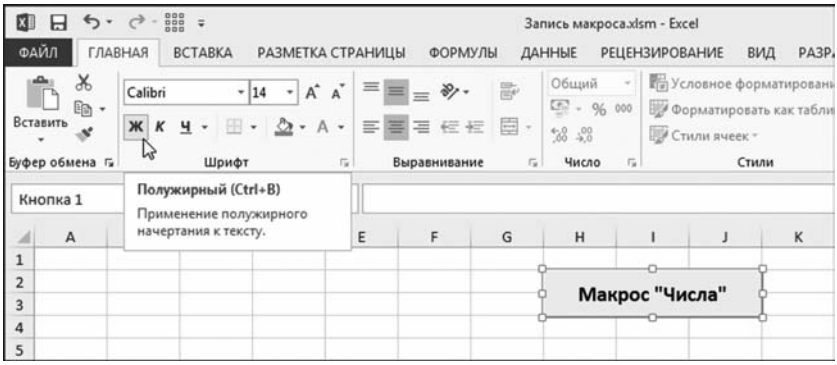


Рис. 9.40
Изменение свойств текста кнопки

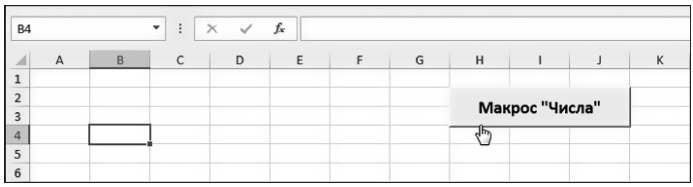


Рис. 9.41
Нажатие на кнопку приводит к выполнению макроса **Числа**

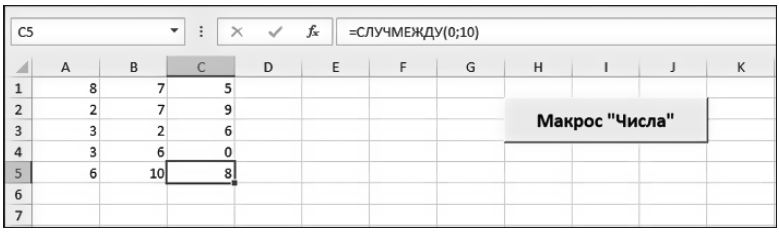


Рис. 9.42
Результат выполнения макроса **Числа** после щелчка на кнопке

ПРОГРАММНЫЙ КОД МАКРОСА

— И откуда из Вас латынь-то эта выскакивает?
Сами-то Вы вроде не из латинцев.
— Да барин у нас прежний всех мужиков заставлял латынь учить.
Желаю, говорит, думать, будто я в Древнем Риме.
Большой просветитель был!

Из к/ф «Формула любви»

Мы видели, как четко и слаженно выполняются макросы. Мы их вызывали и сами, и с помощью пиктограммы на панели быстрого доступа, и с помощью пиктограмм на созданной вкладке, и с помощью кнопки в рабочей области. Каждый раз макрос выполнял свою задачу точно и быстро — так сказать, в соответствии с прейскурантом. Естественным образом возникает

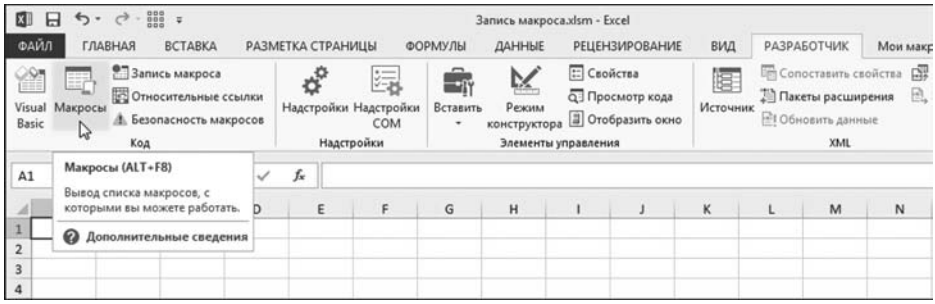


Рис. 9.43
На вкладке **Разработчик** щелкаем пиктограмму **Макросы**

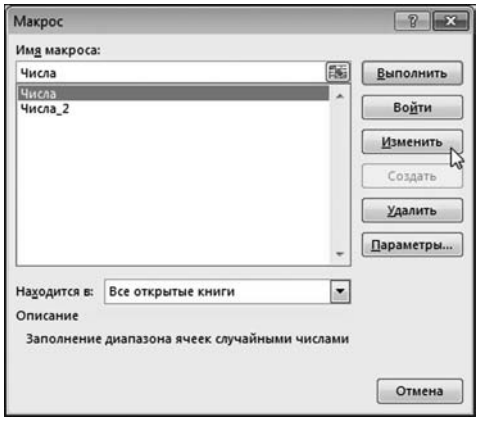


Рис. 9.44
В окне **Макрос** выделяем имя макроса и щелкаем вкладку **Изменить**

вопрос: образно выражаясь, кто же стоит за кулисами? Другими словами, как и где определяются те действия, что должны быть выполнены макросом? Ответы на все наши вопросы даст программный код макроса.

Чтобы увидеть программный код макроса, поступаем следующим образом. На вкладке **Разработчик** в группе **Код** щелкаем пиктограмму **Макросы** (рис. 9.43).

Такую процедуру мы уже выполняли и знаем, что в результате открывается окно **Макрос** со списком макросов. Мы, как и ранее, выбираем в списке нужный макрос (здесь это макрос Числа), но щелкаем, в отличие от предыдущих случаев, не на кнопке **Выполнить**, а на кнопке **Изменить** (рис. 9.44).

Открывается окно редактора кодов Visual Basic Editor (VBE) с программным кодом макроса Числа, а заодно и с кодом макроса Числа_2, как это видно из рисунка 9.45 (просто записаны макросы в одном месте).

На заметку

Того же результата можно добиться проще — достаточно щелкнуть на пиктограмме **Visual Basic** в группе **Код** на вкладке **Разработчик** (рис. 9.43).

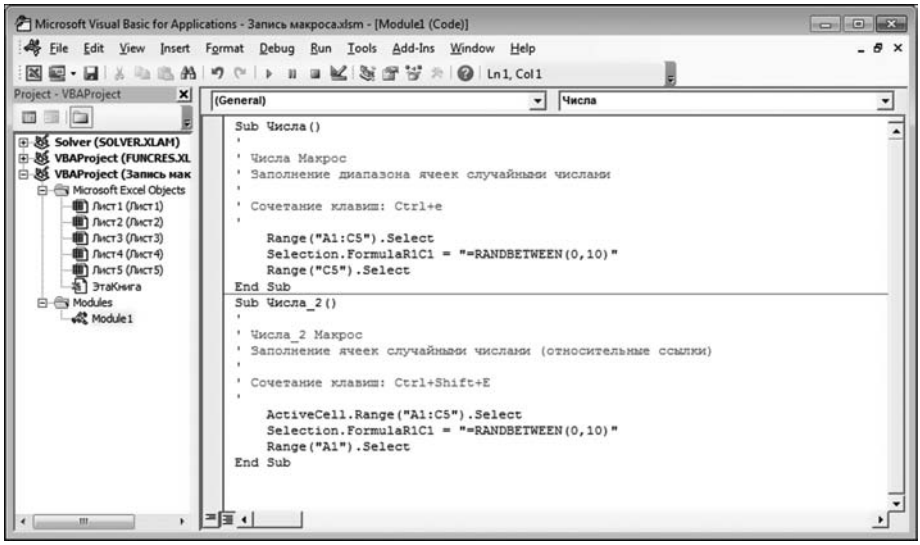


Рис. 9.45
Окно редактора VBE с программным кодом макросов

Все, что нам осталось сделать — разобраться с программным кодом. В листинге 9.1 представлен программный код макроса Числа.

Листинг 9.1. Программный код макроса Числа

```
Sub Числа()
 '
 ' Числа Макрос
 ' Заполнение диапазона ячеек случайными числами
 '
 ' Сочетание клавиш: Ctrl+e
 '
 Range("A1:C5").Select
 Selection.FormulaR1C1 = "=RANDBETWEEN(0,10)"
 Range("C5").Select
End Sub
```

Разумеется, для анализа программного кода необходимо иметь хотя бы минимальное представление об особенностях языка VBA, на котором, собственно, и написан код. Знания в этой области будем принимать как яд — небольшими порциями, для тренировки иммунной системы. Итак, правило первое: все, что начинается с апострофа (символ ') называется *комментарием* и при выполнении макроса игнорируется. Комментарии предназначены для пользователя/программиста. Они важны, поскольку обычно содержат ценные пояснения к коду. Первый комментарий в макросе особенно ценен, поскольку на его основе формируется подсказка для макроса. Но не более. На функциональность кода комментарий не влияет.

Правило второе: начинается описание макроса ключевым словом Sub, а заканчивается конструкцией End Sub. После начальной инструкции Sub указывается имя макроса и пустые круглые скобки. Между начальным Sub и финальной конструкцией End Sub находится непосредственно код макроса.

На заметку

Напомним, что макрос является процедурой VBA. Особенность этой процедуры в том, что она не имеет аргументов. Другими словами, макрос является частным случаем процедуры. При описании процедур в VBA в общем случае после имени процедуры в круглых скобках указываются аргументы. У макроса аргументов нет. Поэтому после имени макроса круглые скобки пустые.

Фактически, весь код макроса состоит из трех команд:

- первая команда: Range("A1:C5").Select;
- вторая команда: Selection.FormulaR1C1 = "=RANDBETWEEN(0,10)";
- третья команда: Range("C5").Select.

Дальше углубляемся в технические подробности. Но сначала небольшой комментарий общего характера.

В VBA используется так называемый *точечный синтаксис*. На сегодня это стандарт для объектно-ориентированных языков программирования, к которым в известном смысле относится и VBA. Данное обстоятельство означает, что в VBA используются такие штуки, как *объекты*. У объектов есть *свойства* и *методы*. Свойство — это то, чем объект обладает, а метод — это то, что объект может делать. Практически все, что может представлять интерес в VBA (в том числе отдельные ячейки и диапазоны ячеек), реализовано через объекты. На практике нас будут интересовать не столько сами объекты, сколько их свойства и методы. Свойства и методы указываются в программных кодах вместе с объектами и отделяются от имени объекта точкой.

На заметку

Деление атрибутов на свойства и методы является несколько условным, поскольку нередко разница между свойством и методом достаточно иллюзорная. Поэтому к такой системе градации следует относится с известной долей философского скептицизма.

Например, инструкция Range("A1:C5") означает не что иное, как диапазон ячеек A1:C5. Это объект. У объекта есть метод Select, который означает выделение диапазона ячеек. Поэтому первая команда Range("A1:C5").Select является инструкцией выделить диапазон ячеек A1:C5. Теперь становится понятной третья команда Range("C5").Select. Несложно догадаться, что это инструкция выделить ячейку C5. Со второй командой Selection.FormulaR1C1 = "=RANDBETWEEN(0,10)" дела обстоят несколько сложнее. Инструкция Selection является объектом, который на данный момент выделен. Поскольку перед этой командой выполнялась команда Range("A1:C5").Select, то в данной конкретной ситуации Selection означает диапазон ячеек A1:C5. У такого объекта как диапазон ячеек (или отдельная ячейка) есть свойство FormulaR1C1. Это то, что в ячейки (или ячейку) записано. Командой Selection.FormulaR1C1 = "=RANDBETWEEN(0,10)" свойству FormulaR1C1

объекта Selection присваивается значение. Присваиваемое значение указывается после оператора присваивания (знак равенства =) и заключается в двойные кавычки. А именно, во все ячейки диапазона записывается формула =RANDBETWEEN(0,10). Основу ее составляет функция RANDBETWEEN() — англоязычный аналог русифицированной функции СЛУЧМЕЖДУ(). Собственно, все.

На заметку

Здесь мы неожиданно столкнулись с самым, пожалуй, печальным мотивом в истории развития Excel. Видимо из лучших побуждений при русификации Excel русифицировали и функции рабочего листа. При этом в языке VBA используются, так сказать, оригинальные англоязычные аналоги этих функций. Нечто подобное произошло и в нашей ситуации: в программном коде используется функция RANDBETWEEN(), а в рабочем документе это будет СЛУЧМЕЖДУ().

Что касается макроса Числа_2, программный код которого приведен в листинге 9.2, то от макроса Числа он принципиально отличается первой командой: вместо инструкции Range("A1:C5").Select использована инструкция ActiveCell.Range("A1:C5").Select. Кроме того, последней командой выделяется не ячейка C5, а ячейка A1.

Листинг 9.2. Программный код макроса Числа_2

```
Sub Числа_2()
 '
 ' Числа_2 Макрос
 ' Заполнение ячеек случайными числами (относительные ссылки)
 '
 ' Сочетание клавиш: Ctrl+Shift+E
 '
 ActiveCell.Range("A1:C5").Select
 Selection.FormulaR1C1 = "=RANDBETWEEN(0,10)"
 Range("A1").Select
End Sub
```

Объект ActiveCell — это активная (выделенная) ячейка (или левая верхняя ячейка выделенного диапазона). Свойство Range("A1:C5") объекта ActiveCell — это диапазон ячеек из пяти строк и трех столбцов, который начинается в активной ячейке. Другими словами, диапазон ActiveCell.Range ("A1:C5") — это диапазон ячеек такого же размера, как A1:C5, но только его левая верхняя ячейка совпадает с активной. Вызов метода Select означает выделение этого диапазона.

Резюмируя, заметим, что при запуске макроса Числа (в строгом соответствии с программным кодом) выполняются следующие действия:

- выделяется диапазон ячеек A1:C5;
- диапазон ячеек A1:C5 заполняется случайными числами (вставляется формула для генерирования случайных чисел);
- выделяется ячейка C5.

При запуске макроса Числа_2 выполняются такие действия:

- начиная с активной ячейки, выделяется диапазон ячеек из трех столбцов и пяти строк;
- выделенный диапазон ячеек заполняется случайными числами;
- выделяется ячейка A1.

Не нужно быть гуру в программировании, чтобы догадаться: макросы выполняют свою работу не оптимально. Действительно, учитывая, для чего мы создавали эти макросы, можно предложить такую схему выполнения макроса для заполнения диапазона ячеек A1:C5 случайными числами с последующим выделением ячейки C5:

- заполняется случайными числами (через вставку формулы) диапазон ячеек A1:C5;
- выделяется ячейка C5.

Также неплохо было бы, чтобы макрос, в котором мы использовали относительные ссылки, выполнялся так:

- заполняется случайными числами диапазон ячеек из трех столбцов и пяти строк, начиная с активной ячейки;
- выделяется ячейка A1.

На заметку

Выше мы небезосновательно исходили из того, что заполнять ячейки рабочего документа с помощью программного кода можно без предварительного выделения заполняемой ячейки.

Обратимся к листингу 9.3, в котором приведена альтернативная версия макроса Числа (новый макрос называется Числа_Новый).

Листинг 9.3. Программный код макроса Числа_Новый

```
Sub Числа_Новый()
 Range("A1:C5").FormulaR1C1 = "=RANDBETWEEN(0,10)"
 Range("C5").Select
End Sub
```

Кроме формальных атрибутов в виде инструкций начала и окончания макроса, он состоит из двух команд. Командой Range("A1:C5").FormulaR1C1 = "=RANDBETWEEN(0,10)" диапазон ячеек A1:C5 заполняется выражениями вида =СЛУЧМЕЖДУ(0;10). Командой Range("C5").Select выделяется ячейка C5.

Для макроса Числа_2 создаем «оптимизированного собрата», который называется Числа_2_Новый. Код этого макроса представлен в листинге 9.4.

Листинг 9.4. Программный код макроса Числа_2_Новый

```
Sub Числа_2_Новый()
 ActiveCell.Range("A1:C5").FormulaR1C1 = "=RANDBETWEEN(0,10)"
 Range("A1").Select
End Sub
```

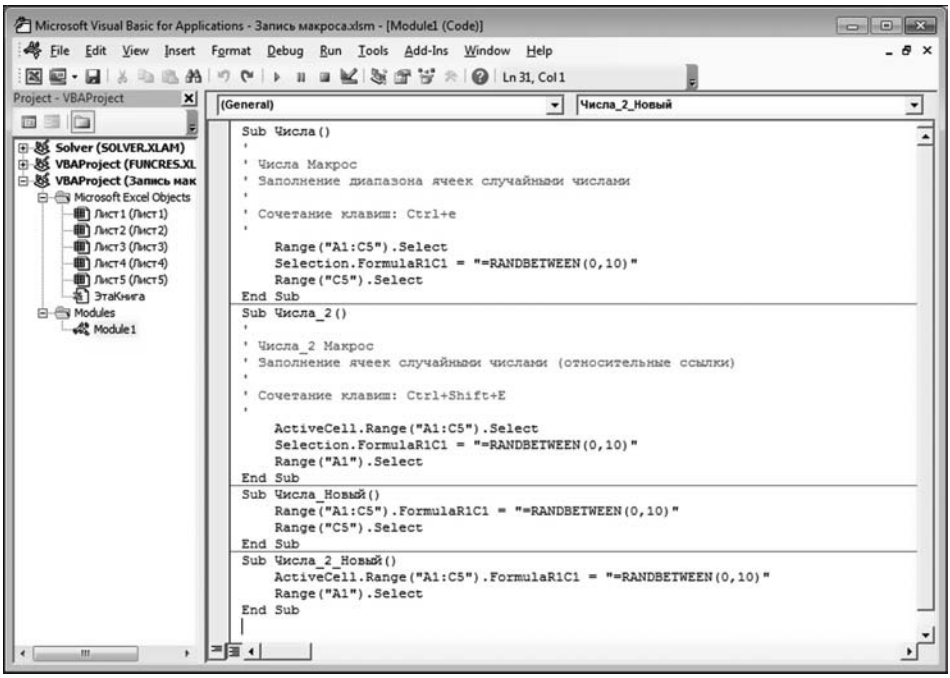


Рис. 9.46
Окно редактора VBE с кодом вновь созданных макросов

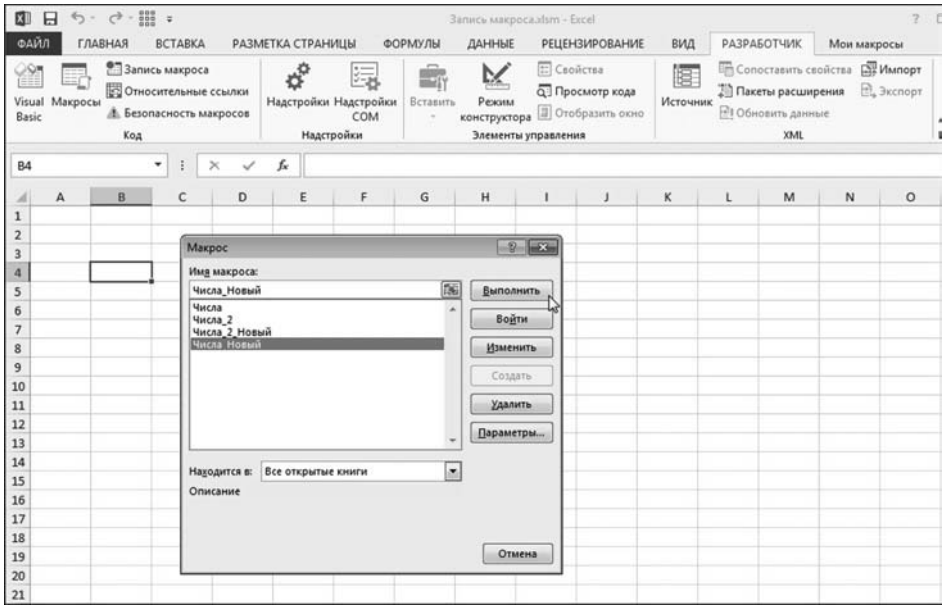


Рис. 9.47
Запуск вновь созданного макроса на выполнение

Здесь ActiveCell.Range(“A1:C5”).FormulaR1C1 = ”=RANDBETWEEN(0,10)” заполняется диапазон ячеек в три столбца и пять строк, начиная с активной ячейки (левая верхняя ячейка заполняемого диапазона). Ячейка A1 выделяется с помощью команды Range(“A1”).Select. Осталось проверить работоспособность созданных макросов. Для этого код каждого из макросов копируем (или вводим с клавиатуры) в окне редактора VBA сразу под программными кодами ранее созданных макросов (рис. 9.46).

После того как новые макросы внесены в модуль проекта, их можно запустить из рабочего документа приложения Excel. Напомним: на вкладке **Разработчик** в группе **Код** щелкаем пиктограмму **Макросы**. Откроется диалоговое окно **Макрос**, в котором выбираем макрос для выполнения (рис. 9.47).

Помимо уже знакомых нам макросов Числа и Числа_2, теперь в списке макросов диалогового окна **Макрос** появились еще и макросы Числа_Новый и Числа_2_Новый. Желающие могут убедиться самостоятельно, что макросы функционируют, причем так, как мы и предполагали при их создании.

Более подробно о программных кодах мы поговорим в следующей главе.

ГЛАВА 10 ОСНОВЫ ПРОГРАММИРОВАНИЯ В VBA

Позвольте Вас спросить,
какое такое понятие Вы про меня имеете?
Я спрашиваю Вас!

Из к/ф «За двумя зайцами»

В этой главе мы расширим свои познания в области программирования в VBA и, более конкретно, в области создания процедур и функций пользователя. Чтобы справиться с этой задачей, необходимо, во-первых, хотя бы немного знать язык VBA и, во-вторых, хотя бы немного владеть навыками работы со встроенным редактором VBA. Эти два компонента просто необходимы для «приготовления хорошего блюда» в виде работающего программного кода. В предыдущей главе мы уже имели дело как с программными кодами, так и с редактором VBE (точнее, кратко упоминали этот редактор). Также ранее мы освоили некоторые несложные команды языка VBA. Здесь мы познакомимся с особенностями языка программирования VBA более детально и научимся применять его на практике. Вместе с тем, следует понимать, что программирование в VBA — это тема для отдельной книги. Поэтому в этой и следующей главах мы будем обсуждать лишь основные конструкции и приемы, ориентированные, в первую очередь, на получение быстрого и качественного результата.

На заметку

Конечно, для изучения VBA нужно проработать, по крайней мере, несколько специализированных книг. Но самое главное — программировать. Программировать всегда и везде. Без этого не приходится рассчитывать на успех.

Помимо рассмотрения основных приемов и методов программирования, в этой главе мы также получим базовые сведения о встроенном редакторе кодов. Точнее, функциональные элементы редактора кодов мы будем изучать по мере необходимости в процессе решения тех или иных задач. Начнем же с обзора возможностей редактора VBE.

РЕДАКТОР КОДОВ VBE

Если есть на этом Плюке гравицапа — так достанем.
Не такое доставали.

Из к/ф «Кин-дза-дза»

Картинку с изображением встроенного редактора программных кодов VBE мы уже несколько раз видели. Правда, при этом мы практически никак не комментировали само окно. Здесь прежде всего напомним, каким образом можно открыть окно редактора VBE. Самый простой способ — воспользоваться пиктограммой **Visual Basic** в группе **Код** на вкладке **Разработчик** (рис. 10.1).

В результате открывается окно редактора кодов, которое может выглядеть так, как показано на рисунке 10.2.

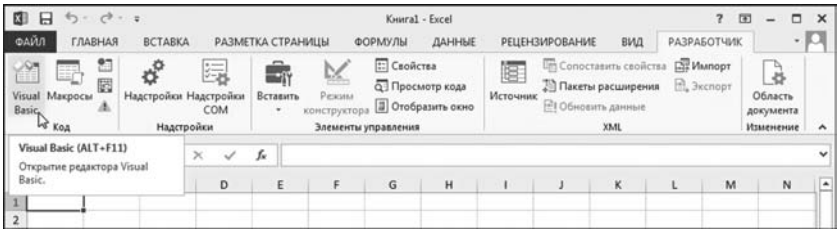


Рис. 10.1
Щелчок на пиктограмме **Visual Basic**
позволяет открыть окно редактора программных кодов

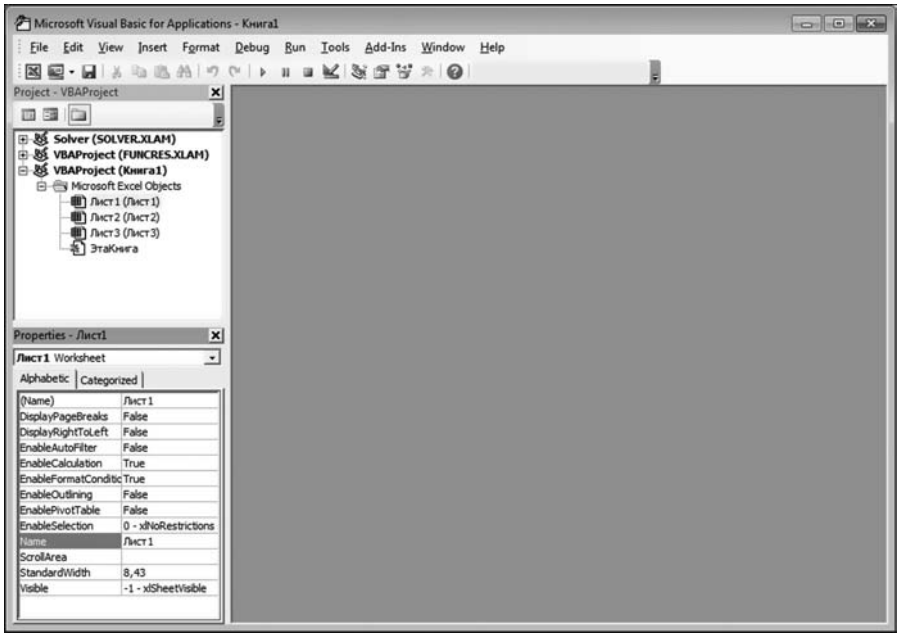


Рис. 10.2
Окно редактора VBE для пустого документа

На заметку

Редактор кодов VBE имеет «архаичный» нелентоочный интерфейс. Для VBE характерно наличие панели меню и панели инструментов. Панель меню содержит команды для работы как непосредственно с редактором, так и с программными кодами, которые обрабатываются в этом редакторе. Панель инструментов содержит набор пиктограмм для основных команд. На рисунке 10.2 отображается стандартная панель инструментов. При желании можно отобразить и другие панели (или убрать стандартную панель).

Кроме того, во внутренней области может отображаться несколько подокон. В документе на рисунке 10.2 два подокна: окно проекта **Project** и окно свойств **Properties**. В окне **Project** отображается объектная структура текущего проекта, который отождествляется с рабочей книгой. В окне свойств **Properties** отображаются свойства того объекта, который выделен в окне проекта **Project**. Кроме того, первостепенное значение имеет окно, в котором непосредственно набираются программные коды. Оно на рисунке 10.2 не отображается, но должно быть в том месте, где пустая серая область — в правой части внутренней области окна редактора VBE.

Для отображения внутренних окон используем меню **View**, в котором представлен список команд, в том числе и команды для отображения подокон.

Основное назначение окна редактора кодов, как несложно догадаться — работа с программными кодами, написанными на VBA. Под «работой» здесь подразумевается создание новых кодов, редактирование уже существующих, их отладка и многое другое. Нас в известном смысле будут интересовать три вопроса:

1. Что (какие команды) написать в программном коде?
2. Куда записать программный код (или где найти уже записанный)?
3. Что можно делать с программным кодом?

Мы стартуем со второго вопроса и плавно переключимся на третий. Затем вернемся к первому вопросу.

Итак, куда можно записывать программный код? В принципе, есть несколько хороших мест, но лучше всего записывать его в отдельный модуль. Модуль можно создать специально или подождать, пока он будет создан автоматически при выполнении действий по созданию функции или процедуры. Начнем с создания модуля средствами редактора VBE. Поможет нам в этом команда **Module** в списке команд меню **Insert** (рис. 10.3).

На заметку

В списке команды меню **Insert**, помимо команды создания модуля, есть очень полезные команды для создания процедур, функций и модулей классов. Фактически, это весь джентльменский набор, который нам понадобится для работы.

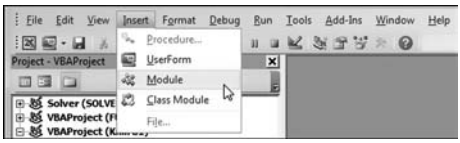


Рис. 10.3
Создание модуля

Собственно на этом основные действия по созданию модуля закончены. Первым признаком того, что модуль действительно создан, будет появление в окне проекта (внутреннее окно с именем **Project** — обычно в левом верхнем углу рабочей области редактора) узла

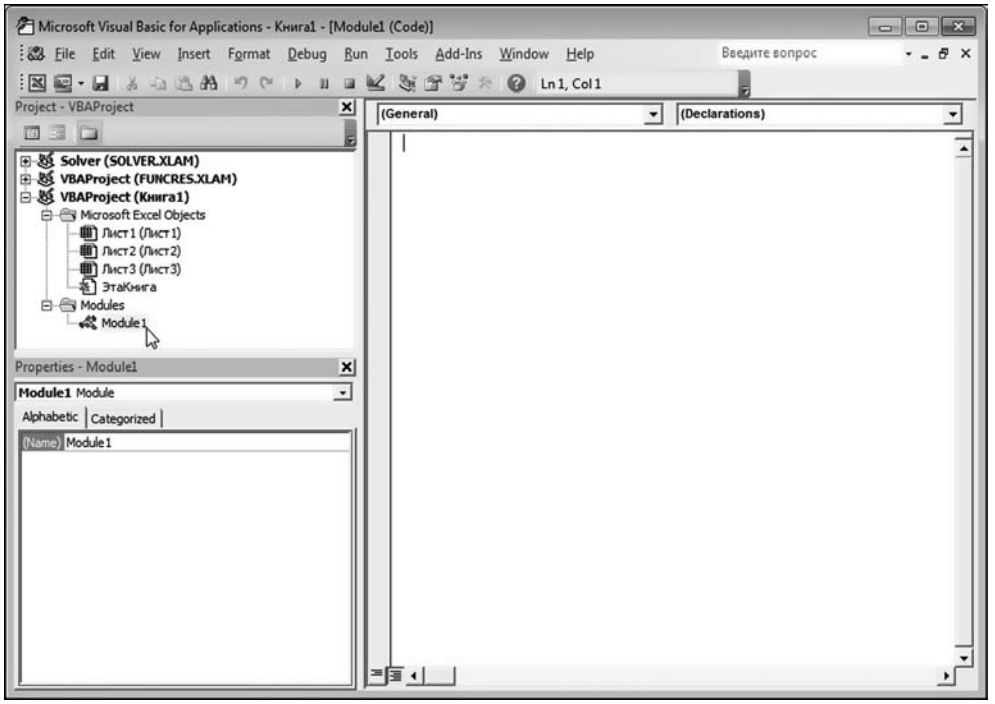


Рис. 10.4
Модуль создан

Modules с пиктограммой созданного модуля (по умолчанию имя **Module1**), как показано на рисунке 10.4.

Обычно автоматически в правой части внутренней области окна появляется белое текстовое поле — в нем отображается содержимое модуля. Новый модуль пустой, поэтому ничего там нет. А в принципе здесь мог бы быть (и будет) наш программный код.

На заметку

Белое поле в правой части окна редактора еще не означает, что открыто содержимое модуля. В окне проекта, кроме модуля, есть узел объектов Excel, который называется **Microsoft Excel Objects** и содержит объекты для рабочих листов и книги как таковой. Каждый из этих объектов может содержать свой код. Чтобы просмотреть код объекта, на пиктограмме этого объекта в окне проекта следует выполнить двойной щелчок мышью. При обычном выделении объекта автоматического переключения на его содержимое в правой части окна редактора не происходит. Другими словами, чтобы просмотреть содержимое модуля, следует выполнить на пиктограмме этого модуля двойной щелчок мышью.

Мы в основном будем программный код вводить или редактировать. Не последнее место также занимает отладка кода, но это уже специальная тема, и затронем мы ее вскользь — приоритетной для нас все же будет процедура ввода программного кода. Редактор кодов в этом случае достаточно удобен и

имеет несколько полезных и приятных свойств. Например, при вводе кода в окне редактора появляется контекстная справка по свойствам и методам (при вводе свойства или метода с использованием точечного синтаксиса в раскрывающемся списке появляется полный список атрибутов объекта). Также весьма удобна справка по аргументам встроенных функций VBA, которая появляется автоматически в процессе ввода аргументов функции. В качестве небольшого примера рассмотрим код макроса, представленный в листинге 10.1.

Листинг 10.1. Макрос для отображения имени рабочего документа

```
Sub ShowWorkbookName()
  MsgBox “Вы работаете с документом “ & ActiveWorkbook.Name
End Sub
```

Макрос (процедура) состоит всего из одной команды, которой отображается диалоговое окно с текстом, содержащим название активной рабочей книги. Вся инструкция имеет вид MsgBox текст. В результате выполнения этой инструкции отображается стандартное диалоговое окно с сообщением текст. В рассматриваемом примере текстовое сообщение получается объединением строки “Вы работаете с документом ” и текстового значения ActiveWorkbook.Name.

На заметку

Для объединения текстовых значений использован оператор конкатенации (объединения) &.

Инструкцией ActiveWorkbook.Name в качестве значения возвращается имя активного рабочего документа. Инструкция представляет собой пример точечного синтаксиса: объект ActiveWorkbook соответствует активной рабочей книге. Свойство Name для объекта ActiveWorkbook означает имя рабочего документа (то, что отображается в строке названия приложения Excel). Процесс ввода программного кода макроса в редакторе кодов проиллюстрирован на рисунке 10.5.

После ввода имени ActiveWorkbook с точкой автоматически появляется список свойств и методов данного объекта. В этом списке есть и свойство Name. В принципе, достаточно удобный режим — особенно если не знаешь, какое именно свойство объекта нужно использовать.

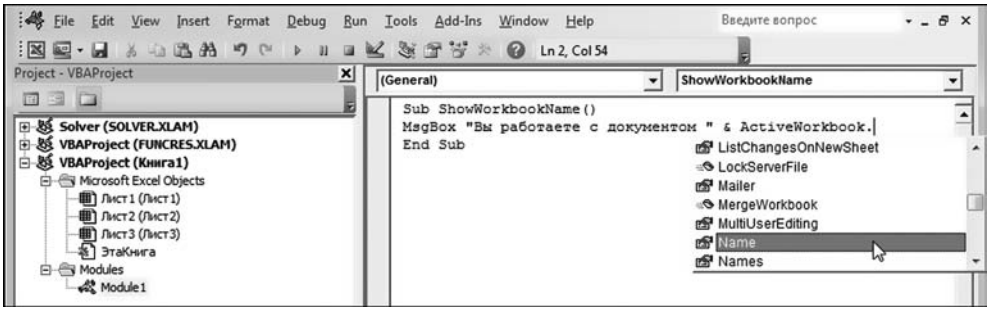


Рис. 10.5
Процесс ввода программного кода макроса

На заметку

Но вернемся к нашему макросу. Желательно проверить его работу. Для этого можно запустить макрос стандартным способом из рабочего окна приложения Excel, а можно воспользоваться специальной утилитой редактора программных кодов (рис. 10.6).

На панели инструментов есть несколько кнопок, предназначенных для работы с макросами. Пиктограмма для запуска макроса на выполнение имеет вид небольшой зеленой стрелки. Щелчок на ней приводит к запуску макроса на выполнение. Результат выполнения макроса показан на рисунке 10.7.

Как и ожидалось, появляется диалоговое окно с кнопкой **OK** и текстом Вы работаете с документом таким-то. Пока на кнопке **OK** не щелкнем, окно не спрячется.

Команды для работы с макросами также доступны в меню **Run**. Полезной для запуска макроса на выполнение может быть команда **Tools ▷ Macros**.

Запустить на выполнение можно макрос. Функции пользователя, о которых речь пойдет чуть позже, не запускаются в принципе — во всяком случае, так прямолинейно, как макросы. Поэтому сказать сразу, корректный ли у них программный код бывает достаточно сложно. Здесь мы плавно

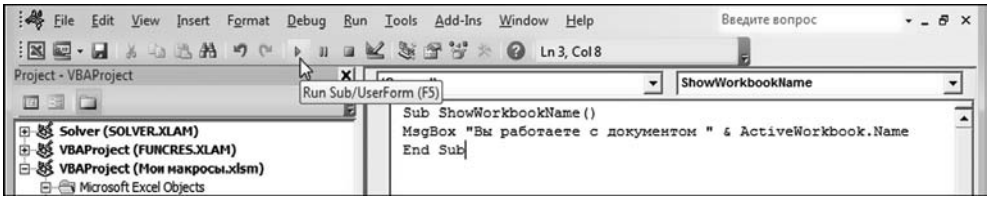


Рис. 10.6
Запуск макроса на выполнение

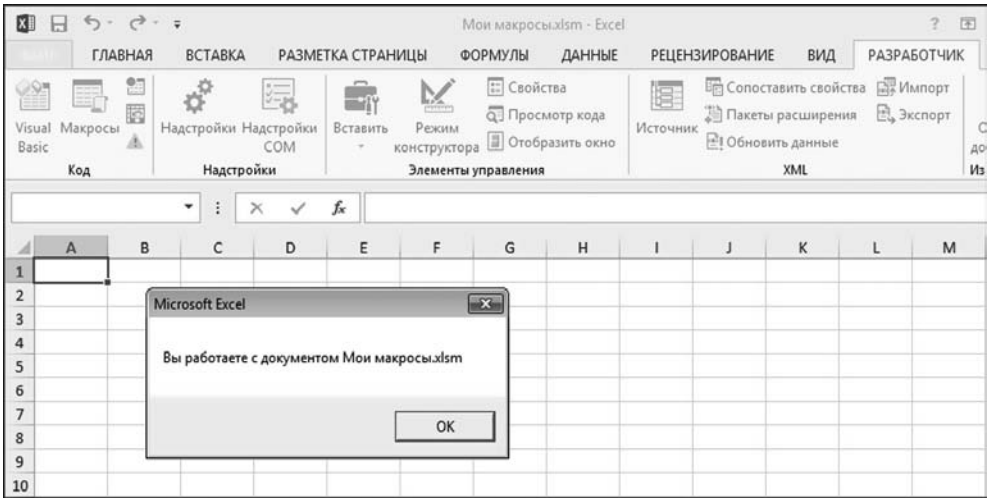


Рис. 10.7
Результат выполнения макроса

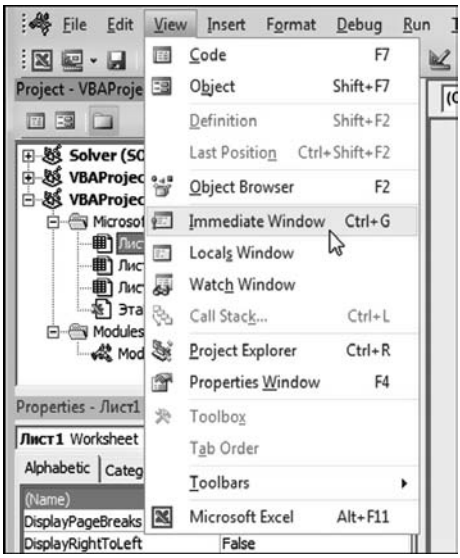


Рис. 10.8
Отображаем окно отладки программного кода

подходим к мысли, что неплохо было бы иметь возможность анализировать программный код на предмет наличия ошибок. Нам нужен отладчик кодов, и такой в VBE есть.

Команды для отладки программных кодов собраны в основном в меню **Debug**. Кроме того, полезным будет окно отладки **Immediate**. Чтобы открыть окно, воспользуемся командой **View ▷ Immediate Window** (рис. 10.8).

На заметку

Для выполнения различных действий в редакторе кодов предлагаются «горячие» сочетания клавиш. Эти комбинации отображаются в списках команд меню. Ими не следует пренебрегать, поскольку они значительно упрощают работу с редактором. Например, открыть окно отладки программного кода можно с помощью комбинации клавиш <Ctrl>+<G>.

В результате (по умолчанию в нижней части) появится окно отладки программных кодов **Immediate**, как показано на рисунке 10.9.

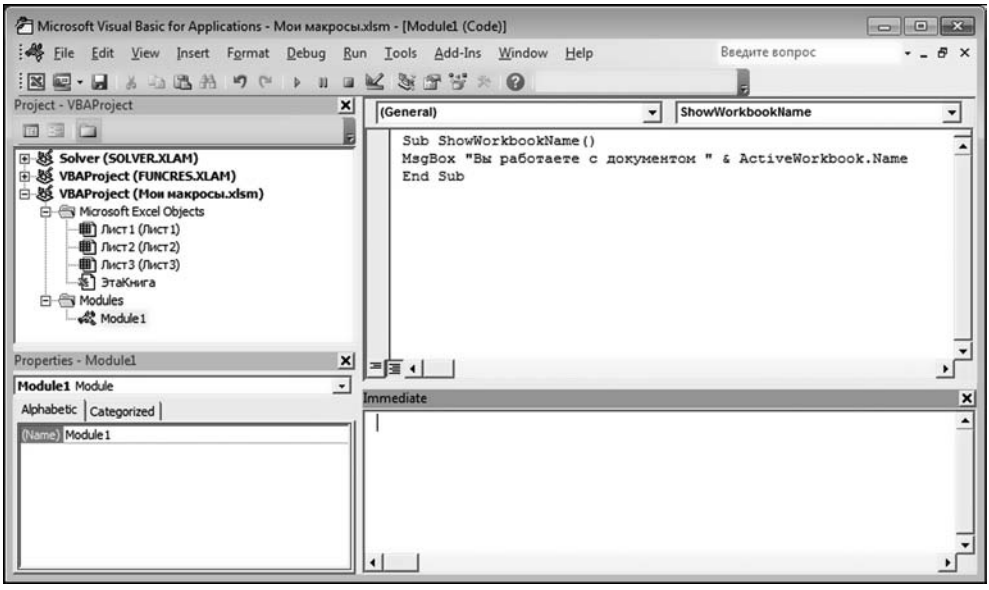


Рис. 10.9
Окно редактора программных кодов с подокном отладки программного кода **Immediate**

Полезность окна отладки кодов в первую очередь связана с тем, что в этом окне можно запускать команды на выполнение. Например, вводим в окне команду ShowWorkbookName и нажимаем клавишу <Enter> (рис. 10.10).

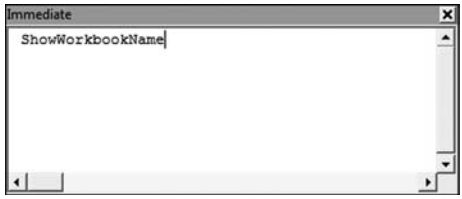


Рис. 10.10
Выполнение команды в окне отладки программного кода

Результат будет таким же, как при запуске макроса из рабочего документа Excel.

Более детально останавливаться на методах работы с редактором кодов не будем. В случае необходимости мы опишем соответствующие действия, когда они окажутся важными с точки зрения достижения нужных результатов. В заключение раздела отметим лишь, что основные настройки редактора кодов выполняются в окне **Options** (рис. 10.11), которое открывается с помощью команды **Tools ▷ Options**.

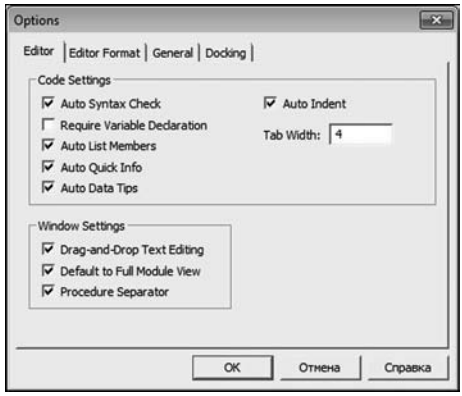


Рис. 10.11
В диалоговом окне **Options** выполняются настройки редактора VBE

У окна четыре вкладки, каждая из которых содержит подборку полезных утилит, которыми задаются основные настройки, режимы и характеристики редактора VBE.

СОЗДАНИЕ ФУНКЦИИ ПОЛЬЗОВАТЕЛЯ

— Я и понятия в этом деле не имею!
— Не беспокойтесь — я имею понятие!

Из к/ф «За двумя зайцами»

Теперь мы переходим к вопросу о том, какие команды/инструкции можно и нужно писать в программном коде. Начнем с простых вещей, а именно, с создания функции пользователя.

Программный код функции по своей структуре напоминает процедуру/макрос. Начинается описание функции ключевым словом Function, а заканчивается инструкцией End Function. После ключевого слова Function, с которого начинается описание функции, следует имя функции, список аргументов и тип результата. Пример несложной функции пользователя приведен в листинге 10.2.

Листинг 10.2. Функция пользователя для отображения типа аргумента

```
Function Get ArgType(arg) As String
 GetArgType = "Тип аргумента: " & TypeName(arg)
End Function
```

Функция называется GetArgName. У функции один аргумент, который в программном коде обозначен как arg и указан в круглых скобках. После имени функции имеется инструкция As String, что означает буквально следующее: функция в качестве результата возвращает текстовое значение (значение типа String).

На заметку

Инструкция As String не является обязательной. В этом случае результат функцией все равно возвращается, но такой результат будет типа Variant. Это несколько особый тип в VBA, и о нем мы поговорим чуть позже.

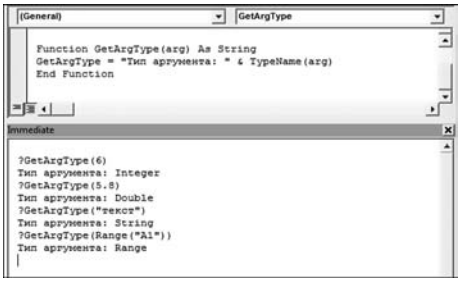


Рис. 10.12
Проверка корректности работы функции в окне **Immediate**

Программный код функции, если не считать внешних атрибутов функции, состоит всего из одной команды GetArgType = "Тип аргумента: " & TypeName(arg). Формально это команда присваивания: в соответствующем выражении слева от оператора присваивания = (знак равенства) имя функции, а справа — текстовое выражение, которое возвращается функцией в качестве результата. Текст-результат получается объединением фразы "Тип аргумента: " и значения выражения TypeName(arg). В последнем использована встроенная функция VBA TypeName, которая в качестве результата возвращает текстовое выражение для типа значения, переданного аргументом функции. Все просто.

Теперь настал черед проверить работу функции. Есть два стратегических подхода: а) можно воспользоваться окном отладки кодов; б) можно проверить работу функции непосредственно в рабочем документе. Сделаем и то, и другое. Для проверки работы функции в окне отладки кодов **Immediate** вводим команду такого формата: ?функция(аргументы). Например, это может быть команда ?GetArgType(6), в результате выполнения которой получаем текстовое значение Тип аргумента: Integer (рис. 10.12).

Там же представлены результаты вызова функции с другими аргументами.

На заметку

Ключевое слово Integer означает целочисленный тип данных в VBA. Кроме этого, встречаются следующие ключевые слова: Double (действительное число с плавающей точкой), String (текстовое значение) а также Range (диапазон ячеек/ячейка). Причем для того, чтобы сослаться на ячейку (т. е. передать аргументом функции ячейку) мы использовали инструкцию Range("A1").

Что касается рабочего документа, то здесь нас ждет несколько приятных неожиданностей. Во-первых, созданная нами функция есть в списке встроенных (точнее, пользовательских) функций рабочего листа. Во-вторых, при вводе функции в формуле появляется контекстная справка с названием функции. В-третьих, при вызове функции с целочисленным аргументом получа-

ем несколько неожиданный результат, как это можно видеть из документа на рисунке 10.13.

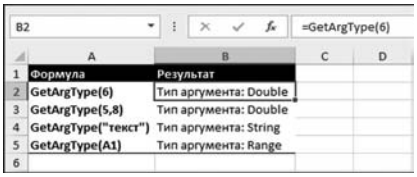


Рис. 10.13
Вызов функции пользователя из рабочего документа

Диапазон ячеек A2:A5 содержит текстовые значения формул, которыми (с добавлением знака равенства в начале) заполнены смежные ячейки. Есть несколько обстоятельств, на которые стоит обратить внимание. В первую очередь, это команда GetArgType(6), результатом которой в данном случае является тип Double для аргумента функции. Объяснение такое, что в рабочем листе и целые, и действительные числа обрабатываются как значения с плавающей точкой, отсюда и результат. Также обращаем внимание читателя, что в рабочем документе десятичным разделителем является запятая (в отличие от редактора VBE, в котором разделителем целой и десятичной части является точка).

На заметку

Символ, который используется в качестве десятичного разделителя в рабочем документе Excel, определяется настройками приложения. По умолчанию используются системные разделители. Для русифицированной версии Excel это запятая. Для англоязычной версии Excel это точка.

Наконец, ссылку на ячейку мы передаем аргументом функции, как обычно это делается при работе с функциями рабочего листа — просто указываем адрес соответствующей ячейки.

Несмотря на внешнюю простоту рассмотренного примера, он вполне может навести на некоторые размышления, за которыми кроется нетривиальная проблема. Связана она с тем, что для эффективного использования функций пользователя обычно приходится предусмотреть варианты передачи аргументом функции значений разных типов. Обстоятельства нередко складываются так, что определение типа аргумента функции имеет первоочередное значение.

На заметку

Тип аргумента функции можно указать в явном виде. Для этого после имени аргумента указывается его тип. Между аргументом и идентификатором типа указывается ключевое слово As.

ЗНАКОМСТВО С СИНТАКСИСОМ VBA

— Силь ву пле, дорогие гости! Силь ву пле!
Же ву при авек плезир! Господи, от страха все слова повыскакивали.
Алексис, они что, по-нашему совсем не понимают?
— Они понимают!

Из к/ф «Формула любви»

Здесь мы познакомимся с особенностями синтаксиса VBA — в таком объеме, чтобы можно было составлять несложные программные коды. Точнее, мы рассмотрим ряд важных вопросов, которые касаются общих правил описания функций и процедур и ряд второстепенных, на первый взгляд, вопросов, которые, тем не менее, не такие уж и второстепенные.

На заметку

Мы столкнулись с двойной проблемой. Во-первых, нас интересует непосредственно синтаксис языка VBA, т. е. правила составления команд и инструкций, типы и правила использования инструкций управления и прочие смежные вопросы. Во-вторых, для эффективной работы с приложением Excel (точнее, для составления кодов, которые были бы полезны при работе с Excel) принципиально важно иметь хотя бы общие представления об объектной модели, которая реализуется для приложения Excel. Здесь мы начнем (или продолжим — это как посмотреть) знакомство непосредственно с синтаксисом. Объектная модель обсуждается по мере необходимости.

Мы уже примерно представляем, как описать функцию или процедуру (в том смысле, что знаем с какой инструкции начать описание и какой инструкцией закончить), но пока не очень хорошо представляем, что можно или нужно писать в теле процедуры/функции.

Есть несколько небольших тем, которые мы обсудим в этом разделе. Разумеется, будут и небольшие примеры. Начнем же мы с переменных и базовых типов данных.

Что же такое переменная и для чего она нужна? В принципе, переменная — это область памяти, которая имеет имя и может хранить данные определенного типа. Благодаря тому, что область именованная, к ней можно обращаться в программе просто по имени, без углубления во всякие технические подробности.

Таблица 10.1

Типы данных VBA

Ключевое слово (тип данных)	Описание
Byte	Целые неотрицательные числа в диапазоне от 0 до 255
Integer	Целые числа в диапазоне от –32 768 до 32 767
Long	Большие целые числа — несколько миллиардов значений
Single	Действительные числа
Double	Действительные числа (большой диапазон, большая точность)
Decimal	Подтип типа Variant для действительных чисел. Переменные с таким типом не объявляются
Currency	Тип данных для действительных чисел с четырьмя цифрами после десятичной запятой. Данные этого типа обычно используются в финансовых вычислениях, поскольку обеспечивают высокую точность расчетов
Boolean	Логический тип с двумя возможными значениями True (*Истина*) и False (*Ложь*)
Date	Тип для работы с датой/временем. Диапазон дат от 1 января 100 г. до 31 декабря 9999 г.
String	Тип для работы с текстовыми данными
Object	Ссылка на объект
Variant	Универсальный тип данных, позволяющий записывать в переменную фактически любые разумные значения

На заметку

В известном смысле переменная напоминает банковскую ячейку с номером. Переменной можно присвоить значение и можно считать значение переменной — аналогично, в банковскую ячейку можно что-то положить или посмотреть, что там лежит (конечно, если это наша банковская ячейка).

Тип переменной определяет, какого размера ячейку в памяти нужно выделять для хранения данных. И хотя язык VBA достаточно демократичен в том смысле, что при объявлении переменной ее тип можно не указывать, без крайней необходимости к такому демократизму лучше не прибегать. Причины этого объясняются далее.

Общая команда по объявлению переменной в VBA начинается с ключевого слова Dim и имеет такую структуру: Dim *переменная* As *тип*. Имена для переменных лучше подбирать информативные, а в качестве типа переменной указывается одно из ключевых слов, обозначающих, как несложно догадаться, тип данных. Здесь мы плавно и подошли к тому, какие в VBA есть типы данных. Они перечислены в таблице 10.1.

Как видим, для работы с числовыми данными в VBA предназначено сразу несколько типов. Мы обычно для работы с целыми числами будем использовать тип Integer. Для работы с действительными числами будем использовать тип Double. Также стоит обратить внимание на тип String для работы с текстом и тип Date для работы с датой/временем.

На заметку

Тип Object предназначен для работы с объектами. Переменные, которые являются ссылками на объекты, имеют некоторые особенности в плане использования. Они будут обсуждаться отдельно.

Как уже отмечалось, тип Variant является в известном смысле «универсальным». Обычно этот тип данных используют (явно или неявно) в случаях, когда тип соответствующей переменной предугадать проблематично. С одной стороны, это удобно. С другой, следует понимать, что платой за такое удобство является не очень экономное использование системных ресурсов. Однако в большинстве случаев здесь нет проблемы.

Следует также иметь в виду, что в VBA имена переменных (и прочих команд) нечувствительны к состоянию регистра (т. е. не имеет значения, какие буквы использованы в названии переменной). Поэтому переменные MyVar, myvar и MYVAR, например, являются одной и той же переменной. Однако если в программном коде переменная объявлена как MyVar, то все прочие варианты написания переменной в окне редактора кодов при наборе кода по умолчанию будут автоматически приводиться к такому способу написания. Пикантность ситуации, однако, в том, что переменные в VBA можно не только объявлять без типа данных, но и вообще не объявлять. В этом случае «правила приведения» имени переменной определяются «правилами первого написания» имени этой переменной.

На заметку

Если быть более точным, в VBA существует режим (и он используется по умолчанию), при котором переменные разрешено не объявлять. Вообще-то

это удобно. Хотя, разумеется, не все так просто. Представим себе такую ситуацию. Мы ввели (объявляя или не объявляя — в данном случае неважно) в программном коде переменную. Затем через некоторое время мы ее используем, допустив при этом ошибку в имени переменной. Результат, скорее всего, будет плачевный, поскольку компилятор воспримет переменную с «неправильным» именем как еще одну, новую (и необъявленную) переменную. Такое положение дел само по себе уже не очень хорошее. Кроме того, код с объявлением переменных более читабельный. В силу этих обстоятельств имеет смысл перейти в режим обязательного объявления переменных. Для этого в редакторе кодов VBE выбираем команду **Tools ▷ Options**, в результате чего откроется диалоговое окно **Options**, представленное на рисунке 10.14.

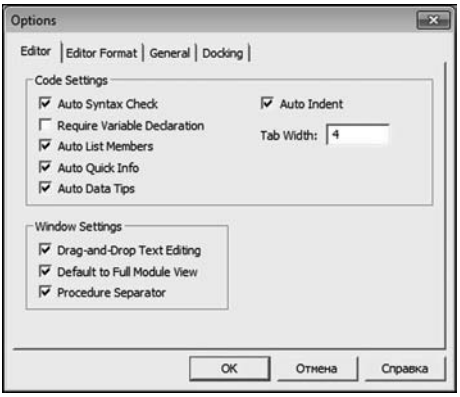


Рис. 10.14
Переключение между режимами обязательного/необязательного объявления переменных

На вкладке **Editor** этого окна есть опция **Require Variable Declaration**, установив для которой флажок, переведем редактор в режим обязательного объявления переменных. В этом режиме редактор нас предупредит, если мы попытаемся использовать в программном коде необъявленную переменную. Еще один способ перейти в данный режим — разместить в начале модуля с программным кодом инструкцию Option Explicit.

После того как переменная объявлена, с ней, как правило, что-то делают. Тут многое зависит от фантазии разработчика, но самая первая операция всегда банальна — это присваивание значения переменной. В качестве оператора присваивания в VBA используется знак равенства (т. е. =). В выражении вида *переменная = значение* выражение, указанное справа от оператора присваивания, присваивается в качестве значения переменной, указанной слева от оператора присваивания.

Что касается операций, которые могут выполняться с переменными, то многое зависит от их типа. Нас, в первую очередь, будут интересовать числовые форматы, текст, логические выражения и еще, пожалуй, объекты (которые мы оставим «на десерт»).

Основные математические операторы VBA перечислены в таблице 10.2.

На заметку

Для объединения текстовых значений помимо оператора + можно использовать оператор конкатенации &. Что касается знака «минус» -, то, кроме операций вычитания, он еще используется для обозначения отрицательных чисел.

Мы также будем использовать операторы сравнения, которые представлены в таблице 10.3. Операторы возвращают в качестве результата значения логических типов в зависимости от истинности соответствующего соотношения.

Таблица 10.2

Основные математические операторы

Оператор	Описание
+	Сложение: бинарный оператор, с помощью которого вычисляется сумма двух числовых операндов или выполняется конкатенация (объединение) текстовых строк
-	Вычитание: бинарный оператор, с помощью которого вычисляется разность двух операндов
*	Умножение: бинарный оператор, используемый для вычисления произведения двух числовых операндов
/	Деление: бинарный оператор для вычисления частного двух числовых операндов
^	Возведение в степень: бинарный оператор, с помощью которого результат вычисляется путем возведения первого числового операнда в степень, определяемую вторым числовым операндом
\	Целочисленное деление: бинарный оператор для вычисления целочисленного деления первого числового операнда на второй числовой операнд
Mod	Остаток от деления: бинарный оператор для вычисления остатка от деления целочисленного первого числового операнда на второй числовой операнд

Таблица 10.3

Операторы сравнения

Оператор	Описание	Оператор	Описание
<	Оператор «меньше»	>=	Оператор «больше или равно»
<=	Оператор «меньше или равно»	<>	Оператор «не равно»
>	Оператор «больше»	=	Оператор «равно»

На заметку

Обратите внимание: оператор сравнения «равно» формально совпадает с оператором присваивания. Какой из двух операторов имеется в виду, определяется в контексте команды, в которой использован данный оператор.

У представленных выше операторов различный приоритет, т. е. если в выражении несколько различных операторов, то порядок выполнения операций определяется с учетом приоритета операторов. Самый высокий приоритет имеет оператор возведения в степень, затем идут операторы умножения и деления, после них операторы сложения и вычитания и, наконец, операторы сравнения. Вообще же, если нет уверенности в порядке вычисления выражения — используйте круглые скобки. Они лишними не бывают!

Что касается ввода инструкций, то, хотя в строке кода в принципе может быть несколько инструкций, для повышения читабельности кода рекомендуется размещать в одной строке не более одной команды. Если все же мы хотим разместить несколько команд в строке, их следует разделять между собой двоеточиями (т. е. :). Когда команда не помещается в одну строку, ее (команду) можно разбить на несколько строк, использовав в качестве индикатора переноса символ подчеркивания (т. е. _).

На заметку

В целях экономии можно объединять в одну команду объявление нескольких переменных. Однако следует помнить, что инструкция As **тип** относится только к одной переменной. Например, командой Dim x, y As Double объявляются две переменных x и y, но только переменная y будет иметь тип Double. Переменная x по умолчанию интерпретируется как такая, что имеет тип Variant. Чтобы обе переменные имели тип Double, используем команду Dim x As Double, y As Double. Вполне «законной» будет, например, и такая команда: Dim a As Integer, b As String.

Далее рассмотрим небольшой пример, в котором, помимо прочего, объявляется несколько переменных. Обратимся к листингу 10.3.

Листинг 10.3. Процедура с объявлением переменных

```
Sub GetInfo()
  ‘ Размер шрифта ячейки
  Dim FSize As Integer
  ‘ Название шрифта
  Dim FName As String
  ‘ Ширина и высота ячейки
  Dim CWidth As Integer, CHeigh As Integer
  ‘ Соотношение сторон ячейки
  Dim CScale As Double
  ‘ Текущая дата
  Dim Today As Date
  ‘ Текст для отображения в окне
  Dim txt As String
  ‘ Обработка активной ячейки
  With ActiveCell
    ‘ Считывание названия шрифта
    FName = .Font.name
    ‘ Считывание размера шрифта
    FSize = .Font.size
    ‘ Считывание ширины ячейки
    CWidth = .Width
    ‘ Считывание высоты ячейки
    CHeight = .Height
    ‘ Вычисление соотношения сторон ячейки
    CScale = CWidth / CHeight
    ‘ Определение текущей даты
    Today = Date
    ‘ Формирование текста для отображения в окне
    txt = ”Сегодня “ & Today & “, и у активной ячейки такие параметры: “ _
    & vbLf & vbLf
    txt = txt & “Шрифт - ” & FName & vbLf
    txt = txt & “Размер - ” & FSize & vbLf
    txt = txt & “Ширина ячейки - ” & CWidth & vbLf
```

```
    txt = txt & "Высота ячейки - " & CHeight & vbLf
    txt = txt & "Соотношение сторон - " & CScale
  End With
  ' Звуковое приветствие
  Application.Speech.Speak("Hello, glad to see you!")
  ' Отображение окна с сообщением
  MsgBox txt, vbInformation, "Информация об активной ячейке"
  ' Завершающая озвучка
  Application.Speech.Speak("Good bye!")
End Sub
```

Это код макроса, который называется GetInfo, и в идеале при его выполнении происходят следующие события (предполагается, что при запуске макроса в рабочем листе выделена ячейка). Сначала на английском языке идет звуковое приветствие. Затем появляется диалоговое окно, в котором содержится информация об основных параметрах форматирования ячейки (тип и размер шрифта, а также ширина и высота ячейки). После закрытия окна можно будет услышать еще одно звуковое напутствие (тоже на английском языке).

Начальный блок программного кода содержит объявление нескольких переменных разного типа. В эти переменные будут записываться значения всевозможных данных. Здесь есть переменные типа Integer, Double, String и Date.

Основу «вычислительной» части кода составляет оператор With. Оператор используется для обработки объектов. В данном случае «обрабатывается» активная ячейка, ссылка на которую выполняется как ActiveCell. Заканчивается оператор инструкцией End With. Все инструкции вида *.ссылка*, которые находятся внутри блока оператора и начинаются с точки, эквивалентны инструкциям ActiveCell.*ссылка*. Например, команда .Font.Name в теле оператора With ActiveCell означает на самом деле ссылку ActiveCell.Font.Name (название шрифта активной ячейки).

На заметку

Активная ячейка — это ActiveCell. Шрифт отображения данных в активной ячейке — это ActiveCell.Font. Название шрифта активной ячейки — это ActiveCell.Font.Name. Аналогично «расшифровываются» ссылки ActiveCell.Font.Size (размер шрифта активной ячейки), ActiveCell.Width (ширина активной ячейки) и ActiveCell.Height (высота активной ячейки).

Идея очень простая: соответствующий параметр активной ячейки «считывается» и записывается в соответствующую переменную. Затем эти переменные используются при формировании текста, отображаемого в диалоговом окне. Хотя, если быть до конца откровенным, необходимости в этом нет — можно было при формировании текста сразу использовать ссылки на свойства активной ячейки.

На заметку

Ширина и высота ячейки определяются в пикселях — это целые числа. Соответствующие переменные имеют тип Integer. Отношение сторон в общем случае есть число действительное, поэтому переменная для записи этого соотношения имеет тип Double.

Текущая дата определяется с помощью встроенной функции Date. Результат записывается в переменную Today (которая имеет тип Date). При формировании текста для диалогового окна (переменная txt) используется оператор конкатенации & и встроенная константа vbLf, которая является инструкцией начала новой строки. Формирование значения переменной txt проходит в несколько этапов путем добавления к текущему значению переменной добавочного текстового «хвостика».

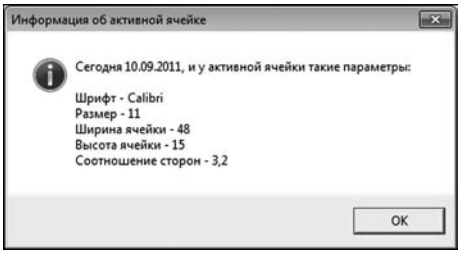


Рис. 10.15
Диалоговое окно отображается при выполнении макроса

Помимо оператора With в программном коде есть еще несколько команд. Например, командой Application.Speech. Speak("Hello, glad to see you!") *озвучивается* фраза Hello, glad to see you. Аналогично, командой Application. Speech. Speak("Good bye!") озвучивается фраза Good bye. Здесь мы использовали функцию Speak, аргументом которой передается озвучиваемый текст. Между этими командами находится инструкция отображения диалогового окна. Функции MsgBox передается три аргумента: текст непосредственно сообщения, константа vbInformation, определяющая тип диалогового окна (окно с информационной пиктограммой), и текст для строки названия диалогового окна.

Чтобы проверить работу макроса, выделяем в рабочем документе ячейку и запускаем макрос. Если все нормально, то сначала услышим голос, затем увидим окно, примерно такое, как на рисунке 10.15 и, после закрытия окна, снова услышим голос «за кадром».

Выше мы, кроме всего прочего, познакомились с оператором With. Несложно заметить, что оператор немного упрощает работу по организации кода. Есть и другие полезные операторы, с которыми также имеет смысл познакомиться.

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ ИНСТРУКЦИИ

Вот, значит, Вам задание.
Как только начнут топить корабль,
сразу же садитесь вот в эту лодку
и энергично гребите к месту происшествия.
Понятно? Выбросьте спасательный круг
и так с выражением скажете двустишие:
«Вы панике не поддавайтесь — организованно спасайтесь!»

Из к/ф «Старый знакомый»

С управляющими инструкциями и операторами в этом разделе мы познакомимся достаточно кратко — в объеме, минимально необходимом для понимания несложных программных кодов. Более основательное знакомство, основанное на практическом использовании, состоится в следующей части книги, которая посвящена решению вычислительных задач средствами Excel.

Очень часто при написании программ полезным (а иногда и просто незаменимым) оказывается условный оператор If. Этот оператор позволяет выполнять разные действия в зависимости от того, выполняется или нет некоторое условие. Пример простого макроса, в котором есть условный оператор (даже два!), приведен в листинге 10.4.

Листинг 10.4. Макрос с условным оператором

```
Sub WhoAreYou()
 ‘ Имя пользователя
 Dim name As String
 ‘ Название для окна
 Dim title As String
 title = ”Служба знакомств”
 ‘ Будем знакомиться?
 Result = MsgBox(“Хотите познакомиться?”, vbYesNo, title)
 ‘ Проверка результата
 If Result = vbYes Then
  name = InputBox(“Как Вас зовут?”, title)
 ‘ Если не указано имя, завершаем макрос
 If name = ”” Then End
  ‘ Если имя указано, отображаем приветствие
  MsgBox “Добрый день, “ & name & “! Приятно познакомиться!”, _
  vbOKOnly, title
 Else
  ‘ Знакомство не состоялось
  MsgBox “Очень жаль! Пока!”, vbOKOnly, title
 End If
End Sub
```

С этим макросом все просто — он предлагает пользователю познакомиться. Для этого отображается диалоговое окно с соответствующим вопросом. Окно отображается функцией MsgBox. Аргумент vbYesNo означает, что у окна две кнопки — **Да** и **Нет** (рис. 10.16).

На заметку

Обратите внимание, что в данном случае у вызова функции есть результат, который запоминается в переменную Result. Переменная предварительно в программном коде не объявлялась.

Для продолжения работы макроса важно, на какой кнопке щелкнул пользователь. Собственно, для проверки выбора пользователя и используем условный оператор. В условном операторе проверяется значение переменной Result. Если пользователь щелкнул на кнопке **Да**, переменная получит значение vbYes. Если пользователь щелкнул на кнопке **Нет**, переменная

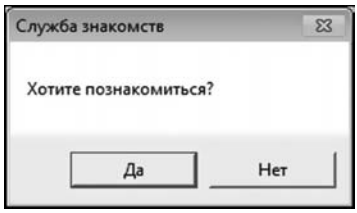


Рис. 10.16
Приглашение к знакомству

получит значение vbNo. Мы проверяем условие Result = vbYes. Если условие выполнено, то управление передается командам, размещенным после ключевого слова Then. У нас это команда name = InputBox("Как Вас зовут?", title). Функцией InputBox отображается окно с полем ввода (рис. 10.17).

Первый аргумент функции определяет текст в области окна (над полем ввода). Заголовок окна определяется аргументом title. В поле ввода пользователь вводит, например, имя и щелкает кнопку **OK**. В результате содержимое поля ввода в текстовом формате будет присвоено переменной name. Окно при этом закрывается.

На заметку

Если пользователь щелкнет на кнопке **Cancel**, окно будет закрыто, а переменная name получит в качестве значения пустую текстовую строку. Этот случай также обрабатывается в программе. После считывания значения переменной name во внутреннем условном операторе проверяется условие name = "". Если это условие выполнено, то работа макроса завершается — для этого после ключевого слова Then (это ключевое слово относится к внутреннему условному оператору) размещена инструкция End. Инструкция завершает работу макроса (в нормальном режиме).

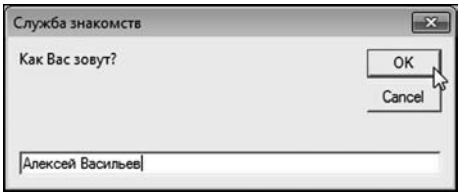


Рис. 10.17
Для знакомства вводим имя пользователя в поле ввода

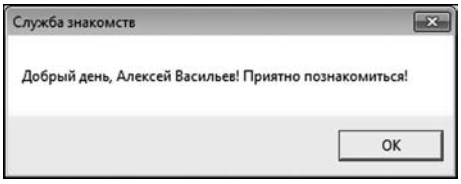


Рис. 10.18
Знакомство состоялось

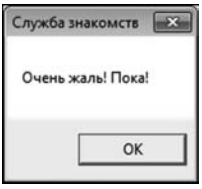


Рис. 10.19
Знакомство не состоялось

После того как значение переменной name считано из поля ввода, отображается диалоговое окно с приветствием, в котором использовано и значение переменной name (рис. 10.18).

На этом работа внешнего условного оператора (а вместе с ним и всего макроса) будет завершена — блок команд после ключевого слова Else не выполняется. Напомним, все это происходит, если было выполнено условие Result = vbYes. Если же условие не выполнено, то не выполняются команды после ключевого слова Then, но выполняются команды после ключевого слова Else. В нашем макросе эта ветка кода предназначена для отображения окна, представленного на рисунке 10.19.

Таким образом, условный оператор позволяет проверить условие и выполнить или блок команд после ключевого слова Then (если условие истинно), или блок команд после ключевого слова Else (если условие ложно). Можно использовать и сокращенные формы условных операторов без Else-блока.

Идеологически близок к условному оператору If оператор выбора Select. Он

позволяет выполнять множественную проверку значения выражения с выполнением разных блоков программного кода. Обратимся к примеру — программный код макроса с оператором выбора представлен в листинге 10.5.

Листинг 10.5. Макрос с оператором выбора

```
Sub SetColor()
 ' Переменная для обозначения ячейки
 Dim Cell As Range
 ' Перебор ячеек в выделенном диапазоне
 For Each Cell In Selection
  With Cell
   ' Оператор выбора
   Select Case .Value
   Case 0
    .Interior.Color = vbYellow
     .Font.Bold = True
     .Font.Color = vbBlue
   Case Is > 0
    .Interior.Color = vbGreen
     .Font.Italic = True
     .Font.Color = vbWhite
   Case Is < 0
    .Interior.Color = vbRed
     .Font.Bold = True
     .Font.Italic = True
     .Font.Color = vbBlack
   End Select
  End With
 Next Cell
End Sub
```

Макрос работает так: в выделенном диапазоне перебираются все ячейки, и в зависимости от значения (а предполагается, что значения все числовые) ячейка заливается каким-то цветом, а к шрифту применяется несложное форматирование. Выделяем три случая:

- ячейка содержит нулевое значение (или пустая). В этом случае ячейка заливается желтым цветом, а также применяется жирный шрифт синего цвета;
- ячейка содержит положительное значение. В этом случае цвет заливки зеленый, шрифт белый курсивный;
- ячейка содержит отрицательное значение. В этом случае цвет заливки красный, цвет шрифта черный, а шрифт жирный курсивный.

В макросе объявляется переменная Cell типа Range — эту переменную мы будем использовать в качестве ссылки на ячейки выделенного диапазона. Выделенный диапазон обозначается как Selection. Для перебора ячеек, которые входят в этот диапазон, используем оператор цикла For. Инструкция

For Each Cell In Selection означает, что переменная Cell будет последовательно «перебирать» все ячейки диапазона Selection.

На заметку

Заканчивается оператор цикла инструкцией Next Cell. Более того, использованный выше формат для оператора цикла далеко не единственный. Нередко оператор цикла используют с индексной переменной, которая пробегает определенный набор значений. Например, если нужно, чтобы переменная n пробегала значения, скажем, от 1 до 100 (с шагом 1), можем воспользоваться оператором цикла, который начинается инструкцией For n = 1 To 100, а заканчивается инструкцией Next n.

В операторе цикла выполняется обработка ячейки на предмет применения параметров форматирования. Поскольку форматирование применяется в зависимости от значения ячейки, срезу же вступает в дело оператор выбора. После ключевой инструкции Select Case указывается ссылка .Value на значение ячейки. Случаю, когда в ячейке нулевое значение, соответствует блок команд, обозначенный инструкцией Case 0. Соответственно, для случая положительных значений предназначен блок команд с инструкцией Case Is>0, а для отрицательных — блок команд с инструкцией Case Is<0. Заканчивается блок оператора выбора инструкцией End Select.

На заметку

Свойство Cell.Interior.Color позволяет задать цвет заливки ячейки, цвет шрифта задается с помощью свойства Cell.Font.Color, жирный/нежирный или курсивный/прямой шрифт задаем соответственно через свойства Cell.Font.Bold и

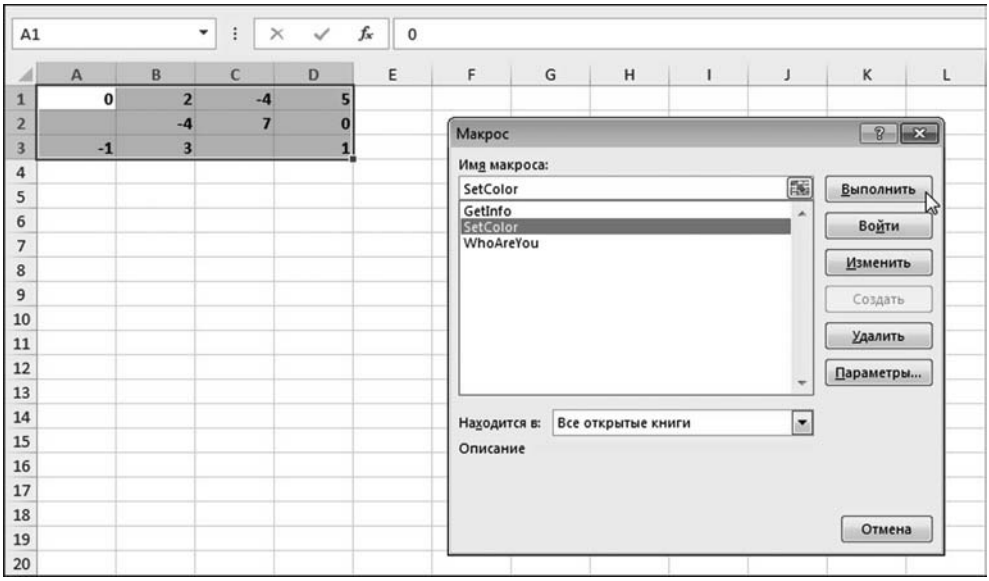


Рис. 10.20
Документ перед запуском макроса на выполнение

Cell.Font.Italic (логические значения True или False). Константы vbRed, vbGreen, vbWhite, vbBlack, vbBlue, vbYellow обозначают цвета (красный, зеленый, белый, черный, синий, желтый).

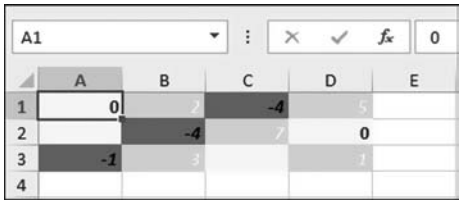


Рис. 10.21
Диапазон ячеек с числами после выполнения макроса

Для проверки работы макроса в рабочем документе выделяем диапазон ячеек с разными числовыми значениями и запускаем макрос на выполнение (рис. 10.20).

После выполнения макроса диапазон ячеек становится красочным, как новогодняя елка (рис. 10.21).

Для удобства восприятия выделение с диапазона снято. Правда, в черно-белом цвете картинка не такая эффектная, но, в конце концов, можно прибегнуть к силе воображения. Что касается нашего знакомства с премудростями программирования в VBA, мы продолжим его в следующей главе.

ГЛАВА 11

СОЗДАНИЕ ФОРМ И ОБРАБОТКА СОБЫТИЙ

То есть по натуре, по хворме, ну, по телу — как водится!
Но по уму, по образованности мы уже не та хворма,
не тот центр тяжести.

Из к/ф «За двумя зайцами»

До этого мы с успехом использовали диалоговые окна. Правда, приходилось довольствоваться достаточно скромным набором возможностей по настройке вида диалогового окна. К счастью, в Excel, а точнее, в VBA пользователю предоставлены широчайшие возможности по созданию собственных графических окон, которые принято называть *формами*. С этими самыми формами мы познакомимся в данной главе. Мы узнаем, как их создают и что с ними делают. Кроме того, мы продолжим знакомство с некоторыми синтаксическими конструкциями, которые весьма полезны и которые мы обошли вниманием в предыдущих главах.

Работа с формами не ограничивается их созданием (хотя и это довольно увлекательный процесс). Обычно форма содержит всякие функциональные элементы — кнопки, переключатели, поля и т. д. И если разместить кнопку на форме, в принципе, не сложно, то «научить» ее «правильному поведению» при определенных действиях пользователя намного сложнее. Тем не менее мы справимся и с этой задачей.

На заметку

«Правильное поведение» — это реакция формы или ее элемента на события. Понятие «событие» имеет вполне конкретный смысл в программировании. Для нас важно понять общую идею, и состоит она в следующем. Допустим, есть некоторое окно, а у окна есть какие-то элементы — скажем, кнопка или поле ввода. С этими элементами управления пользователь может выполнять или не выполнять некоторые действия — например, щелкнуть на кнопке или ввести текст в поле ввода, изменить состояние переключателя и пр. Все это относится к событиям. Если элемент должен реагировать как-то на событие, необходимо прописать программный код для обработчика этого события для данного элемента. Например, если мы хотим, чтобы при щелчке кнопки выполнялись определенные команды, эти команды должны быть включены в обработчик события щелчка кнопки. Обработчик события для элемента — это процедура со специальным названием. Она вызывается автоматически, если с элементом произошло соответствующее событие. Как это все реализуется на практике, мы рассмотрим немного позже на конкретных примерах.

ЗНАКОМСТВО С ФОРМАМИ

Вот, примером, бруки. Как труба стоять.
Не так поставь, и шо? Физиономии вже нету.
Или вот желётка. Тоже хитрая штука.
Не тот цвет — и вже проиграл, симпатии нету.
Ну, не тот парад на лице.
Пинжак — это главная хворма.

Из к/ф «За двумя зайцами»

Знакомство с формами начнем с того, что создадим достаточно простую форму с минимальным набором функциональных элементов. Для этого в редакторе кодов VBE выбираем команду **Insert ▷ UserForm**, как показано на рисунке 11.1.

В результате в проект добавляется новая форма — это серого цвета окно с названием по умолчанию **UserForm1**, которое находится в режиме редактирования, так что для него можно выполнить самые разнообразные настройки (от изменения внешнего вида и до добавления программных кодов обработчиков событий). Окно редактора VBE с проектом, в который добавлена новая форма, показано на рисунке 11.2.

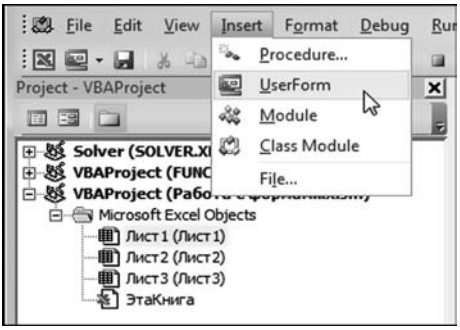


Рис. 11.1
Вставка формы в проект

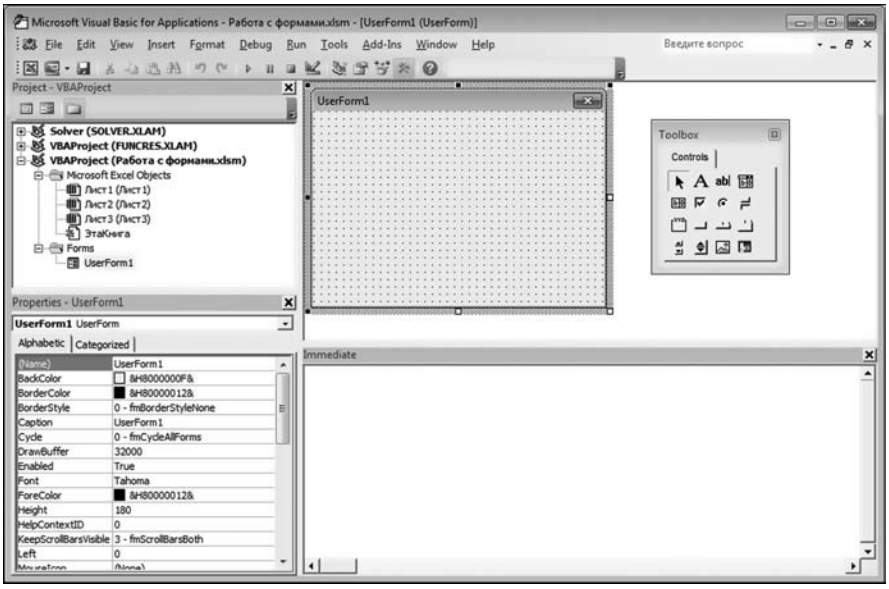


Рис. 11.2
Новая форма добавлена и готова к редактированию

На заметку

При добавлении формы в проект в окне проекта **Project** отображается узел (папка) форм **Forms**, внутри которого собраны все имеющиеся в проекте формы — их, кстати, может быть несколько. Чтобы перейти к форме, необходимо выполнить двойной щелчок мышью на пиктограмме этой формы в окне проекта. В данном случае щелкать нужно на пиктограмме **UserForm1**. Если все сделано правильно, справа, в окне, где обычно мы наблюдали программный код, должна появиться форма. При выделенной форме в окне свойств **Properties** отображаются свойства этой формы. Настройки формы можно выполнять прямо в этом окне. В первую очередь, речь идет о внешнем виде формы. Определять значения свойств формы можно также программными методами, т. е. с помощью команд в программном коде.

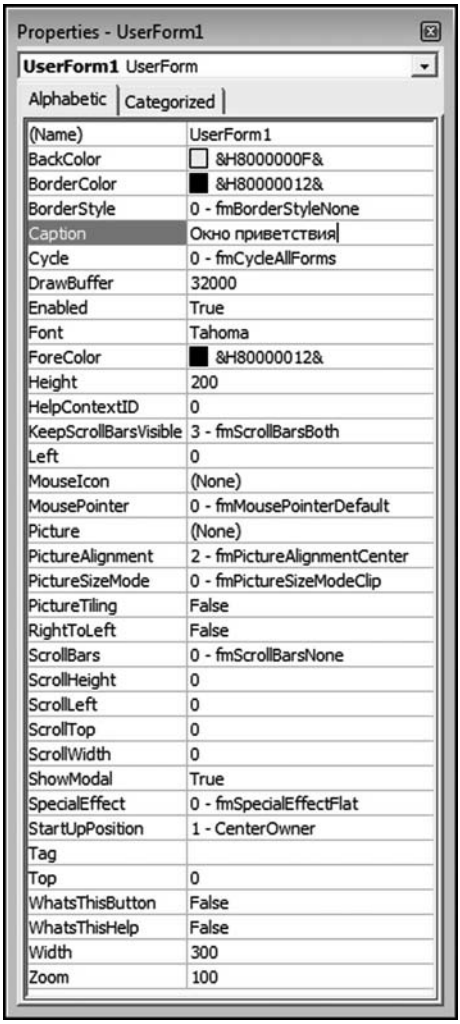


Рис. 11.3
Настройка параметров формы в окне свойств

Окно **Toolbox**, которое можем видеть на рисунке 11.2 рядом с формой, предназначено для размещения на форме различных элементов управления (кнопок, переключателей, меток, полей и т. д.). Если окно не отображается, следует воспользоваться командой **View ▷ Toolbox**.

Итак, форма есть, осталось ее теперь настроить. Некоторые простые настройки, такие, например, как размеры окна формы, выполняются без особых усилий. Скажем, установить размер окна можно простым перетаскиванием рамки формы, как мы это делаем с обычными окнами. А можно воспользоваться окном свойств формы. Например, в окне свойств **Properties** находим поле **Height** и устанавливаем там значение 200. В поле **Width** устанавливаем значение 300. Это соответственно высота и ширина окна формы. Затем, в поле **Caption** вводим текст Окно приветствия, как показано на рисунке 11.3.

Каждое из полей соответствует свойству формы. Свойство **Caption** — это текст полосы заголовка окна формы.

На заметку

Помимо свойства **Caption** у формы есть еще свойство **Name**. Это название объекта формы. Без крайней необходимости значение этого свойства лучше не менять.

Что касается окна свойств **Properties**, то еще раз обращаем внимание читателя, что в этом окне отображаются свойства выделенного в окне проекта объекта. Если по каким-то причинам окно свойств не отображается вообще, следует воспользоваться командой **View ▷ Properties Window** или нажать клавишу <F4>.

После внесения изменений в настройки окно формы будет выглядеть так, как показано на рисунке 11.4.

Правда, практически весь эффект от внесенных изменений свелся к изменению названия окна (обратите внимание на текст в строке названия окна формы). Но это как раз тот случай, когда важен сам процесс, а не результат.

В принципе, уже на этом этапе мы можем проверить нашу форму в деле. Для этого создадим макрос, который будет отображать окно формы. Код макроса впишем в специальный модуль, который добавим в проект с помощью команды **Insert ▷ Module**. Процесс вставки модуля в проект проиллюстрирован на рисунке 11.5.

После того как в проект вставлен модуль, выделяем этот модуль двойным щелчком и в открывшемся окне кода модуля вводим программный код макроса. На рисунке 11.6 пред-

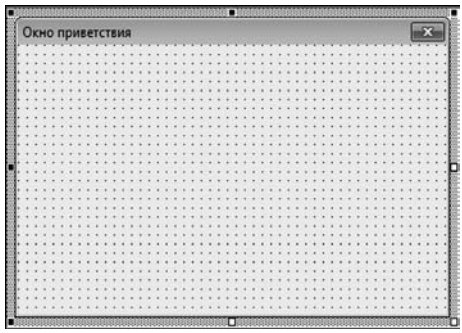


Рис. 11.4
Окно формы после внесения изменений в настройки

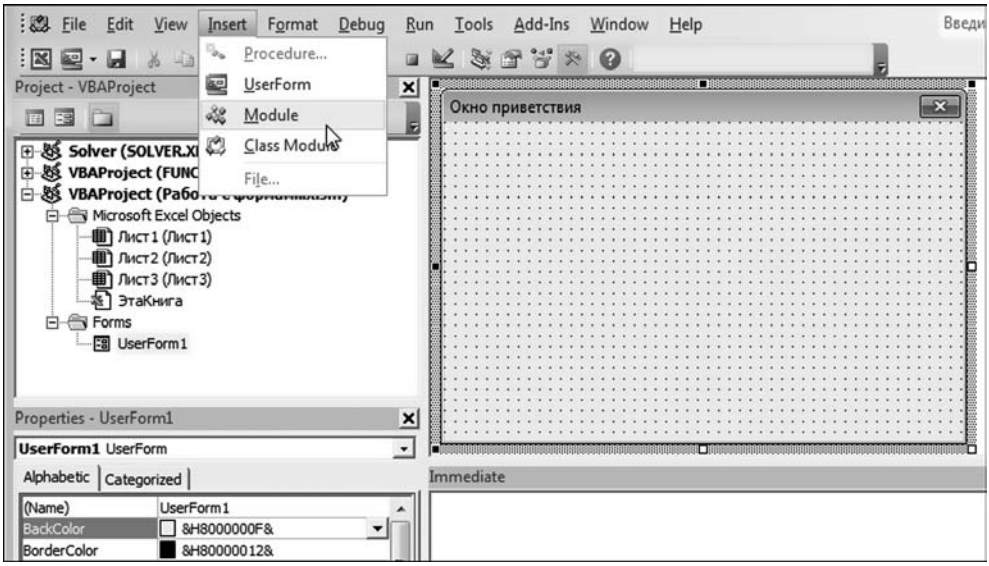


Рис. 11.5
Вставка модуля в проект

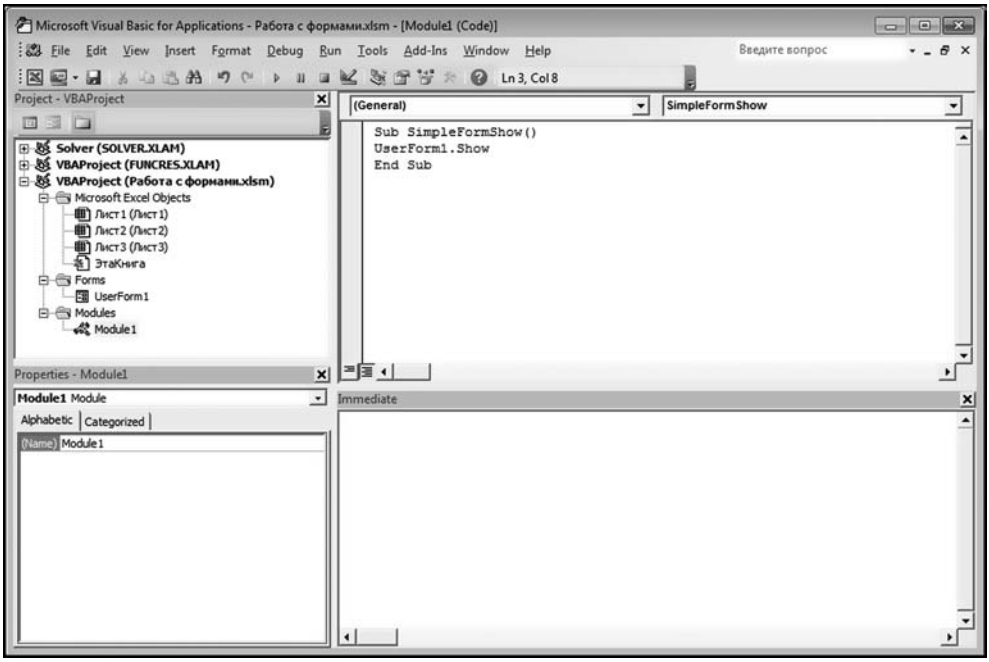


Рис. 11.6
В модуле проекта размещен программный код макроса для отображения формы

ставлено окно редактора VBE с модулем и программным кодом макроса SimpleFormShow.

Код макроса SimpleFormShow приведен в листинге 11.1.

Листинг 11.1. Макрос для отображения окна формы

```
Sub SimpleFormShow()
 UserForm1.Show
End Sub
```

У макроса очень простой код: не считая обязательных инструкций начала/окончания кода макроса, состоит всего из одной команды UserForm1.Show, которой, собственно, и отображается форма. В частности, из объекта формы UserForm1 вызывается метод отображения формы Show.

Наш макрос готов к работе. После запуска макроса на выполнение появится окно, которое, откровенно говоря, диалоговым назвать сложно. Тем не менее это окно перспективное (в том смысле, что мы с ним еще будем работать). Поэтому все же имеет смысл на окно взглянуть — оно показано на рисунке 11.7.

Пользы от этого окна пока что никакой. Поэтому убираем окно с экрана, щелкнув на системной пиктограмме с крестиком (стандартная в Windows пиктограмма для закрытия окна) в правом верхнем углу окна. Займемся усовершенствованием нашего проекта.

Первая задача, которую мы решим — сделаем окно более информативным. Другими словами, мы хотим, чтобы в области окна отображался какой-то текст. Реализовать эту идею мы сможем, если добавим в область окна формы текстовую метку. Последовательность действий для достижения положительного результата вкратце такая:

- добавляем в окно формы текстовую метку;
- выполняем настройки параметров метки;
- для проверки результата запускаем макрос, которым отображается окно формы (это наш старый знакомый — макрос SimpleFormShow).

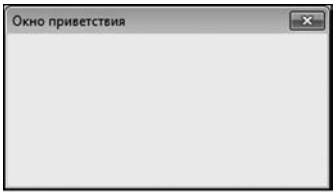


Рис. 11.7
Это окно отображается в результате выполнения макроса

Теперь рассмотрим более подробно два первых этапа. Итак, для добавления метки в форму в окне **Toolbox** щелкаем пиктограмму с большой литерой **A**, как показано на рисунке 11.8.

Рис. 11.8
В окне **Toolbox** выбираем пиктограмму для вставки метки в окно формы

На заметку

При наведении курсора мыши на пиктограммы в окне **Toolbox** отображается контекстная подсказка. Эта подсказка помогает определить элемент, для вставки которого предназначена пиктограмма. Метка является элементом **Label**. Перечень пиктограмм диалогового окна **Toolbox** представлен в таблице 11.1.

Методы работы с некоторыми из этих объектов обсуждаются далее.

После этого необходимо выбрать место для размещения метки в окне формы. Соответствующая область в окне формы захватывается с помощью мыши, как показано на рисунке 11.9.

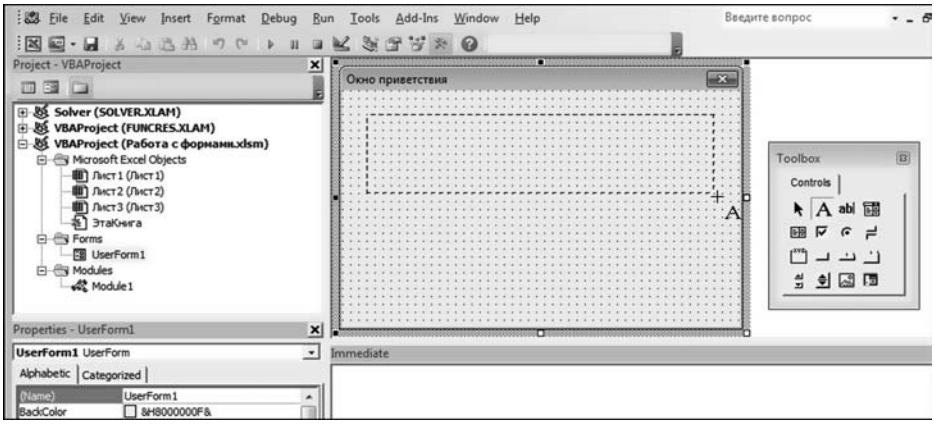


Рис. 11.9
Вставка метки в форму

Таблица 11.1

Пиктограммы диалогового окна Toolbox

Пиктограмма	Объект	Пояснения
	Выбор объекта	Пиктограмма предназначена для перехода в режим выбора объекта
	Label	Текстовая метка. Основное предназначение — отображать текст
	TextBox	Текстовое поле. Этот элемент предназначен для хранения и отображения текста. В отличие от метки в текстовое поле текст можно вводить с клавиатуры
	ComboBox	Раскрывающийся список (с редактируемым текстовым полем). Позволяет выбирать элементы из раскрывающегося списка и вводить значения в текстовое поле
	ListBox	Раскрывающийся список (с недоступным для редактирования полем выбора). Позволяет выбирать в раскрывающемся списке элементы
	CheckBox	Опция. Этот элемент представляет собой классическую опцию: небольшую квадратную область для установки флажка (галочки) и связанную текстовую метку
	OptionButton	Логический переключатель. Элемент представляет собой круглую область для установки положения переключателя со связанной текстовой меткой
	ToggleButton	Кнопка-переключатель. От обычной кнопки отличается тем, что может быть в двух состояниях — нажатом и отжатом
	Frame	Рамка. Этот элемент управления применяется для группировки прочих элементов. Рамку также используют для создания группы переключателей
	CommandButton	Кнопка. Элемент для размещения в окне формы классической кнопки
	TabStrip	Строка корешков вкладок. Используется для создания вкладок
	MultiPage	Элемент для создания страниц с несколькими вкладками. В отличие от элемента TabStrip, элемент MultiPage является контейнером для вкладок
	ScrollBar	Полоса прокрутки. У полосы прокрутки есть ползунок, который можно перетаскивать мышью
	SpinButton	Числовой переключатель. Представляет собой совмещенные пиктограммы с разнонаправленными стрелками. Щелчок на стрелках приводит к увеличению/уменьшению числового значения счетчика. Смежного поля для отображения значения счетчика у элемента нет
	Image	Элемент для вставки графического изображения в окно формы
	RefEdit	Поле ввода адресов ячеек. Позволяет вводить адреса ячеек и диапазонов путем выбора ячеек непосредственно в рабочем листе

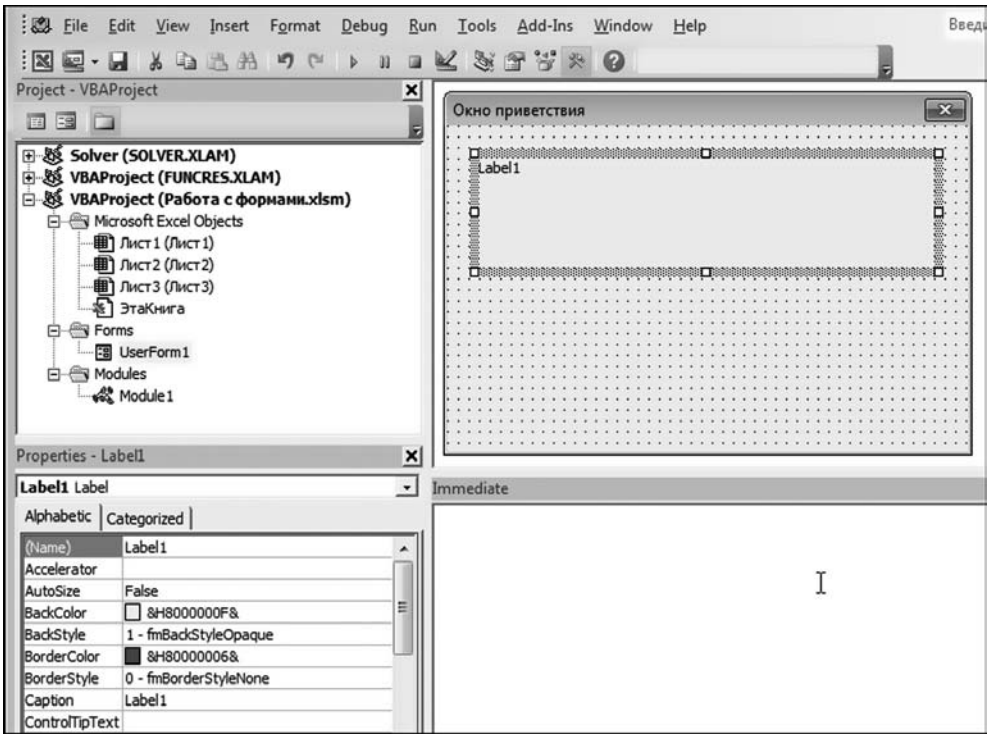


Рис. 11.10
В форму вставлена текстовая метка

Отпускаем кнопку мыши, что означает завершение процесса вставки метки в форму. На рисунке 11.10 показано, как может выглядеть окно формы с только что вставленной в него меткой.

Основное назначение метки — отображать текст. По умолчанию текст, который отображается меткой — это ее название (т. е. название объекта метки). По умолчанию первая созданная метка имеет название **Label1**, что мы собственно и видим в области метки. Далее переходим ко второму этапу — редактированию метки и настройке ее параметров. Все эти операции выполним в окне свойств метки, которое показано на рисунке 11.11.

На заметку

Окно свойств одно, а объектов в проекте уже достаточно много. Чтобы в окне свойств отображались свойства именно метки, необходимо в окне формы выделить метку. Можно пойти и другим путем — выделить форму, а затем воспользоваться раскрывающимся списком в верхней части окна свойств и выбрать в этом списке объект метки.

В окне свойств для метки вносим следующие изменения. Во-первых, меняем текст метки. В поле значения свойства **Caption** вводим текст Всем большой привет!. Во-вторых, изменяем шрифт, используемый для отображения текста метки. Для этого в поле значения свойства **Font** щелкаем пиктограмму

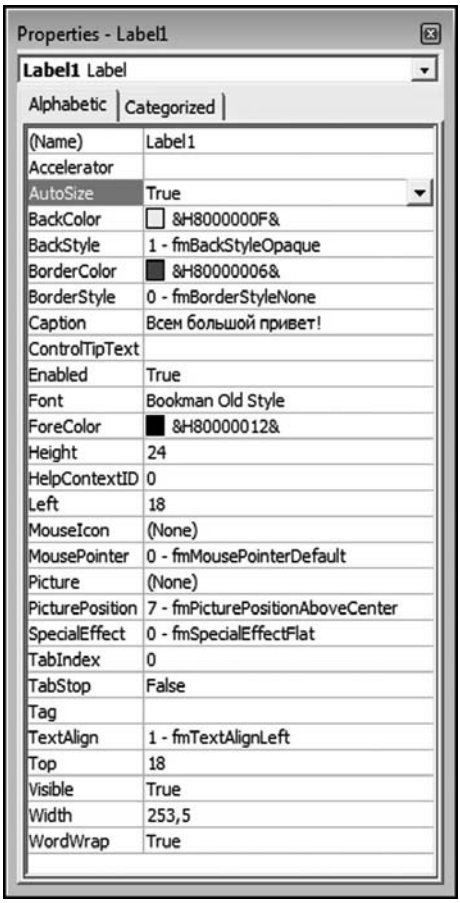


Рис. 11.11
Окно свойств для настройки параметров текстовой метки

с тремя точками в правой части поля. Откроется стандартное окно выбора шрифта. Мы выбираем шрифт Bookman Old Style жирный, размера 20. Еще устанавливаем значение True для свойства **AutoSize**. Мы делаем так, чтобы размер области метки автоматически масштабировался по размеру текста. Это не необходимо, но достаточно удобно. Как будет выглядеть окно формы с меткой после выполнения указанных настроек, показано на рисунке 11.12.

На третьем этапе для проверки результатов наших усилий запускаем макрос на выполнение. В результате увидим окно, как на рисунке 11.13.

Продолжаем эксперименты с формой. Добавим в форму кнопку, чтобы окно можно было закрыть не только щелчком на системной пиктограмме, но и щелчком кнопки. Для этого добавляем кнопку в форму, а затем напишем код для обработки события щелчка на кнопке.

Чтобы добавить кнопку в окно формы, в окне **Toolbox** выбираем пиктограмму для вставки кнопки (объект **CommandButton**), как показано на рисунке 11.14.

Затем выбираем место для размещения кнопки в окне формы (выделяем в области формы место для вставки кнопки). На рисунке 11.15 показано размещение кнопки по центру в нижней части окна формы.

После того как кнопка в форму добавлена, окно формы будет иметь вид, как показано на рисунке 11.16.

По умолчанию кнопка имеет название CommandButton1. Такой же текст отображается в области кнопки. Мы переходим к окну свойств кнопки и устанавливаем шрифт Arial полужирный, размера 12 (свойство **Font** кнопки). Текст кнопки (свойство **Caption**) устанавливаем Закрыть окно. Окно свойств кнопки с выполненными настройками показано на рисунке 11.17.

На заметку

Легко заметить, что схожие свойства разных объектов имеют одинаковые названия. Например, свойство, отвечающее за используемый шрифт, традиционно называется **Font**. Свойство **Caption** определяет заголовок элемента,

и обычно это текст, который отображается в области элемента. Высота и ширина соответственно определяются свойствами **Height** и **Width**, и т. д. Это удобно, поскольку позволяет достаточно легко ориентироваться в списке свойств объекта.

На этом настройка внешнего вида кнопки закончена, пора приступить к более серьезным задачам. А именно, нам необходимо написать программный код, который будет выполняться при щелчке на кнопке. Но важно еще знать и куда его записать. Ответ на второй вопрос получим, если выполним двойной щелчок на кнопке в форме. Автоматически будет создан код процедуры-обработчика (без команд), и откроется соответствующий модуль для ввода команд, как показано на рисунке 11.18.

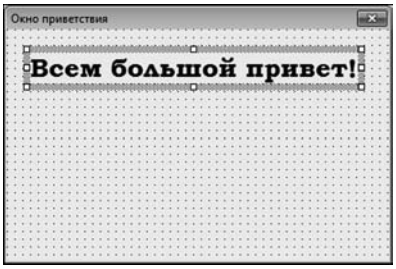


Рис. 11.12
Окно формы с текстовой меткой после выполнения настроек

На заметку

Название процедуры получается объединением имени объекта CommandButton1 и, через символ подчеркивания, ключевого слова Click (что означает «щелчок»). Ключевое слово Private означает, что процедура доступна только для внутреннего вызова — вызвать эту процедуру как макрос из рабочего документа Excel не получится.

В теле процедуры необходимо ввести те команды, которые предлагаются для выполнения в случае щелчка на кнопке. Мы вводим всего одну команду Unload UserForm1. Смысл команды простой — выгрузить из памяти форму, что означает закрытие формы. На рисунке 11.19 показано окно редактора VBE с кодом процедуры-обработчика события щелчка на кнопке.

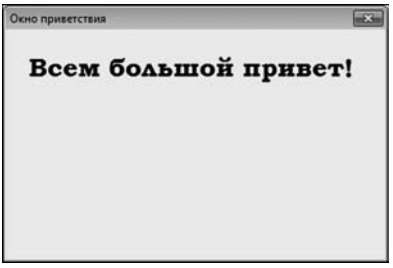


Рис. 11.13
После добавления текстовой метки в форму окно, которое отображается при запуске макроса, будет иметь такой вид

Рис. 11.14
В окне **Toolbox** выбираем пиктограмму для вставки кнопки

Рис. 11.15
Добавление кнопки в форму

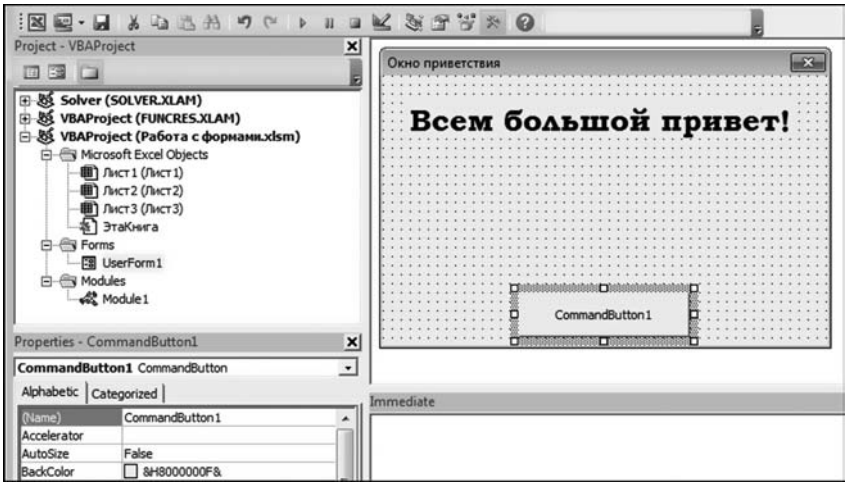


Рис. 11.16
Кнопка добавлена в форму

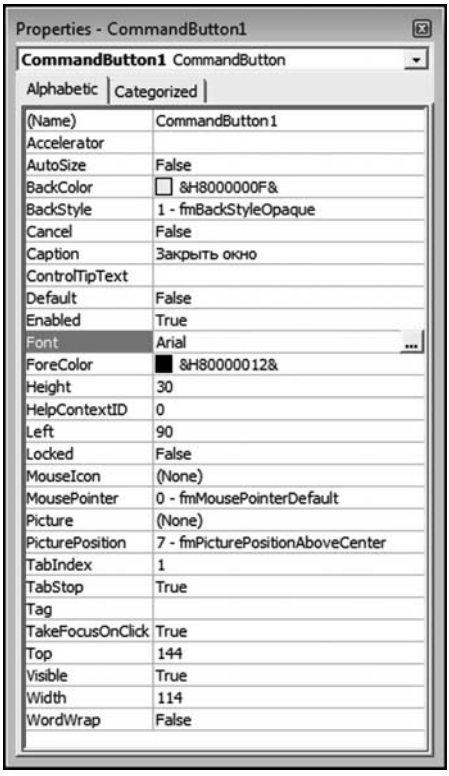


Рис. 11.17
Окно свойств для настройки параметров кнопки

Полностью код процедуры-обработчика приведен в листинге 11.2.

Листинг 11.2. Процедура обработки щелчка на кнопке

```
Private Sub CommandButton1_Click()
  Unload UserForm1
End Sub
```

После запуска макроса увидим окно, показанное на рисунке 11.20.

Теперь в нижней центральной части окна есть кнопка с названием **Закрыть окно**, щелчок на которой приводит к закрытию окна (и завершению работы макроса).

Все, что мы сделали — замечательно, но возникает естественный вопрос: какое это отношение имеет к Excel? Поэтому поставим «жирную точку» — свяжем диалоговое окно с рабочим листом приложения Excel. А именно, будем отображать в диалоговом окне адрес ячейки, которая активна на момент запуска макроса на выполнение. Для этого в окно формы добавляем еще одну метку с начальным текстом На данный момент активна ячейка (см. рис. 11.21).

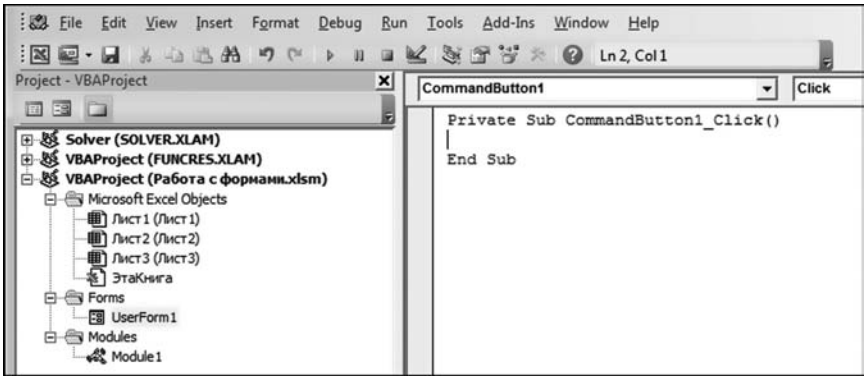


Рис. 11.18
Ввод программного кода для обработчика события щелчка на кнопке

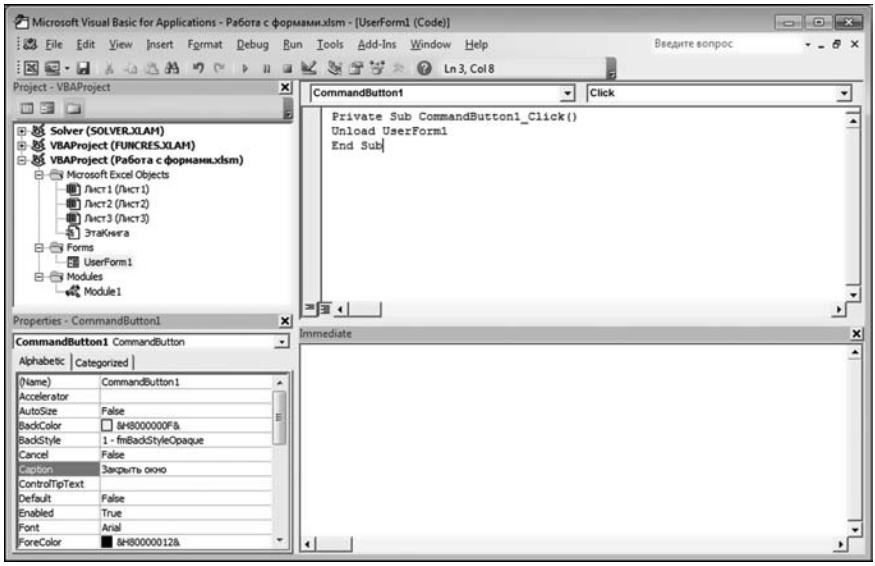


Рис. 11.19
Окно редактора VBE с программным кодом обработчика щелчка на кнопке

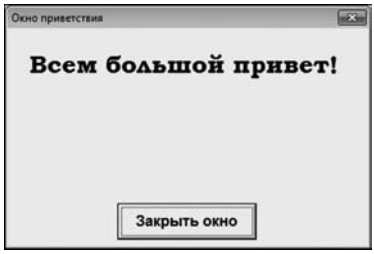


Рис. 11.20
Теперь при выполнении макроса отображается окно с кнопкой

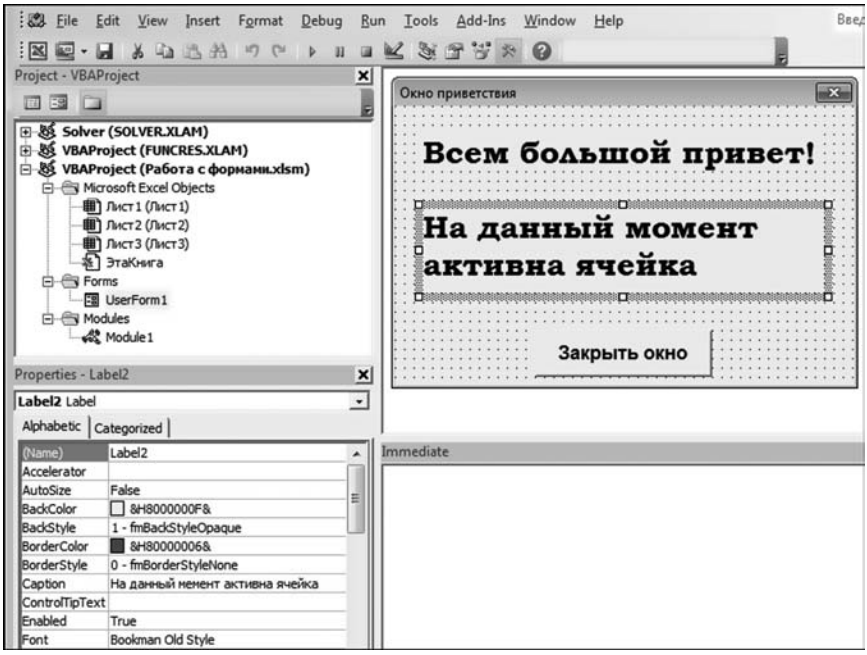


Рис. 11.21
В окно формы добавлена еще одна текстовая метка

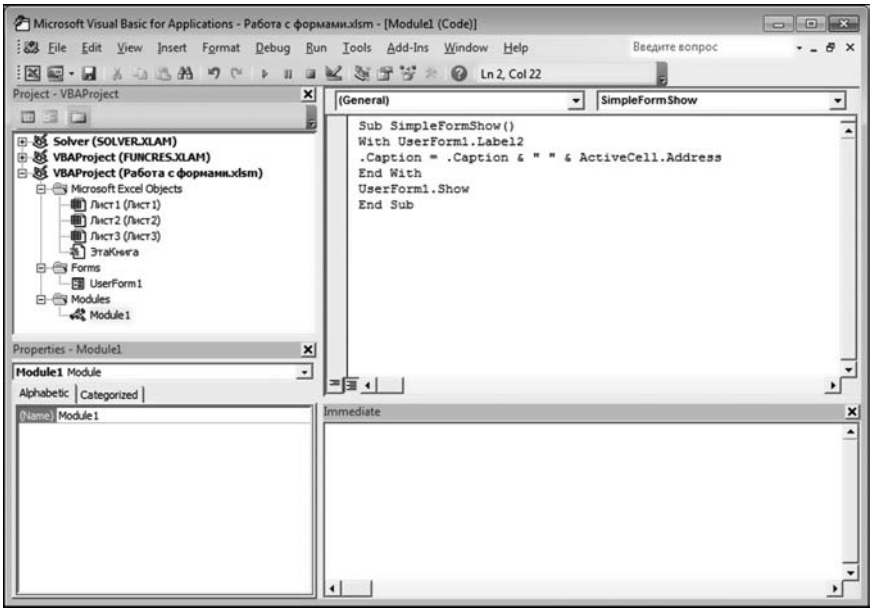


Рис. 11.22
В программный код макроса SimpleFormShow вносим изменения для отображения адреса активной ячейки

Объект этой метки называется Label2, а настройки для метки выполняются такие же, как и для ее предшественницы — за тем лишь исключением, что свойство **AutoSize** осталось со значением по умолчанию False. Нам также придется внести изменения в программный код макроса SimpleFormShow. В этот макрос перед командой отображения формы добавлен командный блок, который предназначен для добавления к тексту второй метки адреса активной ячейки. Редактор кода VBE с измененным кодом макроса SimpleFormShow представлен на рисунке 11.22.

Новый код макроса читатель может увидеть в листинге 11.3.

Листинг 11.3. Код макроса SimpleFormShow после внесения изменений

```
Sub SimpleFormShow()
 With UserForm1.Label2
   .Caption = .Caption & " " & ActiveCell.Address
 End With
 UserForm1.Show
End Sub
```

В макрос добавлен оператор With UserForm1.Label2, в теле которого всего одна команда .Caption = .Caption & " " & ActiveCell.Address. Этой командой к тексту метки (свойство UserForm1.Label2.Caption) добавляется пробел и адрес активной ячейки. Для определения адреса ячейки используем ссылку ActiveCell.Address. После того как текст метки определен, отображается форма — эту команду мы уже рассматривали.

На заметку

Объект формы — это UserForm1. Метки Label1 и Label2 размещены в окне этой формы. Поэтому обращение к метке, например, Label2 выглядит как UserForm1.Label2. Если нас интересует свойство Caption метки Label2, которая находится в окне формы UserForm1, то ссылка на свойство выполняется как UserForm1.Label2.Caption. Вот такая простая арифметика.

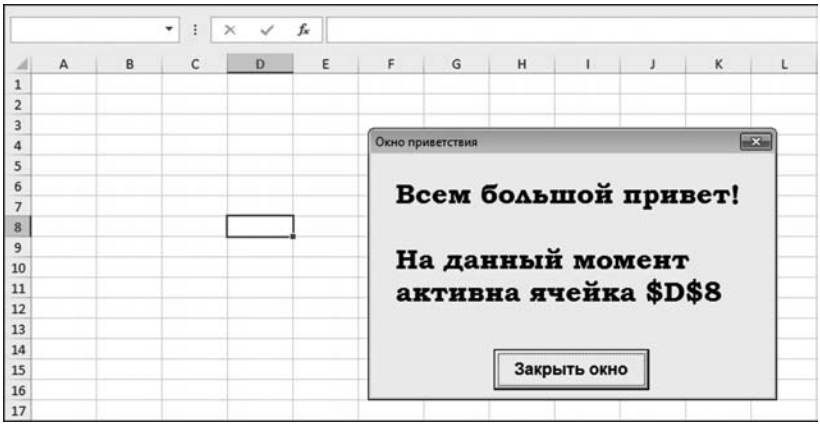


Рис. 11.23
В диалоговом окне отображается адрес активной ячейки

Если мы теперь выделим ячейку и запустим макрос на выполнение, получим результат, как, например, на рисунке 11.23.

Здесь выделена ячейка D8, о чем и говорит информация в окне сообщения. Наличие знаков доллара связано с тем, что отображается абсолютная ссылка на ячейку.

ОБРАБОТКА СОБЫТИЙ

Главное у человека — не деньги, а, натурально, хворма.
Вченость. Потому, ежели человек вченый,
так ему уже свет переворачивается вверх ногами.
Пардон, вверх дыбом.

Из к/ф «За двумя зайцами»

Здесь мы продолжим обсуждать методы работы с формами и, в частности, способы использования различных функциональных элементов в форме. Мы рассмотрим еще один небольшой пример, в котором создается форма с несколькими функциональными элементами. Форма нужна для заполнения диапазона ячеек числовыми значениями, а в окне формы выполняется настройка режима заполнения. Также в этом примере мы узнаем, как в программном коде используются массивы. На этом фоне произойдет знакомство с обработкой событий, выполняемой для элементов формы.

Что касается непосредственно нашего проекта, то он предназначен для заполнения ячеек числами Фибоначчи.

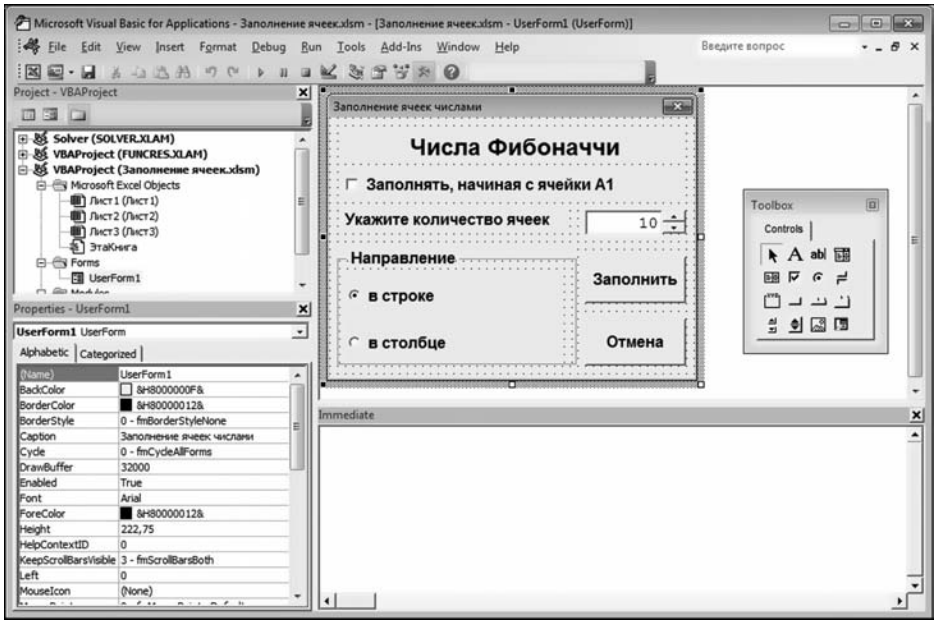


Рис. 11.24
Главная форма проекта

На заметку

Последовательность чисел Фибоначчи формируется так: первые два числа равны единице, а каждое следующее равно сумме двух предыдущих. Таким образом, получаем числа 1, 1, 2, 3, 5, 8, 13, 21 и т. д.

Ячейки рабочего листа могут заполняться или в строке, или в столбце. Также заполнение можно начинать из активной ячейки или из ячейки A1. В окне формы можно выполнять такие настройки:

- установить режим начальной ячейки заполнения — установив флажок специальной опции, можно перейти в режим заполнения документа, начиная с ячейки A1. Если флажок опции не установлен (по умолчанию), ячейки заполняются, начиная с активной;
- в специальном поле с числовым переключателем (пиктограмма с разнонаправленными стрелками) указывается количество заполняемых ячеек (диапазон возможных значений для количества ячеек от 3 до 20);
- переключатель (логический, на два положения), с помощью которого определяем, по строке или по столбцу заполнять ячейки. По умолчанию заполняется строка.

На рисунке 11.24 представлено окно редактора VBE с проектом, у которого есть форма. Именно эта форма отображается при выполнении макроса Fibonacci.

Программный код используемого нами макроса представлен в листинге 11.4.

Листинг 11.4. Код макроса для заполнения ячеек числами Фибоначчи

```
Sub Fibonacci()
 ' Отображение формы
 UserForm1.Show
End Sub
```

В макросе всего одна команда, которой отображается окно формы. Поэтому вся изюминка проекта, очевидно, связана с этой самой формой. Попробуем с ней разобраться, что называется, по полной программе.

В окне формы размещено несколько элементов управления, а именно: две текстовые метки, опция, текстовое поле с числовым переключателем, рамка с двумя элементами логического переключателя, а также две кнопки. Более подробно элементы формы описаны в таблице 11.2.

Что касается текстовых меток, то тут все просто — в них отображается текст. В частности, для элемента Label1 значение свойства **Caption** устанавливаем (в окне свойств) равным Числа Фибоначчи. Значение **Caption** для объекта Label2 задаем равным Укажите количество ячеек.

На заметку

Для всех элементов, кроме текстового поля (объект TextBox1), установлен шрифт **Arial** полужирный, размера **12**.

Значение **Caption** объекта CheckBox1 установлено равным Заполнять, начиная с ячейки A1. Свойство **Value** должно быть равно False, но, поскольку оно такое по умолчанию, то мы на этом свойстве особо не зацикливаемся.

Таблица 11.2

Элементы окна формы

Элемент	Описание
Label1	Текстовая метка со значением **Числа Фибоначчи**. Предназначена для отображения текстового заголовка в верхней части окна формы
Label2	Текстовая метка со значением **Укажите количество ячеек**. Метка содержит информацию о предназначении текстового поля и числового переключателя справа от метки
CheckBox1	Опция с текстом **Заполнить, начиная с ячейки A1** сразу под текстовой меткой Label1. Наличие/отсутствие флажка этой опции определяет начальную ячейку для заполнения: активная ячейка или ячейка A1
TextBox1	Текстовое поле для отображения/ввода значения числового переключателя SpinButton1. Числовое значение в текстовом поле определяет количество заполняемых ячеек. Значение может вводиться непосредственно в поле с клавиатуры или задаваться с помощью числового переключателя справа от поля
SpinButton1	Числовой переключатель для изменения значения (количество заполняемых ячеек) в текстовом поле TextBox1
Frame1	Рамка с заголовком **Направление** для группировки элементов переключателей OptionButton1 и OptionButton2
OptionButton1	Элемент переключателя с текстом **в строке**. Если переключатель установлен в этой позиции, заполняется строка ячеек. Режим используется по умолчанию
OptionButton2	Элемент переключателя с текстом **в столбце**. Если переключатель установлен в этой позиции, заполняется столбец ячеек
CommandButton1	Кнопка с текстом **Заполнить**. После щелчка на этой кнопке выполняется заполнение (в соответствии с настройками в окне формы) ячеек числами Фибоначчи. После заполнения ячеек окно формы закрывается
CommandButton2	Кнопка с текстом **Отмена**. После щелчка на кнопке окно формы закрывается без заполнения ячеек

А вообще свойство отвечает за режим выделения опции (выделена — значение True, а не выделена — значение False). Другие настройки особого интереса, пожалуй, не представляют.

Немного больше настроек придется выполнить для объекта TextBox1. В частности, шрифт устанавливаем обычный **Courier New** размера **12**. Свойство **Text** устанавливаем равным 10 — при этом автоматически такое же значение будет и у свойства **Value**. Это значение, которое будет отображаться в текстовом поле при отображении окна формы. Свойство **TextAlign** должно быть установлено равным 3-fmTextAlignRight (значение выбирается из раскрывающегося списка), что означает выравнивание текста в поле по правому краю.

На заметку

Важны с точки зрения функциональности элемента еще несколько свойств. По умолчанию значение свойства **Locked** установлено равным False. Если установить значение этого свойства True, в текстовое поле нельзя будет ввести значение с клавиатуры — при этом значение в поле отображается и его можно менять программными методами. Есть еще свойство **Enabled** (по умолчанию значение True), которое отвечает за полную блокировку объекта. Если устано-

вить значение свойства False, объект (поле в данном случае) будет полностью заблокирован (неактивен). За «видимость» элемента отвечает свойство **Visible**, значение которого True по умолчанию означает, что элемент отображается.

Для объекта SpinButton1 важно установить следующие значения для свойств объекта. Так, свойству **Max** устанавливается значение 20, свойству **Min** устанавливаем значение 3. Эти свойства определяют соответственно максимальное и минимальное возможные значения для счетчика. Текущее значение счетчика задается через свойство **Value** и равно 10 (как и значение для текстового поля).

На заметку

Состояние (значение свойства **Value**) текстового счетчика определяет количество заполняемых ячеек. Такие значения для минимального и максимального значений объясняются просто. Минимальное значение **3** облегчит нам работу — не придется проверять экзотические случаи с очень маленьким количеством заполняемых ячеек. Очень много ячеек тоже не заполнишь — последовательность Фибоначчи быстро нарастает. Поэтому ограничиваем верхний предел в **20** ячеек.

Стоит также обратить внимание на свойство **SmallChange**, которое определяет дискретность изменения значения числового счетчика. По умолчанию при щелчке на пиктограмме счетчика значение счетчика увеличивается/уменьшается на единицу.

Для элемента Frame1 задаем значение Направление для свойства **Caption**. Внутри элемента размещаем два элемента-переключателя. Для элемента OptionButton1 задаем свойство **Caption** равным в строке и устанавливаем значение свойства **Value** равным True. Последнее означает, что элемент будет выделен при первом отображении окна формы. Поскольку только один элемент переключателя может быть выделен, у элемента OptionButton2 это свойство равно False. Свойство **Caption** у объекта OptionButton2 устанавливаем равным в столбце. Также у обоих элементов задаем выравнивание по левому краю (свойство TextAlign равно 1-fmTextAlignLeft — выбирается из раскрывающегося списка). Для кнопок (объекты CommandButton1 и CommandButton2) задаем лишь названия.

По большому счету, с внешними эффектами мы закончили. Переходим к программным кодам. Нам необходимо решить несколько задач. В основном они касаются «связки» элементов. А именно, нам настоятельно необходимо:

- написать код для заполнения ячеек рабочего листа с учетом состояния опции CheckBox1 и элементов переключателей OptionButton1 и OptionButton2 (на самом деле, достаточно контролировать лишь один из этих двух элементов);
- связать числовой переключатель SpinButton1 с текстовым полем TextBox1 так, чтобы щелчок на счетчике приводил к изменению текста в текстовом поле;
- связать текстовое поле TextBox1 со счетчиком SpinButton1 так, чтобы при вводе в текстовое поле нового значения оно применялось и для счетчика;
- написать код для кнопки закрытия окна формы (без заполнения ячеек).

Самое простое задание последнее, с него и начнем. Код процедуры-обработчика щелчка на кнопке CommandButton2 (кнопка с названием **Отмена**) представлен в листинге 11.5.

Листинг 11.5. Обработчик для кнопки закрытия формы

```
Private Sub CommandButton2_Click()
 ' Выгрузка формы из памяти
 Unload UserForm1
End Sub
```

Здесь всего одна знакомая уже команда (это Unload UserForm1), которой форма выгружается из памяти. В листинге 11.6 приведен код процедуры-обработчика щелчка на числовом переключателе.

Листинг 11.6. Обработчик для числового переключателя

```
Private Sub SpinButton1_Change()
 ' Синхронизация текстового поля со значением числового переключателя
 TextBox1.Text = SpinButton1.Value
End Sub
```

Здесь тоже всего одна команда, причем достаточно простая: TextBox1.Text = SpinButton1.Value. Этой командой свойству Text объекта TextBox1 присваивается значение свойства Value объекта SpinButton1.

На заметку

Здесь важно хотя бы примерно представлять, как все это работает. Так, при щелчке на пиктограмме числового счетчика, в зависимости от того, какая из двух стрелок «щелкнута», значение переключателя (значение свойства Value) увеличивается/уменьшается на величину, определяемую свойством SmallChange (в нашем случае это единица) — если новое значение не выходит за допустимые пределы (задаются свойствами Max и Min). Процедура-обработчик запускается каждый раз при щелчке на счетчике. Поэтому, если мы изменяем состояние счетчика, это автоматически скажется на состоянии текстового поля.

Несколько замысловатее выглядит обработчик для текстового поля. Причем здесь встает вопрос о том, какое событие обрабатывать. Предлагаемое по умолчанию событие/процедура TextBox1_Change не подходит — это событие происходит каждый раз при изменении содержимого поля, т. е. в процессе редактирования содержимого. Нас же интересует ситуация, когда пользователь закончил редактировать поле. На этот случай предусмотрена процедура с ключевым словом *AfterUpdate*, т. е. нам нужно описать процедуру с именем TextBox1_AfterUpdate. Программный код этой процедуры представлен в листинге 11.7.

Листинг 11.7. Обработчик для обновления текстового поля

```
Private Sub TextBox1_AfterUpdate()
 ' Переменная для запоминания значения в текстовом поле
```

```
    Dim num As Integer
    ' Считывание числового значения текстового поля
    num = Val(TextBox1.Text)
    With SpinButton1
      ' Проверка попадания значения в допустимый диапазон
      If num >= .Min And num <= .Max Then
        ' Корректное значение присваивается числовому переключателю
        .Value = num
      Else
        ' Если значение некорректное, в текстовое поле
        ' вписывается значение числового переключателя
        TextBox1.Text = .Value
      End If
    End With
  End Sub
```

Причина такого немаленького кода при решении достаточно простой задачи в том, что нам важно проверить, попадает ли введенное пользователем в текстовое поле значение в допустимый диапазон значений. Алгоритм проверки следующий. На момент начала редактирования текстового поля свойство Text текстового поля TextBox1 и свойство Value числового счетчика SpinButton1 одинаковы. Следующий шаг — пользователь ввел в текстовое поле новое значение — теперь это значение свойства Text объекта TextBox1. Мы его проверяем, и если оно попадает в допустимый интервал значений, присваиваем соответствующее значение свойству Value числового счетчика SpinButton1. Если же новое значение в текстовом поле не попадает в допустимый интервал значений, в текстовое поле необходимо вернуть старое значение. Про это значение «помнит» свойство Value числового счетчика SpinButton1. Здесь действие обратное — свойству Text текстового поля TextBox1 присваивается значение свойства Value числового счетчика SpinButton1.

В теле процедуры TextBox1_AfterUpdate() объявляется целочисленная переменная num, которой затем командой `num = Val(TextBox1.Text)` присваивается значение. Мы использовали функцию Val(), которой в качестве аргумента передается текстовое представление числа, а функция в качестве результата возвращает реальное число. Текстовое представление числа считывается инструкцией TextBox1.Text из текстового поля. Таким образом, в переменную num записано новое значение в текстовом поле. Затем на сцену выходит условный оператор, в котором проверяется условие принадлежности значения переменной num интервалу от SpinButton1.Min до SpinButton1.Max. При этом мы использовали логический оператор And. Если условие выполнено, свойству SpinButton1.Value присваивается значение num. В противном случае свойству TextBox1.Text присваивается значение SpinButton1.Value.

Осталось поставить финальную точку — поработать над кодом обработчика события щелчка на кнопке CommandButton1. Код представлен в листинге 11.8.

Листинг 11.8. Обработчик для кнопки заполнения ячеек

```
Private Sub CommandButton1_Click()
 ‘ Размер массива для чисел Фибоначчи
 Dim N As Integer
 ‘ Объявление динамического массива
 Dim F() As Integer
 ‘ Размер массива задается значением в текстовом поле
 N = Val(TextBox1.Text)
 ‘ Определение размера массива
 ReDim Preserve F(1 To N)
 ‘ Первые два числа в последовательности Фибоначчи
 F(1) = 1
 F(2) = 1
 ‘ Заполнение массива числами Фибоначчи
 For i = 3 To N
   F(i) = F(i - 1) + F(i - 2)
 Next i
 ‘ Переменная для ссылки для начальной ячейки
 Dim StartCell As Range
 ‘ Определение начальной ячейки
 If CheckBox1.Value Then
   Set StartCell = Range(“A1”)
 Else
   Set StartCell = ActiveCell
 End If
 ‘ Определение “направления” заполнения
 If OptionButton1.Value Then
    For i = 1 To N
     ‘ Заполнение строки ячеек
     StartCell.Offset(0, i - 1).Value = F(i)
   Next i
 Else
   ‘ Заполнение столбца ячеек
    For i = 1 To N
     StartCell.Offset(i - 1, 0).Value = F(i)
   Next i
 End If
 ‘ Выгрузка формы из памяти
 Unload UserForm1
End Sub6
```

В известном смысле это «самодостаточный» код — он мало зависит от всего, что мы написали ранее. В коде есть несколько относительно интересных мест, на которые стоит обратить внимание. Важный идеологический момент связан с тем, что мы сначала создаем массив из чисел Фибоначчи, а затем копируем числа из массива в ячейки рабочего листа. Поэтому пред-

варительно определяются размеры массива (количество заполняемых ячеек), и этот массив заполняется числами. Затем мы определяем начальную ячейку заполнения и «направление» заполнения — в строке или в столбце. Теперь немного подробнее о том же самом. Целочисленная переменная N объявляется для записи в нее размера массива. Командой Dim F() As Integer объявляется динамический массив F целых чисел. Затем из текстового поля инструкцией N = Val(TextBox1.Text) считывается количество заполняемых ячеек (размер массива). Чтобы выделить место под массив соответствующего размера, выполняем инструкцию ReDim Preserve F(1 To N).

На заметку

Здесь мы столкнулись с достаточно пикантной ситуацией. Она касается массивов. Существует два их типа — статические и динамические. Размер статического массива должен быть известен на момент компиляции кода. Размер динамического массива определяется в процессе выполнения программы. Мы имеем дело именно со вторым случаем. С практической точки зрения разница между статическим и динамическим массивами в том, что размеры статического массива — константы. Например, командой Dim MyArray(1 To 100) As Integer объявляется целочисленный массив MyArray из 100 элементов (индексы элементов массива изменяются от 1 до 100). Мы себе такую роскошь позволить не можем, поэтому сначала инструкцией Dim F() As Integer декларируем, что F — это массив из целых чисел, неизвестно пока какого размера. А уж командой ReDim Preserve F(1 To N) подтверждаем, что массив из N элементов. Предварительно, разумеется, мы присваиваем значение переменной N. Ключевое слово Preserve используется, если при изменении размера массива необходимо сохранить уже существующие элементы. В данном случае его можно было не использовать.

Командами F(1) = 1 и F(2) = 1 заполняются первые два элемента массива. Для заполнения прочих элементов массива используем оператор цикла с командой F(i) = F(i-1)+F(i-2) (i — индексная переменная) в теле оператора цикла (каждый следующий элемент равен сумме двух предыдущих).

Следующий этап связан с определением начальной ячейки. Для этого командой Dim StartCell As Range мы объявляем переменную StartCell типа Range. Это будет ссылка на начальную ячейку. В условном операторе проверяем, установлен ли флажок опции CheckBox1 (значение свойства Value равно True или False). В зависимости от результата проверки выполняется команда Set StartCell = Range("A1") (начальная ячейка A1) или Set StartCell = ActiveCell (начальная ячейка — активная на момент запуска макроса).

На заметку

При присваивании значений объектным ссылкам используется ключевое слово Set.

«Направление» заполнения определяем путем проверки значения свойства OptionButton1.Value (выясняем, установлен переключатель OptionButton1 или нет). Если переключатель установлен, то запускается оператор цикла с командой StartCell.Offset(0,i-1).Value = F(i) в теле цикла. В противном случае запускается другой оператор цикла, в котором выполняется команда StartCell.Offset(i-1,0).Value = F(i) (здесь и выше i — индексная переменная). Этими командами ячейки рабочего документа заполняются значениями из массива F.

На заметку

Выше мы использовали метод Offset(). Метод вызывается из объекта ячейки и имеет два аргумента. Эти аргументы определяют смещение вдоль столбца и вдоль строки. В качестве результата возвращается ссылка на соответствующую ячейку. Так, например, инструкция StartCell.Offset(2,5).Value означает свойство Value ячейки, которая находится на 2 строки вниз и 5 столбцов вправо по отношению к ячейке StartCell.

Выгрузка формы из памяти осуществляется командой Unload UserForm1. На этом, собственно, все. Осталось проверить, как работает макрос, код которого состоит из команды вызова окна формы (см. листинг 11.4).

Запускаем макрос, в результате чего появится окно, представленное на рисунке 11.25.

Оставляем те настройки, что предлагаются по умолчанию, и щелкаем кнопку **Заполнить**. Результат показан на рисунке 11.26.

Несложно проверить, как функционирует окно формы и каков результат вычислений для других настроек. Желающие могут проделать это самостоятельно.

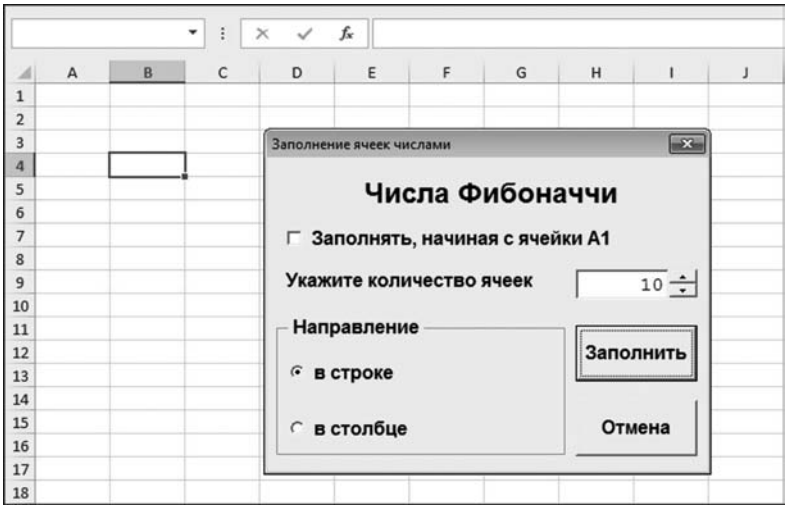


Рис. 11.25
Окно формы на фоне рабочего документа

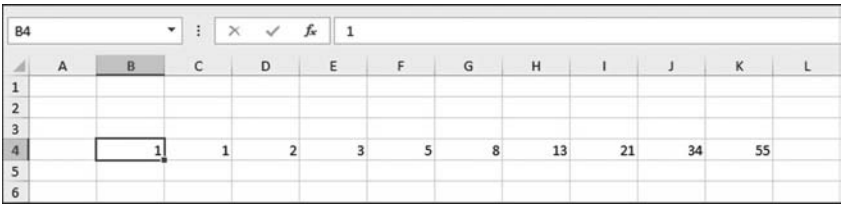


Рис. 11.26
Результат выполнения макроса Fibonacci

ЧАСТЬ ЧЕТВЕРТАЯ

ПРИКЛАДНЫЕ ЗАДАЧИ

ГЛАВА 12 РЕШЕНИЕ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ И СИСТЕМ

Я как-нибудь обойдусь
и без высшей математики,
и без низшей!

Из к/ф «Большая перемена»

С алгебраическими уравнениями мы уже имели дело в предыдущих главах. Но там нас, в первую очередь, интересовал результат. Тем не менее нередко случаются ситуации, когда сам процесс получения результата не менее важен, чем этот самый результат. Другими словами, здесь мы будем рассматривать не просто методы решения уравнений (и систем уравнений) с помощью Excel, а реализацию определенных алгоритмов решения алгебраических уравнений средствами Excel.

На заметку

Даже один и тот же алгоритм в Excel может реализовываться совершенно различными способами, причем количество этих способов практически неограниченно (или, по крайней мере, очень большое). Особенно это актуально, если учесть наличие такого мощного средства разработки, как VBA. Очевидно, что практически в каждом из рассматриваемых примеров одним из возможных вариантов решения задачи может быть составление программного кода средствами VBA, реализующего соответствующий алгоритм. Но даже в этом случае остается открытым вопрос о той степени «взаимодействия» программного кода макроса или функции с рабочим документом Excel, которую предполагается реализовать. Наконец, можно так реализовать программный код, что это будет фактически отдельная независимая программа — связь с документом Excel ограничивается лишь тем, что макрос, например, выполняется под управлением приложения Excel, но не более. Далее мы будем обычно решать задачи сразу несколькими способами, чтобы наиболее полно «охватить» общую картину.

МЕТОД ПОЛОВИННОГО ДЕЛЕНИЯ

Это — тебе. Это — мне. Это — тебе.
Это — обратно тебе. Это — все время тебе.
Посмотрим, я себя не обделил?

Из к/ф «Свадьба в Малиновке»

Метод половинного деления используется для поиска решения уравнения вида $f(x) = 0$ на указанном интервале $x_1 < x < x_2$ при условии, что функция $f(x)$ на этом интервале непрерывна и принимает на границах интервала значения разных знаков. Если все означенные условия выполнены, то мы с

помощью метода половинного деления можем найти решение уравнения с любой заданной (отличной от нуля) точностью за конечное количество итераций.

На заметку

Если функция на интервале непрерывна и на границах интервала принимает значения разных знаков, то на этом интервале у функции имеется хотя бы один нуль (т. е. хотя бы в одной точке функция принимает нулевое значение). Но таких нулей может быть и больше одного. Другими словами, необходимые для применения метода половинного деления условия гарантируют, что корень есть, но не гарантируют, что он один. И если корней несколько, то какой именно будет найден — сказать достаточно сложно (все зависит от начальных границ интервала поиска решения). Локализация корней уравнения — отдельная сложная задача, решение которой выходит за рамки книги. В данном конкретном случае, чтобы гарантировать единственность решения на указанном интервале на функцию, определяющую уравнение, можно наложить условие неизменности знака производной (на всем интервале поиска решения). Таким образом, если мы решаем уравнение $f(x) = 0$ и функция $f(x)$ на интервале поиска решения непрерывна, монотонна (монотонно спадает или монотонно убывает) и на границах интервала принимает значения разных знаков, то уравнение имеет решение, и это решение единственно.

Алгоритм поиска решения алгебраического уравнения $f(x) = 0$ методом половинного деления состоит в следующем:

- вычисляется значение функции $f(x)$ в центральной точке интервала поиска решения. Интерес представляет знак функции в этой точке;
- если функция в центральной точке меньше нуля, в центральную точку смещается та граница интервала поиска решения, на которой функция меньше нуля;
- если функция в центральной точке больше нуля, в центральную точку смещается та граница интервала поиска решения, на которой функция больше нуля;
- в результате интервал поиска решения уменьшается вдвое, а задача, фактически, сводится к предыдущей (с поправкой на измененный интервал) — нужно найти корень уравнения $f(x) = 0$ на интервале, на границах которого функция $f(x)$ принимает значения разных знаков;
- процесс продолжается до тех пор, пока ширина интервала не станет достаточно малой для того, чтобы обеспечить необходимую точность вычисления корня.

Процесс поиска решения алгебраического уравнения методом половинного деления иллюстрируется на рисунке 12.1.

Логическая схема, реализуемая при решении задачи о поиске корня алгебраического уравнения методом половинного деления, в Excel может реализовываться самыми разными способами. Мы рассмотрим наиболее интересные варианты. Начнем с ситуации, когда метод половинного деления реализуется исключительно средствами встроенных функций рабочего документа (т. е. без привлечения программного кода VBA). Для конкретики рассмотрим кубическое уравнение $x^3 - 8x^2 + x + 42 = 0$, которое имеет три решения:

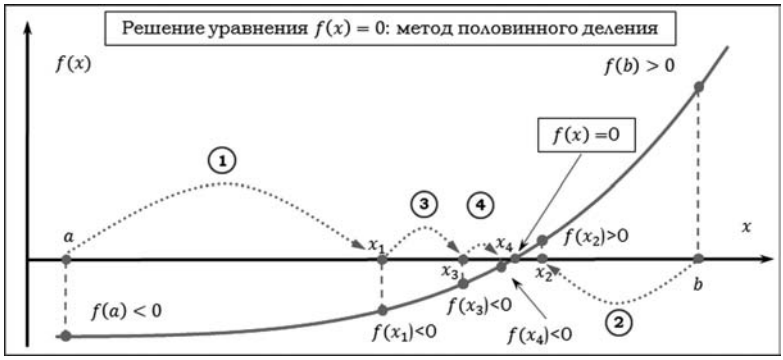


Рис. 12.1
Метод половинного деления: цифрами и штрихованными линиями со стрелкой показана последовательность перемещения границ интервала поиска решения

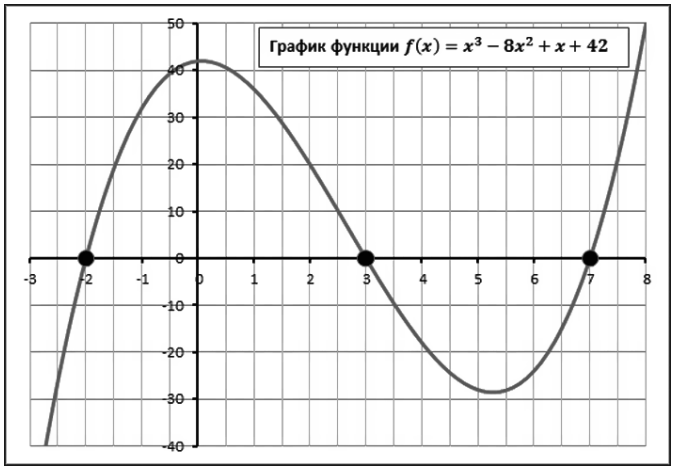


Рис. 12.2
График функции, определяющей решаемое уравнение

№	a	b	x=(a+b)/2	f(a)	f(b)	f(x)
0	0	5	2,5	42	-28	10,125
1	2,5	5	3,75	10,125	-28	-14,015625

Рис. 12.3
Документ для решения уравнения методом половинного деления перед началом основных расчетов

$x_1 = -2$, $x_2 = 3$ и $x_3 = 7$. График функции $f(x) = x^3 - 8x^2 + x + 42$ (созданный средствами Excel) представлен на рисунке 12.2.

Мы будем решать уравнение $x^3 - 8x^2 + x + 42 = 0$ на разных интервалах, и начнем с поиска второго корня $x_2 = 3$ — искать корень будем на интервале $0 \le x \le 5$. Рабочий документ в «начальном состоянии» представлен на рисунке 12.3.

На начальном этапе, помимо «декоративных» элементов, числами и формулами заполняется диапазон ячеек A5:G6. Общая структура рабочего документа такая:

- в столбце A вычисляется и отображается номер итерации;
- в столбце B вычисляется и отображается левая граница поиска интервала решения;
- в столбце C вычисляется и отображается правая граница интервала поиска решения;
- в столбце D вычисляется и отображается значение центральной точки интервала, на котором ищется решение;
- в столбцах E, F и G отображаются соответственно значения функции $f(x) = x^3 - 8x^2 + x + 42$ на границах интервала поиска решения и в его центре.

Более детально вводимые в ячейки диапазона A5:G6 значения описаны в таблице 12.1.

Принципиально важно заполнить 5-ю и 6-ю строки в рабочем документе. Следующий процесс достаточно прост — выделяем диапазон ячеек A6:G6 и используем процедуру автоматического заполнения ячеек. На рисунке 12.4 показано, как может выглядеть документ после копирования значений диапазона A6:G6 в диапазон A7:G7.

Таким образом, заполнив новую строку в рабочем документе, получаем очередную итерацию. Номер итерации, как отмечалось, отображается в столбце A. Текущее приближенное значение корня уравнения отображается в столбце D (т. е. это центральная точка интервала поиска корня). На рисунке 12.5 показан результат вычисления корня уравнения по 20-ти итерациям.

При этом в столбце G для приближенного значения корня отображается значение функции, определяющей решаемое уравнение. В идеале, если корень вычислен точно, соответствующее значение в столбце G должно равняться нулю.

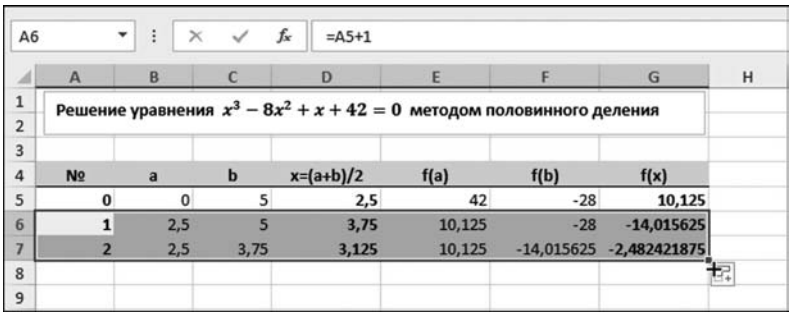
На заметку

Чем меньше в точке предполагаемого корня отличается от нуля функция $f(x)$, тем точнее найден корень. Но это не одно и то же, что точность вычисления корня (хотя данные понятия и взаимосвязаны). Обычно в методе половинного деления в качестве точности вычисления корня берется величина, равная половине длины интервала, на котором локализован корень уравнения. Объяснение простое: если корень локализован на интервале от a до b, то центральная точка этого интервала $c = (a + b)/2$ отстоит от корня на расстояние, не большее, чем $(b - a)/2$. Что касается рассматриваемого примера и, более конкретно, метода его решения в Excel, то здесь имеется одна особенность, связанная с тем, как обрабатывается ситуация «попадания» в граничную точку интервала поиска корня. Об этом речь еще будет идти далее.

Таблица 12.1

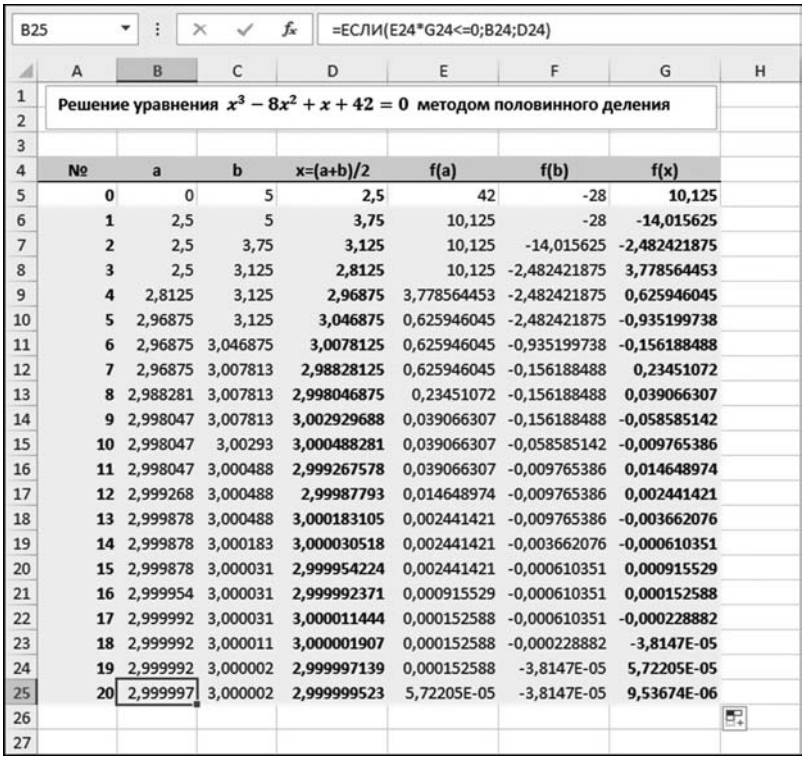
Метод половинного деления

Ячейка	Значение	Комментарий
A5	0	Номер начальной итерации (полагаем стадию начала поиска корня нулевым приближением)
A6	=A5+1	Формула для вычисления номера итерации на основе номера предыдущей итерации (номер итерации увеличивается на единицу)
B5	0	Начальное значение для левой границы интервала, на котором ищется решение уравнения
B6	=ЕСЛИ(E5*G5<=0;B5;D5)	Формула для вычисления значения левой границы интервала, на котором ищется корень уравнения. Если произведение значения функции на левой границе (ячейка E5) и значения функции в центре интервала (ячейка G5) меньше либо равно нулю, то для текущей итерации значение левой границы интервала не меняется и остается таким же, как для предыдущей итерации (ячейка B5). В противном случае значение левой границы для текущей итерации определяется значением центральной точки на предыдущей итерации (ячейка D5)
C5	5	Начальное значение для правой границы интервала поиска корня уравнения
C6	=ЕСЛИ(F5*G5<=0;C5;D5)	Формула для вычисления значения правой границы интервала, на котором ищется корень уравнения. Если произведение значения функции на левой границе (ячейка F5) и значения функции в центре интервала (ячейка G5) меньше либо равно нулю, то для текущей итерации значение правой границы интервала не меняется и остается таким же, как для предыдущей итерации (ячейка C5). В противном случае значение левой границы для текущей итерации определяется значением центральной точки на предыдущей итерации (ячейка D5)
D5	=(B5+C5)/2	Центральная точка интервала, на котором ищется решение уравнения
D6	=(B6+C6)/2	Формула для вычисления центральной точки интервала для поиска корня уравнения для первой итерации. Ячейка заполняется копированием формулы из ячейки D5
E5	=B5^3-8*B5^2+B5+42	Формула для вычисления значения функции на левой границе интервала поиска корня уравнения для нулевой итерации (в начале вычислений)
E6	=B6^3-8*B6^2+B6+42	Формула для вычисления значения функции на левой границе интервала поиска корня уравнения для первой итерации. Ячейка заполняется копированием формулы из ячейки E5
F5	=C5^3-8*C5^2+C5+42	Формула для вычисления значения функции на правой границе интервала поиска корня уравнения для нулевой итерации (начало вычислений). Ячейка заполняется копированием формулы из ячейки E5
F6	=C6^3-8*C6^2+C6+42	Формула для вычисления значения функции на правой границе интервала поиска корня уравнения для первой итерации. Ячейка заполняется копированием формулы из ячейки E6 или F5
G5	=D5^3-8*D5^2+D5+42	Формула для вычисления значения функции в центральной точке интервала поиска корня уравнения для нулевой итерации (начало вычислений). Ячейка заполняется копированием формулы из ячейки F5
G6	=D6^3-8*D6^2+D6+42	Формула для вычисления значения функции в центральной точке интервала поиска корня уравнения для первой итерации. Ячейка заполняется копированием формулы из ячейки F6 или G5



№	a	b	x=(a+b)/2	f(a)	f(b)	f(x)
0	0	5	2,5	42	-28	10,125
1	2,5	5	3,75	10,125	-28	-14,015625
2	2,5	3,75	3,125	10,125	-14,015625	-2,482421875

Рис. 12.4
Вторая итерация при вычислении корня уравнения методом половинного деления



№	a	b	x=(a+b)/2	f(a)	f(b)	f(x)
0	0	5	2,5	42	-28	10,125
1	2,5	5	3,75	10,125	-28	-14,015625
2	2,5	3,75	3,125	10,125	-14,015625	-2,482421875
3	2,5	3,125	2,8125	10,125	-2,482421875	3,778564453
4	2,8125	3,125	2,96875	3,778564453	-2,482421875	0,625946045
5	2,96875	3,125	3,046875	0,625946045	-2,482421875	-0,935199738
6	2,96875	3,046875	3,0078125	0,625946045	-0,935199738	-0,156188488
7	2,96875	3,007813	2,98828125	0,625946045	-0,156188488	0,23451072
8	2,988281	3,007813	2,998046875	0,23451072	-0,156188488	0,039066307
9	2,998047	3,007813	3,002929688	0,039066307	-0,156188488	-0,058585142
10	2,998047	3,00293	3,000488281	0,039066307	-0,058585142	-0,009765386
11	2,998047	3,000488	2,999267578	0,039066307	-0,009765386	0,014648974
12	2,999268	3,000488	2,99987793	0,014648974	-0,009765386	0,002441421
13	2,999878	3,000488	3,000183105	0,002441421	-0,009765386	-0,003662076
14	2,999878	3,000183	3,000030518	0,002441421	-0,003662076	-0,000610351
15	2,999878	3,000031	2,999954224	0,002441421	-0,000610351	0,000915529
16	2,999954	3,000031	2,999992371	0,000915529	-0,000610351	0,000152588
17	2,999992	3,000031	3,000011444	0,000152588	-0,000610351	-0,000228882
18	2,999992	3,000011	3,000001907	0,000152588	-0,000228882	-3,8147E-05
19	2,999992	3,000002	2,999997139	0,000152588	-3,8147E-05	5,72205E-05
20	2,999997	3,000002	2,999999523	5,72205E-05	-3,8147E-05	9,53674E-06

Рис. 12.5
Результат вычисления корня уравнения методом половинного деления по 20-ти итерациям

Созданный нами документ удобен тем, что позволяет вычислять корень практически на любом интервале (конечно, при условии, что на указанном интервале корень есть). Для этого достаточно в ячейки B5 и C5 ввести новые значения для границ интервала. На рисунке 12.6 показано, как рабочий документ используется для нахождения корня уравнения $x_1 = -2$.

D25 | =(B25+C25)/2

Решение уравнения $x^3 - 8x^2 + x + 42 = 0$ методом половинного деления

№	a	b	x=(a+b)/2	f(a)	f(b)	f(x)
0	-6	-1	-3,5	-468	32	-102,375
1	-3,5	-1	-2,25	-102,375	32	-12,140625
2	-2,25	-1	-1,625	-12,140625	32	14,95898438
3	-2,25	-1,625	-1,9375	-12,140625	14,95898438	2,758056641
4	-2,25	-1,9375	-2,09375	-12,140625	2,758056641	-4,34262085
5	-2,09375	-1,9375	-2,015625	-4,34262085	2,758056641	-0,706546783
6	-2,01563	-1,9375	-1,9765625	-0,706546783	2,758056641	1,047009945
7	-2,01563	-1,97656	-1,99609375	-0,706546783	1,047009945	0,175567687
8	-2,01563	-1,99609	-2,005859375	-0,706546783	0,175567687	-0,264152728
9	-2,00586	-1,99609	-2,000976563	-0,264152728	0,175567687	-0,043958665
10	-2,00098	-1,99609	-1,998535156	-0,043958665	0,175567687	0,065887931
11	-2,00098	-1,99854	-1,999755859	-0,043958665	0,065887931	0,010985494
12	-2,00098	-1,99976	-2,000366211	-0,043958665	0,010985494	-0,01648137
13	-2,00037	-1,99976	-2,000061035	-0,01648137	0,010985494	-0,002746634
14	-2,00006	-1,99976	-1,999908447	-0,002746634	0,010985494	0,004119756
15	-2,00006	-1,99991	-1,999984741	-0,002746634	0,004119756	0,000686642
16	-2,00006	-1,99998	-2,000022888	-0,002746634	0,000686642	-0,001029976
17	-2,00002	-1,99998	-2,000003815	-0,001029976	0,000686642	-0,000171662
18	-2	-1,99998	-1,999994278	-0,000171662	0,000686642	0,000257492
19	-2	-1,99999	-1,999999046	-0,000171662	0,000257492	4,29153E-05
20	-2	-2	-2,000001431	-0,000171662	4,29153E-05	-6,4373E-05

Рис. 12.6
Вычисление еще одного корня уравнения

F25 | =C25^3-8*C25^2+C25+42

Решение уравнения $x^3 - 8x^2 + x + 42 = 0$ методом половинного деления

№	a	b	x=(a+b)/2	f(a)	f(b)	f(x)
0	-3	10	3,5	-60	252	-9,625
1	3,5	10	6,75	-9,625	252	-8,203125
2	6,75	10	8,375	-8,203125	252	76,67773438
3	6,75	8,375	7,5625	-8,203125	76,67773438	24,54125977
4	6,75	7,5625	7,15625	-8,203125	24,54125977	5,94619751
5	6,75	7,15625	6,953125	-8,203125	5,94619751	-1,659038544
6	6,953125	7,15625	7,0546875	-1,659038544	5,94619751	2,00779295
7	6,953125	7,054688	7,00390625	-1,659038544	2,00779295	0,140823424
8	6,953125	7,003906	6,978515625	-1,659038544	0,140823424	-0,767446898
9	6,978516	7,003906	6,991210938	-0,767446898	0,140823424	-0,31540271
10	6,991211	7,003906	6,997558594	-0,31540271	0,140823424	-0,087813154
11	6,997559	7,003906	7,000732422	-0,087813154	0,140823424	0,026374162
12	6,997559	7,000732	6,999145508	-0,087813154	0,026374162	-0,030752227
13	6,999146	7,000732	6,999938965	-0,030752227	0,026374162	-0,002197217
14	6,999939	7,000732	7,000335693	-0,002197217	0,026374162	0,012086426
15	6,999939	7,000336	7,000137329	-0,002197217	0,012086426	0,004944093
16	6,999939	7,000137	7,000038147	-0,002197217	0,004944093	0,00137331
17	6,999939	7,000038	6,999988556	-0,002197217	0,00137331	-0,000411986
18	6,999989	7,000038	7,000013351	-0,000411986	0,00137331	0,000480654
19	6,999989	7,000013	7,000000954	-0,000411986	0,000480654	3,43323E-05
20	6,999989	7,000001	6,999994755	-0,000411986	3,43323E-05	-0,000188827

Рис. 12.7
Начальный интервал поиска решения содержит три корня уравнения

D25 =(B25+C25)/2

	A	B	C	D	E	F	G	H
1	Решение уравнения $x^3 - 8x^2 + x + 42 = 0$ методом половинного деления							
2								
3								
4	№	a	b	x=(a+b)/2	f(a)	f(b)	f(x)	
5	0	-3	8	2,5	-60	50	10,125	
6	1	-3	2,5	-0,25	-60	10,125	41,234375	
7	2	-3	-0,25	-1,625	-60	41,234375	14,95898438	
8	3	-3	-1,625	-2,3125	-60	14,95898438	-15,46020508	
9	4	-2,3125	-1,625	-1,96875	-15,46020508	14,95898438	1,392608643	
10	5	-2,3125	-1,96875	-2,140625	-15,46020508	1,392608643	-6,607761383	
11	6	-2,14063	-1,96875	-2,0546875	-6,607761383	1,392608643	-2,502971172	
12	7	-2,05469	-1,96875	-2,01171875	-2,502971172	1,392608643	-0,529267967	
13	8	-2,01172	-1,96875	-1,990234375	-0,529267967	1,392608643	0,438118912	
14	9	-2,01172	-1,99023	-2,000976563	-0,529267967	0,438118912	-0,043958665	
15	10	-2,00098	-1,99023	-1,995605469	-0,043958665	0,438118912	0,197483624	
16	11	-2,00098	-1,99561	-1,998291016	-0,043958665	0,197483624	0,076863413	
17	12	-2,00098	-1,99829	-1,999633789	-0,043958665	0,076863413	0,016477615	
18	13	-2,00098	-1,99963	-2,000305176	-0,043958665	0,016477615	-0,013734214	
19	14	-2,00031	-1,99963	-1,999969482	-0,013734214	0,016477615	0,001373278	
20	15	-2,00031	-1,99997	-2,000137329	-0,013734214	0,001373278	-0,006180074	
21	16	-2,00014	-1,99997	-2,000053406	-0,006180074	0,001373278	-0,002403299	
22	17	-2,00005	-1,99997	-2,000011444	-0,002403299	0,001373278	-0,000514986	
23	18	-2,00001	-1,99997	-1,999990463	-0,000514986	0,001373278	0,000429152	
24	19	-2,00001	-1,99999	-2,000000954	-0,000514986	0,000429152	-4,29154E-05	
25	20	-2	-1,99999	-1,999995708	-4,29154E-05	0,000429152	0,000193119	
26								

Рис. 12.8
Изменение границ интервала поиска решения приводит к тому, что вычисляется еще один корень уравнения

B25 =ЕСЛИ(E24*G24<=0;B24;D24)

	A	B	C	D	E	F	G	H
1	Решение уравнения $x^3 - 8x^2 + x + 42 = 0$ методом половинного деления							
2								
3								
4	№	a	b	x=(a+b)/2	f(a)	f(b)	f(x)	
5	0	-2	0	-1	0	42	32	
6	1	-2	-1	-1,5	0	32	19,125	
7	2	-2	-1,5	-1,75	0	19,125	10,390625	
8	3	-2	-1,75	-1,875	0	10,390625	5,408203125	
9	4	-2	-1,875	-1,9375	0	5,408203125	2,758056641	
10	5	-2	-1,9375	-1,96875	0	2,758056641	1,392608643	
11	6	-2	-1,96875	-1,984375	0	1,392608643	0,699710846	
12	7	-2	-1,98438	-1,9921875	0	0,699710846	0,350708485	
13	8	-2	-1,99219	-1,99609375	0	0,350708485	0,175567687	
14	9	-2	-1,99609	-1,998046875	0	0,175567687	0,087837227	
15	10	-2	-1,99805	-1,999023438	0	0,087837227	0,043931962	
16	11	-2	-1,99902	-1,999511719	0	0,043931962	0,021969319	
17	12	-2	-1,99951	-1,999755859	0	0,021969319	0,010985494	
18	13	-2	-1,99976	-1,99987793	0	0,010985494	0,005492955	
19	14	-2	-1,99988	-1,999938965	0	0,005492955	0,00274653	
20	15	-2	-1,99994	-1,999969482	0	0,00274653	0,001373278	
21	16	-2	-1,99997	-1,999984741	0	0,001373278	0,000686642	
22	17	-2	-1,99998	-1,999992371	0	0,000686642	0,000343322	
23	18	-2	-1,99999	-1,999996185	0	0,000343322	0,000171661	
24	19	-2	-2	-1,999998093	0	0,000171661	8,58306E-05	
25	20	-2	-2	-1,999999046	0	8,58306E-05	4,29153E-05	
26								

Рис. 12.9
Результат вычислений в случае, когда одна из границ начального интервала поиска решения совпадает с корнем уравнения

Мы ввели в ячейку B5 значение -6, а в ячейку C5 — значение -1. В результате получили достаточно неплохое приближение для корня $x_1 = -2$ (см. ячейку D25 в рабочем документе на рисунке 12.6).

Ситуация может быть и более сложной. Например, на рисунке 12.7 показано, каков будет результат вычислений корня, если в качестве начальных значений для границ интервала поиска решения указать значения -3 и 10.

Хотя формально все условия для применимости метода половинного деления выполнены, начальный интервал поиска решения содержит не один, как ранее, а все три корня. Несмотря на то, что решение найдено, вопросы остаются. Главный из них: почему найден именно корень $x_3 = 7$, а не какой-то другой? На самом деле все зависит от того, какие именно границы интервала выбраны. Скажем, если вместо верхней границы взять 8, а не 10, в результате вычислений «найдется» первый корень (см. рис. 12.8).

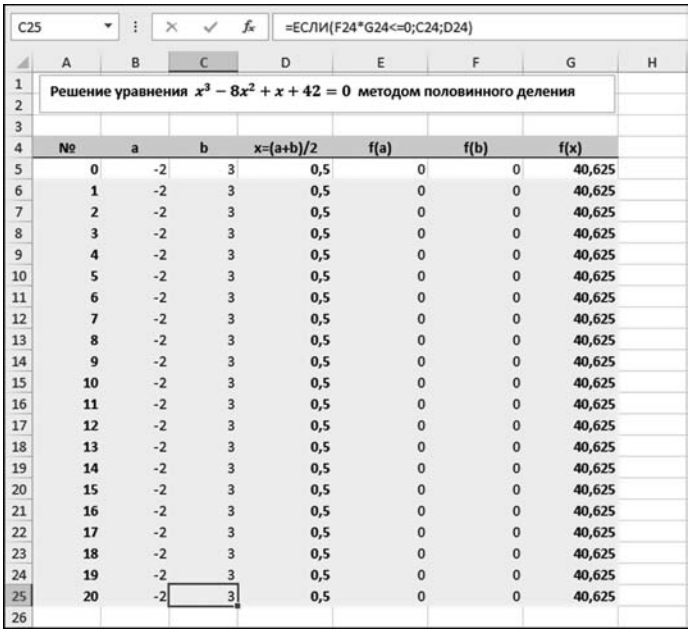
Достойна некоторого пояснения и ситуация, когда одна или обе начальных границы интервала совпадают с корнем полинома. На рисунке 12.9 показано, что будет, если на одной из границ функция $f(x)$ обращается в ноль.

Мы указали для левой границы начального интервала поиска решения значение -2 (корень уравнения), а для другой границы — значение 0. Специфика формул, по которым для каждой новой итерации вычисляются значения для границ интервала поиска решения, такова, что если на этой границе функция обращается в нуль, то значение границы не изменится. Поэтому при вычислениях «нулевая» граница остается неизменной. Это же происходит, если обе границы совпадают с корнями уравнения. Такая ситуация показана на рисунке 12.10.

И если в первом случае (когда лишь одна граница совпадает с корнем уравнения) центральная точка интервала поиска корня дает некоторое представление о корне уравнения, то во втором случае (когда обе границы совпадают с корнями уравнения) центральная точка с очевидностью не может рассматриваться как корень уравнения. Могут быть и другие «интересные» случаи. Например, на рисунке 12.11 некорректно указан начальный интервал для поиска корня (не выполняется необходимое условие различия знаков функции на границах интервала).

Очевидно, что в этом случае решение не найдено. Каков общий вывод? Он, пожалуй, состоит в том, что хотя предложенный подход прост и удобен, он не до конца автоматизирован. Во всяком случае, контролировать процесс вычислений необходимо в ручном режиме. Более того, есть один очень неудобный момент. И связан он с тем, что если нам понадобится изменить функцию $f(x)$ (т. е. если мы захотим решить другое уравнение), то придется заново вводить формулы в несколько ячеек, а это чревато ошибками (даже с учетом возможности автоматического заполнения ячеек). Конечно, можно по-иному организовать рабочий документ, но принципиально это ситуацию навряд ли изменит. Поэтому далее мы прибегнем к помощи VBA.

Создадим документ, в котором функциональная зависимость для уравнения и интервал поиска решения будут задаваться непосредственно в рабочем документе Excel, а само вычисление корня уравнения выполняется с помощью пользовательской процедуры, написанной в VBA.

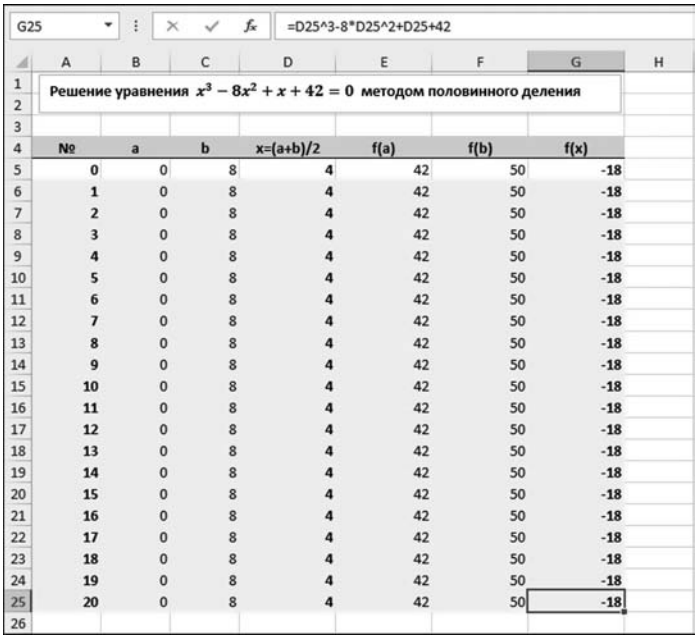


C25 | =ЕСЛИ(F24*G24<=0;C24;D24)

Решение уравнения $x^3 - 8x^2 + x + 42 = 0$ методом половинного деления

№	a	b	x=(a+b)/2	f(a)	f(b)	f(x)
0	-2	3	0,5	0	0	40,625
1	-2	3	0,5	0	0	40,625
2	-2	3	0,5	0	0	40,625
3	-2	3	0,5	0	0	40,625
4	-2	3	0,5	0	0	40,625
5	-2	3	0,5	0	0	40,625
6	-2	3	0,5	0	0	40,625
7	-2	3	0,5	0	0	40,625
8	-2	3	0,5	0	0	40,625
9	-2	3	0,5	0	0	40,625
10	-2	3	0,5	0	0	40,625
11	-2	3	0,5	0	0	40,625
12	-2	3	0,5	0	0	40,625
13	-2	3	0,5	0	0	40,625
14	-2	3	0,5	0	0	40,625
15	-2	3	0,5	0	0	40,625
16	-2	3	0,5	0	0	40,625
17	-2	3	0,5	0	0	40,625
18	-2	3	0,5	0	0	40,625
19	-2	3	0,5	0	0	40,625
20	-2	3	0,5	0	0	40,625

Рис. 12.10
Результат вычислений в случае, когда границы начального интервала поиска решения совпадают с корнями уравнения



G25 | =D25^3-8*D25^2+D25+42

Решение уравнения $x^3 - 8x^2 + x + 42 = 0$ методом половинного деления

№	a	b	x=(a+b)/2	f(a)	f(b)	f(x)
0	0	8	4	42	50	-18
1	0	8	4	42	50	-18
2	0	8	4	42	50	-18
3	0	8	4	42	50	-18
4	0	8	4	42	50	-18
5	0	8	4	42	50	-18
6	0	8	4	42	50	-18
7	0	8	4	42	50	-18
8	0	8	4	42	50	-18
9	0	8	4	42	50	-18
10	0	8	4	42	50	-18
11	0	8	4	42	50	-18
12	0	8	4	42	50	-18
13	0	8	4	42	50	-18
14	0	8	4	42	50	-18
15	0	8	4	42	50	-18
16	0	8	4	42	50	-18
17	0	8	4	42	50	-18
18	0	8	4	42	50	-18
19	0	8	4	42	50	-18
20	0	8	4	42	50	-18

Рис. 12.11
Результат вычислений, если начальный интервал поиска корня указан некорректно

Важный концептуальный момент связан с тем, как мы будем определять функциональную зависимость для решаемого уравнения. Поступим следующим образом: в одну ячейку рабочего документа введем «пробную» точку для аргумента функции (например, центральную точку интервала поиска решения), а в другую ячейку введем формулу, которая на основе ячейки с аргументом вычисляет значение функции в соответствующей точке. Таким образом, в документе будет две ячейки: одна содержит значение аргумента функции, а вторая содержит значение функции для этого аргумента. Следовательно, если изменять значение в ячейке с аргументом, изменится и значение в ячейке с функцией. На этой особенности рабочего документа Excel будет базироваться наш алгоритм вычисления приближенного значения для корня уравнения. Центральным местом в решении задачи, разумеется, будет программный код процедуры VBA, вычисляющей корень уравнения.

На заметку

Вообще, само собой напрашивается желание создать не процедуру, а функцию. Эта функция в качестве результата могла бы возвращать значение корня уравнения, и у нее могло бы быть, например, пять аргументов:

- ссылка на ячейку с аргументом функции;
- ссылка на ячейку со значением функции;
- два аргумента, определяющие значения для границ начального интервала поиска решения уравнения;
- аргумент, определяющий точность, с которой должен быть вычислен корень уравнения.

Тем не менее имеется одно обстоятельство, которое существенно влияет на наши радужные перспективы. Дело в том, что функции пользователя, созданные в VBA, не могут менять структуру рабочего документа. Поэтому из тела функции не получится менять значение во внешней ячейке. Процедура это делать может.

Прежде, чем приступить к обсуждению процедуры, необходимо определиться с тем, как именно она будет работать. О способе определения функциональной зависимости для уравнения мы уже упоминали. Однако кроме функциональной зависимости, необходимо еще определиться с интервалом поиска решения и критерием приемлемости найденного решения. Что касается границ интервала поиска корня уравнения, то разумно предположить, что эти значения будут содержаться в ячейках рабочего документа. Критерием приемлемости найденного решения будет погрешность вычисления корня уравнения, которую можно рассчитать на основе ширины интервала поиска корня. Условимся, что точность, с которой вычисляется корень, будет определяться внутренним параметром процедуры.

Итак, для выполнения процедуры необходимо иметь четыре ячейки с такими значениями:

- две ячейки с границами интервала поиска решения;
- одна ячейка с аргументом для функциональной зависимости, определяющей решаемое уравнение — в этой ячейке будет отображаться решение уравнения;

- ячейка с формулой, определяющей решаемое уравнение и вычисляемой на основе значения ячейки с аргументом.

Помимо этого, необходимо условиться о взаимном расположении этих ячеек. Примем следующие условия:

- ячейка с аргументом находится справа от ячейки с функциональной зависимостью, определяющей уравнение;
- значение для левой границы интервала поиска решения находится под ячейкой с формулой;
- значение для правой границы интервала поиска решения находится под ячейкой с аргументом для формулы, определяющей уравнение;
- на момент запуска процедуры активной должна быть ячейка с формулой, определяющей уравнение;
- вычисленное значение для корня уравнения отображается в ячейке для аргумента функциональной зависимости, определяющей уравнение.

На заметку

Фактически, нам нужна процедура с аргументами. Однако с практической точки зрения при работе с документом Excel удобнее пользоваться макросом — процедурой без аргументов. Поэтому мы аргументы процедуре передаем как бы неявно, через структуру ячеек документа, «отталкиваясь» от той ячейки, которая активна на момент запуска макроса.

Программный код процедуры для вычисления корня уравнения методом половинного деления представлен в листинге 12.1.

Листинг 12.1. Функция для вычисления корня уравнения методом половинного деления

```
Sub Дихотомия()
 ' Погрешность вычисления корня
 Dim epsilon As Double
 epsilon = 0.000001
 ' Запоминаем активную на момент запуска макроса ячейку
 Dim cellF As Range
 Set cellF = ActiveCell.Range("A1")
 ' Запоминаем ячейку справа от активной
 Dim cellX As Range
 Set cellX = cellF.Offset(0, 1)
 ' Переменные для начальных значений границ интервала поиска решения
 Dim a As Double, b As Double
 ' Считываем значения из ячеек рабочего листа
 a = cellF.Offset(1, 0).Value
 b = cellF.Offset(1, 1).Value
 ' Переменные для значений функции на границах интервала
 Dim Fa As Double, Fb As Double
 ' Переменная для запоминания начального значения
 ' ячейки с аргументом функции
 Dim start As Variant
```

```
' Запоминаем начальное значение ячейки с аргументом функции
start = cellX.FormulaLocal
' Меняем содержимое ячейки с аргументом на значение левой границы
cellX.Value = a
' Запоминаем значение функции на левой границе
Fa = cellF.Value
' Меняем содержимое ячейки с аргументом на значение правой границы
cellX.Value = b
' Запоминаем значение функции на правой границе
Fb = cellF.Value
' Если на границах интервала функция принимает
' значения одинаковых знаков
If Fa * Fb > 0 Then
  ' Значение ячейки-аргумента возвращается в исходное состояние
  cellX.FormulaLocal = start
  ' Отображается диалоговое окно с сообщением
  MsgBox "Указан неверный диапазон поиска корня!"
  ' Завершается работа процедуры
  Exit Sub
  ' Другие варианты
Else
  ' Если на левой границе у функции нулевое значение
  If Fa = 0 Then
    ' В ячейку-аргумент записывается значение левой границы
    cellX.Value = a
    ' Завершается работа процедуры
    Exit Sub
    ' Если на левой границе функция не обращается в ноль
  Else
    ' Если на правой границе у функции нулевое значение
    If Fb = 0 Then
      ' В ячейку-аргумент записывается значение правой границы
      cellX.Value = b
      ' Завершается работа процедуры
      Exit Sub
    End If
  End If
End If
' Переменные для запоминания центральной точки интервала и
' значения функции в этой точке
Dim x As Double, Fx As Double
' Оператор цикла - вычисляем корень уравнения
Do
  ' Точка в центре интервала поиска корня уравнения
  x = (a + b) / 2
  ' Записываем значение в ячейку с аргументом
```

```
    cellX.Value = x
    ' Запоминаем значение функции в центральной точке
    Fx = cellF.Value
    ' Если знаки функции на левой границе и в центре совпадают
    If Fx * Fa > 0 Then
      ' Левая граница интервала переносится в центр
       a = x
      ' Значение функции на левой границе
       Fa = Fx
      ' Другие варианты
    Else
      ' Если знаки функции на правой границе и в центре совпадают
      If Fx * Fb > 0 Then
        ' Правая граница переносится в центр
         b = x
        ' Значение функции на правой границе
         Fb = Fx
        ' Если значение функции в центральной точке интервала
        ' равно нулю
      Else
        ' Завершение работы процедуры
        Exit Sub
      End If
    End If
    ' Циклы выполняется до тех пор, пока половинная
    ' ширина интервала поиска не станет меньше погрешности
  Loop Until (Abs(b - a) / 2 < epsilon)
End Sub
```

Созданный нами макрос называется Дихотомия. Это процедура без аргументов. Алгоритм ее выполнения следующий. Вначале командой Dim epsilon As Double объявляется числовая переменная epsilon, значение которой определяет погрешность, с которой вычисляется точность. В данном случае мы используем значение epsilon = 0.000001. Далее, поскольку при выполнении макроса принципиально важно знать, какая ячейка на момент запуска макроса была активной, командой Set cellF = ActiveCell.Range("A1") ссылку на активную ячейку записываем в переменную cellF. Предварительно переменная cellF описана как такая, что имеет объектный тип Range (команда Dim cellF As Range). В активной ячейке содержится значение функции, определяющей уравнение. Кроме активной ячейки, запоминается ячейка справа от активной (переменной cellX присваивается значение cellF.Offset(0,1) — в этой ячейке записано значение аргумента функции). В числовые переменные a и b записываются значения ячеек: под активной (переменная a) и под ячейкой с аргументом (переменная b). Для считывания значений из указанных ячеек используем команды a = cellF.Offset(1,0).Value и b = cellF.Offset(1,1).Value. Таким образом, на начальном этапе переменные a и b содержат значения

границ интервала, на котором ищется решение уравнения. Чтобы определить значения функции на границах этого интервала, нам необходимо последовательно записать значение границ в ячейку с аргументом функции cellX и запомнить, какое при этом будет значение в активной ячейке (ячейка функции cellF). Значения функции на границах интервала записываются в переменные Fa и Fb.

На заметку

Напомним, что необходимым условием применимости метода половинного деления является различие знаков определяющей уравнение функции на границах интервала поиска решения. Чтобы вычислить эти значения и проверить выполнение необходимого условия, мы планируем в ячейку с аргументом функции cellX записывать значения границ интервала поиска решения. Если условие применимости метода выполнено, будет запущен процесс вычисления корня. А вот если функция на границах интервала поиска корня принимает значения одинаковых знаков, мы хотим вернуть все в исходное состояние, т. е. в ячейку cellX следует записать то значение, которое в ячейке было до начала выполнения макроса. Причем в ячейке могло быть не только число, но и, например, формула. Начальное значение ячейки мы записываем в переменную start, которая объявлена как такая, что имеет тип Variant. Значение присваивается командой start = cellX.FormulaLocal.

Командой cellX.Value = a меняем содержимое ячейки cellX на значение левой границы. Значение функции на левой границе записываем в переменную Fa с помощью команды Fa = cellF.Value. Аналогичная процедура проделывается для правой границы интервала поиска решения. После этого «в игру» вступает условный оператор (а точнее, несколько вложенных условных операторов). С его помощью проверяется необходимое условие для применения метода половинного деления. В частности, если функция на границах имеет значения одинаковых знаков, то произведение значений должно быть больше нуля (условие Fa*Fb>0). В этом случае командой cellX.FormulaLocal = start значение ячейки cellX возвращается в исходное состояние, отображается диалоговое окно с сообщением о том, что метод для данного диапазона неприменим (команда MsgBox “Указан неверный диапазон поиска корня!”), и завершается работа макроса (команда Exit Sub).

Помимо варианта, когда функция на границах интервала поиска корня принимает значения одинаковых знаков, возможны такие варианты: на одной или обеих границах у функции нулевое значение, или на границах интервала она принимает значения разных знаков. Что делать в последнем случае, мы знаем — используем метод половинного деления. Варианты с нулевыми границами отслеживаем отдельно. В частности, если на левой границе значение функции равно нулю (условие Fa = 0), в ячейку cellX в качестве результата записывается значение левой границы (команда cellX.Value = a), и работа макроса завершается. Если на левой границе значение у функции ненулевое, проверяется (на предмет равенства нулю) правая граница интервала. Если на правой границе значение функции нулевое (при ненулевом значении на левой границе), значение правой границы интервала становится решением уравнения и заносится в ячейку cellX.

Основная часть кода макроса выполняется в случае, если на границах интервала поиска решения функция, определяющая уравнение, принимает значения разных знаков. Нам при этом понадобится еще несколько числовых переменных: x (для запоминания текущего приближения для корня уравнения) и Fx (для запоминания значения функции уравнения в этой точке).

Вычисление приближенного значения для корня уравнения выполняется с помощью оператора цикла Do-Until. В теле оператора цикла (который выполняется до тех пор, пока не станет истинным условие Abs(b-a)/2<epsilon) переменная x получает значение центральной точки интервала (команда x = (a+b)/2), после чего это значение записывается в ячейку cellX (команда cellX.Value = x). На следующем этапе командой Fx = cellF.Value в переменную Fx записывается значение функции в центральной точке интервала. С помощью условного оператора проверяется условие Fx*Fa>0, которое означает, что на левой границе и в центре интервала функция уравнения принимает ненулевые значения одинаковых знаков. В этом случае левая граница переносится в центр интервала (команда a = x), причем новое значение Fx получает и переменная Fa, поскольку она должна содержать значение функции уравнения на левой границе, которая предыдущей командой изменилась.

Если знаки функции на правой границе и в центре совпадают (условие Fx*Fb>0), аналогичные действия выполняются по отношению к правой границе интервала поиска решения.

Возможен также вариант, когда оба указанных условия не выполняются. Это означает, что в центре интервала поиска решения функция принимает нулевое значение. Следовательно, решение найдено, и выполнение макроса нужно прекратить.

На заметку

Теоретически, если не выполнено ни одно из условий Fx*Fa>0 и Fx*Fb>0, то это может означать, что, по крайней мере, одно из трех значений Fa, Fb или Fx равно нулю. Однако соответствующая ветка кода выполняется, только если значения Fa и Fb не равны нулю. Поэтому нулю может быть равно только значение Fx.

Проиллюстрируем использование созданного макроса. На рисунке 12.12 показан документ перед началом вычислений.

В ячейку G9 вводится формула =H9^3-8*H9^2+H9+42, которая и определяет решаемое уравнение. Значение в этой ячейке вычисляется на основании значения ячейки H9. В начале вычислений в этой ячейке нулевое значение. В ячейки G10 и H10 вводятся значения для границ интервала поиска решения (-20 и 1 соответственно). Перед запуском макроса на выполнение активной должна быть ячейка G9.

На заметку

Понятно, что предварительно в рабочем документе с помощью редактора VBA должен быть создан модуль с процедурой Дихотомия, программный код которой приведен выше. Что касается запуска макроса, то это можно делать стандартными средствами (например, через вкладку **Разработчик**). Другие варианты — создать кнопку на панели быстрого запуска или разместить в рабочем документе кнопку для запуска макроса. В общем, вариантов достаточно много.

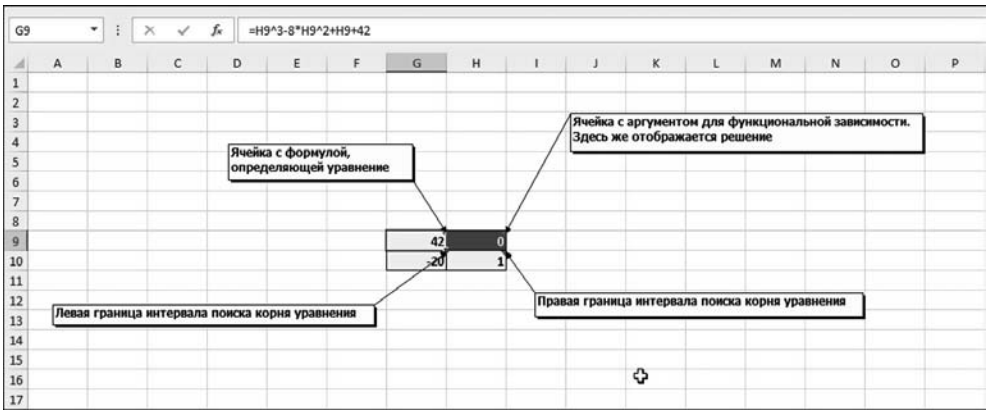


Рис. 12.12
Документ Excel перед началом выполнения макроса Дихотомия

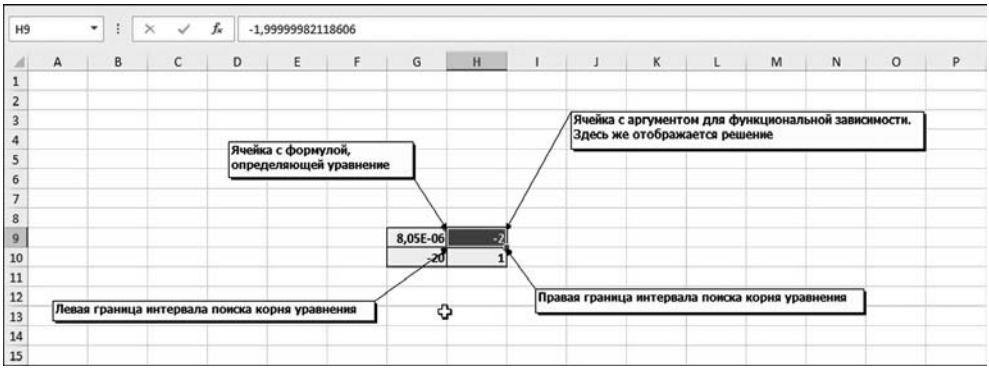


Рис. 12.13
Результат вычисления корня уравнения с помощью метода Дихотомия

Рис. 12.14
Границы диапазона поиска корня указаны некорректно

После запуска макроса получаем результат, как на рисунке 12.13.

В данном случае найден корень $x_1 = -2$. Если в качестве границ интервала поиска решения указать иные значения, можем найти другой корень.

На заметку

В ячейке H9 отображается значение -2, на самом деле значение вычислено несколько иное (в ячейке отображается округленное значение). То значение, которое вычислено, можно увидеть в строке формул, если выделить ячейку H9 (как показано на рисунке 12.13).

На рисунке 12.14 показана ситуация, когда перед запуском макроса границы диапазона поиска корня указаны некорректно — на границах диапазона функция, определяющая уравнение, принимает значения одинаковых знаков.

После запуска макроса на выполнение получаем сообщение, как на рисунке 12.15.

Желающие могут поэкспериментировать на предмет того, что произойдет, если на одной или обеих границах диапазона поиска корня уравнения функция, определяющая уравнение, будет равняться нулю.

Прежде, чем перейти к рассмотрению других методов и алгоритмов решения алгебраических уравнений, рассмотрим еще один концептуальный подход по реализации метода половинного деления средствами VBA. В данном случае мы создаем две функции: одна определяет решаемое уравнение (функция уравнения), а вторая предназначена непосредственно для реализации метода половинного деления. Суть подхода в том, что функция, определяющая решаемое уравнение, передается в качестве аргумента функции, которая «решает» уравнение. При этом функция для решения уравнения создается в отдельном модуле, а функция, определяющая уравнение — в модуле класса.

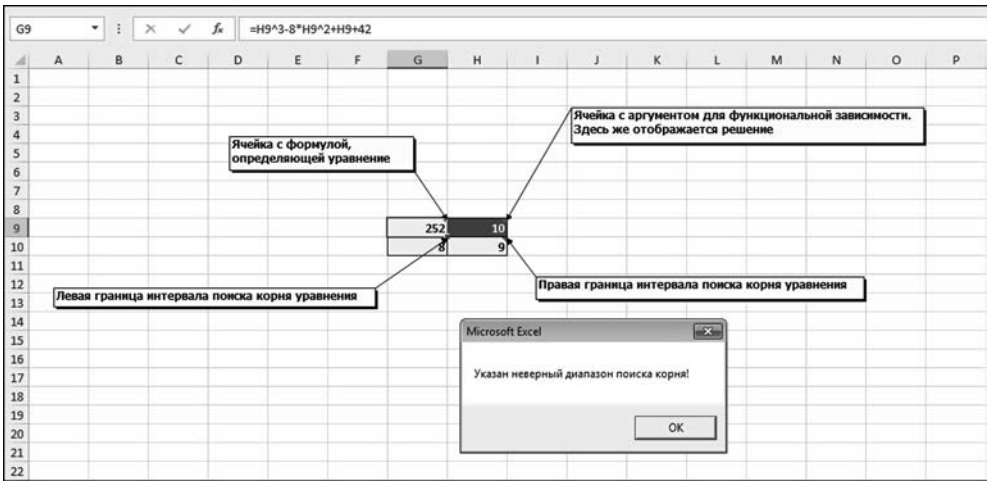


Рис. 12.15
Диалоговое окно с сообщением о том, что границы указаны некорректно, отображается в результате выполнения макроса

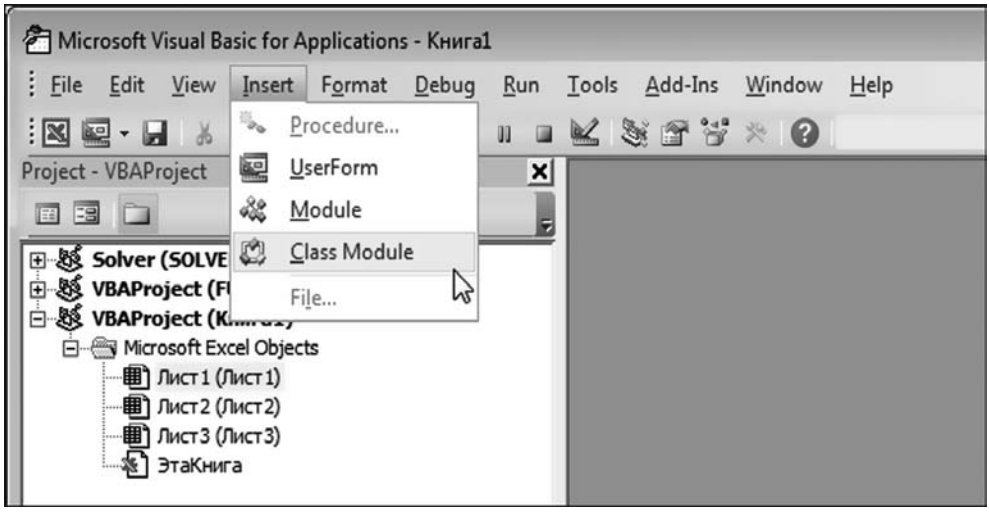


Рис. 12.16
Создание модуля класса

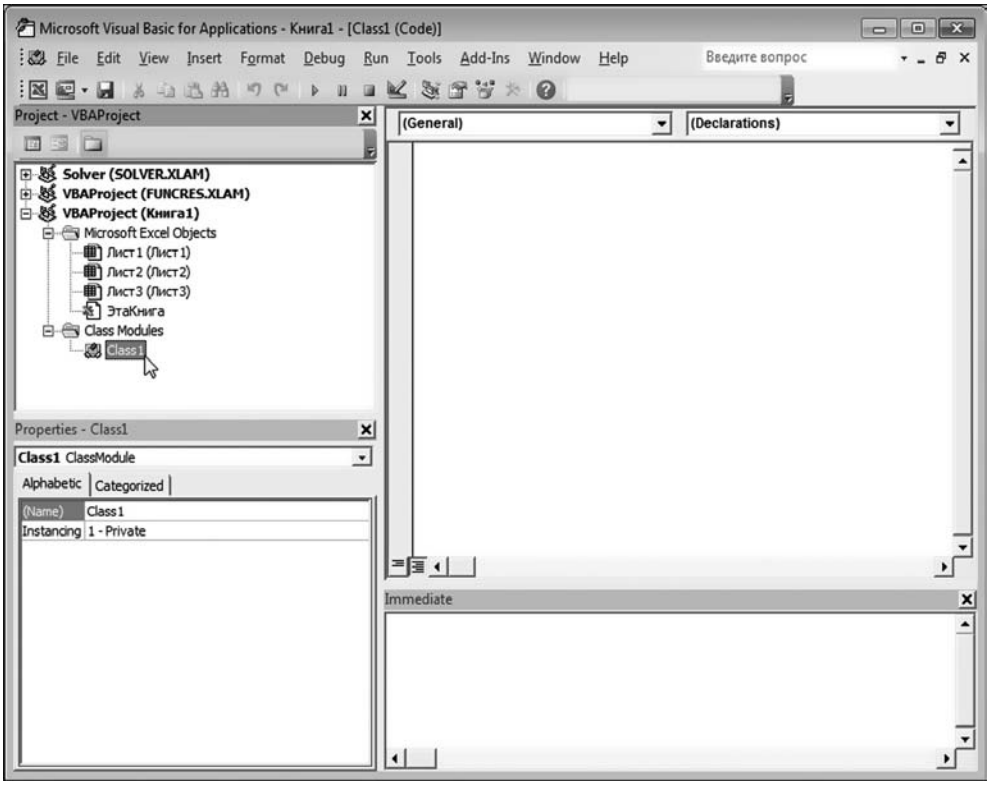


Рис. 12.17
В проекте создан модуль класса с названием по умолчанию Class1

На заметку

Чтобы создать модуль класса, в окне редактора VBA в меню **Insert** выбираем команду **Class Module** (рис. 12.16).

В результате в проект добавляется модуль класса, который отображается в окне проекта **Project** в группе **Class Modules** и по умолчанию имеет название **Class1** (рис. 12.17).

После двойного щелчка на элементе модуля класса в окне **Project** открывается окно для ввода и редактирования кода модуля класса. В этом окне вводится программный код функции, определяющей уравнение (обсуждается далее). Кроме того, название модуля класса можно изменить. Делается это в поле **Name** в окне свойств **Properties**. На рисунке 12.18 показана ситуация, когда название по умолчанию модуля класса Class1 изменено на Equations, в рабочем документе оставлен один рабочий лист (листы добавляются/удаляются в рабочем окне приложения Excel), а в окно кода модуля класса введен программный код функции уравнения eqn().

Для добавления обычного модуля в меню **Insert** редактора VBA выбираем команду **Module**, как это показано на рисунке 12.19.

В результате в проект добавляется модуль (по умолчанию название Module1), в который мы непосредственно и добавляем программный код функции, предназначенной для решения уравнений (рис. 12.20). Специфика программных кодов обеих упомянутых выше функций обсуждается далее, равно как и причины, по которым нам понадобилось создавать модуль класса.

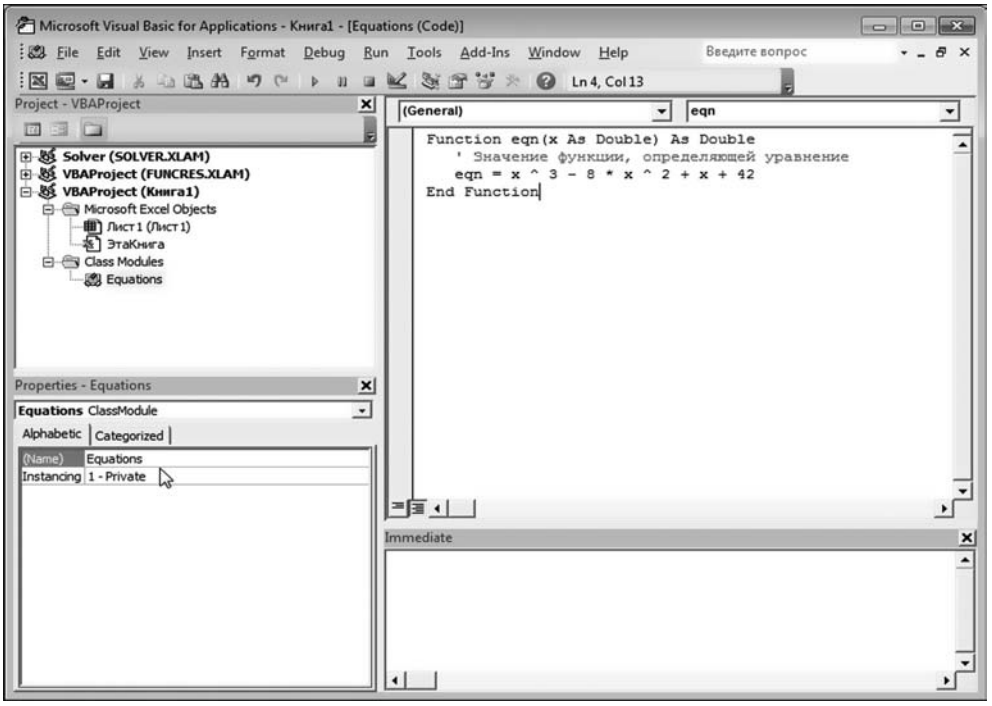


Рис. 12.18
Изменение названия модуля класса (на Equations)
и добавление в модуль класса функции eqn(), определяющей уравнение

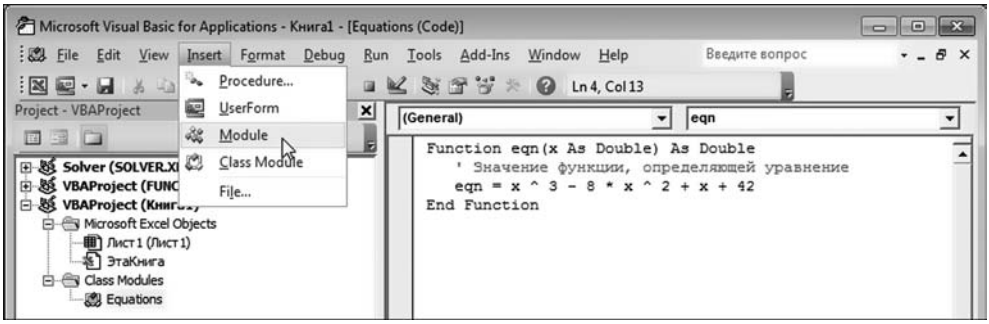


Рис. 12.19
Добавление в проект обычного модуля

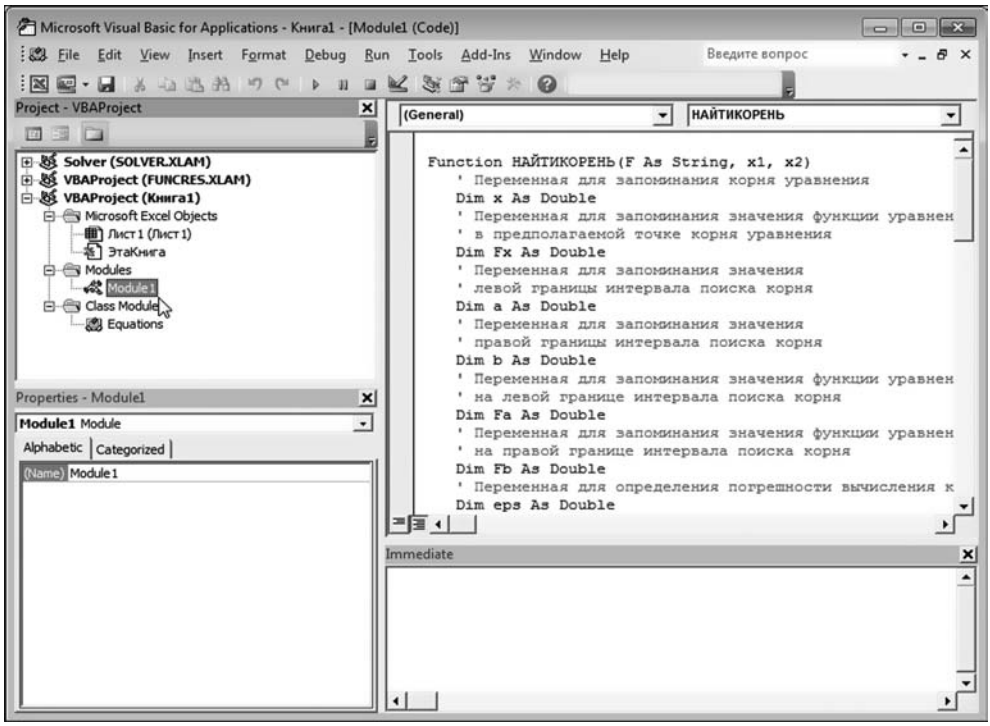


Рис. 12.20
Добавление в модуль программного кода функции
для решения уравнений методом половинного деления

Функцию для решения уравнений мы собираемся использовать в рабочем документе Excel. В связи с этим функция размещается в модуле (обычном, не модуле класса). Название у функции будет «кириллическое», чтобы она не очень уж выделялась на фоне прочих функций русифицированной версии Excel. В частности, функция будет называться НАЙТИКОРЕНЬ, и у нее будет три параметра (или аргумента):

- «ссылка» на функцию, определяющую решаемое уравнение;
- левая граница диапазона поиска корня;
- правая граница диапазона поиска корня.

Подход, основанный на передаче аргументом функции, определяющей уравнение, весьма удобен, поскольку теоретически позволяет использовать одну и ту же функцию для решения разных уравнений — достаточно лишь при вызове поменять аргумент. Проблема, однако, в том, что такого понятия, как «ссылка на функцию» в VBA нет. Приходится идти окольными путями. В частности, передавать аргументом мы будем имя функции, которую следует использовать при решении уравнения. Поэтому технически первый аргумент функции НАЙТИКОРЕНЬ() — это текстовое значение. Что касается второго и третьего аргументов, то можно было бы ожидать, что это числовые значения типа Double. Тем не менее мы хотим, чтобы в качестве соответствующих аргументов можно было использовать не только числовые литералы, но и ссылки на ячейки рабочего листа, поэтому тип этих аргументов явно не указываем (как результат, они будут относиться к типу Variant).

В качестве результата функция НАЙТИКОРЕНЬ() должна возвращать корень уравнения. Но это в случае, если метод половинного деления применим. Если метод неприменим, функция будет возвращать значение ошибки #ЗНАЧ!. Алгоритм выполнения функции достаточно простой: сначала проверяются «особые» случаи, вроде равенства нулю функции на границе интервала поиска корня и условие применимости метода половинного деления. Если предпосылки для применения метода есть, производится вычисление корня уравнения. При этом используется *рекурсия*: в теле функции НАЙТИКОРЕНЬ() вызывается эта же функция НАЙТИКОРЕНЬ(), но с измененными аргументами (уменьшен вдвое диапазон поиска корня). Программный код функции НАЙТИКОРЕНЬ() представлен в листинге 12.2.

Листинг 12.2. Функция для решения уравнений методом половинного деления

```
Function НАЙТИКОРЕНЬ(F As String, x1, x2)
 ‘ Переменная для запоминания корня уравнения
 Dim x As Double
 ‘ Переменная для запоминания значения функции уравнения
 ‘ в предполагаемой точке корня уравнения
 Dim Fx As Double
 ‘ Переменная для запоминания значения
 ‘ левой границы интервала поиска корня
 Dim a As Double
 ‘ Переменная для запоминания значения
 ‘ правой границы интервала поиска корня
 Dim b As Double
 ‘ Переменная для запоминания значения функции уравнения
 ‘ на левой границе интервала поиска корня
 Dim Fa As Double
 ‘ Переменная для запоминания значения функции уравнения
```

```
‘ на правой границе интервала поиска корня
Dim Fb As Double
‘ Переменная для определения погрешности вычисления корня
Dim eps As Double
‘ Значение погрешности вычисления корня
eps = 0.000001
‘ Значение левой границы интервала поиска корня уравнения
 a = x1
‘ Значение правой границы интервала поиска корня уравнения
 b = x2
‘ Объектная переменная класса Equations
Dim m As Equations
‘ Создаем объект класса Equations
Set m = New Equations
‘ Вычисляем значение функции уравнения на левой границе
Fa = CallByName(m, F, VbMethod, a)
‘ Если на левой границе функция уравнения принимает нулевое значение
If Fa = 0 Then
  ‘ Значение левой границы - корень уравнения
  НАЙТИКОРЕНЬ = a
  ‘ Завершение выполнения функции
  Exit Function
End If
‘ Вычисление значения функции на правой границе
Fb = CallByName(m, F, VbMethod, b)
‘ Если на правой границе функция уравнения равна нулю
If Fb = 0 Then
  ‘ Значение правой границы - корень уравнения
  НАЙТИКОРЕНЬ = b
  ‘ Завершение выполнения функции
  Exit Function
End If
‘ Если на границах интервала поиска корня функция уравнения
‘ принимает значения одинаковых знаков
If Fa * Fb > 0 Then
  ‘ Функция возвращает ошибку
  НАЙТИКОРЕНЬ = CVErr(xlErrValue)
  ‘ Завершение выполнения функции
  Exit Function
End If
‘ Вычисляем центральную точку интервала поиска корня
x = (a + b) / 2
‘ Вычисляем значение функции в центральной точке интервала
‘ поиска корня
Fx = CallByName(m, F, VbMethod, x)
‘ Если корень локализован с достаточной точностью или
```

```
    ' функция уравнения в центральной точке интервала равна нулю
    If (Abs(b - a) < 2 * eps) Or (Fx = 0) Then
      ' Центральная точка интервала поиска корня - решение уравнения
      НАЙТИКОРЕНЬ = x
      ' Если корень не найден
    Else
      ' Если в центре интервала и на левой границе функция уравнения
      ' принимает значения одинаковых знаков
      If Fa * Fx > 0 Then
        ' Рекурсивный вызов функции поиска корня уравнения.
        ' Левая граница интервала поиска корня смещена в центр
        НАЙТИКОРЕНЬ = НАЙТИКОРЕНЬ(F, x, b)
        ' Если в центре интервала и на правой границе функция уравнения
        ' принимает значения одинаковых знаков
      Else
        ' Рекурсивный вызов функции поиска корня уравнения.
        ' Правая граница интервала поиска корня смещена в центр
        НАЙТИКОРЕНЬ = НАЙТИКОРЕНЬ(F, a, x)
      End If
    End If
End Function
```

Как уже отмечалось, у функции НАЙТИКОРЕНЬ() три аргумента. Через F обозначено имя функции, которая определяет решаемое уравнение. Предполагается, что у этой функции один аргумент и в качестве значения она возвращает число. Аргументы x1 и x2 обозначают границы поиска решения. Это могут быть как числовые значения, так и адреса ячеек. Ячейки, в свою очередь, должны содержать числовые значения. Хотя идеологически разница не очень большая, числа и ссылки в программном коде обрабатываются по-разному. Чтобы избежать неоднозначности, мы в программном коде функции вводим две переменные a и b типа Double и в качестве значений им присваиваем x1 и x2. Такой прием срабатывает благодаря тому, что для ссылок на ячейки имеется «свойство по умолчанию»: значение ячейки. Если формально числовой переменной присваивается ссылка на ячейку, то на самом деле будет использовано свойство Value соответствующей ячейки, т. е. ее значение. В дальнейшем мы будем «оперировать» именно переменными a и b.

На заметку

Как и в предыдущем примере, точность вычисления корня уравнения определяется локальной переменной. В данном случае это переменная eps.

Для «внутренних» расчетов мы планируем вызывать функцию уравнения, имя которой передается через аргумент функции НАЙТИКОРЕНЬ(). С этой целью воспользуемся встроенной функцией VBA, которая называется CallByName и позволяет делать именно то, что мы собираемся делать — вызвать функцию, если известно ее имя. Но для этого вызываемая функция должна быть методом некоторого объекта. Чтобы создать объект, мы сначала

создаем класс (на основе которого потом создаем объект). Технически создание класса состоит в том, что мы добавляем модуль класса и меняем его название на Equations. Как это делается, показано выше. В программном коде функции НАЙТИКОРЕНЬ() командой Dim m As Equations объявляется объектная переменная класса Equations. Для создания объекта класса Equations мы используем команду New Equations. Результат этой команды (ссылка на созданный объект) присваивается в качестве значения переменной m. Все вместе выглядит как Set m = New Equations. Созданный таким образом объект будет использован нами для передачи первым аргументом функции CallByName().

На заметку

Функции CallByName() передаются такие аргументы:
- ссылка на объект, из которого вызывается метод;
- имя вызываемого метода (функции);
- константа, определяющая режим вызова метода;
- аргументы для передачи методу (если есть).

Для вычисления значения функции уравнения на левой границе мы используем команду Fa = CallByName(m,F,VbMethod,a). Команда означает, что в переменную Fa записывается результат вызова функции/метода с именем F, который вызывается из объекта m с аргументом a. Константа VbMethod, переданная третьим аргументом функции CallByName(), определяет режим вызова метода.

На заметку

В общем случае могут использоваться такие константы: VbGet (вызывается Get-процедура свойства), VbSet (вызывается Set-процедура свойства), VbLet (вызывается Let-процедура свойства) и VbMethod (вызывается метод).

Сразу после того, как вычислено значение функции уравнения на левой границе, проверяется, равна или нет функция уравнения нулю на левой границе. Если на левой границе функция уравнения принимает нулевое значение (условие Fa = 0), то значение левой границы — корень уравнения. Поэтому выполняются команды НАЙТИКОРЕНЬ = a (значение, возвращаемое функцией) и Exit Function (завершение выполнения функции). В случае, когда на левой границе функция уравнения принимает ненулевое значение, командой Fb = CallByName(m,F,VbMethod,b) вычисляется значение функции уравнения на правой границе, и выполняется проверка, аналогичная той, что описана выше для левой границы.

В случае, когда на обеих границах функция уравнения принимает ненулевые значения, проверяется условие применимости (точнее, условие неприменимости) метода половинного деления. Если знаки значений функции уравнения на границах интервала поиска корня одинаковые (условие Fa*Fb>0), в качестве результата функцией НАЙТИКОРЕНЬ() возвращается ошибка. Соответствующая команда выглядит как НАЙТИКОРЕНЬ = CVErr(xlErrValue). Здесь для «генерирования» ошибки использована встроенная функция VBA CVErr(), аргументом которой передается константа xlErrValue, соответствующая ошибке #ЗНАЧ!.

На заметку

Можно использовать такие константы для «обозначения» типа ошибки: xlErrNA (ошибка #Н/Д), xlErrRef (ошибка #ССЫЛКА!), xlErrDiv0 (ошибка #ДЕЛ/0!), xlErrValue (ошибка #ЗНАЧ!), xlErrName (ошибка #ИМЯ?), xlErrNum (ошибка #ЧИСЛО!) и xlErrNull (ошибка #ПУСТО!).

В том случае, когда все «экзотические» ситуации рассмотрены, и выяснилось, что метод половинного деления применим, командой x = (a+b)/2 определяем центральную точку интервала поиска решения уравнения и вычисляем значение функции уравнения в этой центральной точке командой Fx = CallByName(m,F,VbMethod,x). Далее возможно несколько вариантов. Во-первых, теоретически возможна ситуация, когда ширина интервала поиска корня достаточно мала для того, чтобы локализовать корень уравнения с необходимой точностью. Соответствующее условие можно записать как Abs(b-a)<2*eps. Во-вторых, выбрав центральную точку интервала, мы могли «случайно» угадать корень уравнения (при том, что интервал поиска корня достаточно большой). Условие равенства нулю функции уравнения в центральной точке интервала записывается как Fx = 0. В обоих случаях имеет смысл завершить выполнение функции НАЙТИКОРЕНЬ() со значением переменной x в качестве результата. Общее условие может быть записано как (Abs(b-a)<2*eps) Or (Fx = 0). В этом выражении мы использовали логический оператор Or, а результатом выражения является значение True, если истинно хотя бы одно из двух условий. Если ни одно из условий не выполняется, следует продолжить половинное деление интервала поиска корня уравнения. Здесь также возможны два варианта:

- если значение функции уравнения в центре совпадает со значением функции уравнения на левой границе, левая граница сдвигается в центр, и по отношению к этому новому интервалу применяется процедура половинного деления — команда НАЙТИКОРЕНЬ = НАЙТИКОРЕНЬ(F,x,b) выполняется, если истинно условие Fa*Fx>0;
- если значение функции уравнения в центре совпадает со значением функции уравнения на правой границе, правая граница сдвигается в центр, и по отношению к этому новому интервалу применяется процедура половинного деления — команда НАЙТИКОРЕНЬ = НАЙТИКОРЕНЬ(F,a,x) выполняется, если ложно условие Fa*Fx>0 (а, значит, с учетом всего вышерассмотренного, истинно условие Fb*Fx>0).

Таким образом, мы воспользовались рекурсивным вызовом функции, что достаточно точно соответствует общему алгоритму реализации метода половинного деления.

На заметку

Напомним, что рекурсия подразумевает вызов в теле функции этой же функции. Такой подход удобен в случае, если общий алгоритм, реализуемый в функции, задан рекурсивным образом, т. е. когда после некоторых вычислений задача сводится к такой, что аналогична исходной, но с некоторыми измененными начальными параметрами. В методе половинного деления мы имеем дело с чем-то подобным: на каждом шаге интервал поиска корня уменьшается в два раза, и при этом задача «сводится к предыдущей».

Чтобы понять, как выполняется программный код функции с рекурсией, следует учесть, что каждый раз, когда при выполнении кода встречается инструкция вызова функции, код этой функции загружается в память и начинает выполняться (с теми аргументами, что переданы функции при вызове). При выполнении этого кода снова загружается код функции (в месте рекурсивного вызова) и т. д. Понятно, что код должен быть организован так, чтобы в какой-то момент этот рекурсивный вызов прекратился, иначе получим бесконечную рекурсивную ссылку. В нашем случае условиями выхода из рекурсивного вызова являются условия малости интервала поиска корня или равенство нулю функции уравнения в центральной точке интервала.
Обычно программные коды с рекурсивным вызовом функций выглядят достаточно элегантно, но с точки зрения скорости вычислений они не самые оптимальные.

В листинге 12.3 приведен программный код функции eqn(), которая описана в модуле класса Equations и определяет решаемое уравнение (речь, как и ранее, идет об уравнении $x^3 - 8x^2 + x + 42 = 0$).

Листинг 12.3. Функция для определения решаемого уравнения

```
Function eqn(x As Double) As Double
  ' Значение функции, определяющей уравнение
  eqn = x ^ 3 - 8 * x ^ 2 + x + 42
End Function
```

В документе на рисунке 12.21 приведены примеры вычислений, выполненных с помощью разработанных выше функций.

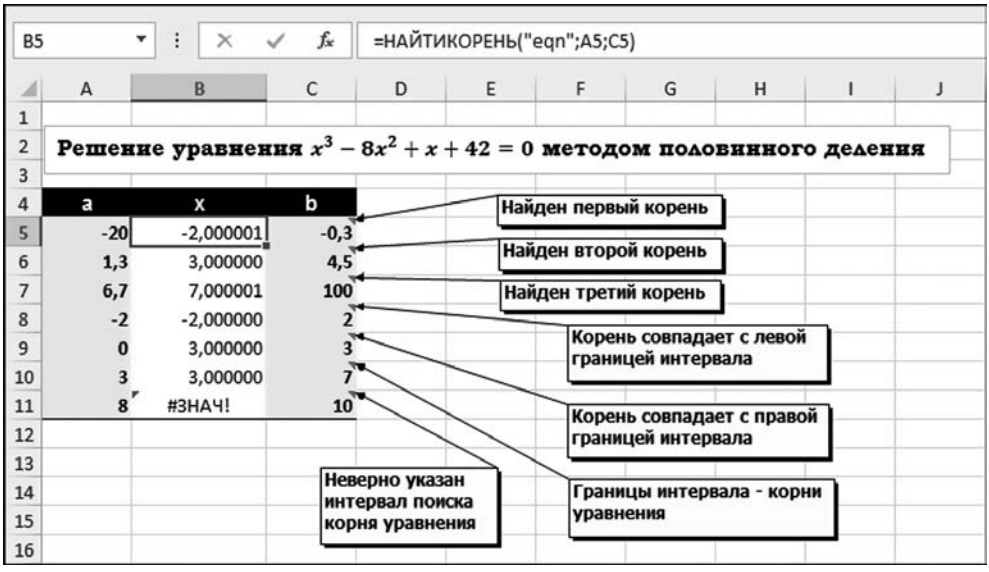


Рис. 12.21
Результат вычисления корня уравнения методом половинного деления для разных интервалов локализации корня

В представленном документе диапазон ячеек A5:A11 содержит значения левой границы интервала поиска решения, а диапазон ячеек C5:C11 содержит значения правой границы интервала поиска решения. В ячейках B5:B11 с помощью созданной нами функции НАЙТИКОРЕНЬ() вычисляется корень уравнения $x^3 - 8x^2 + x + 42 = 0$ на соответствующем интервале (если он там есть). Для вычисления корней уравнения в ячейку B5 вводится формула =НАЙТИКОРЕНЬ("eqn";A5;C5), после чего она копируется в прочие ячейки диапазона B5:B11. Нами вычислены все три корня уравнения, рассмотрены случаи, когда одна или обе границы интервала поиска корня совпадают с решением уравнения, а также рассмотрен случай некорректно указанного интервала.

На заметку

Если в какой-то момент мы захотим решить методом половинного деления другое уравнение, мы сможем воспользоваться созданной выше функцией НАЙТИКОРЕНЬ(). Достаточно будет в модуле класса Equations описать функцию для этого нового уравнения и указать ее имя первым аргументом при вызове функции НАЙТИКОРЕНЬ().

МЕТОД ХОРД

А хотя бы я и жадничаю.
Зато от чистого сердца!

Из м/ф «Падал прошлогодний снег»

Метод хорд во многом похож на метод половинного деления. Принципиальное отличие состоит в том, как выбирается очередное приближение для корня уравнения. Так, если в методе половинного деления «проверяемой» является центральная точка интервала поиска корня, то в методе хорд эта точка определяется как пересечение оси абсцисс с прямой, соединяющей граничные точки на графике функции $f(x)$. В частности, если ищется корень уравнения $f(x) = 0$ на интервале от a до b (при условии, что функция $f(x)$ принимает на границе интервала значения разных знаков), то вместо точки в центре интервала вычисляется значение функции в точке $x = \frac{af(b) - bf(a)}{f(b) - f(a)}$. Далее в эту точку переносится одна из границ интервала поиска корня, а именно та, на которой знак функции совпадает со знаком функции в точке $x = \frac{af(b) - bf(a)}{f(b) - f(a)}$. На рисунке 12.22 схематически представлен процесс поиска корня уравнения методом хорд на интервале, на котором функция, определяющая уравнение, принимает на границах значения разных знаков.

На заметку

По сравнению с методом половинного деления, метод хорд обычно обеспечивает более быструю сходимость, т. е. для вычисления корня уравнения приходится выполнять меньшее количество итераций, чем при использовании метода половинного деления. Вместе с тем, погрешность вычисления корня в методе хорд определить значительно сложнее. Другими словами, если задача

состоит в том, чтобы вычислить корень с определенной точностью, то связать оценку точности вычисления корня с количеством итераций в методе хорд достаточно сложно. Для погрешности решения $\varepsilon = |x - x_0|$ (здесь x и x_0 обозначают приближенное и точное решения уравнения) существует оценка $\varepsilon \le |f(x)|/M$, где через M обозначено минимальное значение модуля производной $f'(x)$ на интервале, на котором локализован корень уравнения. Это соотношение можно использовать на практике, для чего, правда, придется сделать оценку для производной от функции уравнения.

Сначала рассмотрим, как для решения алгебраического уравнения метод хорд может быть реализован в рабочем документе Excel. Для определенности будем решать уже знакомое нам уравнение $x^3 - 8x^2 + x + 42 = 0$. Рабочий документ Excel на начальной стадии вычислений представлен на рисунке 12.23.

Этот документ очень сильно напоминает тот, что рассматривался при решении этого же уравнения методом половинного деления. Фактически, по сравнению с документом, который создавался для реализации метода половинного деления, здесь поменялись формулы в ячейках D5 и D6, которые определяют приближенные значения для корня уравнения соответственно

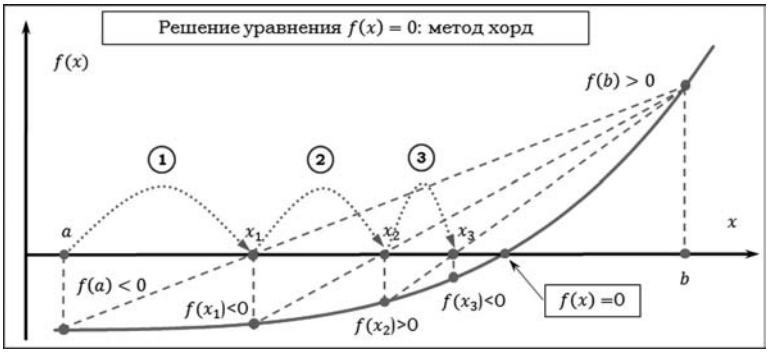


Рис. 12.22
Метод хорд:

цифрами и штрихованными линиями со стрелкой показана последовательность перемещения границ интервала, на котором ищется решение.

D5 | =(B5*F5-C5*E5)/(F5-E5)

Решение уравнения $x^3 - 8x^2 + x + 42 = 0$ методом хорд

№	a	b	x	f(a)	f(b)	f(x)
0	-10	0	-0,232044199	-1768	42	41,32470541
1	-10	-0,23204	-0,455142847	-1768	41,32470541	39,79333194

Рис. 12.23
Рабочий документ Excel перед началом вычислений
для поиска корня уравнения по методу хорд

на нулевой (начало вычислений) и первой итерациях. А именно, в ячейку D5 вводится формула =(B5*F5-C5*E5)/(F5-E5), после чего она (формула) копируется в ячейку D6. Полный перечень вводимых в рабочий документ формул и числовых значений представлен в таблице 12.2. Большинство из них читателю должно быть уже знакомо по предыдущему разделу.

Что касается «базовой» формулы =(B5*F5-C5*E5)/(F5-E5), то она является реализацией выражения $x = \dfrac{af(b) - bf(a)}{f(b) - f(a)}$, только «на языке» адресов ячеек. При этом ячейка F5 содержит значение функции $f(b)$ на правой границе, ячейка E5 содержит значение функции $f(a)$ на левой границе, а значения левой a и правой b границ содержатся в ячейках B5 и C5 соответственно.

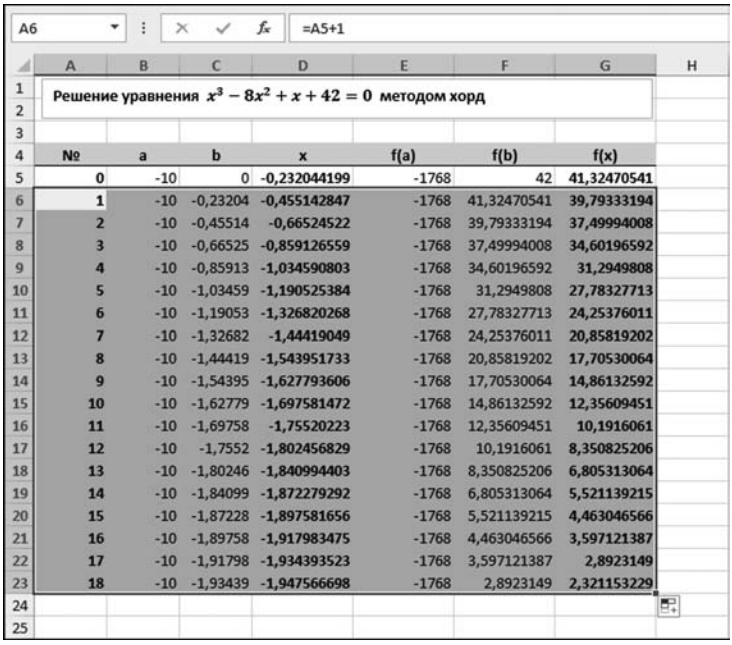
Для выполнения вычислений выделяем ячейки A6:G6 и с помощью маркера автоматического заполнения заполняем ячейки внизу, как это показано на рисунке 12.24.

Искомое приближение для корня уравнения вычисляется в столбце D. Как и в случае проведения вычислений методом половинного деления, для того, чтобы изменить границы интервала, на котором ищется корень, достаточно

Таблица 12.2

Метод хорд

Ячейка	Значение	Комментарий
A5	0	Номер начальной итерации
A6	=A5+1	Вычисления номера следующей итерации
B5	-10	Начальное значение для левой границы интервала поиска корня уравнения
B6	=ЕСЛИ(E5*G5<=0;B5;D5)	Вычисление значения левой границы интервала
C5	0	Начальное значение для правой границы интервала поиска корня уравнения
C6	=ЕСЛИ(F5*G5<=0;C5;D5)	Вычисление значения правой границы интервала
D5	=(B5*F5-C5*E5)/(F5-E5)	Приближение для корня на нулевой итерации
D6	=(B6*F6-C6*E6)/(F6-E6)	Приближение для корня на первой итерации. Формула копируется из ячейки D5
E5	=B5^3-8*B5^2+B5+42	Значение функции на левой границе интервала для нулевой итерации
E6	=B6^3-8*B6^2+B6+42	Значение функции на левой границе интервала для первой итерации. Формула копируется из ячейки E5
F5	=C5^3-8*C5^2+C5+42	Значение функции на правой границе интервала для нулевой итерации. Формула копируется из ячейки E5
F6	=C6^3-8*C6^2+C6+42	Значение функции на правой границе интервала для первой итерации. Формула копируется из ячейки E6 или F5
G5	=D5^3-8*D5^2+D5+42	Значение функции в точке приближенного решения для корня уравнения (нулевая итерация). Формула копируется из ячейки F5
G6	=D6^3-8*D6^2+D6+42	Значение функции в точке приближенного решения для корня уравнения (первая итерация). Формула копируется из ячейки F6 или G5

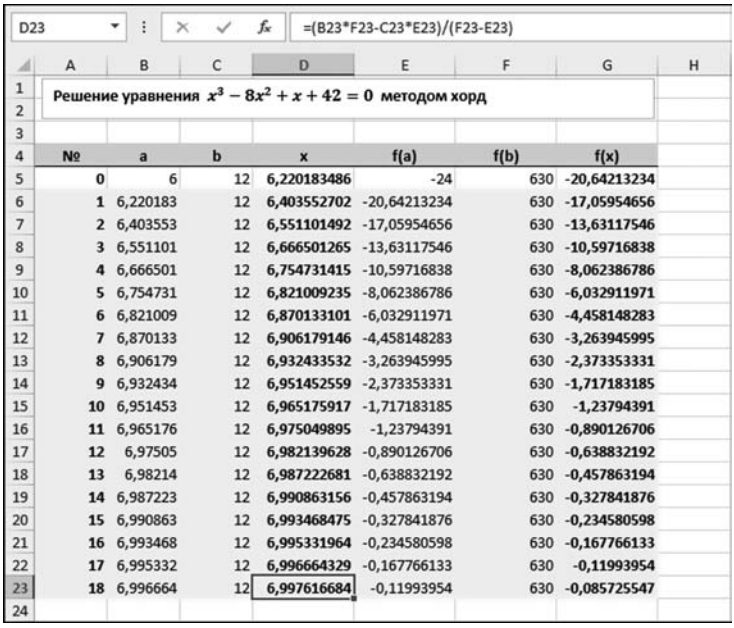


A6 =A5+1

Решение уравнения $x^3 - 8x^2 + x + 42 = 0$ методом хорд

№	a	b	x	f(a)	f(b)	f(x)
0	-10	0	-0,232044199	-1768	42	41,32470541
1	-10	-0,23204	-0,455142847	-1768	41,32470541	39,79333194
2	-10	-0,45514	-0,66524522	-1768	39,79333194	37,49994008
3	-10	-0,66525	-0,859126559	-1768	37,49994008	34,60196592
4	-10	-0,85913	-1,034590803	-1768	34,60196592	31,2949808
5	-10	-1,03459	-1,190525384	-1768	31,2949808	27,78327713
6	-10	-1,19053	-1,326820268	-1768	27,78327713	24,25376011
7	-10	-1,32682	-1,44419049	-1768	24,25376011	20,85819202
8	-10	-1,44419	-1,543951733	-1768	20,85819202	17,70530064
9	-10	-1,54395	-1,627793606	-1768	17,70530064	14,86132592
10	-10	-1,62779	-1,697581472	-1768	14,86132592	12,35609451
11	-10	-1,69758	-1,75520223	-1768	12,35609451	10,1916061
12	-10	-1,7552	-1,802456829	-1768	10,1916061	8,350825206
13	-10	-1,80246	-1,840994403	-1768	8,350825206	6,805313064
14	-10	-1,84099	-1,872279292	-1768	6,805313064	5,521139215
15	-10	-1,87228	-1,897581656	-1768	5,521139215	4,463046566
16	-10	-1,89758	-1,917983475	-1768	4,463046566	3,597121387
17	-10	-1,91798	-1,934393523	-1768	3,597121387	2,8923149
18	-10	-1,93439	-1,947566698	-1768	2,8923149	2,321153229

Рис. 12.24
Результат решения уравнения методом хорд



D23 =(B23*F23-C23*E23)/(F23-E23)

Решение уравнения $x^3 - 8x^2 + x + 42 = 0$ методом хорд

№	a	b	x	f(a)	f(b)	f(x)
0	6	12	6,220183486	-24	630	-20,64213234
1	6,220183	12	6,403552702	-20,64213234	630	-17,05954656
2	6,403553	12	6,551101492	-17,05954656	630	-13,63117546
3	6,551101	12	6,666501265	-13,63117546	630	-10,59716838
4	6,666501	12	6,754731415	-10,59716838	630	-8,062386786
5	6,754731	12	6,821009235	-8,062386786	630	-6,032911971
6	6,821009	12	6,870133101	-6,032911971	630	-4,458148283
7	6,870133	12	6,906179146	-4,458148283	630	-3,263945995
8	6,906179	12	6,932433532	-3,263945995	630	-2,373353331
9	6,932434	12	6,951452559	-2,373353331	630	-1,717183185
10	6,951453	12	6,965175917	-1,717183185	630	-1,23794391
11	6,965176	12	6,975049895	-1,23794391	630	-0,890126706
12	6,97505	12	6,982139628	-0,890126706	630	-0,638832192
13	6,98214	12	6,987222681	-0,638832192	630	-0,457863194
14	6,987223	12	6,990863156	-0,457863194	630	-0,327841876
15	6,990863	12	6,993468475	-0,327841876	630	-0,234580598
16	6,993468	12	6,995331964	-0,234580598	630	-0,167766133
17	6,995332	12	6,996664329	-0,167766133	630	-0,11993954
18	6,996664	12	6,997616684	-0,11993954	630	-0,085725547

Рис. 12.25
Чтобы вычислить корень на другом интервале достаточно внести изменения в ячейки B5 и C5

изменить значения в ячейках B5 и C5. В приведенном на рисунке 12.24 документе корень вычисляется на интервале от -10 до 0. Пример вычисления корня на интервале от 6 до 12 представлен на рисунке 12.25.

На заметку

Созданный выше документ, разумеется, не лишен недостатков. Например, если для начальных границ интервала указать одинаковые значения, в ячейках для вычисления приближенных значений для корня уравнения возникает ошибка деления на ноль. Такие ситуации в принципе отслеживаются и обрабатываются, причем с помощью встроенных функций Excel. Однако это делает выражения для формул слишком громоздкими — несоизмеримо громоздкими по сравнению с тем «положительным» эффектом, который достигается благодаря усложнению выражений.

Разумеется, кроме вычислений, непосредственно в рабочем листе Excel мы всегда можем создать программный код, наподобие того, что рассматривался при решении уравнения методом половинного деления. В листинге 12.4 представлен программный код функции, определяющей решаемое уравнение.

Листинг 12.4. Функция для определения решаемого уравнения

```
Function УРАВНЕНИЕ(a3, a2, a1, a0, x) As Double
 ' Функция уравнения - полином третьей степени
 УРАВНЕНИЕ = a3 * x ^ 3 + a2 * x ^ 2 + a1 * x + a0
End Function
```

Эта функция предназначена для решения уравнений с функцией полиномиального типа третьей степени, т. е. для решения уравнений вида $a_3x^3 + a_2x^2 + a_1x + a_0 = 0$. Коэффициенты полиномиального выражения a_k ($k = 0, 1, 2, 3$) и переменная x передаются аргументами функции. Это могут быть как числа, так и ссылки. Функцию УРАВНЕНИЕ() мы будем использовать в другой функции, которая называется МЕТОДХОРД() и в которой непосредственно и реализован алгоритм решения уравнения методом хорд. В листинге 12.5 приведен программный код этой функции.

На заметку

Программные коды обеих функций размещаются в одном модуле — модуле проекта.

Листинг 12.5. Функция для решения уравнения методом хорд

```
Function МЕТОДХОРД(a3, a2, a1, a0, x1, x2, num)
 ' Переменные для запоминания корня уравнения,
 ' а также левой и правой границ интервала поиска корня
 Dim x As Double, a As Double, b As Double
 ' Значение левой границы интервала поиска корня уравнения
 a = x1
 ' Значение правой границы интервала поиска корня уравнения
 b = x2
```

```
' Переменная определяет количество итераций
Dim N As Integer
' Количество итераций при вычислении корня
 N = num
' Индексная переменная
Dim i As Integer
' Переменные для запоминания значения функции уравнения
' в предполагаемой точке корня уравнения и на границах
' интервала поиска корня
Dim Fx As Double, Fa As Double, Fb As Double
' Оператор цикла для вычисления корня уравнения
 For i = 0 To N
  ' Вычисляем значение функции уравнения на левой границе
   Fa = УРАВНЕНИЕ(a3, a2, a1, a0, a)
 ' Если на левой границе функция уравнения принимает нулевое значение
  If Fa = 0 Then
    ' Значение левой границы - корень уравнения
   МЕТОДХОРД  =  a
    ' Завершение выполнения функции
    Exit Function
  End If
 ' Вычисление значения функции на правой границе
  Fb = УРАВНЕНИЕ(a3, a2, a1, a0, b)
 ' Если на правой границе функция уравнения равна нулю
 If Fb = 0 Then
    ' Значение правой границы - корень уравнения
   МЕТОДХОРД  =  b
    ' Завершение выполнения функции
    Exit Function
  End If
 ' Если на границах интервала поиска корня функция уравнения
 ' принимает значения одинаковых знаков
 If Fa * Fb > 0 Then
    ' Функция возвращает ошибку
   МЕТОДХОРД = CVErr(xlErrValue)
    ' Завершение выполнения функции
    Exit Function
    ' На границах интервала поиска корня функция уравнения
    ' принимает значения разных знаков
  Else
    ' Вычисляем приближение для корня
     x = (a * Fb - b * Fa) / (Fb - Fa)
    ' Вычисляем значение функции в точке
    ' предполагаемого корня
     Fx = УРАВНЕНИЕ(a3, a2, a1, a0, x)
    ' Если корень найден
```

```
      If Fx = 0 Then
       ' Значение функции
       МЕТОДХОРД  =  x
       ' Завершение выполнения функции
       Exit Function
       ' Корень не найден
      Else
       ' Если в точке предполагаемого корня
       ' и на левой границе функция уравнения
       ' принимает значения одинаковых знаков
       If Fa * Fx > 0 Then
        ' Левая граница переносится в точку предполагаемого корня
          a = x
        ' В точке предполагаемого корня
        ' и на правой границе функция уравнения
        ' принимает значения одинаковых знаков
       Else
        ' Правая граница переносится в точку предполагаемого корня
          b = x
       End If
      End If
     End If
    Next i
    ' Значение функции
    МЕТОДХОРД  =  x
End Function
```

Аргументами функции передаются коэффициенты полиномиальной зависимости, границы интервала для поиска корня уравнения, а также количество итераций, которые нужно выполнить. В теле функции объявляется несколько переменных, предназначенных для запоминания значений границ интервала поиска корня (a и b), корня уравнения (переменная x), значений функции уравнения на границах интервала поиска корня и в точке предполагаемого корня уравнения (переменные Fa, Fb и Fx), а также целочисленная переменная N, в которую записывается значение для количества итераций, выполняемых при вычислении корня уравнения.

После того как на основании аргументов функции МЕТОДХОРД() определены значения переменных a, b и N, запускается оператор цикла, в котором целочисленная индексная переменная i пробегает значения от 0 до N. Каждая итерация соответствует вычислению нового приближения для корня уравнения.

В теле оператора цикла командой Fa = УРАВНЕНИЕ(a3,a2,a1,a0,a) вычисляется значение функции уравнения на левой границе диапазона поиска решения. При этом мы вызываем рассмотренную выше функцию УРАВНЕНИЕ(), аргументами которой передаем коэффициенты, определяющие полиномиальное выражение для функции уравнения (переменные a3, a2, a1 и a0,

которые передавались аргументами функции МЕТОДХОРД()), а также значение левой границы (переменная a) интервала поиска корня — именно в этой точке вычисляется функция уравнения.

После того как значение функции уравнения на левой границе вычислено, выполняется проверка на предмет равенства этого значения нулю. Если функция уравнения на левой границе равна нулю, значение левой границы становится результатом функции, и вычисление функции завершается. Если на левой границе функция нулю не равна, вычисляется значение функции на правой границе (команда Fb = УРАВНЕНИЕ(a3,a2,a1,a0,b)). Если на границе справа функция уравнения равна нулю — корень уравнения найден, и работа функции завершается. В противном случае (т. е. если ни на одной из границ функция уравнения не обращается в нуль) возможны два варианта: функция уравнения на границах интервала принимает значения разных знаков, или она принимает на границах интервала значения одного знака. В последнем случае (условный оператор с проверкой выражения Fa*Fb>0) функция в качестве результата возвращает ошибку #ЗНАЧ! (команда МЕТОДХОРД = CVErr(xlErrValue) и завершает свою работу. Наконец, в случае, если функция уравнения на границах интервала принимает значения разных знаков, вычисляется очередное приближение для корня уравнения (команда x = (a*Fb-b*Fa)/(Fb-Fa)), а также значение для функции уравнения в этой точке (команда Fx = УРАВНЕНИЕ(a3,a2,a1,a0,x)). Вычисленное значение проверяется: не равно ли оно нулю? Если да, то мы решили уравнение. Если нет, то нам предстоит сдвинуть одну из границ внутрь интервала. Сдвигается та граница, на которой значение функции уравнения совпадает со значением функции в точке внутри интервала. Для проверки традиционно используем условный оператор. По окончании работы оператора цикла значение переменной x возвращается как значение функции МЕТОДХОРД(). Пример использования этой функции в рабочем документе Excel представлен на рисунке 12.26.

В этом рабочем документе ячейки B5:B8 содержат коэффициенты для полиномиального выражения, определяющего решаемое уравнение. Границы интервала, на котором вычисляется корень, задаются в ячейках D5 и D6. Ячейка D8 содержит значение для количества итераций, по которым вычисляется корень уравнения. Результат (решение уравнения) вычисляется в ячейке F5 по формуле =МЕТОДХОРД(B8;B7;B6;B5;D5;D6;D8).

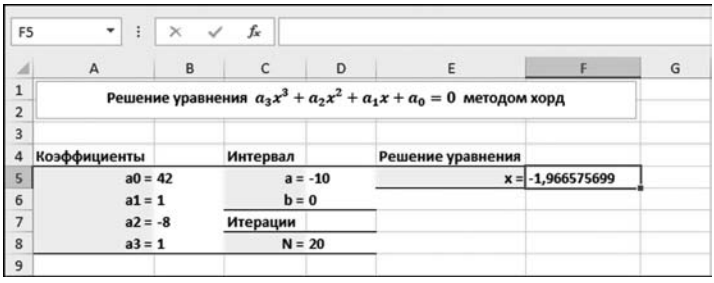


Рис. 12.26
Решение уравнения с помощью функции пользователя методом хорд

МЕТОД КАСАТЕЛЬНЫХ

Мы не будем полагаться на случай.
Мы пойдем простым логическим ходом.

Из к/ф «Ирония судьбы, или С легким паром!»

При решении алгебраического уравнения вида $f(x) = 0$ методом касательных (или методом Ньютона) необходимо указывать не границы интервала поиска корня, а начальное приближение для поиска корня уравнения (например, некоторое значение x_0). Через точку $(x_0, f(x_0))$ на кривой, определяемой функцией $f(x)$, с абсциссой x_0 проводится касательная. Точка пересечения этой касательной с осью абсцисс — новое приближение для корня уравнения (назовем его x_1). Затем через точку $(x_1, f(x_1))$ проводим касательную, и, определив точку пересечения этой касательной с осью абсцисс, находим новое, второе приближение для корня уравнения x_2 и т. д. На рисунке 12.27 этот процесс проиллюстрирован графически.

Если на каком-то этапе вычислено приближение x_n, то следующее приближение x_{n+1} может быть вычислено по формуле $x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$, определяющей рекуррентное соотношение для приближений корня уравнения. В этом выражении через $f'(x) \equiv \dfrac{df(x)}{dx}$ обозначена производная по аргументу от функции $f(x)$, определяющей решаемое уравнение.

Понятно, что описанный выше итерационный процесс совсем не обязательно должен сходиться к решению уравнения. Другими словами, даже если мы можем формально реализовать итерационную процедуру, совсем не факт, что полученный результат будет решением уравнения. Обычно это сразу заметно: если итерационный процесс сходится к корню, то значение функции $f(x)$ приближается к нулю, а приращение между последовательными приближениями для корня уменьшается. Есть и более точные и надежные

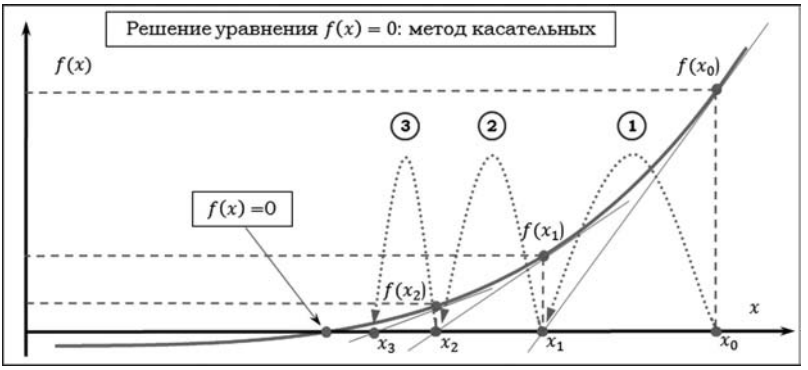


Рис. 12.27
Общая схема решения уравнения методом касательных:
цифрами и пунктирными линиями со стрелкой показана последовательность перемещения границ интервала, на котором ищется решение, тонкие прямые линии обозначают касательные, на основе которых вычисляются приближения для решений.

методы определения того, применим ли метод касательных для решения уравнения. Критерий формируется на основе оценки для значений первой и второй производных для функции $f(x)$ на интервале поиска решения. Если известно, что корень локализован на интервале значений от a до b и на этом интервале первая производная $f'(x)$ строго больше нуля и не превышает по абсолютной величине некоторого значения M (т. е. $0 < |f'(x)| \le M$ для любого $a \le x \le b$), а вторая производная $f''(x)$ по абсолютной величине не превышает некоторого значения N (т. е. $|f''(x)| \le N$ для любого $a \le x \le b$), то метод касательных сходится при условии $\frac{N}{M} \cdot \frac{b-a}{2} < 1$. В качестве начального приближения выбирается точка, в которой функция и вторая производная имеют одинаковые знаки (т. е. $f(x_0)f''(x_0) > 0$).

В качестве примера реализации метода касательных покажем, как с помощью этого метода в рабочем документе Excel решаются полиномиальные уравнения, т. е. уравнения вида $a_0 + a_1x + a_2x^2 + \ldots + a_nx^n = 0$. Это означает, что функция уравнения $f(x)$ задана в виде полинома $f(x) = \sum_{k=0}^{n} a_k x^k$. Кроме того, при вычислениях нам понадобится значение производной $f'(x) = \sum_{k=1}^{n} k a_k x^{k-1}$, которая также является полиномом.

На заметку

Несложно заметить, что полином (как функциональная зависимость) однозначно определяется набором коэффициентов. Чтобы вычислить значение полинома, достаточно знать его коэффициенты и аргумент (точка, для которой вычисляется значение полинома). Производная от полинома — это тоже полином. Причем коэффициенты полинома-производной легко вычисляются на основе коэффициентов исходного полинома. Таким образом, обычно нетривиальная процедура вычисления производной сводится к несложным алгебраическим расчетам.

B14 : {=B13-СУММ((B13^$B$3:$E$3)*$B$4:$E$4)/СУММ((B13^$B$3:$E$3)*$B$5:$E$5)}

	A	B	C	D	E	F	G	H
1	Решение уравнения полиномиального вида методом касательных							
2								
3	Степень	0	1	2	3			
4	Полином	42	1	-8	1			
5	Производная	1	-16	3	0			
6								
7								
8	Итерация	Корень						
9	0	10,000000						
10	1	8,212766						
11	2	7,315353						
12	3	7,030463						
13	4	7,000329						
14	5	7,000000						
15								

Рис. 12.28
Вычисление корня уравнения методом касательных

Рассмотрим рабочий документ, представленный на рисунке 12.28.

В этом документе полиномиальные выражения вычисляются с помощью встроенных функций рабочего документа Excel. В качестве «тестового» мы используем полином $f(x) = x^3 - 8x^2 + x + 42$. В ячейки диапазона B4:E4 вводятся соответствующие коэффициенты (в порядке возрастания показателя степени аргумента). Сами показатели степени заносятся в ячейки B3:E3.

На заметку

В ячейку B3 вводится значение 0, в ячейку C3 — формула =B3+1. Затем эта формула копируется во все прочие ячейки диапазона B3:E3.

В ячейках B5:E5 вычисляются и отображаются коэффициенты полинома-производной. Для этого в ячейку B5 вводится формула =C4*C3, после чего копируется в остальные ячейки диапазона B5:E5.

На заметку

При вычислении коэффициентов полинома-производной мы использовали следующее свойство: если в исходной полиномиальной функции коэффициент a_n соответствует множителю x^n, то в производной это трансформируется в слагаемое na_nx^{n-1}. Поэтому, чтобы вычислить коэффициент в производной для слагаемого с множителем x^n, нужно коэффициент возле множителя x^{n+1} в выражении для исходного (дифференцируемого) полинома умножить на $n + 1$. Если дифференцируемый полином был степени N, то производная является полиномом степени $N - 1$. Формально это означает, что коэффициент возле множителя x^N в полиноме-производной равен нулю. Как следствие, мы должны были бы в ячейку E5 не копировать формулу, а просто ввести ноль. Мы все же ячейку E5 заполняли копированием формулы. В результате эта ячейка содержит выражение =F4*F3. В этой формуле имеются ссылки на пустые ячейки. По умолчанию пустые ячейки в Excel (при вычислениях) интерпретируются как такие, что содержат нулевое значение. Как следствие, в ячейке E5 вычисляется нулевое значение, что нам собственно и нужно.

В столбце A, начиная с ячейки A9, вводятся номера итераций для вычисления корня. В ячейку A9 вводится значение 0, а в ячейку A10 вводим формулу =A9+1. В результате при автоматическом заполнении (копированием) нижних ячеек получим последовательность неотрицательных целых чисел. В ячейку B9 вводится начальное приближение 10 для поиска корня уравнения. В ячейку B10 вводим формулу массива =B9-СУММ((B9^$B$3:$E$3)*$B$4:$E$4)/СУММ((B9^$B$3:$E$3)*$B$5:$E$5). После того как ячейки A10:B10 заполнены, они выделяются и перетаскиванием маркера автоматического заполнения формулы из этих ячеек копируются в ячейки снизу. Как видим, за очень небольшое количество итераций корень уравнения найден (см. ячейку B14 на рисунке 12.28).

На заметку

Напомним, что формула массива вводится нажатием комбинации клавиш <Ctrl>+<Shift>+<Enter>. Это важно, поскольку иначе формула, которая вводится в ячейку B10 и копируется в остальные ячейки, «работать» не будет.

В этой формуле от значения корня, вычисленного на предыдущем шаге (ячейка сверху от той, в которой содержится формула), отнимается частное от значения полинома и производной полинома в этой же точке. Из ячейки B10 ссылка на ячейку с предыдущей итерацией для корня уравнения выглядит как B9. Значение полинома вычисляется инструкцией СУММ((B9^$B$3:$E$3)*$B$4:$E$4). Значение производной вычисляется выражением СУММ((B9^$B$3:$E$3)*$B$5:$E$5). Поскольку они отличаются только ссылкой на массив, достаточно проанализировать одно из этих выражений. Например, в выражении СУММ((B9^$B$3:$E$3)*$B$4:$E$4) выполняется суммирование по элементам массива (B9^$B$3:$E$3)*$B$4:$E$4. Это «вычисляемый» массив. Команда формирования этого массива состоит из произведения двух массивов: B9^$B$3:$E$3 и $B$4:$E$4. Массив $B$4:$E$4 — это диапазон ячеек с коэффициентами полинома. Массив B9^$B$3:$E$3 — набор значений аргумента массива (ячейка B9) в степени, определяемой диапазоном ячеек $B$3:$E$3. Поскольку вся формула в ячейку B9 вводится как формула массива, то операции с массивами выполняются «поэлементно»: если операндами являются массивы одинакового размера (как, например, (B9^$B$3:$E$3)*$B$4:$E$4), то результатом является массив того же размера, элементы которого получаются последовательным применением соответствующего бинарного оператора к парам элементов из массивов-операндов; если один операнд скалярный, а другой — массив (как, например, в выражении B9^$B$3:$E$3), то результатом является массив, и его элементы формируются путем последовательного применения бинарного оператора к скалярному операнду и элементу из массива-операнда. Поэтому результат формируется так: создается массив, элементы которого получаются возведением значения ячейки B9 в степень, показатели степени содержатся в ячейках $B$3:$E$3. Потом полученный массив поэлементно умножается на массив $B$4:$E$4. Элементы полученного массива суммируются — это и есть значение полинома.

Значение производной для полинома вычисляется практически так же, с поправкой на то, что коэффициенты производной-полинома записаны в ячейках $B$5:$E$5.

Хотя процесс вычисления полиномиального выражения без применения специальных утилит не очень громоздкий, иногда все же имеет смысл воспользоваться некоторой автоматизацией. Как минимум, имеет смысл описать функцию пользователя для вычисления полиномиального выражения на основе значений коэффициентов полинома (правда, имеется встроенная функция РЯД.СУММ(), которая позволяет вычислять полиномиальные суммы — но в данном случае это не принципиально). В листинге 12.6 приведен программный код такой функции.

Листинг 12.6. Функция для вычисления полиномиального выражения

```
Function ПОЛИНОМ(C, z) As Double
 ' Переменная для записи значения полинома
 Dim P As Double
 ' Переменная для запоминания степенного множителя
 Dim q As Double
 ' Переменная для запоминания аргумента
```

```
 Dim x As Double
 ' Переменная для оператора цикла (коэффициент полинома)
 Dim a As Variant
 ' Начальное значение для полиномиальной суммы
  P = 0
 ' Аргумент в нулевой степени
  q = 1
 ' Аргумент
  x = z
 ' Вычисление полинома
 For Each a In C
   ' Добавка к текущему значению полиномиальной суммы
    P = P + a * q
   ' Новый степенной множитель
    q = q * x
 Next a
 ' Значение полинома
  ПОЛИНОМ = P
End Function
```

У функции ПОЛИНОМ() два аргумента: как предполагается, первым аргументом функции передается ссылка на диапазон ячеек с коэффициентами для полиномиальной зависимости, а второй аргумент функции — аргумент полинома (точка, в которой вычисляется значение полинома).

В теле функции объявляется несколько переменных: в переменную P будет записываться полиномиальная сумма (начальное значение 0), переменная q нужна для запоминания «аргумента в степени», т. е. степенного слагаемого (начальное значение переменной равно 1, что соответствует слагаемому без аргумента, или аргументу в нулевой степени). Значение аргумента записывается в переменную x.

Вычисление полинома (полиномиальной суммы) выполняется в теле оператора цикла. Там использована индексная переменная a, которой перебираются элементы из «множества» C (первый аргумент функции ПОЛИНОМ() — напомним, что мы предполагаем передавать в качестве этого аргумента массив ячеек).

В теле оператора цикла всего две команды. Командой P = P+a*q к текущему значению полиномиальной суммы прибавляется очередное слагаемое (добавка вида $a_k x^k$). Добавка a*q вычисляется как произведение коэффициента a_k (переменная a) на x^k (переменная q). Командой q = q*x вычисляется новый степенной множитель (слагаемое x^{k+1} для следующего итерационного шага получается умножением на аргумент полинома текущего значения степенного множителя). Вычисленное таким образом значение переменной P возвращается как результат функции ПОЛИНОМ().

Кроме функции для вычисления полинома, создаем функцию для вычисления производной от полинома. Программный код этой функции представлен в листинге 12.7.

Листинг 12.7. Функция для вычисления производной от полиномиального выражения

```
Function ПОЛИНОМПРОИЗВ(C, z) As Double
 ' Объявляем массив для записи коэффициентов
 ' полинома-производной
 Dim a() As Double
 ' Переменная для запоминания количества коэффициентов
 Dim n As Integer
 ' Количество коэффициентов полинома-производной
  n = C.Count - 1
 ' Создание массива для коэффициентов
 ' полинома-производной
 ReDim a(1 To n) As Double
 ' Индексная переменная для оператора цикла
 Dim i As Integer
 ' Оператор цикла для заполнения элементов массива
  For i = 1 To n
   ' Вычисление элемента массива на основе коэффициента
   ' дифференцируемого полинома
    a(i) = i * C(i + 1)
 Next i
 ' Для вычисления производной вызывается
 ' функция вычисления полинома
 ПОЛИНОМПРОИЗВ = ПОЛИНОМ(a, z)
End Function
```

Функция называется ПОЛИНОМПРОИЗВ() и аргументы у нее такие же, как и у функции ПОЛИНОМ(), но с той лишь разницей, что теперь первый аргумент определяет не набор коэффициентов конечного полинома (который получается в результате дифференцирования), а набор коэффициентов дифференцируемого полинома. Поскольку у нас уже есть функция для вычисления полинома, а производная от полинома также является полиномом, нам достаточно вычислить его коэффициенты. Основные вычисления в теле функции ПОЛИНОМПРОИЗВ() связаны с решением этой задачи. Идея очень простая: создается числовой массив a, который заполняется значениями (на основании коэффициентов дифференцируемого полинома), а затем этот массив передается аргументом функции вычисления полиномиального значения ПОЛИНОМ().

На заметку

Массив a является динамическим — его размер определяется на основании первого аргумента, переданного функции ПОЛИНОМПРОИЗВ(). Поэтому массив объявляется с инструкцией Dim, а затем размер массива задается (изменяется) с помощью инструкции ReDim.

Пример использования функций ПОЛИНОМ() и ПОЛИНОМПРОИЗВ() для решения уравнения методом касательных представлен на рисунке 12.29.

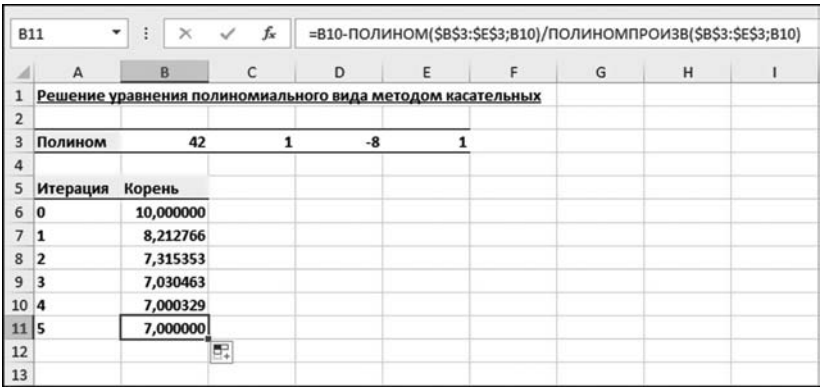


Рис. 12.29
Вычисление корня уравнения методом касательных
с использованием пользовательских функций

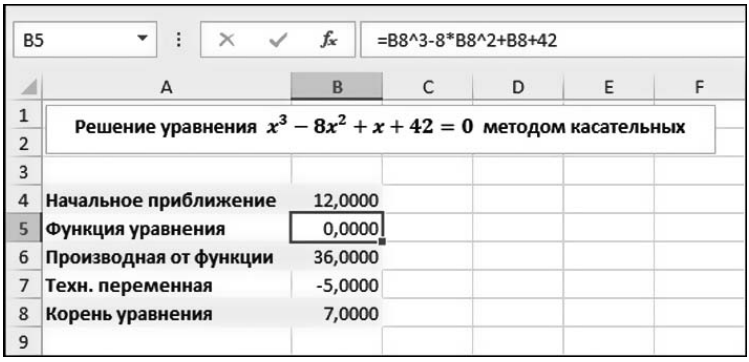


Рис. 12.30
Решение уравнения методом касательных с использованием циклических ссылок

По структуре документ очень напоминает тот, что рассматривался ранее (рис. 12.28), однако есть некоторые отличия. Во-первых, в данном случае нет необходимости в рабочем документе вычислять коэффициенты для полинома-производной — только в ячейках B3:E3 представлены коэффициенты исходного полинома (как и ранее, решаем уравнение $x^3 - 8x^2 + x + 42 = 0$). Во-вторых, в ячейку B7, на основании значения которой заполняются все прочие ячейки внизу, вводится достаточно простая формула =B6-ПОЛИНОМ ($B$3:$E$3;B6)/ПОЛИНОМПРОИЗВ($B$3:$E$3;B6), в которой использованы описанные выше функции для вычисления полиномиальной функции и производной от полиномиальной функции. Причем формула вводится обычная (не формула массива, как было до этого).

Если отслеживание приближенного корня на каждой итерации не является первоочередной задачей, для решения уравнения методом касательных можно воспользоваться циклическими ссылками. Пример таких вычислений представлен в документе на рисунке 12.30.

На заметку

Нелишним будет напомнить, что для использования циклических ссылок в окне настроек **Параметры Excel** в разделе **Формулы** следует установить флажок опции **Включить итерационные вычисления** (рис. 12.31).
Если этого не сделать, при вводе циклической ссылки появится сообщение об ошибке.

Для решения уравнения $x^3 - 8x^2 + x + 42 = 0$ методом касательных в рабочем документе заполняем пять ячеек, как это описано в таблице 12.3.

В этом документе содержатся как явные, так и неявные циклические ссылки. Но перед их обсуждением имеет смысл кратко остановиться на общем алгоритме проведения вычислений. Отправной точкой служит рекуррентное соотношение $x_{n+1} = x_n - f(x_n)/f'(x_n)$ для вычисления очередного приближения для корня уравнения. Поэтому самый простой и очевидный способ решения — ввести в ячейку рабочего документа формулу, вычисляющую выражение $x - f(x)/f'(x)$, причем в качестве x использовать ссылку на ячейку,

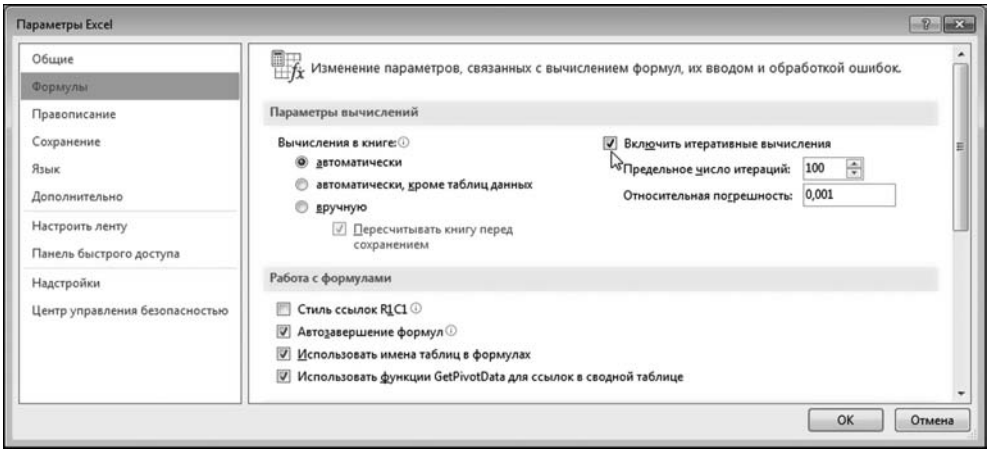


Рис. 12.31
Переход в режим использования циклических ссылок

Таблица 12.3

Метод касательных

Ячейка	Значение	Комментарий
B4	12	Начальное приближение для поиска корня уравнения
B5	=B8^3-8*B8^2+B8+42	Формула для вычисления функции уравнения в точке предполагаемого корня
B6	=3*B8^2-16*B8+1	Формула для вычисления производной от функции уравнения в точке предполагаемого корня
B7	=B7-B5/B6	Ячейка с явной циклической ссылкой для вычисления итерационным методом корня уравнения. Значение ячейки — вспомогательная переменная, используемая при поиске решения
B8	=B7+B4	Корень уравнения

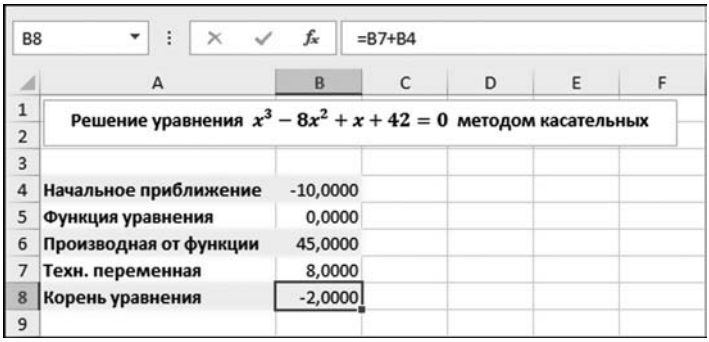


Рис. 12.32
Найден еще один корень уравнения

в которую вводится формула. В результате получим циклическую ссылку и, в идеале, корень уравнения в ячейке с этой циклической ссылкой. Однако помимо громоздкости такой формулы, здесь имеется еще одна, более принципиальная проблема: вычисления начинаются с нулевого значения, т. е. при решении уравнения автоматически будет использовано нулевое начальное приближение для корня. Это далеко не всегда приемлемо, поскольку начальное приближение в методе касательных является важным фактором для успешного решения уравнения. Поэтому необходимо предусмотреть возможность явно задавать точку начального приближения для корня уравнения. Наша стратегия состоит в том, чтобы выполнить замену переменных в соотношении $x_{n+1} = x_n - f(x_n)/f'(x_n)$, введя в рассмотрение величину $z_n = x_n - x_0$, где x_0 обозначает начальное приближение для корня уравнения. Тогда рекуррентное соотношение в терминах новой переменной может быть записано как $z_{n+1} = z_n - f(z_n + x_0)/f'(z_n + x_0)$. Наш «выигрыш» в том, что для переменной z начальное приближение нулевое. То есть мы, фактически, будем решать уравнение относительно z, а потом вычислим корень уравнения как $x = z + x_0$.

Начальное приближение для корня уравнения (величина x_0) указывается в ячейке B4. В ячейке B7 будет вычисляться «технический» параметр z. Корень уравнения (параметр x, вычисляется в ячейке B8), таким образом, должен представлять собой сумму значений в ячейках B4 и B7. Отсюда в ячейке B8 появляется формула =B7+B4. Ячейка B5 содержит значение функции уравнения (величина $f(z + x_0)$), которое вычисляется по формуле =B8^3-8*B8^2+B8+42. Здесь мы воспользовались тем обстоятельством, что по определению $z + x_0 = x$, а этот параметр вычисляется в ячейке B8. В ячейке B6 вычисляется производная от функции уравнения $f'(x) = 3x^2 - 16x + 1$. Значение в ячейке B7 вычисляется по формуле =B7-B5/B6 в соответствии с рекуррентным соотношением $z_{n+1} = z_n - f(z_n + x_0)/f'(z_n + x_0)$. Как видим (рис. 12.30), результат вычислений вполне приемлемый. Если изменить начальное приближение для корня уравнения в ячейке B4, можем получить иной корень уравнения. На рисунке 12.32 показан результат вычисления корня уравнения при начальном приближении для корня -10.

Вместе с тем, еще раз хочется обратить внимание на критическую важность выбора корректного начального приближения для корня уравнения.

На заметку

Работа с циклическими ссылками требует некоторой изобретательности. Например, в рассмотренном документе достаточно сложная структура циклических (прямых и непрямых) ссылок. Если в какой-то из ячеек произойдет ошибка, она мгновенно «распространяется» на все связанные ячейки, и вернуть ситуацию к нормальной может быть весьма проблематично. Причина в том, что при вычислении циклических ссылок используются текущие значения ячеек. Если текущее значение ячейки — «ошибка», то какого результата можно добиться? В самой критической ситуации может помочь такая последовательность действий: выходим из режима использования циклических ссылок (убираем флажок опции **Включить итерационные вычисления** в окне настроек **Параметры Excel**), редактируем (если надо) документ и снова включаем режим использования циклических ссылок.

Эта же причина может привести к некоторым недоразумениям, связанным с результатом вычислений по циклическим ссылкам при изменении начального приближения для корня уравнения. Еще раз подчеркнем, что схема вычислений такая: если изменяется какая-то ячейка в рабочем документе, то пересчитываются зависимые от нее ячейки и дальше по цепочке. Если в документе имеется явная или неявная циклическая ссылка, то в расчетах используется текущее значение ячейки. Например, в созданном нами документе при изменении значения ячейки B4 (начальное приближение для корня уравнения), пересчитывается значение ячейки B8. Как следствие изменения значения ячейки B8, пересчитываются ячейки B5 и B6. Это приводит к пересчету по явной циклической ссылке ячейки B7. Изменение ячейки B7 приводит к пересчету ячейки B8 и т. д.

Выше мы использовали аналитические выражения для функции уравнения и ее производной. Но в принципе, если аналитическое выражение для производной неизвестно, оно громоздкое, его в силу каких-то причин нет возможности вычислить, можно прибегнуть к процедуре вычисления производной в числовом виде. Далее проиллюстрирован именно такой подход.

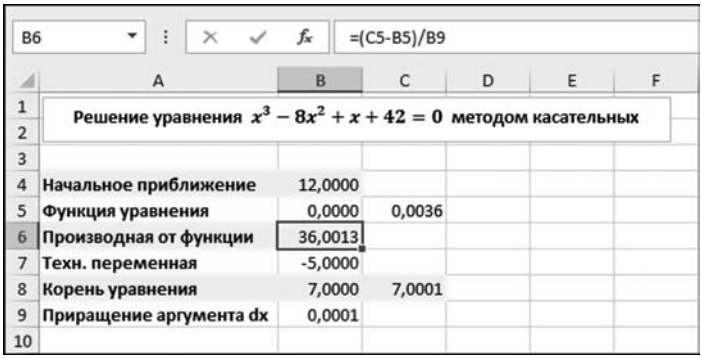


Рис. 12.33
В процессе вычисления корня производная рассчитывается в числовом виде

На рисунке 12.33 представлена модификация рассмотренного выше документа (с циклическими ссылками): изменения касаются способа вычисления производной $f'(x)$ — вместо аналитического выражения для производной мы вычисляем производную в числовом виде.

Идея состоит в том, что производная $f'(x)$ от функции $f(x)$ приближенно вычисляется в виде $f'(x) \approx \frac{f(x+dx)-f(x)}{dx}$, где через dx обозначено малое приращение аргумента функции. Соответствующий параметр (т. е. приращение dx) указывается в ячейке B9 рабочего документа (мы использовали значение 0,0001 для этого параметра). В ячейку C8 вводится формула =B8+B9. В результате в этой ячейке вычисляется «аргумент с приращением» $x + dx$. Чтобы вычислить значение функции уравнения для «аргумента с приращением», в ячейку C5 из ячейки B5 копируется формула вычисления функции уравнения. В результате ячейка C5 будет содержать формулу =C8^3-8*C8^2+C8+42. Производная в ячейке B6 вычисляется по формуле =(C5-B5)/B9 как приращение функции, отнесенное к приращению аргумента.

Преимущество подхода с вычислением производной в числовом виде очевидно связано с тем, что нет необходимости в явном виде вычислять выражение для производной и вводить еще одну дополнительную формулу. Недостаток же связан с тем, что мы используем приближенное выражение для производной, а это в некоторых случаях может быть критичным.

МЕТОД ПОСЛЕДОВАТЕЛЬНЫХ ПРИБЛИЖЕНИЙ

— Это серьезно, Билли!
Это восемьдесят!
— Вчера это стоило пятьдесят!
— Инфляция, инфляция...

Из к/ф «Человек с бульвара Капуцинов»

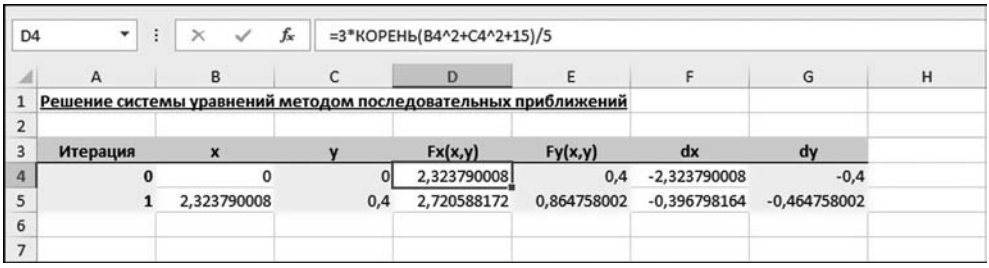
Метод последовательных приближений состоит в том, что уравнение вида $x = \varphi(x)$ решается на основе использования рекуррентного соотношения $x_{k+1} = \varphi(x_k)$ (при условии, что задано начальное приближение x_0 для корня). Мы этот метод уже использовали для решения скалярных уравнений. Здесь изучим вопрос о том, как метод последовательных приближений может использоваться для решения системы алгебраических уравнений.

Предположим, имеется n уравнений вида $x_k = \varphi_k(x_1, x_2, \ldots, x_n)$, индекс $k = 1, 2, \ldots, n$. В векторной форме систему можно записать как $\vec{x} = \vec{\varphi}(\vec{x})$, где обозначено $\vec{x} = (x_1, x_2, \ldots, x_n)$ и $\vec{\varphi}(\vec{x}) = (\varphi_1(\vec{x}), \varphi_2(\vec{x}), \ldots, \varphi_n(\vec{x}))$. Если задано начальное приближение $\vec{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)})$, то каждое последующее приближение вычисляется через рекуррентное векторное соотношение $\vec{x}^{(m+1)} = \vec{\varphi}(\vec{x}^{(m)})$ (индекс $m = 0, 1, 2, \ldots$). Те же соотношения в невекторной форме записи будут иметь вид $x_k^{(m+1)} = \varphi_k(x_1^{(m)}, x_2^{(m)}, \ldots, x_n^{(m)})$ (нижний индекс $k = 1, 2, \ldots, n$ определяет переменную и уравнение, а верхний индекс $m = 0, 1, 2, \ldots$ определяет итерацию). Наша задача состоит в том, чтобы реализовать такой процесс в рабочем документе Excel.

На заметку

Вопрос о сходимости итерационного процесса мы оставляем за рамками этой книги. Те из читателей, кого интересует строгая математическая постановка задачи и критерии ее разрешимости, могут обратиться к специальной литературе. Если ограничиться нестрогим объяснением, то сводится оно к тому, что производные от функций в правых частях уравнений системы по абсолютной величине в области поиска решений должны быть меньше единицы.

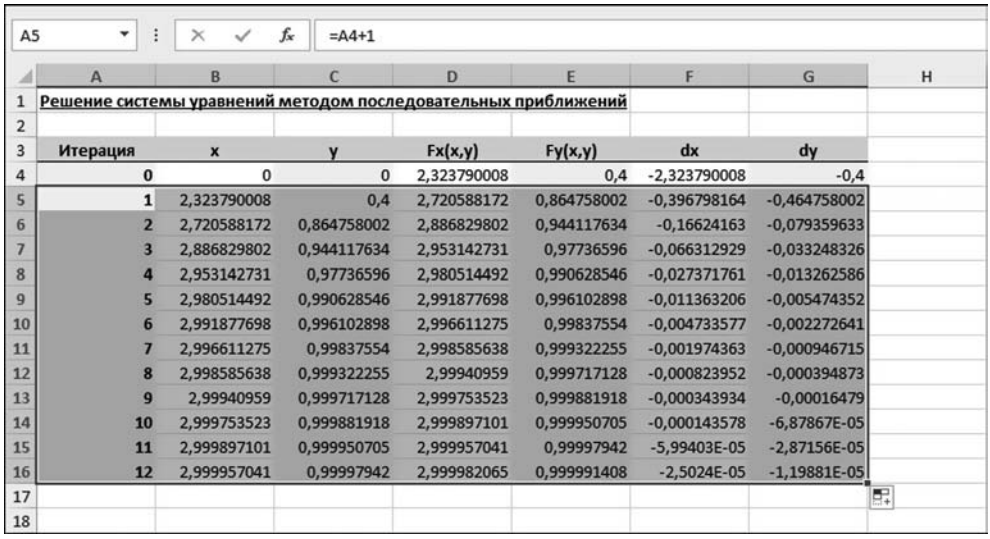
Рассмотрим систему из двух уравнений: $5x - 3\sqrt{x^2 + y^2 + 15} = 0$ и $5y - x = 2$. Для применения метода последовательных приближений запишем эти уравнения в виде $x = \frac{3\sqrt{x^2 + y^2 + 15}}{5}$ и $y = \frac{x + 2}{5}$ соответственно. Рабочий документ, в котором мы будем выполнять вычисления, показан на рисунке 12.34.



D4 | =3*КОРЕНЬ(B4^2+C4^2+15)/5

	A	B	C	D	E	F	G	H
1	Решение системы уравнений методом последовательных приближений							
2								
3	Итерация	x	y	Fx(x,y)	Fy(x,y)	dx	dy	
4	0	0	0	2,323790008	0,4	-2,323790008	-0,4	
5	1	2,323790008	0,4	2,720588172	0,864758002	-0,396798164	-0,464758002	
6								
7								

Рис. 12.34
Рабочий документ перед началом вычисления решения системы уравнений методом последовательных приближений



A5 | =A4+1

	A	B	C	D	E	F	G	H
1	Решение системы уравнений методом последовательных приближений							
2								
3	Итерация	x	y	Fx(x,y)	Fy(x,y)	dx	dy	
4	0	0	0	2,323790008	0,4	-2,323790008	-0,4	
5	1	2,323790008	0,4	2,720588172	0,864758002	-0,396798164	-0,464758002	
6	2	2,720588172	0,864758002	2,886829802	0,944117634	-0,16624163	-0,079359633	
7	3	2,886829802	0,944117634	2,953142731	0,97736596	-0,066312929	-0,033248326	
8	4	2,953142731	0,97736596	2,980514492	0,990628546	-0,027371761	-0,013262586	
9	5	2,980514492	0,990628546	2,991877698	0,996102898	-0,011363206	-0,005474352	
10	6	2,991877698	0,996102898	2,996611275	0,99837554	-0,004733577	-0,002272641	
11	7	2,996611275	0,99837554	2,998585638	0,999322255	-0,001974363	-0,000946715	
12	8	2,998585638	0,999322255	2,99940959	0,999717128	-0,000823952	-0,000394873	
13	9	2,99940959	0,999717128	2,999753523	0,999881918	-0,000343934	-0,00016479	
14	10	2,999753523	0,999881918	2,999897101	0,999950705	-0,000143578	-6,87867E-05	
15	11	2,999897101	0,999950705	2,999957041	0,99997942	-5,99403E-05	-2,87156E-05	
16	12	2,999957041	0,99997942	2,999982065	0,999991408	-2,5024E-05	-1,19881E-05	
17								
18								

Рис. 12.35
Результат решения системы уравнений

Перед началом вычислений числами и формулами заполняются ячейки диапазона A4:G5. Столбец A содержит номер итерации. В столбцах B и C отображаются вычисляемые значения для переменных, относительно которых решается система уравнений. В столбцах D и F вычисляются и отображаются значения для функций в правых частях уравнений системы. Если решение найдено, то значения в столбцах B и D, а также C и E должны совпадать. Для удобства и наглядности в столбцах F и G отображается разница между текущим значением переменной в левой части уравнения и функцией в правой части уравнения. Более подробно вводимые в ячейки документа значения описаны в таблице 12.4.

Процедура дальнейших вычислений нам уже знакома: выделяем диапазон ячеек A5:G5 и заполняем ячейки внизу перетягиванием маркера заполнения. Результат вычислений представлен на рисунке 12.35.

Мы получили достаточно неплохое приближение для точного решения системы уравнений $x = 3$ и $y = 1$.

Таблица 12.4

Метод последовательных приближений

Ячейка	Значение	Комментарий
A4	0	Начальная итерация
A5	=A4+1	Вычисление следующей итерации
B4	0	Начальное приближение для переменной x
B5	=D4	Следующее приближение для переменной x. Вычисляется как значение функции $F_x(x,y)=\frac{3\sqrt{x^2+y^2+15}}{5}$ (первое уравнение) на предыдущей итерации
C4	0	Начальное приближение для переменной y
C5	=E4	Следующее приближение для переменной y. Вычисляется как значение функции $F_y(x,y)=\frac{x+2}{5}$ (второе уравнение) на предыдущей итерации
D4	=3*КОРЕНЬ(B4^2+C4^2+15)/5	Формула для вычисления значения функции $F_x(x,y)=\frac{3\sqrt{x^2+y^2+15}}{5}$ (первое уравнение)
D5	=3*КОРЕНЬ(B5^2+C5^2+15)/5	Формула копируется из ячейки D4
E4	=(B4+2)/5	Формула для вычисления функции $F_y(x,y)=\frac{x+2}{5}$ (второе уравнение)
E5	=(B5+2)/5	Формула копируется из ячейки E4
F4	=B4-D4	Разница между переменной x и функцией $F_x(x, y)$ для нулевой итерации
F5	=B5-D5	Формула копируется из ячейки F4
G4	=C4-E4	Разница между переменной y и функцией $F_y(x, y)$ для нулевой итерации
G5	=C5-E5	Формула копируется из ячейки G4

МЕТОД НЬЮТОНА

Нет, он сомнителен...
Он сомнителен. Я бы ему не доверял.

Из к/ф «Покровские ворота»

Метод касательных, который мы рассматривали выше для решения отдельных алгебраических уравнений, может быть расширен на случай системы алгебраических уравнений. Постановка задачи может быть следующей: необходимо решить систему уравнений $f_k(x_1, x_2, \ldots, x_n) = 0$ (индекс $k = 1, 2, \ldots, n$) относительно переменных x_m (индекс $m = 1, 2, \ldots, n$). В векторном виде эта система может быть записана как $\vec{f}(\vec{x}) = 0$, где введены обозначения $\vec{x} = (x_1, x_2, \ldots, x_n)^T$ и $\vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \ldots, f_n(\vec{x}))^T$ (символ T означает «транспонирование»). Для решения системы уравнений методом Ньютона используется рекуррентное соотношение $\vec{x}^{(m+1)} = \vec{x}^{(m)} - \hat{\mathbf{A}}^{-1}(\vec{x}^{(m)})\vec{f}(\vec{x}^{(m)})$. Тут через $\hat{\mathbf{A}}(\vec{x})$ обозначена матрица, элементы которой $a_{ij}(\vec{x}) = \dfrac{\partial f_i(\vec{x})}{\partial x_j}$ представляют собой частные производные по аргументам от функций, определяющих уравнения системы. Соответственно, $\hat{\mathbf{A}}^{-1}(\vec{x})$ обозначает матрицу, обратную к матрице $\hat{\mathbf{A}}(\vec{x})$.

Таким образом, для итерационного вычисления приближений для решения системы нам необходимо вычислить матрицу производных, найти к ней обратную матрицу, а затем реализовать вычисления с матричными произведениями. Такой алгоритм может быть реализован в Excel, чем мы далее и займемся.

На заметку

Чтобы понять, откуда берется приведенное выше рекуррентное соотношение, воспользуемся соображениями следующего характера. Допустим, что $\vec{x} = (x_1, x_2, \ldots, x_n)^T$ — точное решение уравнения $\vec{f}(\vec{x}) = 0$, а $\Delta\vec{x} = (\Delta x_1, \Delta x_2, \ldots, \Delta x_n)^T$ — малое отклонение от этого точного решения. Положим по определению $\vec{z} = \vec{x} - \Delta\vec{x}$ и рассмотрим выражение $f_k(\vec{z}) = f_k(\vec{x} - \Delta\vec{x}) = f_k(x_1 - \Delta x_1, x_2 - \Delta x_2, \ldots, x_n - \Delta x_n)$ для некоторого фиксированного индекса k. В первом приближении разложения по векторному параметру $\Delta\vec{x}$ и с учетом тождества $f_k(\vec{x}) = 0$ получаем $f_k(\vec{x} - \Delta\vec{x}) \approx f(\vec{x}) - \sum_{i=1}^{n} \dfrac{\partial f_k(\vec{x})}{\partial x_i}\Delta x_i = -\sum_{i=1}^{n} \dfrac{\partial f_k(\vec{x})}{\partial x_i}\Delta x_i$. В векторной форме эти соотношения обобщаются в виде соотношения $\vec{f}(\vec{z}) + \hat{\mathbf{A}}(\vec{x})\Delta\vec{x} = 0$. Это, в свою очередь, позволяет нам записать $\Delta\vec{x} = -\hat{\mathbf{A}}^{-1}(\vec{x})\vec{f}(\vec{z})$. Если теперь учесть, что $\Delta\vec{x} = \vec{x} - \vec{z}$ и принять $\hat{\mathbf{A}}^{-1}(\vec{x}) \approx \hat{\mathbf{A}}^{-1}(\vec{z})$, получим соотношение $\vec{x} = \vec{z} - \hat{\mathbf{A}}^{-1}(\vec{z})\vec{f}(\vec{z})$.

Решим с помощью Excel методом Ньютона систему уравнений вида $4x^2 + y^2 + 2xy - y - 2 = 0$ и $2x^2 + y^2 + 3xy - 3 = 0$. Эта система имеет два решения. Одно «аналитическое» решение $x = 0{,}5$ и $y = 1$ (точное решение), а также «приближенное» решение $x \approx -0{,}428$ и $y \approx 2{,}388$. В зависимости от на-

чального приближения для решения системы, мы сможем найти оба этих решения.

Для проведения вычислений нам понадобятся функции $f_1(x, y) = 4x^2 + y^2 + 2xy - y - 2$ и $f_2(x, y) = 2x^2 + y^2 + 3xy - 3$, определяющие систему уравнений, а также частные производные от этих функций, определяющие матрицу производных

$$\hat{\mathbf{A}}(x,y) = \begin{bmatrix} \dfrac{\partial f_1(x,y)}{\partial x} & \dfrac{\partial f_1(x,y)}{\partial y} \\ \dfrac{\partial f_2(x,y)}{\partial x} & \dfrac{\partial f_2(x,y)}{\partial y} \end{bmatrix} = \begin{bmatrix} 8x+2y & 2x+2y-1 \\ 4x+3y & 3x+2y \end{bmatrix}.$$

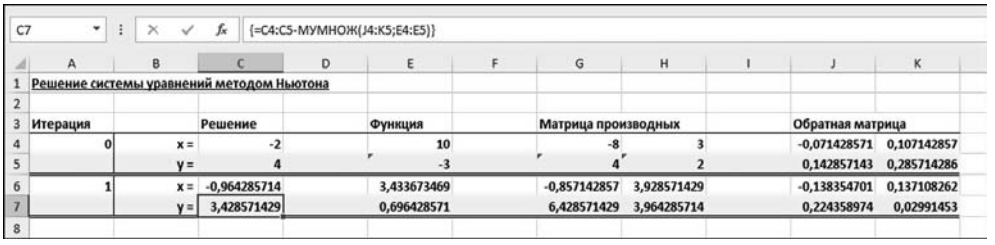
Формулы для вычисления функций уравнений системы, а также формулы для матрицы производных непосредственно вводятся в ячейки рабочего документа. Для вычисления решений системы создаем документ, представленный на рисунке 12.36.

На начальном этапе предстоит заполнить достаточно много ячеек. Описание того, как заполняются ячейки, можно найти в таблице 12.5.

Таким образом, столбец A содержит номера итераций (начиная с нулевого значения, соответствующего начальному приближению). В столбце C отображаются итерационные приближения для решения уравнения. Столбец E содержит значения функций, определяющих систему уравнений, в предполагаемых точках решения (столбец C). Кроме этого, в рабочем документе отображается для каждой итерации матрица производных (от функций, определяющих уравнения системы) и матрица, обратная к ней.

Откровенно говоря, количество заполняемых ячеек можно уменьшить — например, крайней необходимости в явном виде вычислять обратную матрицу для частных производных нет. Вместе с тем, если мы хотим контролировать вычислительный процесс поэтапно (а мы хотим), то информация о значении элементов обратной матрицы будет нелишней.

Процедура вычислений достаточно простая: выделяем диапазон ячеек, соответствующих первой итерации (ячейки A6:K7), и с помощью маркера автоматического заполнения копируем содержимое этих ячеек в ячейки снизу.



	A	B	C	D	E	F	G	H	I	J	K
1	Решение системы уравнений методом Ньютона										
2											
3	Итерация		Решение		Функция		Матрица производных			Обратная матрица	
4	0	x =	-2		10		-8	3		-0,071428571	0,107142857
5		y =	4		-3		4	2		0,142857143	0,285714286
6	1	x =	-0,964285714		3,433673469		-0,857142857	3,928571429		-0,138354701	0,137108262
7		y =	3,428571429		0,696428571		6,428571429	3,964285714		0,224358974	0,02991453
8											

Рис. 12.36
Документ для решения системы уравнений методом Ньютона перед началом вычисления

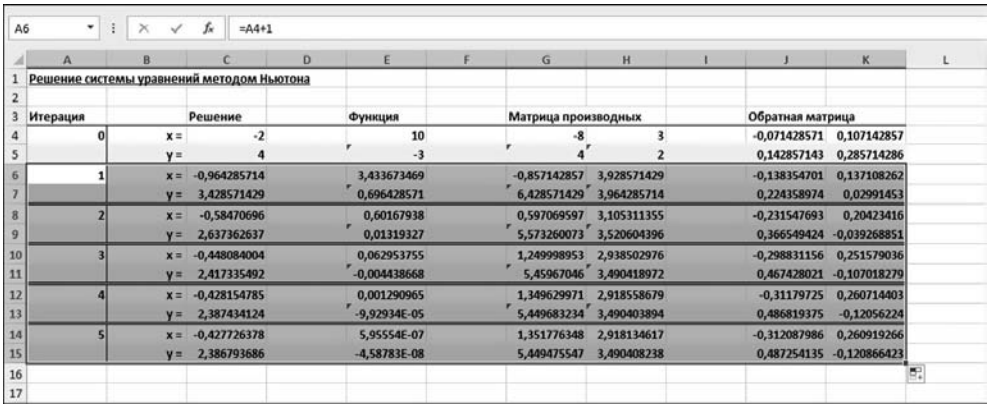
Таблица 12.5

Метод Ньютона для системы уравнений

Ячейка	Значение	Комментарий
A4	0	Начальная итерация
A6	=A4+1	Вычисление номера новой итерации на основе номера предыдущей итерации
C4	-2	Начальное приближение для переменной x
C5	4	Начальное приближение для переменной y
E4	=4*C4^2+C5^2+2*C4*C5-C5-2	Формула для вычисления значения функции первого уравнения системы $f_1(x, y) = 4x^2 + y^2 + 2xy - y - 2$ для нулевой итерации
E5	=2*C4^2+C5^2+3*C4*C5-3	Формула для вычисления функции второго уравнения системы $f_2(x, y) = 2x^2 + y^2 + 3xy - 3$ для нулевой итерации
G4	=8*C4+2*C5	Формула для вычисления частной производной $\frac{\partial f_1(x,y)}{\partial x} = 8x + 2y$ — элемента матрицы производных
H4	=2*C4+2*C5-1	Формула для вычисления частной производной $\frac{\partial f_1(x,y)}{\partial y} = 2x + 2y - 1$ — элемента матрицы производных
G5	=4*C4+3*C5	Формула для вычисления частной производной $\frac{\partial f_2(x,y)}{\partial x} = 4x + 3y$ — элемента матрицы производных
H5	=3*C4+2*C5	Формула для вычисления частной производной $\frac{\partial f_2(x,y)}{\partial y} = 3x + 2y$ — элемента матрицы производных
J4:K5	=МОБР(G4:H5)	Формула для вычисления обратной матрицы на основе матрицы производных (ячейки G4:H5). Формула вводится в диапазон ячеек J4:K5 как формула массива — нажатием комбинации клавиш <Ctrl>+<Shift>+<Enter>
C6:C7	=C4:C5-МУМНОЖ(J4:K5;E4:E5)	Вычисление приближенного значения для решения системы для новой итерации. Формула вводится как формула массива (комбинация клавиш <Ctrl>+<Shift>+<Enter>) в диапазон ячеек C6:C7
E6	=4*C6^2+C7^2+2*C6*C7-C7-2	Формула для вычисления значения функции первого уравнения для первой итерации. Формула копируется из ячейки E4
E7	=2*C6^2+C7^2+3*C6*C7-3	Формула для вычисления значения функции второго уравнения для первой итерации. Формула копируется из ячейки E5
G6	=8*C6+2*C7	Формула для вычисления частной производной от функции первого уравнения по первой переменной. Формула копируется из ячейки G4

Продолжение табл. 12.5

Ячейка	Значение	Комментарий
H6	=2*C6+2*C7-1	Формула для вычисления частной производной от функции первого уравнения по второй переменной. Формула копируется из ячейки H4
G7	=4*C6+3*C7	Формула для вычисления частной производной от функции второго уравнения по первой переменной. Формула копируется из ячейки G5
H7	=3*C6+2*C7	Формула для вычисления частной производной от функции второго уравнения по второй переменной. Формула копируется из ячейки H5
J6:K7	=МОБР(G6:H7)	Формула (массива) для вычисления обратной матрицы к матрице производных для первой итерации. Формула копируется из диапазона ячеек J4:K5



A6 =A4+1

	A	B	C	D	E	F	G	H	I	J	K	L
1	Решение системы уравнений методом Ньютона											
2												
3	Итерация		Решение		Функция		Матрица производных			Обратная матрица		
4	0	x =	-2		10		-8	3		-0,071428571	0,107142857	
5		y =	4		-3		4	2		0,142857143	0,285714286	
6	1	x =	-0,964285714		3,433673469		-0,857142857	3,928571429		-0,138354701	0,137108262	
7		y =	3,428571429		0,696428571		6,428571429	3,964285714		0,224358974	0,02991453	
8	2	x =	-0,58470696		0,60167938		0,597069597	3,105311355		-0,231547693	0,20423416	
9		y =	2,637362637		0,01319327		5,573260073	3,520604396		0,366549424	-0,039268851	
10	3	x =	-0,448084004		0,062953755		1,249998953	2,938502976		-0,298831156	0,251579036	
11		y =	2,417335492		-0,004438668		5,45967046	3,490418972		0,467428021	-0,107018279	
12	4	x =	-0,428154785		0,001290965		1,349629971	2,918558679		-0,31179725	0,260714403	
13		y =	2,387434124		-9,92934E-05		5,449683234	3,490403894		0,486819375	-0,12056224	
14	5	x =	-0,427726378		5,95554E-07		1,351776348	2,918134617		-0,312087986	0,260919266	
15		y =	2,386793686		-4,58783E-08		5,449475547	3,490408238		0,487254135	-0,120866423	
16												
17												

Рис. 12.37
Результат поиска решения системы уравнений методом Ньютона

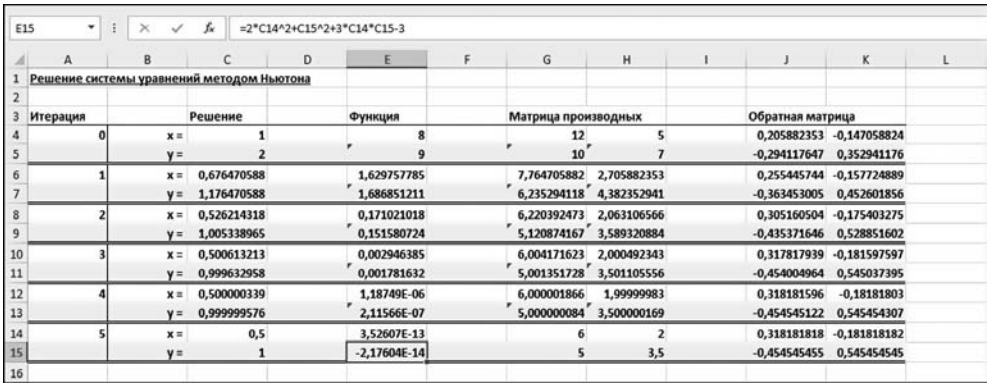
На заметку

В отличие от рассмотренных ранее примеров, в данном случае одной итерации соответствуют две строки в рабочем листе. Поэтому при копировании формул с помощью маркера автоматического заполнения необходимо «захватывать» парное количество строк.

На рисунке 12.37 показан результат вычисления решения системы уравнений по нескольким итерациям с начальными приближениями $x = -2$ и $y = 4$. В результате получаем приближенное решение $x \approx -0{,}428$ и $y \approx 2{,}387$.

На заметку

Насколько полученные оценки для решения системы приемлемы, можно судить по значениям в ячейках столбца E, где отображаются значения функции уравнений системы в точке предполагаемого решения. Чем меньше отличается от нуля значение ячеек в столбце E, тем точнее найдено решение. И хотя это не то же самое, что точность решения, тем не менее «корреляция» здесь очевидна.



E15 =2*C14^2+C15^2+3*C14*C15-3

	A	B	C	D	E	F	G	H	I	J	K	L
1	Решение системы уравнений методом Ньютона											
2												
3	Итерация		Решение		Функция		Матрица производных			Обратная матрица		
4	0	x =	1		8		12	5		0,205882353	-0,147058824	
5		y =	2		9		10	7		-0,294117647	0,352941176	
6	1	x =	0,676470588		1,629757785		7,764705882	2,705882353		0,255445744	-0,157724889	
7		y =	1,176470588		1,686851211		6,235294118	4,382352941		-0,363453005	0,452601856	
8	2	x =	0,526214318		0,171021018		6,220392473	2,063106566		0,305160504	-0,175403275	
9		y =	1,005338965		0,151580724		5,120874167	3,589320884		-0,435371646	0,528851602	
10	3	x =	0,500613213		0,002946385		6,004171623	2,000492343		0,317817939	-0,181597597	
11		y =	0,999632958		0,001781632		5,001351728	3,501105556		-0,454004964	0,545037395	
12	4	x =	0,500000339		1,18749E-06		6,000001866	1,99999983		0,318181596	-0,18181803	
13		y =	0,999999576		2,11566E-07		5,000000084	3,500000169		-0,454545122	0,545454307	
14	5	x =	0,5		3,52607E-13		6	2		0,318181818	-0,181818182	
15		y =	1		-2,17604E-14		5	3,5		-0,454545455	0,545454545	
16												

Рис. 12.38
Для вычисления еще одного решения системы изменяем значения начального приближения для решения

Чтобы найти другое решение системы (напомним, что рассматриваемая система имеет два решения), в ячейках C4:C5 изменяем значения для начального приближения для решения системы, как это показано на рисунке 12.38.

В данном случае для получения решения $x = 0{,}5$ и $y = 1$ мы вводим в ячейки C4 и C5 значения 1 и 2 соответственно.

МЕТОД ГРАДИЕНТНОГО СПУСКА

Почему вы меня все время роняете?

Из к/ф «Ирония судьбы, или С легким паром!»

К задаче поиска решения системы уравнений $f_k(x_1, x_2, \ldots, x_n) = 0$ (индекс $k = 1, 2, \ldots, n$) относительно переменных x_m (индекс $m = 1, 2, \ldots, n$) можно подойти с несколько иной стороны. А именно, рассмотрим выражение $\Phi(x_1, x_2, \ldots, x_n) = \sum_{k=1}^{n} f_k^2(x_1, x_2, \ldots, x_n)$, которое, очевидно, определяет некоторую функцию от переменных x_m (индекс $m = 1, 2, \ldots, n$). Что можно сказать о функции $\Phi(x_1, x_2, \ldots, x_n)$? Поскольку это сумма квадратов функций, определяющих систему уравнений, то, очевидно, функция $\Phi(x_1, x_2, \ldots, x_n)$ не может быть отрицательной. Другими словами, функция не может быть меньше нуля, причем нулю она равняется, только если каждая функция $f_k(x_1, x_2, \ldots, x_n) = 0$ для всех индексов $k = 1, 2, \ldots, n$. Следовательно, поиск решения системы уравнений можно свести к поиску минимума функции $\Phi(x_1, x_2, \ldots, x_n)$. Один из методов поиска минимума такой функции — метод градиентного (или скорейшего) спуска.

На заметку

В принципе, при «конструировании» функции для последующей минимизации можно взять некоторую положительно определенную матрицу $\hat{\mathbf{B}}$ с эле-

ментами b_{ij}, определив функцию для минимизации как $\Phi(x_1, x_2, \ldots, x_n) = \sum_{i,j=1}^{n} b_{ij} f_i(x_1, x_2, \ldots, x_n) f_j(x_1, x_2, \ldots, x_n)$. В матричном виде это же можно записать как $\Phi(\vec{x}) = \vec{f}^{\,T}(\vec{x})\hat{\mathbf{B}}\vec{f}(\vec{x})$, где $\vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \ldots, f_n(\vec{x}))^T$. Нелишним также будет напомнить, что положительно определенной называется симметричная матрица $\hat{\mathbf{B}}$ такая, что для любого отличного от нуля вектора $\vec{y}$ скаляр $\vec{y}^{\,T}\hat{\mathbf{B}}\vec{y} > 0$.

Для начала поиска решения по методу градиентного спуска необходимо задать начальное приближение для решения системы $\vec{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)})$. Выбор следующего приближения осуществляется в «направлении», противоположном к градиенту от функции $\Phi(\vec{x})$.

На заметку

Как известно, градиент от некоторой скалярной функции $u(x_1, x_2, \ldots, x_n)$ по определению — это составленный из частных производных вектор с компонентами $\left[\frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \ldots, \frac{\partial u}{\partial x_n}\right]$. В пространстве независимых переменных он определяет направление наибольшего возрастания функции $u(x_1, x_2, \ldots, x_n)$.

Если быть более точным, то рекуррентное соотношение, связывающее приближения $\vec{x}^{(k)}$ и $\vec{x}^{(k+1)}$ для решения системы на k-й и $(k+1)$-й итерациях выглядит так:

$$\vec{x}^{(k+1)} = \vec{x}^{(k)} - p_k \nabla\Phi(\vec{x}^{(k)}).$$

Здесь через ∇ обозначен оператор вычисления градиента от функции нескольких аргументов: по определению $\nabla\Phi(\vec{x}) = \left[\frac{\partial\Phi(\vec{x})}{\partial x_1}, \frac{\partial\Phi(\vec{x})}{\partial x_2}, \ldots, \frac{\partial\Phi(\vec{x})}{\partial x_n}\right]^T$. Параметр p_k определяется из условия минимума выражения $\Phi\left(\vec{x}^{(k)} - p_k\nabla\Phi(\vec{x}^{(k)})\right)$. Другими словами, необходимо решить уравнение $\frac{d\Phi\left(\vec{x}^{(k)} - p_k\nabla\Phi(\vec{x}^{(k)})\right)}{dp_k} = 0$ относительно параметра p_k. В общем случае это значительно усложняет расчеты, поэтому вместо решения трансцендентного уравнения относительно параметра p_k обычно используют более простой метод оценки этого параметра, основанный на линейном разложении функциональных зависимостей по параметру p_k. В частности, используем такое приближение:

$$f_m(\vec{x}^{(k)} - p\nabla\Phi(\vec{x}^{(k)})) \approx f_m(\vec{x}^{(k)}) - p\sum_{i=1}^{n}\frac{\partial f_m(\vec{x}^{(k)})}{\partial x_i}\frac{\partial\Phi(\vec{x}^{(k)})}{\partial x_i}.$$

Учитывая, что по определению $\Phi(\vec{x}) = \sum_{m=1}^{n} f_m^2(\vec{x})$, можем записать $\frac{\partial\Phi(\vec{x})}{\partial x_i} = 2\sum_{m=1}^{n} f_m(\vec{x})\frac{\partial f_m(\vec{x})}{\partial x_i}$. Если перейти к векторно-матричному представлению, то приведенные выше соотношения можно записать следующим образом: $\Phi(\vec{x}) = \vec{f}^{\,T}(\vec{x})\vec{f}(\vec{x})$ и $\nabla\Phi(\vec{x}) = 2\hat{\mathbf{A}}(\vec{x})^T\vec{f}(\vec{x})$. Здесь, как и ранее, использованы

обозначения $\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_n(\vec{x})]^T$, символ T означает транспонирование, а через $\hat{\mathbf{A}}(\vec{x})$ обозначена матрица производных с элементами $\frac{\partial f_i(\vec{x})}{\partial x_j}$ (индексы $i, j = 1, 2, \ldots, n$). Тогда линейное разложение по параметру p может быть представлено как $\vec{f}(\vec{x}^{(k)} - p\nabla\Phi(\vec{x}^{(k)})) \approx \vec{f}(\vec{x}^{(k)}) - p\hat{\mathbf{A}}\nabla\Phi(\vec{x}^{(k)})$. Тогда

$$\Phi\left(\vec{x}^{(k)} - p\nabla\Phi(\vec{x}^{(k)})\right) = \vec{f}^{\,2}\left(\vec{x}^{(k)} - p\nabla\Phi(\vec{x}^{(k)})\right) \approx$$
$$\approx \vec{f}^{\,2}\left(\vec{x}^{(k)}\right) - 4p\left(\hat{\mathbf{A}}^T(\vec{x}^{(k)})\vec{f}(\vec{x}^{(k)})\right)^2 + p^2\left(\hat{\mathbf{A}}(\vec{x}^{(k)})\hat{\mathbf{A}}^T(\vec{x}^{(k)})\vec{f}(\vec{x}^{(k)})\right)^2.$$

Дифференцируя это выражение по параметру p и приравнивая полученное значение к нулю, получаем значение для этого параметра для k-й итерации: $p_k = \frac{1}{2}\frac{\left(\hat{\mathbf{A}}^T(\vec{x}^{(k)})\vec{f}(\vec{x}^{(k)})\right)^2}{\left(\hat{\mathbf{A}}(\vec{x}^{(k)})\hat{\mathbf{A}}^T(\vec{x}^{(k)})\vec{f}(\vec{x}^{(k)})\right)^2}$. При этом операцию возведения вектора в квадрат мы понимаем в смысле скалярного произведения транспонированного вектора на самого себя: $\vec{y}^2 \equiv \vec{y}^T\vec{y}$.

Подытоживая, можем выделить основные позиции в алгоритме поиска решения системы методом градиентного спуска:

- выбирается начальное приближение $\vec{x}^{(0)}$ для решения системы;
- вычисляем матрицу производных $\hat{\mathbf{A}}(\vec{x}^{(0)})$, транспонированную матрицу производных $\hat{\mathbf{A}}^T(\vec{x}^{(0)})$, а также элементы вектор-функции $\vec{f}(\vec{x}^{(0)})$ в точке начального приближения для решения системы;
- по формуле $p_0 = \frac{1}{2}\frac{\left(\hat{\mathbf{A}}^T(\vec{x}^{(0)})\vec{f}(\vec{x}^{(0)})\right)^2}{\left(\hat{\mathbf{A}}(\vec{x}^{(0)})\hat{\mathbf{A}}^T(\vec{x}^{(0)})\vec{f}(\vec{x}^{(0)})\right)^2}$ вычисляем значение параметра p для данной итерации;
- по формуле $\vec{x}^{(1)} = \vec{x}^{(0)} - p_0\nabla\Phi(\vec{x}^{(0)})$ вычисляем следующее приближение для решения системы уравнений. При этом используем соотношение $\nabla\Phi(\vec{x}^{(0)}) = 2\hat{\mathbf{A}}(\vec{x}^{(0)})^T\vec{f}(\vec{x}^{(0)})$;
- после того как первое приближение вычислено, оно используется для вычисления второго приближения, и все перечисленные выше действия выполняются снова, но уже для первого приближения и т. д.

Учитывая все перечисленные обстоятельства, можем прийти к выводу, что для реализации метода градиентного спуска в рабочем листе Excel нам нужно предусмотреть возможность расчета функций, определяющих урав-

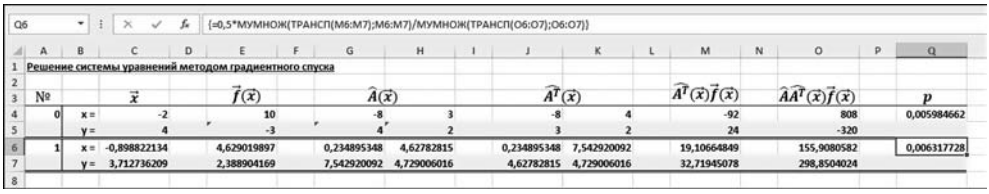


Рис. 12.39
Документ для поиска решения системы уравнений методом градиентного спуска (перед началом вычислений)

нения системы, а также матрицу производных. Все прочие параметры вычисляются на их основе. В этом отношении схема реализации в рабочем листе Excel метода градиентного спуска имеет много общего со схемой реализации метода Ньютона (хотя в первом вычислений все-таки значительно больше). На рисунке 12.39 показан документ для поиска решения системы уравнений методом градиентного спуска.

Мы будем решать ту же самую систему уравнений, что и в предыдущем примере (т. е. систему $4x^2 + y^2 + 2xy - y - 2 = 0$ и $2x^2 + y^2 + 3xy - 3 = 0$). Только теперь, разумеется, вместо метода Ньютона используем метод градиентного спуска. Поэтому вычисление функций уравнения и матрицы производных носит тот же характер, как это было для метода Ньютона. Для удобства мы также в явном виде вычисляем транспонированную матрицу производных $\hat{\mathbf{A}}^T(\vec{x})$, вектор $\hat{\mathbf{A}}^T(\vec{x})\vec{f}(\vec{x})$ — результат произведения транспонированной матрицы производных на вектор-функцию системы уравнений, а также вектор $\hat{\mathbf{A}}(\vec{x})\hat{\mathbf{A}}^T(\vec{x})\vec{f}(\vec{x})$ — результат произведения матрицы производных на транспонированную матрицу производных и на вектор-функцию системы уравнений, и, разумеется, параметр p. Вводимые в ячейки рабочего документа формулы и значения описаны в таблице 12.6.

Здесь обращает на себя внимание блок формул, связанных с вычислением матриц и векторов. Например, для транспонирования матрицы производных используем встроенную функцию ТРАНСП(). Также важно помнить, что возведение вектора (речь идет о векторе-столбике) в квадрат — это скалярное произведение транспонированного вектора на нетранспонированный вектор. Поэтому некоторые формулы выглядят достаточно громоздкими, хотя на самом деле предназначены для вычисления несложных выражений.

На заметку

При вычислении очередного приближения для решения системы на основе рекуррентного соотношения $\vec{x}^{(k+1)} = \vec{x}^{(k)} - p_k \nabla\Phi(\vec{x}^{(k)})$ мы воспользовались тем обстоятельством, что $\nabla\Phi(\vec{x}) = 2\hat{\mathbf{A}}(\vec{x})^T\vec{f}(\vec{x})$. Таким образом, на практике использовалось рекуррентное соотношение $\vec{x}^{(k+1)} = \vec{x}^{(k)} - 2p_k\hat{\mathbf{A}}(\vec{x}^{(k)})^T\vec{f}(\vec{x}^{(k)})$.

На рисунке 12.40 показан результат вычисления решения системы уравнений методом градиентного спуска.

Технология вычислений — традиционная: выделяем диапазон ячеек A6:Q7 и с помощью маркера автоматического заполнения захватываем область снизу под ячейками выделенного диапазона. Как и в предыдущем примере, захватывается парное количество строк в рабочем документе.

Что можно отметить, исследуя приведенный на рисунке 12.40 документ? Очевидно, что решение найдено. Также несложно заметить, что итераций потребовалось больше, чем при использовании метода Ньютона.

На заметку

Обратите внимание, что в документе на рисунке 12.40 строки с 16-й по 49-ю включительно скрыты.

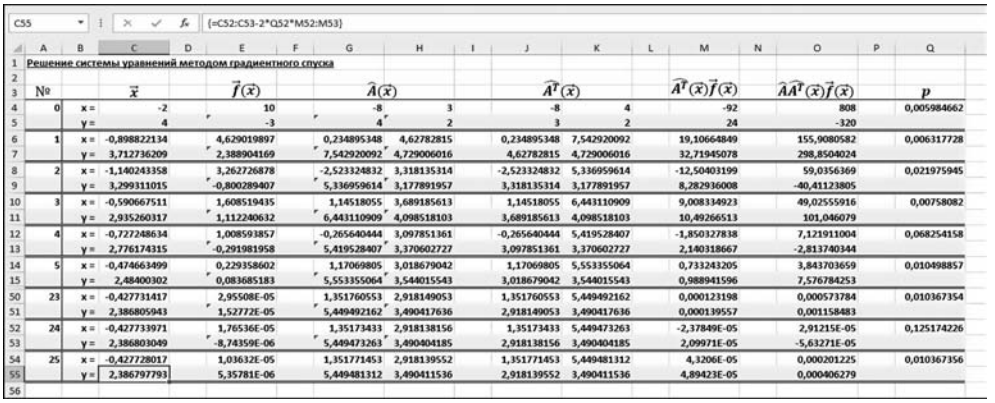
Таблица 12.6

Метод градиентного спуска

Ячейка	Значение	Комментарий
A4	0	Начальная итерация
A6	=A4+1	Номер новой итерации — вычисляется на основе номера предыдущей итерации
C4	–2	Начальное приближение для переменной x
C5	4	Начальное приближение для переменной y
E4	=4*C4^2+C5^2+2*C4*C5-C5-2	Вычисление функции $f_1(x, y) = 4x^2 + y^2 + 2xy - y - 2$ (определяет первое уравнение системы) для нулевой итерации
E5	=2*C4^2+C5^2+3*C4*C5-3	Вычисления функции $f_2(x, y) = 2x^2 + y^2 + 3xy - 3$ (второе уравнение системы) для нулевой итерации
G4	=8*C4+2*C5	Частная производная $\frac{\partial f_1(x,y)}{\partial x} = 8x + 2y$
H4	=2*C4+2*C5-1	Частная производная $\frac{\partial f_1(x,y)}{\partial y} = 2x + 2y - 1$
G5	=4*C4+3*C5	Частная производная $\frac{\partial f_2(x,y)}{\partial x} = 4x + 3y$
H5	=3*C4+2*C5	Частная производная $\frac{\partial f_2(x,y)}{\partial y} = 3x + 2y$
J4:K5	=ТРАНСП(G4:H5)	Транспонированная матрица производных $\hat{\mathbf{A}}^T(\vec{x})$. Вводится как формула массива
M4:M5	=МУМНОЖ(J4:K5;E4:E5)	Вектор $\hat{\mathbf{A}}^T(\vec{x})\vec{f}(\vec{x})$. Вводится как формула массива
O4:O5	=МУМНОЖ(МУМНОЖ (G4:H5;J4:K5);E4:E5)	Вектор $\hat{\mathbf{A}}(\vec{x})\hat{\mathbf{A}}^T(\vec{x})\vec{f}(\vec{x})$. Вводится как формула массива
Q4	=0,5*МУМНОЖ(ТРАНСП(M4:M5);M4:M5)/МУМНОЖ(ТРАНСП(O4:O5);O4:O5)	Вычисление параметра $p = 0,5 \cdot \frac{\left(\hat{\mathbf{A}}^T(\vec{x})\vec{f}(\vec{x})\right)^T\left(\hat{\mathbf{A}}^T(\vec{x})\vec{f}(\vec{x})\right)}{\left(\hat{\mathbf{A}}(\vec{x})\hat{\mathbf{A}}^T(\vec{x})\vec{f}(\vec{x})\right)^T\left(\hat{\mathbf{A}}(\vec{x})\hat{\mathbf{A}}^T(\vec{x})\vec{f}(\vec{x})\right)}$. Формула вводится как формула массива
C6:C7	=C4:C5-2*Q4*M4:M5	Решение системы для первой итерации. Вводится как формула массива
E6	=4*C6^2+C7^2+2*C6*C7-C7-2	Значение функции первого уравнения (первая итерация). Формула копируется из ячейки E4
E7	=2*C6^2+C7^2+3*C6*C7-3	Значение функции второго уравнения (первая итерация). Формула копируется из ячейки E5
G6	=8*C6+2*C7	Частная производная (формула копируется из ячейки G4)
H6	=2*C6+2*C7-1	Частная производная (формула копируется из ячейки H4)
G7	=4*C6+3*C7	Частная производная (формула копируется из ячейки G5)

Продолжение табл. 12.6

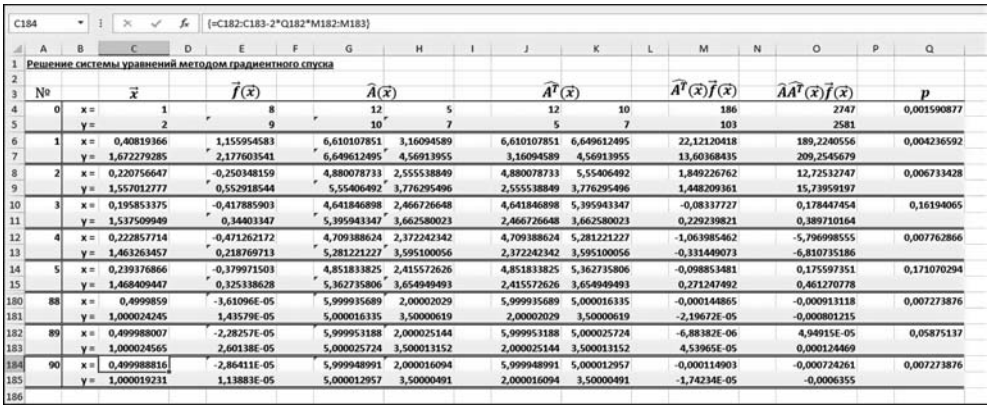
Ячейка	Значение	Комментарий
H7	=3*C6+2*C7	Частная производная (формула копируется из ячейки H5)
J6:K7	=ТРАНСП(G6:H7)	Транспонированная матрица производных (формула массива копируется из диапазона ячеек J4:K5)
M6:M7	=МУМНОЖ(J6:K7;E6:E7)	Вектор $\hat{\mathbf{A}}^T(\vec{x})\vec{f}(\vec{x})$ для первой итерации (формула массива копируется из ячеек M4:M5)
O6:O7	=МУМНОЖ(МУМНОЖ(G6:H7;J6:K7); E6:E7)	Вектор $\hat{\mathbf{A}}(\vec{x})\hat{\mathbf{A}}^T(\vec{x})\vec{f}(\vec{x})$ для первой итерации (формула массива копируется из ячеек O4:O5)
Q6	=0,5*МУМНОЖ(ТРАНСП(M6:M7); M6:M7)/ МУМНОЖ (ТРАНСП(O6:O7);O6:O7)	Вычисление параметра p для первой итерации (формула массива копируется из ячейки Q4)



C55 {=C52:C53-2*Q52*M52:M53}

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Решение системы уравнений методом градиентного спуска																
2																	
3	№		$\vec{x}$		$\vec{f}(\vec{x})$		$\hat{A}(\vec{x})$			$\hat{A}^T(\vec{x})$			$\hat{A}^T(\vec{x})\vec{f}(\vec{x})$		$\hat{A}\hat{A}^T(\vec{x})\vec{f}(\vec{x})$		p
4	0	x =	-2		10		-8	3		-8	4		-92		808		0,005984662
5		y =	4		-3		4	2		3	2		24		-320		
6	1	x =	-0,898822134		4,629019897		0,234895348	4,62782815		0,234895348	7,542920092		19,10664849		155,9080582		0,006317728
7		y =	3,712736209		2,388904169		7,542920092	4,729006016		4,62782815	4,729006016		32,71945078		298,8504024		
8	2	x =	-1,140243358		3,262726878		-2,523324832	3,318135314		-2,523324832	5,336959614		-12,50403199		59,0356369		0,021975945
9		y =	3,299311015		-0,800289407		5,336959614	3,177891957		3,318135314	3,177891957		8,282936008		-40,41123805		
10	3	x =	-0,590667511		1,608519435		1,14518055	3,689185613		1,14518055	6,443110909		9,008334923		49,02555916		0,00758082
11		y =	2,935260317		1,112240632		6,443110909	4,098518103		3,689185613	4,098518103		10,49266513		101,046079		
12	4	x =	-0,727248634		1,008593857		-0,265640444	3,097851361		-0,265640444	5,419528407		-1,850327838		7,121911004		0,068254158
13		y =	2,776174315		-0,291981958		5,419528407	3,370602727		3,097851361	3,370602727		2,140318667		-2,813740344		
14	5	x =	-0,474663499		0,229358602		1,17069805	3,018679042		1,17069805	5,553355064		0,733243205		3,843703659		0,010498857
15		y =	2,48400302		0,083685183		5,553355064	3,544015543		3,018679042	3,544015543		0,988941596		7,576784253		
50	23	x =	-0,427731417		2,95508E-05		1,351760553	2,918149053		1,351760553	5,449492162		0,000123198		0,000573784		0,010367354
51		y =	2,386805943		1,52772E-05		5,449492162	3,490417636		2,918149053	3,490417636		0,000139557		0,001158483		
52	24	x =	-0,427733971		1,76536E-05		1,35173433	2,918138156		1,35173433	5,449473263		-2,37849E-05		2,91215E-05		0,125174226
53		y =	2,386803049		-8,74359E-06		5,449473263	3,490404185		2,918138156	3,490404185		2,09971E-05		-5,63271E-05		
54	25	x =	-0,427728017		1,03632E-05		1,351771453	2,918139552		1,351771453	5,449481312		4,3206E-05		0,000201225		0,010367356
55		y =	2,386797793		5,35781E-06		5,449481312	3,490411536		2,918139552	3,490411536		4,89423E-05		0,000406279		
56																	

Рис. 12.40
Результат вычисления решения системы уравнений методом градиентного спуска (часть ячеек скрыта)



C184 {=C182:C183-2*Q182*M182:M183}

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Решение системы уравнений методом градиентного спуска																
2																	
3	№		$\vec{x}$		$\vec{f}(\vec{x})$		$\hat{A}(\vec{x})$			$\hat{A}^T(\vec{x})$			$\hat{A}^T(\vec{x})\vec{f}(\vec{x})$		$\hat{A}\hat{A}^T(\vec{x})\vec{f}(\vec{x})$		p
4	0	x =	1		8		12	5		12	10		186		2747		0,001590877
5		y =	2		9		10	7		5	7		103		2581		
6	1	x =	0,40819366		1,155954583		6,610107851	3,16094589		6,610107851	6,649612495		22,12120418		189,2240556		0,004236592
7		y =	1,672279285		2,177603541		6,649612495	4,56913955		3,16094589	4,56913955		13,60368435		209,2545679		
8	2	x =	0,220756647		-0,250348159		4,880078733	2,555538849		4,880078733	5,55406492		1,849226762		12,72532747		0,006733428
9		y =	1,557012777		0,552918544		5,55406492	3,776295496		2,555538849	3,776295496		1,448209361		15,73959197		
10	3	x =	0,195853375		-0,417885903		4,641846898	2,466726648		4,641846898	5,395943347		-0,08337727		0,178447454		0,16194065
11		y =	1,537509949		0,34403347		5,395943347	3,662580023		2,466726648	3,662580023		0,229239821		0,389710164		
12	4	x =	0,222857714		-0,471262172		4,709388624	2,372242342		4,709388624	5,281221227		-1,063985462		-5,796998555		0,007762866
13		y =	1,463263457		0,218769713		5,281221227	3,595100056		2,372242342	3,595100056		-0,331449073		-6,810735186		
14	5	x =	0,239376866		-0,379971503		4,851833825	2,415572626		4,851833825	5,362735806		-0,098853481		0,175597351		0,171070294
15		y =	1,468409447		0,325338628		5,362735806	3,654949493		2,415572626	3,654949493		0,271247492		0,461270778		
180	88	x =	0,4999859		-3,61096E-05		5,999935689	2,00002029		5,999935689	5,000016335		-0,000144865		-0,000913118		0,007273876
181		y =	1,000024245		1,43579E-05		5,000016335	3,50000619		2,00002029	3,50000619		-2,19672E-05		-0,000801215		
182	89	x =	0,499988007		-2,28257E-05		5,999953188	2,000025144		5,999953188	5,000025724		-6,88382E-06		4,94915E-05		0,05875137
183		y =	1,000024565		2,60138E-05		5,000025724	3,500013152		2,000025144	3,500013152		4,53965E-05		0,000124469		
184	90	x =	0,499988816		-2,86411E-05		5,999948991	2,000016094		5,999948991	5,000012957		-0,000114903		-0,000724261		0,007273876
185		y =	1,000019231		1,13883E-05		5,000012957	3,50000491		2,000016094	3,50000491		-1,74234E-05		-0,0006355		
186																	

Рис. 12.41
Вычисление решения системы с другим начальным приближением

Для вычисления другого корня в ячейках C4:C5 изменяем значения для начального приближения решения. Например, на рисунке 12.41 проиллюстрирован процесс вычисления решения на основе начального приближения 1 (для переменной x) и 2 (для переменной y).

Для получения решения $x = 0{,}5$ и $y = 1$ понадобилось довольно приличное количество итераций — во всяком случае, намного больше, чем при использовании метода Ньютона для той же самой системы с тем же самым начальным приближением (рис. 12.38). В этом смысле метод градиентного спуска не всегда является оптимальным.

МЕТОДЫ ЛИНЕЙНОЙ АЛГЕБРЫ

— Я надеюсь, вы не собираетесь музицировать?
— Ага, петь хочется!

Из к/ф «Служебный роман»

Существует ряд задач линейной алгебры, имеющих большое с практической точки зрения значение — они достаточно часто встречаются на практике в той или иной постановке. Здесь мы кратко рассмотрим задачу о решении системы линейных уравнений и задачу о вычислении собственных чисел матрицы.

Перечисленные выше задачи достаточно нетривиальны, однако благодаря наличию в Excel встроенных функций для вычисления обратных матриц и определителей, их решение в Excel особо большого труда не представляет.

Сначала рассмотрим систему линейных уравнений. В общем случае имеем дело с уравнениями $a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1$, …, $a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n$, где через a_{ij} (индексы $i, j = 1, 2, \ldots, n$) обозначены числовые коэффициенты линейных уравнений, а b_i (индекс $i = 1, 2, \ldots, n$) определяют правые части этих уравнений. Система уравнений решается относительно независимых переменных $x_1, x_2, \ldots, x_n$. В матричной форме система линейных алгебраических уравнений может быть записана в виде $\hat{\mathbf{A}}\vec{x} = \vec{B}$, где матрица коэффициентов системы уравнений $\hat{\mathbf{A}}$ формируется элементами a_{ij} (индексы $i, j = 1, 2, \ldots, n$), вектор правой части системы $\vec{B} = \left[b_1, b_2, \ldots, b_n\right]^T$, а вектор неизвестных переменных $\vec{x} = \left[x_1, x_2, \ldots, x_n\right]^T$. Векторно-матричная форма записи системы линейных уравнений удобна, поскольку позволяет сразу записать (хотя и несколько формально) решение системы в виде $\vec{x} = \hat{\mathbf{A}}^{-1}\vec{B}$. Здесь через $\hat{\mathbf{A}}^{-1}$ обозначена матрица, обратная к матрице коэффициентов системы уравнений $\hat{\mathbf{A}}$. Учитывая, что в Excel, как отмечалось выше, есть функция МОБР() для вычисления обратных матриц, процесс поиска решения системы алгебраических уравнений превращается в простую и формальную процедуру. Проиллюстрируем это на примере: решим в рабочем документе Excel систему из трех уравнений: $3x_1 - 2x_2 + x_3 = 2$; $5x_1 + x_2 - 3x_3 = -2$ и $-x_1 + x_2 + 2x_3 = 7$. Точное решение этой системы уравнений: $x_1 = 1$, $x_2 = 2$ и $x_3 = 3$. Рассмотрим документ на рисунке 12.42.

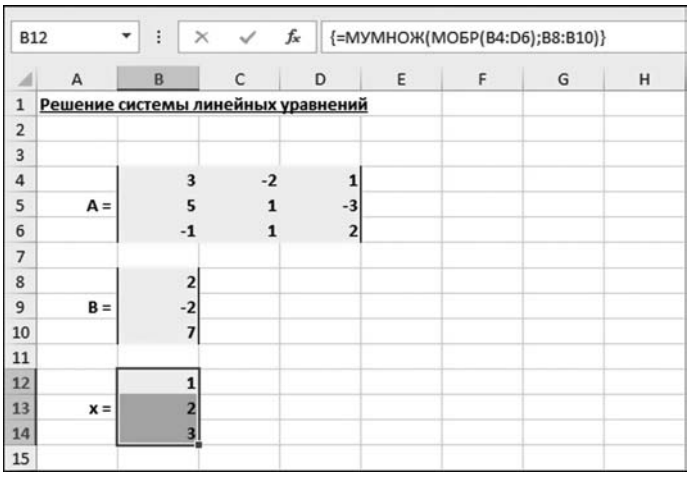


Рис. 12.42
Решение системы линейных уравнений

В ячейки B4:D6 этого документа вводятся элементы матрицы коэффициентов системы уравнений. В ячейки B8:B10 вводим коэффициенты правых частей системы уравнений. Решение системы уравнений вычисляется в ячейках B12:B14 по формуле =МУМНОЖ(МОБР(B4:D6);B8:B10), которая вводится как формула массива (комбинация клавиш <Ctrl>+<Shift>+<Enter>). Как видим, результат вычислен корректно.

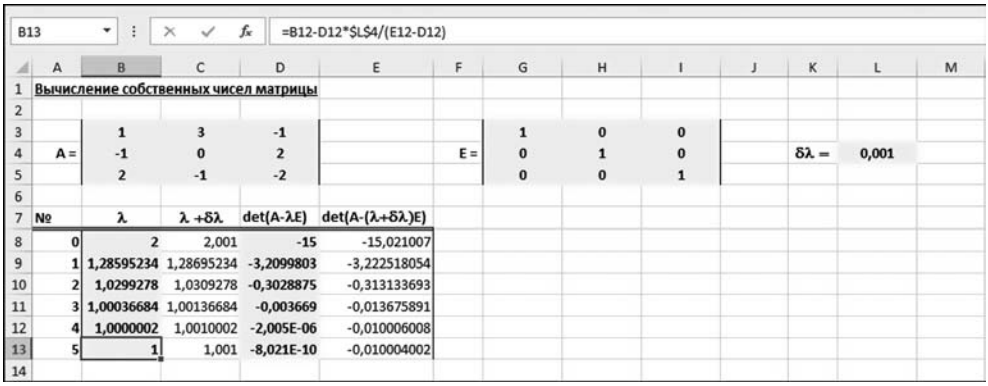
Несколько более сложной является задача о вычислении собственных чисел матрицы. Кратко задача формулируется следующим образом. Имеется матрица (квадратная) $\hat{\mathbf{A}}$ Рассматривается соотношение $\hat{\mathbf{A}}\vec{u} = \lambda\vec{u}$, где λ — число, а $\vec{u}$ — не равный нулю вектор. Данное соотношение выполняется далеко не для всех чисел λ, и не для всех векторов $\vec{u}$. Задача состоит в том, чтобы найти такие числа и соответствующие векторы, для которых выполняется соотношение $\hat{\mathbf{A}}\vec{u} = \lambda\vec{u}$. Эту задачу можно разделить на два этапа: вычисление собственных чисел и вычисление собственных векторов, соответствующих этим собственным числам.

На заметку

Одному собственному числу (в случае кратности) могут соответствовать несколько собственных векторов. Также следует учесть, что собственные числа могут быть комплексными. Все эти случаи мы здесь рассматривать не будем. Как отмечалось выше, мы ограничимся рассмотрением задачи о вычислении собственных чисел матрицы.

Для определения собственных чисел матрицы $\hat{\mathbf{A}}$ относительно λ решается уравнение $\det(\hat{\mathbf{A}} - \lambda\hat{\mathbf{E}}) = 0$. Здесь через $\hat{\mathbf{E}}$ обозначена единичная матрица того же ранга, что и матрица $\hat{\mathbf{A}}$.

Выражение $\det(\hat{\mathbf{A}} - \lambda\hat{\mathbf{E}})$ представляет собой полином относительно переменной λ. Поэтому при поиске собственных чисел матрицы $\hat{\mathbf{A}}$, речь, фактически,



B13 =B12-D12*$L$4/(E12-D12)

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Вычисление собственных чисел матрицы												
2													
3		1	3	-1			1	0	0				
4	A =	-1	0	2		E =	0	1	0		δλ =	0,001	
5		2	-1	-2			0	0	1				
6													
7	№	λ	λ +δλ	det(A-λE)	det(A-(λ+δλ)E)								
8	0	2	2,001	-15	-15,021007								
9	1	1,28595234	1,28695234	-3,2099803	-3,222518054								
10	2	1,0299278	1,0309278	-0,3028875	-0,313133693								
11	3	1,00036684	1,00136684	-0,003669	-0,013675891								
12	4	1,0000002	1,0010002	-2,005E-06	-0,010006008								
13	5	1	1,001	-8,021E-10	-0,010004002								
14													

Рис. 12.43
Вычисление собственного числа матрицы

идет о решении полиномиального уравнения (которое называется *характеристическим уравнением*). Некоторая проблема связана с тем, что коэффициенты соответствующего полинома в явном виде не заданы. Благо, при работе с Excel все эти проблемы проблемами не являются.

Дальше рассмотрим пример вычисления собственных чисел матрицы. Для определенности рассмотрим такую матрицу:

$$\hat{\mathbf{A}} = \begin{bmatrix} 1 & 3 & -1 \\ -1 & 0 & 2 \\ 2 & -1 & -2 \end{bmatrix}.$$

У этой матрицы три собственных числа: $\lambda_1 = 1$, $\lambda_{2,3} = -1 \pm i\sqrt{6}$. Таким образом, у матрицы из трех собственных чисел всего одно действительное. Его и попытаемся найти. Для этого используем документ, представленный на рисунке 12.43.

В ячейки B3:D5 рабочего документа вводятся числовые коэффициенты матрицы $\hat{\mathbf{A}}$. В ячейки G3:I5 вводим (для удобства дальнейших вычислений) элементы единичной матрицы $\hat{\mathbf{E}}$: по диагонали — единицы, а все недиагональные элементы равны нулю. Кроме того, поскольку мы планируем использовать для решения характеристического уравнения метод Ньютона, в ячейку L4 вводится числовое значение для приращения параметра — нам предстоит вычислять производную от функции уравнения, и вычислять ее мы будем в числовом виде как отношение приращения функции к приращению аргумента.

На заметку

Как отмечалось, для поиска собственных чисел матрицы $\hat{\mathbf{A}}$ относительно параметра λ решается алгебраическое уравнение $\det(\hat{\mathbf{A}} - \lambda\hat{\mathbf{E}}) = 0$. Если обозначить $P(\lambda) \equiv \det(\hat{\mathbf{A}} - \lambda\hat{\mathbf{E}})$, то нам предстоит решить уравнение $P(\lambda) = 0$. Рекуррентное соотношение для решения этого уравнения методом Ньютона будет выглядеть как $\lambda^{(k+1)} = \lambda^{(k)} - \dfrac{P(\lambda^{(k)})}{P'(\lambda^{(k)})}$, где через $P'(\lambda)$ обозначена произ-

водная от функции $P(\lambda)$ по параметру λ. Для производной мы используем оценку

$$P'(\lambda) \approx \frac{P(\lambda+\delta\lambda)-P(\lambda)}{\delta\lambda} = \frac{\det(\hat{\mathbf{A}}-(\lambda+\delta\lambda)\hat{\mathbf{E}})-\det(\hat{\mathbf{A}}-\lambda\hat{\mathbf{E}})}{\delta\lambda}.$$

Таким образом, для вычисления корня уравнения (собственного числа матрицы) используем рекуррентное соотношение

$$\lambda^{(k+1)}=\lambda^{(k)}-\frac{\det(\hat{\mathbf{A}}-\lambda^{(k)}\hat{\mathbf{E}})\delta\lambda}{\det(\hat{\mathbf{A}}-(\lambda^{(k)}+\delta\lambda)\hat{\mathbf{E}})-\det(\hat{\mathbf{A}}-\lambda^{(k)}\hat{\mathbf{E}})}.$$

На начальном этапе вычислений заполняются ячейки A8:E9 (нулевая и первая итерации), после чего выделяются ячейки A9:E9, и с помощью маркера заполнения формулы копируются в ячейки снизу. Какие значения и формулы вводятся в ячейки диапазона A8:E9, описано в таблице 12.7.

На заметку

Обратите внимание, что в формулах, вводимых в ячейки A8:E9, используются абсолютные ссылки на ячейки $B$3:$D$5 (ссылка на матрицу $\hat{\mathbf{A}}$), $G$3:$I$5 (ссылка на единичную матрицу $\hat{\mathbf{E}}$) и $L$4 (ссылка на ячейку с приращением для корня). Это делается для того, чтобы они (ссылки) не изменялись при копировании формул в ячейках. Кроме того, мы использовали встроенную функцию МОПРЕД() для вычисления определителя матрицы (матрица передается аргументом функции).

Таблица 12.7

Вычисление собственного числа матрицы

Ячейка	Значение	Комментарий
A8	0	Начальная итерация (номер)
A9	=A8+1	Вычисляем номер новой итерации
B8	2	Начальное приближение для корня уравнения — собственного числа матрицы
C8	=B8+$L$4	Вычисление значения «корня с приращением»: текущая оценка для корня плюс добавка (нулевая итерация)
D8	=МОПРЕД($B$3:$D$5-B8*$G$3:$I$5)	Значение определителя $\det(\hat{\mathbf{A}}-\lambda\hat{\mathbf{E}})$ в точке предполагаемого корня (нулевая итерация)
E8	=МОПРЕД($B$3:$D$5-C8*$G$3:$I$5)	Значение определителя $\det(\hat{\mathbf{A}}-(\lambda+\delta\lambda)\hat{\mathbf{E}})$ в точке предполагаемого «корня с приращением» (нулевая итерация). Формула копируется из ячейки D8
B9	=B8-D8*$L$4/(E8-D8)	Формула для вычисления нового приближения для корня на основе рекуррентного соотношения (метод Ньютона)
C9	=B9+$L$4	Вычисление значения «корня с приращением» (первая итерация). Формула копируется из ячейки C8
D9	=МОПРЕД($B$3:$D$5-B9*$G$3:$I$5)	Значение определителя $\det(\hat{\mathbf{A}}-\lambda\hat{\mathbf{E}})$ в точке предполагаемого корня (первая итерация). Формула копируется из ячейки D8
E9	=МОПРЕД($B$3:$D$5-C9*$G$3:$I$5)	Значение определителя $\det(\hat{\mathbf{A}}-(\lambda+\delta\lambda)\hat{\mathbf{E}})$ для первой итерации. Формула копируется из ячейки E8 или D9

ГЛАВА 13 ИНТЕРПОЛИРОВАНИЕ И АППРОКСИМАЦИЯ

Тот, кто нам мешает,
тот нам и поможет.

Из к/ф «Кавказская пленница»

В этой главе рассматриваются два класса задач: интерполирование и аппроксимация. Как будет показано далее, в обоих случаях приложение Excel может стать незаменимым помощником, благодаря которому решение соответствующих задач становится простым и где-то даже приятным. Традиционно, перед описанием методов и приемов применения приложения Excel для решения задач по интерполированию и аппроксимации функциональных зависимостей кратко рассмотрим теорию. Начнем с интерполирования функций.

В достаточно общем случае задача интерполирования может быть сформулирована следующим образом. Предположим, имеется набор узловых точек x_1, x_2, ..., x_n и набор значений y_1, y_2, ..., y_n некоторой функции в этих точках. Функция может быть как неизвестной (например, когда речь идет о результатах экспериментальных измерений), так и известной. Задача состоит в том, чтобы построить некоторую функциональную зависимость $f(x)$ такую, чтобы соответствующая кривая проходила через все точки (x_k, y_k) (индекс $k = 1, 2, ..., n$), т. е. необходимо, чтобы для всех индексов $k = 1, 2, ..., n$ выполнялись соотношения $f(x_k) = y_k$. Обычно предварительно задается аналитическое выражение для функции $f(x)$, которое содержит набор «подгоночных» параметров в количестве, достаточном для того, чтобы путем их подбора обеспечить выполнение всех условий вида $f(x_k) = y_k$.

На заметку

Обычно задача интерполирования решается для того, чтобы «восстановить» по набору дискретных данных аналитическую функциональную зависимость. Вместе с тем, нередко на практике приходится иметь дело с известными (заданными явно) функциональными зависимостями, настолько сложными однако, что их практическое применение представляется маловозможным. В подобных ситуация разумным выходом может быть интерполяция сложных аналитических зависимостей более простыми. При этом необходимая точность достигается за счет выбора схемы расположения узлов и их общего количества.

Нередко в качестве функции, на основе которой строится интерполяционная зависимость, выбирают полиномиальные выражения.

ИНТЕРПОЛЯЦИОННЫЙ ПОЛИНОМ ЛАГРАНЖА

Замечательная идея!
Что ж она мне самому в голову не пришла?

Из к/ф «Ирония судьбы, или С легким паром!»

Если интерполяционная зависимость строится в виде полинома, то для однозначного определения коэффициентов интерполяционного полинома (на основе соотношений вида $f(x_k) = y_k$) степень полинома должна быть на единицу меньше, чем количество точек, по которым выполняется интерполирование. В этом смысле если мы имеем дело с набором из n точек, то полином степени $n - 1$, проходящий через каждую из этих точек, определяется однозначно. Причина в том, что полином $P_{n-1}(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{n-1}x^{n-1}$ степени $n - 1$ содержит n коэффициентов a_0, a_1, …, a_n, которые однозначно определяются из n условий $P_{n-1}(x_k) = y_k$ (индекс $k = 1, 2, \ldots, n$). В этом смысле интрига связана лишь с тем, как технически вычисляются коэффициенты полинома, т. е. какой алгоритм использован для вычисления параметров интерполяционного полинома. Однако независимо от метода расчетов, результат в каждом из случаев должен быть один и тот же.

Существует несколько «классических» алгоритмов построения интерполяционного полинома, которые дают названия для получаемых в результате полиномов. Можно выделить два наиболее популярных: *полином Лагранжа* и *полином Ньютона*. В этом разделе исследуем вопрос о построении полинома по схеме Лагранжа (с применением Excel), а следующий раздел посвящен вопросу построения с помощью Excel полинома Ньютона.

Схема Лагранжа предполагает, что соответствующее полиномиальное выражение, построенное по точкам (x_0, y_0), (x_1, y_1), …, (x_n, y_n) ищется в виде

$$L_n(x) = y_0\varphi_0(x) + y_1\varphi_1(x) + \ldots + y_n\varphi_n(x) = \sum_{m=0}^{n} y_m\varphi_m(x).$$

На заметку

Обратите внимание, что в данном случае полином имеет степень n, поскольку строится по $n + 1$ точке: индексация точек (x_m, y_m) начинается с нуля, а последний индекс равен n.

Функции $\varphi_m(x)$ являются полиномами степени n, причем такими, что в узловых точках x_k имеют место соотношения: $\varphi_m(x_k) = 0$, если $k \neq m$, и $\varphi_m(x_k) = 1$, если $k = m$. Эти полиномы вычисляются в виде произведений

$$\varphi_m(x) = \frac{(x - x_0)(x - x_1)\cdots(x - x_{m-1})(x - x_{m+1})\cdots(x - x_n)}{(x_m - x_0)(x_m - x_1)\cdots(x_m - x_{m-1})(x_m - x_{m+1})\cdots(x_m - x_n)}$$

для всех индексов $m = 0, 1, 2, \ldots, n$.

На заметку

Если интерполяционный полином ищется в виде $L_n(x) = \sum_{m=0}^{n} y_m \varphi_m(x)$, при условии, что имеет место $\varphi_m(x_k) = \delta_{km}$ (символ Кронекера), то:

а) $L_n(x_k) = \sum_{m=0}^{n} y_k \delta_{km} = y_k$ — интерполяционная функция принимает «правильные» значения в узловых точках;

б) $L_n(x)$ является полиномом степени n — как линейная комбинация полиномиальных выражений $\varphi_m(x)$.

Для того чтобы вычислить полином $\varphi_m(x)$ с упомянутыми выше свойствами, его удобно представить в виде $\varphi_m(x) = \frac{\psi_m(x)}{\psi_m(x_m)}$, где $\psi_m(x)$ также является полиномом степени n, принимающим нулевые значения во всех узловых точках x_k, за исключением индекса $k = m$. Достаточно рассмотреть полином

$$\psi_m(x) = (x - x_0)(x - x_1)\ldots(x - x_{m-1})(x - x_{m+1})\ldots(x - x_n) = \prod_{\substack{k=0,\\ k\neq m}}^{n} (x - x_k).$$

Тогда имеет место очевидное соотношение $\varphi_m(x) = \prod_{\substack{k=0,\\ k\neq m}}^{n} \frac{(x - x_k)}{(x_m - x_k)}$.

Далее исследуем вопрос о том, как описанная выше схема вычисления полинома по методу Лагранжа может быть реализована в рабочем документе Excel. Обратимся к документу на рисунке 13.1.

В данном случае мы вычисляем интерполяционный полином по методу Лагранжа для табулированной по 11 равноудаленным узловым точкам (на интервале значений аргумента от -2π до 2π) функции $f(x) = \frac{\sin(x)}{1 + x^2}$. Данных в документе достаточно много, поэтому имеет смысл дать некоторую общую характеристику документа (а также последовательности заполнения ячеек документа), а уже затем перейти к обсуждению деталей.

На заметку

Обратите внимание, что некоторые строки в документе скрыты. Мы прибегли к этому приему, чтобы можно было оценить весь диапазон ячеек, в котором выполнялись вычисления: он очень большой (до 107 строк включительно), и если не скрыть некоторые строки, в области экрана не помещается.

В ячейках столбца A (начиная с ячейки A7) вводятся значения аргумента, для которых вычисляются значения интерполяционного полинома. Соседние ячейки (в столбце B) содержат значения интерполируемой функции. Необходимости ее вычислять, конечно, нет, но для сравнения значений интерполяционного полинома с «точными» значениями, соответствующие данные нам вполне пригодятся. Рядом с ячейками со значениями интерполируемой функции (столбец C) вычисляются значения интерполяционного полинома (построенного по методу Лагранжа).

На заметку

Значения точек (ячейки A7:A107), в которых вычисляется интерполяционный полином, вводятся в принципе произвольным образом в произвольном количестве — все определяется тем, для каких целей мы собираемся вычислять интерполяционный полином. В наши дальнейшие планы входит построить график для интерполяционного полинома и отобразить на узловые точки, через которые он проходит. Поэтому, как будет показано дальше, значения аргумента для интерполяционного полинома мы вычисляем: соответствующие точки равномерно распределены на интервале, на котором выполняется интерполяция, но частота этих точек значительно выше, чем частота узловых точек, на основе которых полином собственно и создается.

Ячейки D4:N4 содержат значения узловых точек, на основе которых мы создаем интерполяционный полином. Значения полинома в узловых точках отображаются в ячейках D5:N5. Также для удобства в ячейках D3:N3 отображаются номера (индексы) узловых точек — но, надо признать, функциональной нагрузки они в данном случае не несут.

Все основные вычисления выполняются в ячейках D7:N107 и, как результат, в ячейках C7:C107 удается вычислить значение интерполяционного полинома в точках, которые указаны в ячейках A7:A107. В ячейках D7:N107 содержатся значения функций $\varphi_m(x)$ для разных аргументов интерполяционного полинома x и разных индексов узловых точек m. В одной и той же строке диапазона ячеек D7:N107 находятся значения функций $\varphi_m(x)$ для одного и того же аргумента x, но разных индексов m. Напротив, в столбцах

C7 =СУММПРОИЗВ($D$5:$N$5;D7:N7)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Интерполяционный полином Лагранжа													
2														
3			Индекс узл. точки	0	1	2	3	4	5	6	7	8	9	10
4			Узловая точка	-6,28	-5,03	-3,77	-2,51	-1,26	0,00	1,26	2,51	3,77	5,03	6,28
5			Знач. функции	0,0000	0,0362	0,0386	-0,0803	-0,3687	0,0000	0,3687	0,0803	-0,0386	-0,0362	0,0000
6	Аргумент	Функция	Пол. Лагранжа											
7	-6,28	6,0533E-18	0,0000	1,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
12	-5,65	1,7824E-02	0,4781	0,1762	1,7620	-2,6430	4,2287	-5,2859	4,9335	-3,3638	1,6264	-0,5286	0,1036	-0,0093
17	-5,03	3,6208E-02	0,0362	0,0000	1,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
22	-4,40	4,6748E-02	-0,0711	-0,0093	0,2782	1,2519	-1,1128	1,1685	-1,0015	0,6491	-0,3035	0,0963	-0,0185	0,0016
27	-3,77	3,8639E-02	0,0386	0,0000	0,0000	1,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
32	-3,14	-4,1983E-16	0,0649	0,0016	-0,0273	0,3682	0,9819	-0,5728	0,4124	-0,2455	0,1091	-0,0335	0,0063	-0,0005
37	-2,51	-8,0336E-02	-0,0803	0,0000	0,0000	0,0000	1,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
42	-1,88	-2,0888E-01	-0,2812	-0,0005	0,0076	-0,0573	0,4582	0,8019	-0,3208	0,1604	-0,0655	0,0191	-0,0035	0,0003
47	-1,26	-3,6875E-01	-0,3687	0,0000	0,0000	0,0000	0,0000	1,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
52	-0,63	-4,2142E-01	-0,2580	0,0003	-0,0038	0,0238	-0,1057	0,5552	0,6662	-0,1851	0,0634	-0,0170	0,0029	-0,0002
57	0,00	5,5511E-15	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	1,0000	0,0000	0,0000	0,0000	0,0000	0,0000
62	0,63	4,2142E-01	0,2580	-0,0002	0,0029	-0,0170	0,0634	-0,1851	0,6662	0,5552	-0,1057	0,0238	-0,0038	0,0003
67	1,26	3,6875E-01	0,3687	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	1,0000	0,0000	0,0000	0,0000	0,0000
72	1,88	2,0888E-01	0,2812	0,0003	-0,0035	0,0191	-0,0655	0,1604	-0,3208	0,8019	0,4582	-0,0573	0,0076	-0,0005
77	2,51	8,0336E-02	0,0803	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	1,0000	0,0000	0,0000	0,0000
82	3,14	-6,0157E-16	-0,0649	-0,0005	0,0063	-0,0335	0,1091	-0,2455	0,4124	-0,5728	0,9819	0,3682	-0,0273	0,0016
87	3,77	-3,8639E-02	-0,0386	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	1,0000	0,0000	0,0000
92	4,40	-4,6748E-02	0,0711	0,0016	-0,0185	0,0963	-0,3035	0,6491	-1,0015	1,1685	-1,1128	1,2519	0,2782	-0,0093
97	5,03	-3,6208E-02	-0,0362	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	1,0000	0,0000
102	5,65	-1,7824E-02	-0,4781	-0,0093	0,1036	-0,5286	1,6264	-3,3638	4,9335	-5,2859	4,2287	-2,6430	1,7620	0,1762
107	6,28	2,5725E-16	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	1,0000
108														

Рис. 13.1
Вычисление интерполяционного полинома по методу Лагранжа

диапазона D7:N107 содержатся значения для разных аргументов x, но одного и того же индекса m. Аргументы, напомним, содержатся в ячейках столбца A (ячейки A7:A107), а индексы содержатся в строке 3 (ячейки D3:N3).

На заметку

Функции $\varphi_m(x)$ вычисляются построчно, а для начального (нулевого) и последнего (десятого) индексов вычисляются «персонально» — в соответствующие ячейки (в пределах одной строки) вводятся оригинальные формулы. Для заполнения остальных ячеек (в пределах строки) достаточно ввести формулу в одну ячейку, после чего скопировать эту формулу в прочие ячейки строки (кроме начальной ячейки в столбце D и конечной ячейки в столбце N). Сами формулы приводятся далее.

Последовательность заполнения ячеек рабочего документа может быть следующей.

- Заполняем ячейки с надписями и ячейки D3:N3 с индексами узловых точек (в ячейку D3 вводим нулевое значение, в ячейку E3 — формулу =D3+1, и копируем эту формулу в прочие ячейки диапазона).
- Заполняем диапазон ячеек D4:N4 значениями функции: для этого в ячейку D4 вводим формулу =SIN(D4)/(1+D4^2) и копируем ее в прочие ячейки диапазона.
- В ячейку A7 вводим формулу =D4 (левая граница интервала, на котором выполняется интерполирование, совпадает с начальной узловой точкой). В ячейку A8 вводится формула =A11+($N$4-$D$4)/100. В данном случае мы исходим из того, что интервал интерполирования разбивается на 100 одинаковых подынтервалов, длина каждого из которых, очевидно, в 100 раз меньше длины исходного интервала. Последнюю можно вычислить как разность значений конечной и начальной узловых точек.
- Формулу из ячейки A8 с помощью маркера заполнения копируем в нижние ячейки, вплоть до ячейки A107.
- В ячейку B7 вводим формулу =SIN(A7)/(1+A7^2) и на ее основе заполняем (перетаскиванием маркера автоматического заполнения) диапазон B7:B107.
- В ячейку D7 вводится формула =ПРОИЗВЕД($A7-$E$4:$N$4)/ПРОИЗВЕД($D$4-$E$4:$N$4). Формула вводится как формула массива — нажатием комбинации клавиш <Ctrl>+<Shift>+<Enter>.
- В ячейку N7 вводим формулу массива =ПРОИЗВЕД($A7-$D$4:$M$4)/ПРОИЗВЕД($N$4-$D$4:$M$4).
- В ячейку E7 вводится формула массива =ПРОИЗВЕД($A7-$D$4:D$4)/ПРОИЗВЕД(E$4-$D$4:D$4)*ПРОИЗВЕД($A7-F$4:$N$4)/ПРОИЗВЕД(E$4-F$4:$N$4), после чего она перетаскиванием маркера автоматического заполнения копируется в ячейки диапазона F7:M7.
- Для вычисления значения интерполяционного полинома в ячейку C7 вводим формулу =СУММПРОИЗВ($D$5:$N$5;D7:N7).
- Выделяем диапазон ячеек C7:N7, и, перетаскивая маркер автоматического заполнения, копируем содержимое этого диапазона в нижние ячейки —

Рис. 13.2
График табулируемой функции (штрихованная линия)
и интерполяционного полинома (сплошная линия)

вплоть до строки 107. Другими словами, необходимо выделить диапазон C7:N7 и маркером заполнения захватить диапазон C7:N017.

Несложно заметить, что вычисленные нами значения интерполяционного полинома в узловых точках совпадают со значениями табулируемой функции — как и должно быть. Что касается остальных точек, то совпадение не столь оптимистично. Исследовать ситуацию можно более наглядно, если обратиться к графикам. Их легко построить с помощью Excel: графики функции и интерполяционного полинома строятся на основе значений диапазона ячеек A7:C107. Кроме этого, в виде отдельных точек имеет смысл отобразить табулированные значения функции (диапазон ячеек D4:N5). Результат показан на рисунке 13.2.

Если в центральной части диапазона интерполирования интерполяционный полином достаточно «близок» к кривой, построенной на основе «точного» соотношения, то на границах интервала отклонение очень и очень значительное. К сожалению, такая ситуация (значительные осцилляции на границах интервала интерполирования) является характерной для полиномиальной интерполяции.

Хотя приведенные выше вычисления нельзя назвать слишком сложными, все же имеет место некоторое неудобство, связанное с необходимостью выделения большого массива ячеек для хранения промежуточных результатов вычислений. Избежать такого рода неприятностей можно, если воспользоваться VBA и написать несложный программный код — функцию для вычисления значения интерполяционного полинома на основе двух массивов данных: массива ячеек со значениями узловых точек и массива ячеек со значениями табулируемой функции в этих точках. Обратимся к программному коду в листинге 13.1.

Листинг 13.1. Функция для вычисления полинома Лагранжа

```
Function ЛАГРАНЖ(Px As Range, Py As Range, z As Variant) As Double
 ‘ Переменная для запоминания аргумента полинома
 Dim x As Double
 ‘ Значение аргумента полинома
  x = z
 ‘ Переменная для запоминания количества узловых точек
 Dim n As Integer
 ‘ Вычисляем количество узловых точек
 n = Px.Count
 ‘ Целочисленные переменные для операторов цикла
 Dim i As Integer, j As Integer
 ‘ Переменная для вычисления значения полинома
 Dim L As Double
 ‘ Начальное значение для полиномиальной суммы
  L = 0
 ‘ Переменная для вычисления произведения
 Dim phi As Double
 ‘ Внешний цикл
 For i = 1 To n
  ‘ Начальное значение для произведения
   phi = 1
  ‘ Первый внутренний цикл
   For j = 1 To i - 1
    ‘ Умножаем на разность аргумента и узловой точки
    phi = phi * (x - Px.Cells(j).Value)
    ‘ Делим на разность узловых точек
    phi = phi / (Px.Cells(i).Value - Px.Cells(j).Value)
  Next j
  ‘ Второй внутренний цикл
   For j = i + 1 To n
    ‘ Умножаем на разность аргумента и узловой точки
    phi = phi * (x - Px.Cells(j).Value)
    ‘ Делим на разность узловых точек
    phi = phi / (Px.Cells(i).Value - Px.Cells(j).Value)
  Next j
  ‘ К полиномиальной сумме добавляем очередное слагаемое
   L = L + phi * Py.Cells(i).Value
 Next i
 ‘ Результат вычислений
 ЛАГРАНЖ = L
End Function
```

Этим кодом определяется пользовательская функция ЛАГРАНЖ(), у которой три аргумента: диапазон ячеек со значениями узловых точек, диапазон ячеек со значениями интерполируемой функции в узловых точках, а так-

же аргумент, для которого вычисляется значение интерполяционного полинома Лагранжа. Функция возвращает числовой результат типа Double.

Поскольку третий аргумент функции, которым определяется точка, в которой вычисляется значение полинома Лагранжа, может быть как числовым значением, так и ссылкой на ячейку, в теле функции значение этого третьего аргумента функции записываем в числовую переменную x.

На заметку

Если третий аргумент — число, то процесс «переписывания» аргумента в локальную переменную особого смысла не имеет, хотя и вполне «законный». Если же третий аргумент — ссылка на ячейку, то присваивание значения ячейки локальной числовой переменной позволяет снять неоднозначность в определении типа третьего аргумента. Напомним, что у объектов имеются так называемые свойства по умолчанию, которые используются в ссылках на объекты, если свойство объекта явно не указано, но по контексту команды оно там должно быть. К такого рода ситуациям относится и процедура присваивания локальной числовой переменной в качестве значения ссылки на ячейку. При этом свойством по умолчанию является Value, т. е. значение ячейки.

Вычисление значения интерполяционного полинома базируется не только на значениях табулированной функции в узловых точках, но также в некотором смысле на количестве этих узловых точек. Другими словами, тот алгоритм, который мы используем для вычисления полинома, подразумевает явное использование такого параметра, как количество узловых точек. Явно этот параметр мы аргументом функции ЛАГРАНЖ() не передаем. Но он может быть вычислен — например, как количество ячеек в диапазоне, переданном первым аргументом функции ЛАГРАНЖ(). Поэтому мы объявляем целочисленную переменную n, а в качестве значения присваиваем ей выражение Px.Count. Здесь мы воспользовались тем обстоятельством, что свойство Count возвращает количество ячеек в диапазоне — в данном случае в диапазоне Px (первый аргумент функции ЛАГРАНЖ()).

На заметку

В принципе, в диапазоне, который передается вторым аргументом функции ЛАГРАНЖ() и содержит значения табулируемой в узловых точках функции, должно быть такое же количество ячеек, как и в диапазоне со значениями узловых точек. Если это не так, то, скорее всего, произойдет ошибка (хотя и не обязательно). Такого рода ситуации мы в программном коде не отслеживаем и предполагаем, что в обоих диапазонах одинаковое количество ячеек.
Кроме того, следует учесть, что свойство Count возвращает общее количество ячеек в диапазоне: если диапазон состоит из нескольких строк и нескольких столбцов, то общее количество ячеек равняется произведению количества строк на количество столбцов в диапазоне. Это, в свою очередь, позволяет передавать аргументами функции ячейки, сгруппированные как в строке, так и в столбике.

Основные вычисления производятся в блоке из вложенных условных операторов. Специально для использования в этих операторах объявляются две целочисленные переменные i и j. Переменные L и phi (обе типа Double) нужны

соответственно для вычисления полиномиальной суммы и для вычисления выражений вида $\varphi_m(x) = \prod_{\substack{k=0,\\ k\neq m}}^{n} \frac{(x - x_k)}{(x_m - x_k)}$. Перед началом выполнения вложенных операторов цикла переменной L присваивается нулевое значение.

Индексная переменная i во внешнем цикле пробегает значения от 1 до n. В начале каждой итерации переменной phi присваивается единичное значение. После этого последовательно запускаются два внутренних оператора цикла. В теле каждого из этих циклов выполняются практически идентичные (с формальной точки зрения) действия. Основное различие между циклами — диапазон изменения индексной переменной j. Для первого цикла она изменяется от 1 до i-1, а для второго цикла — от i+1 до n. Таким образом, при фиксированном значении i переменная j пробегает все значения от 1 до n, за исключением значения i. За каждую такую итерацию переменная phi сначала умножается на величину (x-Px.Cells(j).Value), а затем делится на величину (Px.Cells(i).Value-Px.Cells(j).Value). Здесь следует учесть, что Px.Cells(*индекс*).Value — это значение ячейки с указанным *индексом* в диапазоне Px (т. е. это значение узловой точки).

На заметку

Две команды в теле внутреннего оператора цикла можно объединить в одну, но такая команда выглядела бы достаточно громоздко.

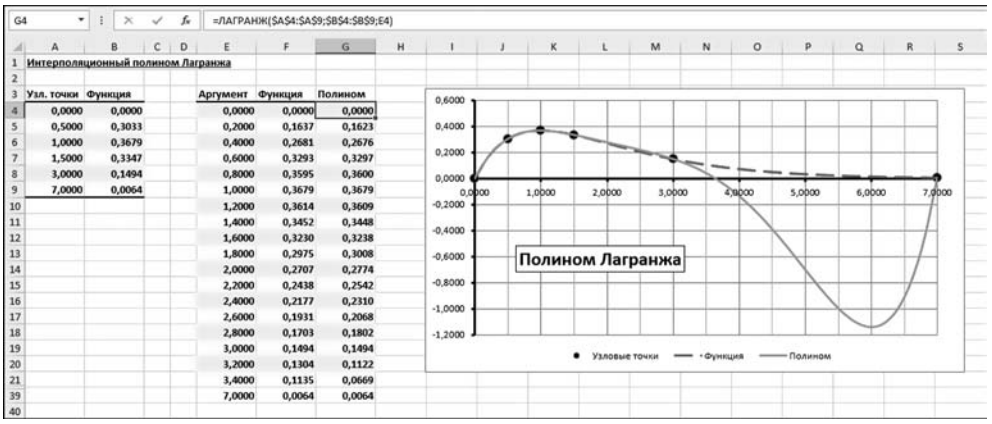
После того как значение переменной phi (для данного значения i) вычислено, командой L = L+phi*Py.Cells(i).Value к полиномиальной сумме добавляем очередное слагаемое. Здесь Py.Cells(i).Value есть ссылка на значение ячейки в диапазоне Py со значениями табулированной функции. В итоге значение переменной L возвращается как результат функции (команда ЛАГРАНЖ = L).

На заметку

Выше для перебора ячеек в диапазонах, которые передаются аргументами функции ЛАГРАНЖ(), использовалось свойство Cells с передачей одного индекса. В этом случае ячейки в диапазоне «перебираются» слева направо, сверху вниз. Поэтому теоретически совсем не обязательно, чтобы оба диапазона (со значениями узловых точек и значениями табулированной функции) имели одинаковую структуру — достаточно, чтобы количество ячеек в первом диапазоне равнялось количеству ячеек во втором диапазоне.

Созданную нами функцию ЛАГРАНЖ() можно использовать в рабочем документе Excel для вычисления значения интерполяционного полинома. В качестве примера рассмотрим документ, представленный на рисунке 13.3.

Ячейки A4:B9 содержат данные об узловых точках и значениях табулируемой в этих точках функции. А именно, в ячейках A4:A9 указаны несколько неравномерно распределенных на интервале от 0 до 7 точек. В соседних ячейках диапазона B4:B9 вычисляются соответствующие узловым точкам значения функции $f(x) = x \cdot \exp(-x)$.



Узл. точки	Функция
0,0000	0,0000
0,5000	0,3033
1,0000	0,3679
1,5000	0,3347
3,0000	0,1494
7,0000	0,0064

Аргумент	Функция	Полином
0,0000	0,0000	0,0000
0,2000	0,1637	0,1623
0,4000	0,2681	0,2676
0,6000	0,3293	0,3297
0,8000	0,3595	0,3600
1,0000	0,3679	0,3679
1,2000	0,3614	0,3609
1,4000	0,3452	0,3448
1,6000	0,3230	0,3238
1,8000	0,2975	0,3008
2,0000	0,2707	0,2774
2,2000	0,2438	0,2542
2,4000	0,2177	0,2310
2,6000	0,1931	0,2068
2,8000	0,1703	0,1802
3,0000	0,1494	0,1494
3,2000	0,1304	0,1122
3,4000	0,1135	0,0669
7,0000	0,0064	0,0064

Рис. 13.3
Использование функции пользователя для вычисления интерполяционного полинома Лагранжа (строки с 22 по 38 скрыты)

На заметку

В ячейку B4 вводится формула =EXP(-A4)*A4 и копируется во все прочие ячейки диапазона B4:B9.

Ячейки E4:E39 содержат значение точек, равномерно распределенных на интервале от 0 до 7, для которых вычисляется значение исходной (табулируемой) функции (ячейки F4:F39) и полинома Лагранжа (ячейки G4:G39).

На заметку

В ячейку E4 вводится формула =A4, в ячейку E5 вводится формула =E4+0,2 и копируется вплоть до ячейки E39. Диапазон ячеек F4:F39 заполняется копированием из ячейки F4 формулы =EXP(-E4)*E4. Для вычисления значений интерполяционного полинома в ячейку G4 вводим формулу =ЛАГРАНЖ ($A$4:$A$9;$B$4:$B$9;E4) и с помощью маркера автоматического заполнения копируем ее в прочие ячейки диапазона G4:G39.

На основе этих данных (включая ячейки A4:B9) строится диаграмма с исходными базовыми точками, кривой для функциональной зависимости $f(x) = x \cdot \exp(-x)$ и интерполяционного полинома.

На заметку

Обратите внимание, что часть строк в документе на рисунке 13.3 скрыта. При этом, чтобы на диаграмме данные в скрытых ячейках продолжали отображаться, необходимо в окне **Настройка скрытых и пустых ячеек** установить флажок опции **Показывать данные в скрытых строках и столбцах** (в окне **Выбор источника данных** для диаграммы нужно щелкнуть на кнопке **Скрытые и пустые ячейки**).

Несложно заметить, что интерполяционный полином далеко не везде дает хорошее приближение для табулированной функции, даже с учетом того, что последняя достаточно плавная.

ИНТЕРПОЛЯЦИОННЫЙ ПОЛИНОМ НЬЮТОНА

А жизнь нельзя подогнать под выверенную схему.

Из к/ф «Ирония судьбы, или С легким паром!»

При построении интерполяционного полинома по методу Ньютона применяемое для интерполирования данных полиномиальное выражение ищется в виде $P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \ldots + a_n(x - x_0)\ldots(x - x_n)$. Другими словами, если необходимо построить интерполяционный полином по методу Ньютона, то «отправной точкой» является полиномиальное выражение вида $P_n(x) = \sum_{k=0}^{n} a_k \prod_{m=0}^{k-1}(x - x_m)$. Коэффициенты a_0, a_1, …, a_n определяются из соотношений $P_n(x_p) = y_p$ (индекс $p = 0, 1, \ldots, n$). Последнее соотношение, с учетом приведенного выше вида для интерполяционного полинома, позволяет записать $P_n(x_p) = \sum_{k=0}^{p-1} a_k \prod_{m=0}^{k-1}(x_p - x_m) = y_p$. В частности, $a_0 = y_0$, $a_1 = y_1 - \frac{a_0}{x_1 - x_0}$. Для вычисления прочих коэффициентов может использоваться рекуррентное соотношение $a_p = \frac{y_p - \sum_{k=0}^{p-1} a_k \prod_{m=0}^{k-1}(x_p - x_m)}{\prod_{m=0}^{p-1}(x_p - x_m)}$. Вычисление интерполяционного полинома фактически сводится к вычислению коэффициентов a_0, a_1, …, a_n. Для решения этой задачи в рабочем документе Excel без привлечения программных средств VBA проведем некоторые предварительные преобразования.

На заметку

В данном случае мы несколько отступаем от «классического» способа вычисления коэффициентов, входящих в выражение для интерполяционного полинома Ньютона. Здесь мы будем в первую очередь исходить из возможностей приложения Excel.

В первую очередь, остановимся на тех соотношениях, которые служат отправными точками для вычисления коэффициентов a_0, a_1, …, a_n. Очевидно, что эти соотношения — совпадение в узловых точках значения интерполяционного полинома и табулированной функции. Выше мы уже приводили соответствующее соотношение (общего вида), здесь же мы его представим несколько иначе, а именно, в матричном виде. В частности, условия вида $P_n(x_p) = y_p$ (индекс $p = 0, 1, \ldots, n$) можно записать в матричном виде как $\hat{\mathbf{X}}\vec{a} = \vec{y}$, где введены такие обозначения: $\vec{a} = [a_1, a_2, \ldots, a_n]^T$, $\vec{y} = [y_1 - y_0, y_2 - y_0, \ldots, y_n - y_0]^T$ (символ T обозначает «транспонирование»), а матрица $\hat{\mathbf{X}}$ размера $n \times n$ имеет «треугольную» структуру с нулевыми элементами выше главной диагонали. Ненулевые элементы вычисляются в виде произведений: на пересечении i-й строки и j-го столбца находится элемент $(x_i - x_0)(x_i - x_1)\ldots(x_i - x_j)$ (индексы $j + 1 \le i = 1, 2, \ldots, n$). Таким образом, вектор

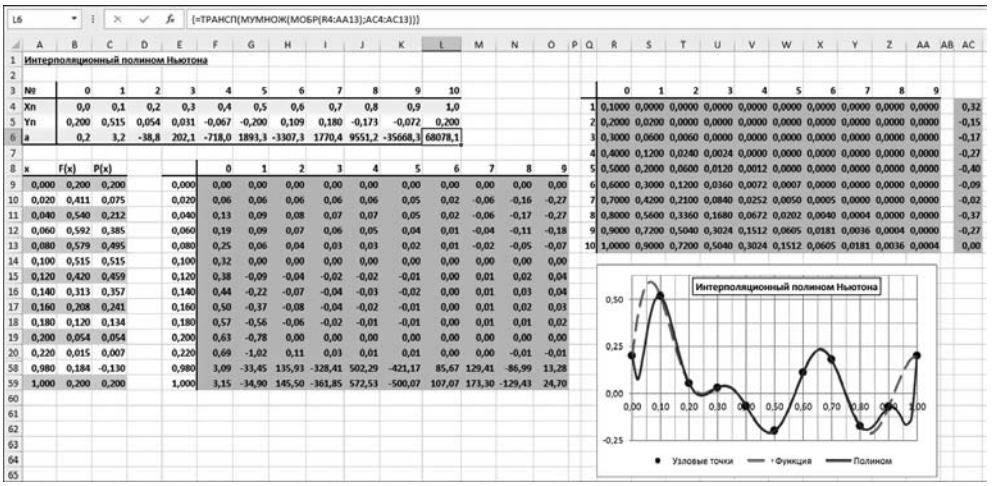


Рис. 13.4
Рабочий документ для вычисления интерполяционного полинома Ньютона (строки с 21-й по 57-ю скрыты)

коэффициентов может быть вычислен как $\vec{a} = \hat{\mathbf{X}}^{-1}\vec{y}$, где через $\hat{\mathbf{X}}^{-1}$ обозначена матрица, обратная к матрице $\hat{\mathbf{X}}$.

На заметку

Вектор $\vec{a}$ состоит из коэффициентов a_1, a_2, …, a_n. Напомним, что коэффициент $a_0 = y_0$.

Приведенная выше схема матричных вычислений может быть реализована в рабочем документе Excel. Рассмотрим документ, представленный на рисунке 13.4.

На заметку

В представленном документе строки с 21-й по 57-ю включительно скрыты. При создании диаграммы необходимо перейти в режим отображения скрытых ссылок.

Документ содержит достаточно большое количество данных, которые условно можно разбить на четыре блока (плюс диаграмма с графиками табулируемой функции, интерполяционного полинома и узловыми точками). Проанализируем отдельно каждый блок.

Диапазон ячеек A3:L6 содержит исходные данные, на основе которых строится интерполяционный полином, а также вычисляемые на основе этих данных коэффициенты полинома. В ячейки B3:L3 вводится индекс узловой точки (и коэффициента, соответственно). Здесь можно ввести в ячейку B3 начальное нулевое значение, в ячейку C3 ввести формулу =B3+1, после чего эта формула копируется в ячейки D3:L3. В ячейки B4:L4 вводятся значения для узловых точек, а в ячейки B5:L5 вводятся значения табулируемой функции в узловых точках.

На заметку

Мы используем равноотстоящие узлы: интервал от 0 до 1 разбивается на 10 равных промежутков. В ячейку B4 вводится нулевое значение, в ячейку C4 вводится формула =B4+1/10, и эта формула копируется в ячейки D4:L4.

Для табулирования мы используем функциональную зависимость $f(x) = \frac{\cos(6\pi x)}{5} + \sin(4\pi x)\exp(-5x)$. Поэтому в ячейку B5 вводим формулу =COS(6*ПИ()*B4)/5+SIN(4*ПИ()*B4)*EXP(-5*B4), и с помощью маркера автоматического заполнения копируем ее в прочие ячейки диапазона B5:L5.

В ячейках B6:L6 вычисляются коэффициенты $a_0, a_1, \ldots, a_{10}$, которые входят в выражение для интерполяционного полинома Ньютона. Принимая во внимание то обстоятельство, что $a_0 = y_0$, в ячейку B6 вводим формулу =B5. В ячейки диапазона C6:L6 вводится формула массива =ТРАНСП(МУМНОЖ(МОБР(R4:AA13);AC4:AC13)).

На заметку

Выделяем диапазон ячеек C6:L6, вводим в строку формул означенную выше формулу и нажимаем комбинацию клавиш <Ctrl>+<Shift>+<Enter>.

В соответствии с этой формулой производятся следующие вычисления:

- вычисляется матрица, обратная к матрице, записанной в ячейки R4:AA13;
- полученная матрица умножается на вектор-столбик, записанный в ячейки AC4:AC13;
- полученный вектор транспонируется — это и есть тот результат, что отображается в ячейках C6:L6.

С вектором в ячейках AC4:AC13 дело обстоит достаточно просто — в эти ячейки вводится формула массива =ТРАНСП(C5:L5)-B5, которой вычисляется вектор $\vec{y} = \left[y_1 - y_0,\ y_2 - y_0,\ \ldots,\ y_{10} - y_0\right]^T$. В данном случае от значений из диапазона ячеек C5:L5 (после транспонирования) отнимается значение ячейки B5.

Несколько сложнее вычисляется значение ячеек диапазона R4:AA13. В приведенных выше обозначениях это матрица $\hat{\mathbf{X}}$. Заполняется диапазон ячеек R4:AA13 построчно: сначала в ячейки R4:AA4 вводится формула массива =ПРОИЗВЕД(СМЕЩ($B$4;0;Q4)-$B$4:СМЕЩ($B$4;0;$B$3:$K$3)), после чего она копируется (с помощью маркера автоматического заполнения ячеек) в прочие ячейки диапазона R4:AA13. Нелишним будет кратко остановиться на особенностях использованной формулы.

Результатом выражения =ПРОИЗВЕД(СМЕЩ($B$4;0;Q4)-$B$4:СМЕЩ($B$4;0;$B$3:$K$3)), очевидно, является произведение элементов массива, который формируется инструкцией СМЕЩ($B$4;0;Q4)-$B$4:СМЕЩ($B$4;0;$B$3:$K$3). Этот массив, в свою очередь, состоит из элементов, каждый из которых — разность значения в ячейке СМЕЩ($B$4;0;Q4) и значений в ячейках $B$4:СМЕЩ($B$4;0;$B$3:$K$3). Здесь нужно учесть, что функция СМЕЩ() возвращает в качестве результата ссылку на ячейку (точнее, диапазон, который может состоять из одной ячейки): адрес этой ячейки можно получить, если отступить от ячейки, указанной первым аргументом

функции, на количество строк и столбцов, указанных соответственно вторым и третьим аргументами функции. Поэтому, например, инструкция СМЕЩ($B$4;0;Q4) есть ссылка на ячейку, находящуюся в той же строке, что и ячейка B4, но на несколько столбцов сдвинутая — на сколько именно, определяется значением в ячейке Q4.

На заметку

Значения в ячейках Q4:Q12, которые определяют «сдвиг», вычисляются формулой массива =ТРАНСП(C3:L3), т. е. это те же значения, что и в ячейках C3:L3.

Немного сложнее понять результат выражения $B$4:СМЕЩ($B$4;0;$B$3:$K$3). В данном случае речь идет о массиве из диапазонов ячеек. Начальная ячейка в каждом из этих диапазонов — это ячейка B4. Каждый диапазон состоит из одной строки, но конечная ячейка у каждого диапазона своя. Количество столбцов, на которые конечная ячейка отстоит от начальной, определяется элементами массива $B$3:$K$3 (числа от 0 до 9 включительно). Таким образом, в результате вычисления выражения $B$4:СМЕЩ($B$4;0;$B$3:$K$3) получаем набор диапазонов, начиная от B4:B4 до B4:K4 с дискретностью увеличения ширины диапазона в один столбец. Выражение ($B$4;0;Q4)-$B$4:СМЕЩ($B$4;0;$B$3:$K$3) — это набор диапазонов, получаемых вычитанием из значения ячейки, отстоящей от ячейки B4 на Q4 столбцов, значений ячеек диапазонов от B4:B4 до B4:K4. Применяя к этой конструкции функцию ПРОИЗВЕД(), получаем массив из произведений, каждое из которых вычисляется по соответствующему диапазону.

На заметку

Вообще в данном случае мы идем по достаточно тонкому льду. Дело в том, что функция ПРОИЗВЕД() перемножает все, что передано ей в качестве аргумента. Предложенный подход «сработал» благодаря функции СМЕЩ(), которая в исходном своем назначении ориентирована на вычисление в качестве результата ссылки на диапазон ячеек.

При копировании формулы массива =ПРОИЗВЕД(СМЕЩ($B$4;0;Q4)-$B$4:СМЕЩ($B$4;0;$B$3:$K$3)) меняется ссылка на ячейку Q4, определяющую сдвиг относительно ячейки B4. В результате каждая строка диапазона R4:AA13 содержит произведения вида $(x_i - x_0)(x_i - x_1)\ldots(x_i - x_j)$, где значение индекса i определяется значениями в ячейках Q4:Q13, а индекса j — значениями ячеек R3:AA3 (диапазон заполняется формулой массива =B3:K3). При этом мы неявно воспользовались тем обстоятельством, что при $j \geq i$ произведение $(x_i - x_0)(x_i - x_1)\ldots(x_i - x_j) = 0$, поскольку там обязательно встречается нулевой множитель. В результате верхняя (от основной диагонали) часть диапазона R4:AA13 содержит нулевые значения (рис. 13.4).

Ячейки диапазона A8:C59 содержат значения аргумента (ячейки A9:A59), для которых вычисляется табулированная функция и интерполяционный полином, значения функции (ячейки B9:B59) и значения интерполяционного полинома (ячейки C9:C59). Все ячейки заполняются пошаговым копированием. Так, в ячейку A9 вводится формула =B4, а в ячейку A10 под ней —

вводим формулу =A9+($L$4-$B$4)/50, после чего ее можно копировать в нижние ячейки. В ячейку B9 вводится формула =COS(6*ПИ()*A9)/5+SIN(4*ПИ()*A9)*EXP(-5*A9) и копируется во все прочие ячейки внизу для заполнения диапазона B9:B59. Наконец, значения в ячейках C9:C59 получаются «размножением» формулы =$B$6+СУММ(F9:O9) из ячейки C9. В этой формуле к значению ячейки B6 (коэффициент a_0) прибавляется сумма значений из диапазона ячеек F9:O9. Учитывая, что формула копируется, несложно догадаться, что значения в ячейках C9:C59, дающие представление о поведении интерполяционного полинома, функционально зависят от значений в ячейках диапазона F9:O59. Этот диапазон — последнее звено в цепи вычислений.

Если кратко, то диапазон ячеек F9:O59 содержит (построчно) значения произведений вида $(x - x_0)(x - x_1)\ldots(x - x_j)$ для разных значений аргумента x и разных индексов j. Значения аргумента x дублируются в ячейках E9:E59 (заполняются формулой массива =A9:A59), а значения индекса j определяются в ячейках F8:O8 (заполняется с помощью формулы массива =B3:K3). Диапазон ячеек F9:O59 заполняется так: в ячейку F9 вводим формулу массива =C$6*ПРОИЗВЕД($E9-$B$4:СМЕЩ($B$4;0;F$8)), которую затем копируем вправо до ячейки O9, а затем построчно копируем формулы из ячеек диапазона F9:O9 во все прочие ячейки диапазона F9:O59.

Проанализируем формулу массива =C$6*ПРОИЗВЕД($E9-$B$4:СМЕЩ($B$4;0;F$8)). Это произведение значения ячейки C6 на результат выражения ПРОИЗВЕД($E9-$B$4:СМЕЩ($B$4;0;F$8)). Выражение ПРОИЗВЕД($E9-$B$4:СМЕЩ($B$4;0;F$8)), в свою очередь, это произведение значений массива, который получается вычитанием из ячейки E9 значений диапазона ячеек $B$4:СМЕЩ($B$4;0;F$8) (начальная ячейка B4, а конечная в той же строке удалена на количество столбцов, определяемых значением в ячейке F8). Ячейка C$6 содержит значение коэффициента интерполяционного полинома (в данном случае это коэффициент a_1), который умножается на величину $(x - x_0)$. После копирования формулы в соседнюю справа ячейку G9 она станет такой (формулой массива): =D$6*ПРОИЗВЕД($E9-$B$4:СМЕЩ($B$4;0;G$8)). Это произведение коэффициента a_2 на $(x - x_0)(x - x_1)$. То есть, копируя формулы слева направо, получаем новые слагаемые для интерполяционного полинома (для одного и того же значения аргумента x).

На заметку

Обратите внимание на использование смешанных ссылок — при копировании формул это достаточно важный момент.

Таким образом, по строкам диапазона F9:O59 содержатся слагаемые для вычисления полинома при фиксированном значении аргумента, а по столбцам — слагаемые для разных аргументов, соответствующие одному и тому же коэффициенту в выражении для интерполяционного полинома.

Если приведенная выше методика вычисления интерполяционного полинома Ньютона кажется слишком сложной, можно воспользоваться более простым подходом — написать специальный программный код. Другими словами, мы всегда можем описать функцию пользователя, которая в качестве результата возвращает значение интерполяционного полинома Ньютона.

На заметку

Еще раз напомним, что интерполяционный полином на самом деле всегда один, речь идет лишь о способе его вычисления.

В листинге 13.2 представлен программный код функции, которая называется НЬЮТОН и решает поставленную задачу.

Листинг 13.2. Функция для вычисления полинома Ньютона

```
Function НЬЮТОН(Px As Range, Py As Range, z As Variant) As Double
 ‘ Переменная для запоминания аргумента полинома
 Dim t As Double
 ‘ Значение аргумента полинома
  t = z
 ‘ Переменная для запоминания количества интервалов
 Dim n As Integer
 ‘ Вычисляем количество интервалов (количество точек минус 1)
 n = Px.Count-1
 ‘ Массив коэффициентов для полинома
 ReDim a(0 To n) As Double
 ‘ Массив узловых точек
 ReDim x(0 To n) As Double
 ‘ Массив значений функции
 ReDim y(0 To n) As Double
 ‘ Целочисленные переменные для операторов цикла
 Dim i As Integer, j As Integer
 ‘ Заполнение массивов
 For i = 0 To n
  ‘ Узловая точка
  x(i) = Px.Cells(i + 1).Value
  ‘ Значение функции в узловой точке
 y(i) = Py.Cells(i + 1).Value
 Next i
 ‘ Первый (с нулевым индексом) коэффициент
 a(0) = y(0)
 ‘ Технические переменные
 Dim b As Double, s As Double
 ‘ Вычисление коэффициентов для полинома - внешний цикл
 For i = 1 To n
  ‘ Начальные значения для технических переменных
   b = y(i) - a(0)
   s = 1
  ‘ Внутренний цикл
  For j = 0 To i - 2
   ‘ Вычисляем произведение
    s = s * (x(i) - x(j))
   ‘ Вычисляем разность
```

```
        b = b - a(j + 1) * s
      Next j
      ' Значение коэффициента для полинома
      a(i) = b / s / (x(i) -x(i - 1))
    Next i
    ' Переменная для вычисления значения полинома
    Dim P As Double
    ' Начальное значение для полиномиальной суммы
    P = a(0)
    ' Начальное значение для произведения
    s = 1
    ' Вычисляем значение полинома
    For i = 1 To n
      ' Вычисляем произведение
      s = s * (t - x(i - 1))
      ' Вычисление полиномиальной суммы
      P = P + a(i) * s
    Next i
    ' Результат вычислений
    НЬЮТОН = P
End Function
```

В некоторых местах код этой функции напоминает программный код функции ЛАГРАНЖ(), которая рассматривалась ранее. Поэтому остановимся лишь на наиболее характерных моментах, связанных с вычислением результата функции НЬЮТОН().

Аргументы у функции НЬЮТОН() фактически такие же, как и у функции ЛАГРАНЖ(): это диапазон ячеек Px со значениями узловых точек, диапазон ячеек Py со значениями табулируемой функции в узловых точках, а также аргумент z, для которого вычисляется значение полинома (т. е. непосредственно аргумент для интерполяционного полинома. Последний может быть как ссылкой на ячейку, так и числовым значением. Поэтому в теле функции выполняется переопределение аргумента полинома: соответствующее значение записывается в переменную t типа Double.

Целочисленная переменная n вычисляется командой n = Px.Count-1, в результате чего значение этой переменной на единицу меньше количества узловых точек. Переменную в данном случае можем интерпретировать как количество интервалов, на которые разбивается диапазон интерполирования. Это же число служит показателем степени интерполяционного полинома. Далее нами будут созданы массивы для вычисления коэффициентов интерполяционного полинома (массив a) и массивы для записи в них узловых точек (массив x) и значений табулируемой функции в узловых точках (массив y). Благодаря такому определению переменной n мы сможем индексировать элементы этих массивов в диапазоне от 0 до n, в полном соответствии с теми обозначениями, которые мы использовали выше при изложении общего теоретического подхода к построению интерполяционного полинома Ньютона.

На заметку

Мы естественным образом предполагаем, что количество ячеек в диапазонах Px и Py одинаково.

Также мы используем ряд «технических» переменных: целочисленные переменные i и j в качестве индексных переменных в операторе цикла, а также две Double-переменные b и s для записи промежуточных результатов вычислений.

Вычисления проводятся в несколько этапов. Сначала на основе аргументов функции НЬЮТОН() заполняются массивы x и y соответственно значениями узловых точек и значениями табулируемой функции в узловых точках. Для этого запускается оператор цикла, в котором индексная переменная i пробегает значение от 0 до n. За каждую итерацию командой x(i) = Px.Cells(i+1).Value значения очередной ячейки из диапазона Px записывается в элемент массива x(i), а командой y(i) = Py.Cells(i+1).Value в элемент y(i) записывается значение из ячейки в диапазоне Py.

На заметку

Обратите внимание, что, если индексация массивов x и y начинается с нуля, то индексация ячеек в диапазонах Px и Py начинается с единицы. Поэтому между индексами соответствующих элементов массивов и ячеек диапазонов существует единичный «сдвиг» и, например, элементу с нулевым индексом соответствует ячейка с индексом 1 в коллекции ячеек диапазона.

После того как заполнены массивы с узловыми точками и значениями табулированной функции в узловых точках, командой a(0) = y(0) вычисляется первый коэффициент (коэффициент с нулевым индексом) для полинома Ньютона. Для вычисления прочих коэффициентов запускаются вложенные операторы цикла.

Индексная переменная i для внешнего оператора цикла на этот раз пробегает значения от 1 до n. За каждую итерацию переменная b получает начальное значение y(i)-a(0), которое по мере выполнения оператора будет уточняться. Переменная s получает вначале значение 1. После этого запускается внутренний оператор цикла с индексной переменной j, пробегающей значения от 0 до i-2.

На заметку

При начальном единичном значении переменной i верхняя граница для диапазона изменения индексной переменной j равна -1 (а нижняя — нулевая). В этом случае оператор цикла по переменной j не выполняется.

Во внутреннем цикле командой s = s*(x(i)-x(j)) мы «уточняем» произведение умножением на очередной множитель (речь идет о произведении $(x_i - x_0)(x_i - x_1)\ldots(x_i - x_j)$, которое, умноженное на коэффициент a_{j+1}, входит в выражение $a_i = \dfrac{y_i - a_0 - \sum_{j=0}^{i-2} a_{j+1}(x_i - x_0)(x_i - x_1)\ldots(x_i - x_j)}{(x_i - x_0)(x_i - x_1)\ldots(x_i - x_{i-1})}$ для определения коэффициента a_i). Соответствующее слагаемое учитывается в числителе

приведенного выше выражения, который записывается в переменную b: речь идет о команде b = b-a(j+1)*s.

После завершения внутреннего оператора цикла командой a(i) = b/s/(x(i)-x(i-1)) вычисляется значение коэффициента для полинома. Здесь мы воспользовались тем обстоятельством, что после выполнения внутреннего оператора цикла значение переменной s есть произведение $(x_i - x_0)(x_1 - x_1)\ldots(x_i - x_{i-2})$. В знаменателе должен быть множитель $(x_i - x_0)(x_1 - x_1)\ldots(x_i - x_{i-1})$. Поэтому числитель (переменная b) можно разделить на s, а потом еще на $(x_i - x_{i-1})$ (т. е. на выражение (x(i)-x(i-1))).

На этом выполнение внешнего оператора цикла заканчивается, а с ним и вычисление коэффициентов для интерполяционного полинома. На следующем этапе предстоит вычислить значение непосредственно полинома в точке, значение которой определяется переменной t.

Переменная, используемая для вычисления значения полинома, называется P, и в качестве начального значения она получает величину a(0) (коэффициент полинома с нулевым индексом).

На заметку

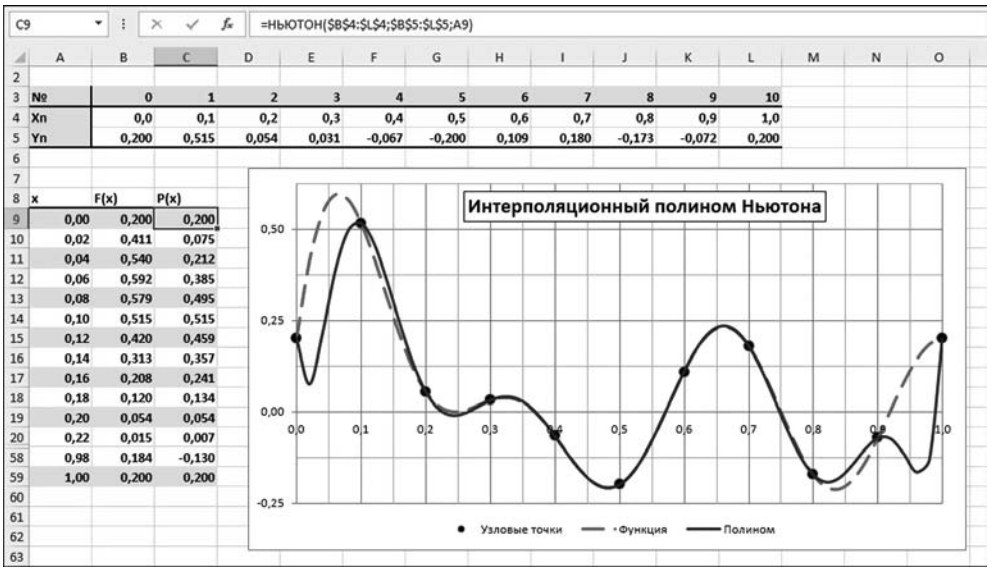
Напомним, что интерполяционный полином Ньютона вычисляется по формуле $P_n(t) = a_0 + a_1(t - x_0) + a_2(t - x_0)(t - x_1) + \ldots + a_n(t - x_0)(t - x_1)\ldots(t - x_{n-1})$. Соответствующая сумма вычисляется последовательным добавлением слагаемых в переменную P. Слагаемые, в свою очередь, получаются умножением коэффициентов a_i (которые уже вычислены) на произведение $(t - x_0)(t - x_1)\ldots(t - x_{i-1})$, для вычисления которого нам понадобится еще один оператор цикла. Значения произведений вида $(t - x_0)(t - x_1)\ldots(t - x_{i-1})$ будем записывать в переменную s, которую мы до этого уже использовали для вычисления несколько иного произведения. Начальное значение для произведения определяется командой s = 1.

При вычислении значения полинома индексная переменная i пробегает значения от 1 до n. В теле оператора командой s = s*(t-x(i-1)) вычисляем произведение для очередного слагаемого, после чего командой P = P+a(i)*s в переменную P добавляем новое слагаемое. В результате после завершения работы оператора цикла переменная P содержит значение интерполяционного полинома в точке, определяемой переменной t. Командой НЬЮТОН = P присваивается в качестве значения функции НЬЮТОН().

Для иллюстрации работы созданной нами функции воспользуемся теми же исходными данными, что и в предыдущем случае — табулируем на интервале от 0 до 1 по 11 равноотстоящим узловым точкам функцию $f(x) = \frac{\cos(6\pi x)}{5} + \sin(4\pi x)\exp(-5x)$. На рисунке 13.5 показан соответствующий документ.

Мы получили такой же результат, как и в предыдущем случае, когда полином вычислялся непосредственно в рабочем документе Excel без применения программных кодов VBA (рис. 13.4).

Сплошной линией показана кривая, построенная на основе полиномиальной зависимости, штрихованная линия дает представление о поведении табулированной функции, а узловые точки обозначены кружками. Как и



№	0	1	2	3	4	5	6	7	8	9	10
Xn	0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0
Yn	0,200	0,515	0,054	0,031	-0,067	-0,200	0,109	0,180	-0,173	-0,072	0,200

x	F(x)	P(x)
0,00	0,200	0,200
0,02	0,411	0,075
0,04	0,540	0,212
0,06	0,592	0,385
0,08	0,579	0,495
0,10	0,515	0,515
0,12	0,420	0,459
0,14	0,313	0,357
0,16	0,208	0,241
0,18	0,120	0,134
0,20	0,054	0,054
0,22	0,015	0,007
0,98	0,184	-0,130
1,00	0,200	0,200

Рис. 13.5
Вычисление интерполяционного полинома Ньютона с помощью функции пользователя

следовало ожидать, интерполяционный полином проходит через все узловые точки и в данном случае очень неплохо приближает исходную функцию, возможно, за исключением граничных интервалов.

Что касается данных в рабочем листе, то диапазон ячеек B4:L5 содержит данные об узловых точках и значениях табулированной функции в узловых точках. В ячейках A9:C59 вычисляются значения для табулированной функции и интерполяционного полинома на интервале значений аргумента от 0 до 1. Все ячейки, за исключением диапазона C9:C59, заполняются так же, как в предыдущем примере (когда вычисления проводились исключительно в рабочем листе). Ячейки C9:C59 заполняем следующим образом: в ячейку C9 вводим формулу =НЬЮТОН($B$4:$L$4;$B$5:$L$5;A9) и затем копируем ее в нижние ячейки диапазона C9:C59. Первым и вторым аргументами функции НЬЮТОН() передаются соответственно ссылки на диапазон с узловыми точками и значениями функции в этих точках. Это те данные, на основе которых строится полином. Ссылки использованы абсолютные, поскольку при копировании формулы они не должны меняться. Третий аргумент функции НЬЮТОН() — это ссылка на ячейку, содержащую значение аргумента (точки), для которого вычисляется полином. Эта ссылка относительная, поскольку при копировании формулы должна меняться, указывая на новое значение аргумента. На основе ячеек диапазона A9:C59 строится диаграмма:

- ячейки A9:A59 содержат значения для аргументов функции и полинома;
- ячейки B9:B59 содержат значения функции;
- ячейки C9:C59 содержат значения интерполяционного полинома.

Узловые точки строятся на основе данных в ячейках B4:L4 (значения узловых точек) и B5:L5 (табличные значения функции).

На заметку

В диаграмму добавляется три ряда данных: функция, полином и узловые точки. Первые две зависимости отображаются линиями (тип ряда диаграммы — **Точечная с гладкими кривыми**), а третья — точками (тип ряда диаграммы выбирается **Точечная с маркерами**).

Кроме того, часть ячеек диапазона A9:C59 скрыта. Чтобы значения из скрытых ячеек отображались на диаграмме, для нее нужно применить режим отображения скрытых ячеек.

ПОЛИНОМИАЛЬНАЯ ИНТЕРПОЛЯЦИЯ

Вот что крест животворящий делает!

Из к/ф «Иван Васильевич меняет профессию»

Большинство «классических» алгоритмов для вычисления интерполяционных выражений рассчитаны на то, что вычислительные средства весьма скудны. Вместе с тем, если в нашем распоряжении имеется такое мощное вычислительное приложение, как Excel, вполне допустимо предпринять «атаку в лоб». Другими словами, если нас интересует конечный результат, а не способ его получения, мы можем не прибегать к замысловатым алгоритмам и провести серию несложных (с учетом возможностей Excel) вычислений. В частности, предположим, что имеется набор узловых точек x_k и значений y_k некоторой функции в этих точках (индекс $k = 0, 1, \ldots, n$). Наша задача состоит в том, чтобы найти такое полиномиальное выражение $Q_n(x) = \sum_{k=0}^{n} \alpha_k x^k$ (т. е. вычислить коэффициенты α_k, $k = 0, 1, \ldots, n$), чтобы выполнялись соотношения $Q_n(x_k) = y_k$ для всех индексов $k = 0, 1, \ldots, n$. Если ввести следующие соотношения — вектор $\vec{\alpha} = [\alpha_0, \alpha_1, \ldots, \alpha_n]^T$, вектор $\vec{y} = [y_0, y_1, \ldots, y_n]^T$ (символ T

L6: {=ТРАНСП(МУМНОЖ(МОБР(B9:L19);ТРАНСП(B5:L5)))}

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Интерполяционный полином															
2																
3	№	0	1	2	3	4	5	6	7	8	9	10		x	F(x)	Q(x)
4	Xn	0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0		0,0	0,200	0,200
5	Yn	0,200	0,515	0,054	0,031	-0,067	-0,200	0,109	0,180	-0,173	-0,072	0,200		0,020	0,411	0,075
6	alpha	0,2	-18,7	819,7	-11123,9	73007,2	-272190,7	616271,2	-863059,4	730236,2	-342019,6	68078,1		0,040	0,540	0,212
7														0,060	0,592	0,385
8		0	1	2	3	4	5	6	7	8	9	10		0,080	0,579	0,495
9	0	1	0	0	0	0	0	0	0	0	0	0		0,100	0,515	0,515
10	1	1	0,1	0,01	0,001	0,0001	0,00001	0,000001	1E-07	1E-08	1E-09	1E-10		0,120	0,420	0,459
11	2	1	0,2	0,04	0,008	0,0016	0,00032	0,000064	1,28E-05	2,56E-06	5,12E-07	1,02E-07		0,140	0,313	0,357
12	3	1	0,3	0,09	0,027	0,0081	0,00243	0,000729	0,000219	6,56E-05	1,97E-05	5,9E-06		0,160	0,208	0,241
13	4	1	0,4	0,16	0,064	0,0256	0,01024	0,004096	0,001638	0,000655	0,000262	0,000105		0,180	0,120	0,134
14	5	1	0,5	0,25	0,125	0,0625	0,03125	0,015625	0,007813	0,003906	0,001953	0,000977		0,200	0,054	0,054
15	6	1	0,6	0,36	0,216	0,1296	0,07776	0,046656	0,027994	0,016796	0,010078	0,006047		0,220	0,015	0,007
16	7	1	0,7	0,49	0,343	0,2401	0,16807	0,117649	0,082354	0,057648	0,040354	0,028248		0,240	0,000	-0,010
17	8	1	0,8	0,64	0,512	0,4096	0,32768	0,262144	0,209715	0,167772	0,134218	0,107374		0,260	0,003	-0,005
18	9	1	0,9	0,81	0,729	0,6561	0,59049	0,531441	0,478297	0,430467	0,38742	0,348678		0,280	0,016	0,012
19	10	1	1	1	1	1	1	1	1	1	1	1		0,300	0,031	0,031
20														0,320	0,038	0,041
53														0,980	0,184	-0,130
54														1,000	0,200	0,200
55																

Рис. 13.6
Вычисление коэффициентов интерполяционного полинома через матрицу Вандермонда

означает «транспонирование») и матрица $\hat{\mathbf{C}}$ с элементами $c_{ij} = x_i^j$ (индексы $i, j = 0, 1, \ldots, n$), то соотношения для определения коэффициентов интерполяционного полинома могут быть записаны в виде линейной системы уравнений $\hat{\mathbf{C}}\vec{\alpha} = \vec{y}$. Отсюда легко записать выражение для вектора коэффициентов полинома $\vec{\alpha} = \hat{\mathbf{C}}^{-1}\vec{y}$. Через $\mathbf{C}^{-1}$ обозначена матрица, обратная к матрице $\hat{\mathbf{C}}$ (которая, кстати, называется *матрицей Вандермонда*). Эта схема может быть с легкостью реализована в рабочем документе Excel. С целью достижения наибольшего методологического эффекта, в качестве исходных данных для построения интерполяционного полинома мы возьмем те же значения, что использовались при вычислении интерполяционного полинома Ньютона. Далее обратимся к документу, представленному на рисунке 13.6.

Ячейки B4:L4 содержат значения узловых точек. Ячейки B5:L5 содержат значения табулированной функции в узловых точках. В ячейках B6:L6 по формуле массива =ТРАНСП(МУМНОЖ(МОБР(B9:L19);ТРАНСП(B5:L5))) вычисляются коэффициенты интерполяционного полинома.

На заметку

В соответствии с формулой массива =ТРАНСП(МУМНОЖ(МОБР(B9:L19); ТРАНСП(B5:L5))) выполняются такие действия. На основе ячеек диапазона B9:L19 (матрица Вандермонда) вычисляется обратная матрица. Эта матрица умножается (по правилам вычисления матричного произведения) на вектор, получающийся транспонированием диапазона ячеек B5:L5 (значения табулированной функции в узловых точках). В результате получается вектор-столбец, который транспонируется и поэлементно записывается в ячейки диапазона B6:L6 (тот диапазон, в который вводится формула массива).

В ячейках диапазона B9:L19 вычисляются элементы матрицы Вандермонда. Заполняется диапазон ячеек так: в ячейки B9:B19 вводится формула массива =1 (хотя можно просто в ячейки ввести значение 1, или, например, выделить ячейки B9:B19, ввести в строке формул значение 1 и нажать комбинацию клавиш <Ctrl>+<Enter>), а в диапазон ячеек C9:L19 вводится формула массива =ТРАНСП(B4:L4)^C8:L8. При этом ячейки B8:L8 заполняются с помощью формулы массива =B3:L3, а ячейки A9:A19 заполняются формулой массива =ТРАНСП(B3:L3). В результате в ячейках B8:L8 и A9:A19 содержатся индексы для элементов матрицы Вандермонда.

На заметку

Смысл формулы массива =ТРАНСП(B4:L4)^C8:L8 весьма прост. Мы берем ячейки B4:L4 (это узловые точки), транспонируем диапазон (в результате получается вектор-столбец из 11 элементов), а затем каждый элемент последовательно возводим в степень с показателем, определяемым ячейками C8:L8. Получаем столбцы матрицы Вандермонда, за исключением самого первого — того, где элементы возводятся в нулевую степень. Теоретически, по этому принципу можно было бы вычислить и значения в первом столбце, но Excel болезненно реагирует на ситуацию, когда ноль возводится в нулевую степень. По нашей логике это должна быть единица. Приложение Excel этой логики не придерживается. Поэтому нам пришлось первый столбец матрицы Вандермонда вычислять в явном виде (мы его заполнили единицами).

После того как коэффициенты интерполяционного полинома вычислены, задача, фактически, решена. Осталось использовать эти коэффициенты (ячейки B6:L6) для вычисления значений интерполяционного полинома в различных точках области интерполирования. Соответствующие вычисления проделаны в ячейках диапазона N4:P54 (на рисунке 13.6 некоторые ячейки этого диапазона скрыты). В ячейках N4:N54 содержатся значения для аргументов интерполяционного полинома и табулированной функции, в ячейках O4:O54 вычисляются значения функции в точках из диапазона N4:N54. Значения интерполяционного полинома вычисляются в ячейках P4:P54. Диапазон заполняется так: в ячейку P4 вводится формула =$B$6+РЯД.СУММ(N4;1;1;$C$6:$L$6), после чего она копируется во все другие ячейки диапазона P4:P54. В данном случае мы воспользовались встроенной функцией РЯД.СУММ() для вычисления значения полиномиального выражения. Первым аргументом функции передается ссылка на аргумент полиномиального выражения. Второй и третий аргументы — соответственно начальная степень аргумента и шаг дискретности изменения степени. Наконец, четвертый аргумент функции РЯД.СУММ() — это набор коэффициентов полинома. Другими словами, если нам нужно вычислить сумму $\sum_{k=0}^{N} a_k x^{n+km}$, то первым аргументом функции РЯД.СУММ() передается переменная x, вторым аргументом передается переменная n, третьим аргументом передается переменная m, а набор коэффициентов a_0, a_1, ..., a_N передается четвертым аргументом функции. В нашем случае роль аргумента x играет ячейка N4, второй и третий аргументы — единицы, а коэффициенты передаются через абсолютную ссылку на диапазон C6:L6 (при копировании формул эта ссылка не должна меняться). Несложно заметить, что диапазон C6:L6 содержит не все коэффициенты полинома — отсутствует коэффициент, соответствующий аргументу в нулевой степени. Это слагаемое мы явно указали в формуле =$B$6+РЯД.СУММ(N4;1;1;$C$6:$L$6). Причина та же, по которой мы отдельно заполняли первый столбец для матрицы Вандермонда — ошибка при вычислении значения «ноль в нулевой степени».

Несложно убедиться, что в узловых точках вычисленный нами полином дает правильные значения — они совпадают со значениями в узловых точках табулированной функции. Желающие могут построить диаграмму для графика функции и интерполяционного полинома. Они будут такими же, как и в случае интерполирования по методу Ньютона. Причина очевидна — в обоих случаях строится один и тот же полином, но только используются разные алгоритмы.

Если желания или необходимости вычислять в явном виде матрицу Вандермонда нет, все промежуточные вычисления можно «спрятать» в пользовательскую функцию. Пример такой функции приведен в листинге 13.3.

Листинг 13.3. Функция для вычисления интерполяционного полинома

```
Function ИНТЕРППОЛИНОМ(Px As Range, Py As Range, z As Variant)
 ' Переменная для определения количества интервалов
 Dim n As Integer
```

```
    ' Значение переменной для определения количества интервалов
    n = Px.Count - 1
    ' Целочисленные переменные для использования в операторах цикла
    Dim i As Integer, j As Integer
    ' Двумерный массив для записи матрицы Вандермонда
    ReDim C(0 To n, 0 To n) As Double
    ' Одномерный массив узловых точек
    ReDim x(0 To n) As Double
    ' Одномерный массив значений функции в узловых точках
    ReDim y(0 To n) As Double
    ' Оператор цикла для заполнения массивов
    For i = 0 To n
      ' Массив узловых точек
      x(i) = Px.Cells(i + 1).Value
      ' Массив значений функции в узловых точках
      y(i) = Py.Cells(i + 1).Value
      ' Первый элемент матрицы Вандермонда
      C(i, 0) = 1
      ' Оператор цикла для заполнения строки в матрице Вандермонда
      For j = 1 To n
        ' Вычисление элементов строки матрицы Вандермонда
        C(i, j) = C(i, j - 1) * x(i)
      Next j
    Next i
    ' Массив для записи степеней аргумента полинома
    ReDim t(0 To n) As Double
    ' Первый элемент массива (аргумент в нулевой степени)
    t(0) = 1
    ' Заполнение массива
    For i = 1 To n
      ' Вычисление следующего элемента на основе предыдущего
      t(i) = t(i - 1) * z
    Next i
    ' Переменная для записи результата
    Dim Q As Variant
    ' Использование функций рабочего листа
    With Application.WorksheetFunction
      ' Вычисление результата
      Q = .MMult(t, .MMult(.MInverse(C), .Transpose(y)))
    End With
    ' Значение функции
    ИНТЕРППОЛИНОМ = Q
  End Function
```

Мы уже рассматривали несколько программных кодов для вычисления интерполяционного полинома, поэтому здесь речь может идти еще об одном

алгоритме его вычисления. В частности, мы в теле функции ИНТЕРППОЛИНОМ() основные вычисления выполняем с помощью функций рабочего листа приложения.

Аргументы у функции простые и понятные: два диапазона со значениями узловых точек и табулированной функции, а также аргумент для интерполяционного полинома (т. е. «точка», в которой вычисляется интерполяционный полином).

Как и ранее, в целочисленную переменную n командой n = Px.Count-1 записывается количество интервалов (количество точек минус один), и, как и ранее, мы предполагаем, что количество ячеек в первых двух аргументах-диапазонах одинаково.

Для записи матрицы Вандермонда командой ReDim C(0 To n,0 To n) As Double объявляется двумерный числовой массив C с индексацией элементов от 0 до n по каждому индексу. Одномерный массив x предназначен для записи узловых точек, одномерный массив y предназначен для записи значений функции в узловых точках, а одномерный массив t предназначен для записи степеней аргумента полинома. У всех этих массивов индексация элементов выполняется от 0 до n.

На заметку

Если обозначить аргумент интерполяционного полинома через z, а его коэффициенты через a_0, a_1, ..., a_n, то значение полинома, очевидно, вычисляется как $Q_n(z) = a_0 + a_1 z + a_2 z^2 + \ldots + a_n z^n$. Если ввести в рассмотрение векторы $\vec{a} = [a_0, a_1, \ldots, a_n]^T$ и $\vec{z} = [1, z, z^2, \ldots, z^n]$, то значение полинома может быть записано как матричное произведение $Q_n(z) = \vec{z} \cdot \vec{a}$, результатом которого является скаляр. Именно такой подход мы и используем. Вектор коэффициентов $\vec{a}$ мы вычислим на основе матрицы Вандермонда $\hat{\mathbf{C}}$ и вектора значений функции в узловых точках $\vec{y} = [y_0, y_1, \ldots, y_n]^T$ с помощью соотношения $\vec{a} = \hat{\mathbf{C}}^{-1}\vec{y}$, а вектор $\vec{z}$ вычисляем в явном виде. Результат записывается в переменную t.

Для заполнения массивов запускается оператор цикла, в котором индексная переменная i пробегает значения от 0 до n. За каждую итерацию командами x(i) = Px.Cells(i+1).Value и y(i) = Py.Cells(i+1).Value заполняются соответствующие элементы массива узловых точек и значений функции, после чего запускается внутренний оператор цикла (с индексной переменной j, изменяющейся от 1 до n), в котором заполняется i-я строка матрицы Вандермонда. Командой C(i,j) = C(i,j-1)*x(i) каждый следующий элемент в строке вычисляется на основе предыдущего умножением на значение i-й узловой точки x(i). Самому первому (начальному) элементу i-й строки в матрице Вандермонда перед запуском внутреннего оператора цикла командой C(i,0) = 1 присваивается единичное значение.

После того как массивы x, y и C заполнены, заполняется массив t. Командой t(0) = 1 первый элемент массива получает единичное значение, а затем в теле оператора цикла командой t(i) = t(i-1)*z вычисляются прочие его элементы (индексная переменная i пробегает значения от 1 до n).

Результат функции записывается в переменную Q. Поскольку при вычислениях вызываются функции рабочего листа, чтобы каждый раз при вы-

зове функции не указывать перед ее именем ссылку Application.Worksheet Function, вычисления выполняются в With-блоке. В этом блоке, в частности, всего одна команда Q = .MMult(t,.MMult(.MInverse(C),.Transpose(y))), в которой использованы функция транспонирования Transpose(), функция вычисления обратной матрицы MInverse() и функция вычисления матричного произведения MMult().

На заметку

Точка перед именем функции означает, что это сокращенный вариант от полной инструкции вызова. Например, инструкция .Transpose() в данном случае — это сокращенный вариант вызова Application.WorksheetFunction. Transpose().

Что касается последовательности вычисления выражения Q = .MMult(t,.MMult (.MInverse(C),.Transpose(y))), то дело обстоит следующим образом. Инструкцией .MInverse(C) вычисляется матрица, обратная к матрице C, и результат умножается (по правилу матричного произведения) на транспонированный вектор y (инструкция .Transpose(y)). Результат этого подвыражения — вектор с коэффициентами полинома. После этого вектор t умножается на неявно вычисленный вектор коэффициентов, и результат записывается в переменную Q.

Наконец, переменная Q присваивается командой ИНТЕРППОЛИНОМ = Q в качестве значения функции.

На рисунке 13.7 приведен пример использования созданной нами функции ИНТЕРППОЛИНОМ() для вычисления значений интерполяционного полинома. Здесь мы используем ту же функциональную зависимость и те же узловые точки, что и в предыдущем примере.

По сравнению с документом на рисунке 13.6, здесь рабочих данных в документ приходится вводить намного меньше. А именно, в ячейках B4:L5

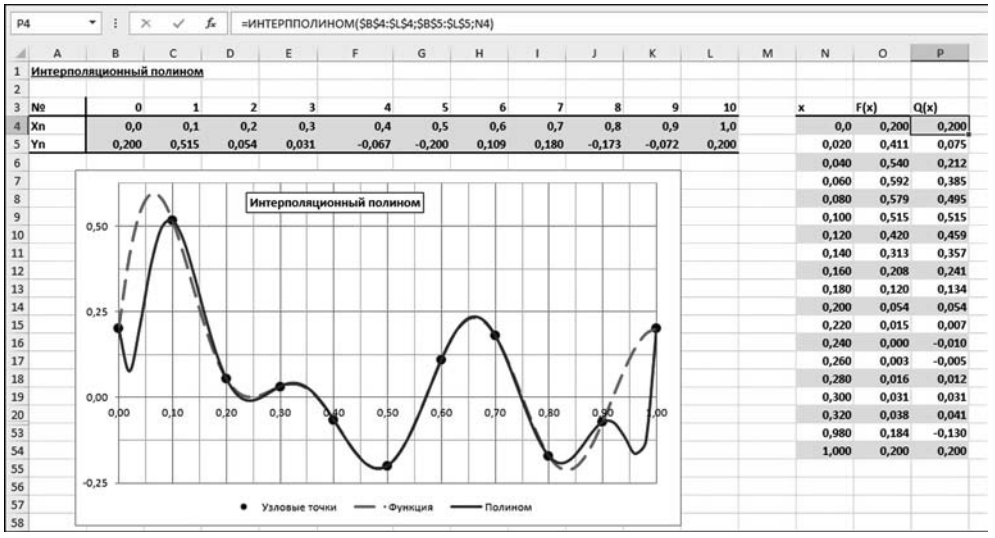


Рис. 13.7
Для вычисления интерполяционного полинома вызывается функция пользователя

указаны значения узловых точек и значения функции в узловых точках. Это та информация, которая необходима для вычисления коэффициентов полинома, т. е. для однозначного определения функциональной зависимости. Значения аргументов полинома указаны в ячейках N4:N54. Для сравнения приводятся значения табулированной функции (ячейки O4:O54), а сам полином вычисляется в ячейках P4:P54 (часть ячеек скрыта). Для заполнения диапазона ячеек P4:P54 в ячейку P4 вводится формула =ИНТЕРППОЛИНОМ ($B$4:$L$4;$B$5:$L$5;N4) и копируется во все остальные ячейки диапазона. Для удобства восприятия также показана диаграмма с графиками функции, полинома и узловыми точками. В этом смысле картинка не изменилась. Выигрыш, как отмечалось ранее, состоит в том, что нет необходимости явно вычислять полиномиальные коэффициенты и матрицу Вандермонда.

ИНТЕРПОЛЯЦИЯ НАБОРОМ ФУНКЦИЙ

Ох, красота-то какая! Лепота!

Из к/ф «Иван Васильевич меняет профессию»

Проведенные в конце предыдущего раздела вычисления наводят на простую мысль, что вместо интерполяционного полинома можно взять линейную комбинацию нескольких априори известных функций. В этом случае мы можем сформулировать задачу следующим образом. Пускай имеется набор из n узловых точек $x_1, x_2, \ldots, x_n$, а также известны значения некоторой функции в этих точках — соответственно $y_1, y_2, \ldots, y_n$. Кроме этого, заданы функции $\varphi_1(x), \varphi_2(x), \ldots, \varphi_n(x)$. Мы рассматриваем линейные комбинации вида $F(x) = a_1\varphi_1(x) + a_2\varphi_2(x) + \ldots + a_n\varphi_n(x)$. Задача следующая: нужно так подобрать коэффициенты $a_1, a_2, \ldots, a_n$ линейной комбинации, чтобы в узловых точках значения функции $F(x)$ совпадали со значениями табулированной функции, т. е. чтобы выполнялись соотношения вида $F(x_k) = y_k$ для всех индексов $k = 1, 2, \ldots, n$.

Эта задача сводится в общем случае к решению системы линейных неоднородных алгебраических уравнений вида $a_1\varphi_1(x_k) + a_2\varphi_2(x_k) + \ldots + a_n\varphi_n(x_k) = y_k$ (индекс $k = 1, 2, \ldots, n$) относительно коэффициентов $a_1, a_2, \ldots, a_n$. Чтобы ее формализовать, введем следующие обозначения: $\vec{a} = \left[a_1, a_2, \ldots, a_n\right]^T$, $\vec{y} = \left[y_1, y_2, \ldots, y_n\right]^T$, а также матрицу $\hat{\Phi}$ с элементами $\Phi_{ij} = \varphi_j(x_i)$. Тогда задача сводится к решению матричного уравнения $\hat{\Phi}\vec{a} = \vec{y}$ относительно вектора $\vec{a}$. Решение может быть записано как $\vec{a} = \hat{\Phi}^{-1}\vec{y}$ — разумеется, при условии, что обратная матрица $\hat{\Phi}^{-1}$ существует. А обратная матрица существует, если определитель матрицы $\hat{\Phi}$ не равен нулю.

Вопрос о выборе подходящего набора функций $\varphi_1(x), \varphi_2(x), \ldots, \varphi_n(x)$ нетривиален. Дело в том, что обычной линейной независимости функций в данном случае мало. Обычно накладывается условие, чтобы определитель матрицы $\hat{\Phi}$ с элементами $\Phi_{ij} = \varphi_j(x_i)$ был отличен от нуля не только для данного фиксированного набора узловых точек $x_1, x_2, \ldots, x_n$, но для любого набора. Далеко не любая система линейно независимых функций удовлетворяет

этому условию. Системы функций, для которых данное условие выполнено, называются *системами Чебышева*.

На заметку

Напомним, что функции $\varphi_1(x)$, $\varphi_2(x)$, ..., $\varphi_n(x)$ называются линейно независимыми (на некотором интервале), если никакая их линейная комбинация не обращается тождественно в нуль если хотя бы один коэффициент в этой комбинации отличен от нуля.

Практические последствия всего сказанного сводятся к тому, что далеко не каждый набор функций $\varphi_1(x)$, $\varphi_2(x)$, ..., $\varphi_n(x)$ подходит для выполнения интерполирования. Поскольку задача выбора подходящих систем функций выходит за рамки книги, ограничимся лишь рассмотрением некоторых частных случаев. В частности, решим задачу об интерполировании периодической на интервале от 0 до 1 функции на основе набора функций 1, $\sin(2\pi x)$, $\cos(2\pi x)$, $\sin(4\pi x)$, $\cos(4\pi x)$, ..., $\sin(2n\pi x)$, $\cos(2n\pi x)$ (всего $2n + 1$ функций, т. е. система функций подходит для интерполирования по $2n + 1$ узловой точке при условии, что в этот набор входит не больше одной граничной точки диапазона от 0 до 1).

На заметку

Желающие убедиться в корректности постановки такой задачи и ее особенностях могут обратиться к специальной литературе по этому вопросу.

Поскольку количество узловых точек при таком подходе должно быть нечетным, несколько изменим систему индексации узловых точек и адаптируем задачу под конкретный набор функций. Итак, узловые точки обозначим как x_0, x_1, ..., x_{2n} и, соответственно, значения табулированной функции в этих точках равны y_0, y_1, ..., y_{2n}. Для определенности будем полагать, что все узловые точки $0 \le x_k < 1$ (индекс $k = 0, 1, ..., 2n$). Тогда интерполяционный многочлен может быть записан в виде $T_n(x) = a_0 + \sum_{k=1}^{n} \left(a_k \cos(2\pi kx) + b_k \sin(2\pi kx)\right)$. Свойство этого многочлена таково, что он периодичен с периодом 1 (т. е. для любого целого числа N имеет место $T_n(x + N) = T_n(x)$), причем $T_n(0) = T_n(1)$.

На заметку

Выше было сказано, что мы строим интерполяционный многочлен для функции, периодичной на интервале от 0 до 1. Формально это так. Но мы можем рассматривать задачу и несколько шире. Предположим, что некоторая функция (не обязательно периодичная) табулирована на интервале от 0 до 1. Если мы подойдем к задаче формально и построим интерполяционный многочлен, то он определит некоторую функциональную зависимость, которая интерполирует табулированную функцию на интервале от 0 до 1, а за пределами этого интервала получаем периодическое продолжение интерполяционной зависимости для табулированной функции.

Из условий $T_n(x_k) = y_k$ (индекс $k = 0, 1, ..., 2n$) получаем $2n + 1$ условие для определения $2n + 1$ коэффициента (имеются в виду коэффициенты a_0, a_1, ..., a_n и b_1, b_2, ..., b_n).

Для представления формального решения задачи введем следующие обозначения: $\vec{q} = [a_0, a_1, \ldots, a_n, b_1, b_2, \ldots, b_n]^T$, $\vec{y} = [y_0, y_1, \ldots, y_{2n}]^T$ и матрица $\hat{\Phi}$ с элементами Φ_{ij} (индексы $i, j = 0, 1, \ldots, 2n$), которые определяются следующей системой соотношений: $\Phi_{ij} = \cos(2\pi j x_i)$ при $0 \le j \le n$ и $\Phi_{ij} = \sin(2\pi(j-n)x_i)$ при $n < j \le 2n$. Тогда вектор коэффициентов может быть найден как $\vec{q} = \hat{\Phi}^{-1}\vec{y}$, а значение интерполяционного многочлена может быть вычислено как произведение двух векторов $T_n(x) = \vec{\varphi}(x)\vec{q}$. Здесь мы использовали обозначение $\vec{\varphi}(x) = (1, \cos(2\pi x), \cos(4\pi x), \ldots, \cos(2\pi n x), \sin(2\pi x), \sin(4\pi x), \ldots, \sin(2\pi n x))$. Далее реализуем означенный подход в рабочем документе Excel. Мы рассмотрим функцию $f(x) = (3x-1)^2$ на интервале $0 \le x < 1$. Интерполяция будет выполняться по пяти узловым точкам (таким образом, $2n + 1 = 5$ и $n = 2$), которые определяются соотношениями $x_k = k/5$ (индекс $k = 0, 1, \ldots, 4$). Документ, в котором решена эта задача, представлен на рисунке 13.8.

В ячейки документа B2:F2 вводятся значения узловых точек, а в ячейках B3:F3 вычисляются значения табулированной функции в узловых точках. Индексы узловых точек отображаются в ячейках B1:F1.

На заметку

В диапазоне ячеек B1:F3 содержится вся исходная информация, необходимая для дальнейших вычислений. Заполняются ячейки так. Сначала в ячейку B1 вводим нулевое значение (начальный индекс узловой точки). В ячейку C1 вводится формула =B1+1 и копируется в прочие ячейки диапазона B1:F1. Для заполнения диапазона ячеек B2:F2 в ячейку B2 вводим формулу =B1/5 и копируем ее в остальные ячейки диапазона. Ячейки диапазона B3:F3 заполняются копированием формулы =(3*B2-1)^2 из ячейки B3.

В ячейках P2:P6 вычисляются коэффициенты для интерполяционного многочлена. Для этого в ячейках K2:O6 вычисляется матрица значений базисной системы функций в узловых точках. Диапазон заполняется «блоками». Сна-

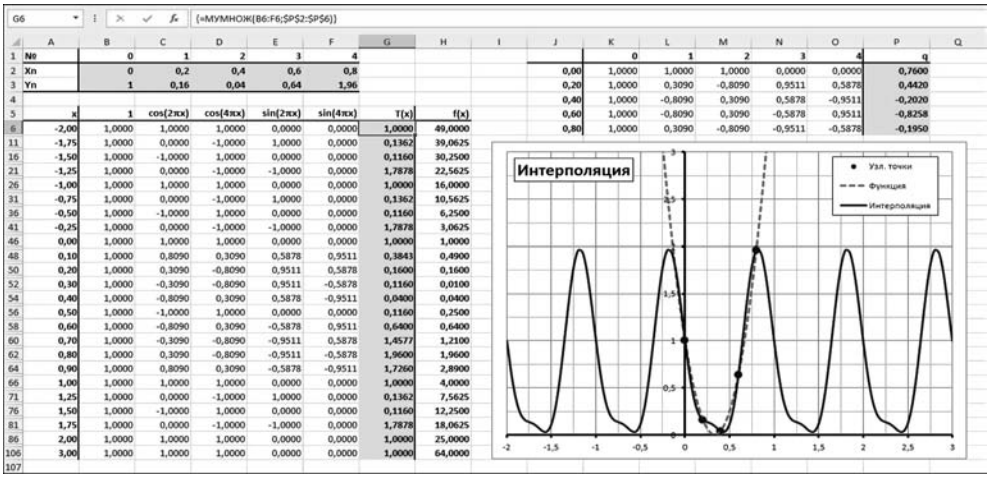


Рис. 13.8
Интерполирование периодической функции (часть ячеек скрыта)

чала в ячейки K2:K6 вводится формула массива =COS(2*K1*ПИ()*$J$2:$J$6) и копируется (всем столбиком) в ячейки L2:M6, а затем в столбик N2:N6 вводится формула массива =SIN(2*(N1-$M$1)*ПИ()*$J$2:$J$6) и копируется в ячейки O2:O6.

На заметку

При вычислении матрицы значений в узловых точках системы базисных функций использованы ссылки на ячейки J2:J6 (значения узловых точек) и K1:O1 (набор индексов для базисных функций — такой же, как и для узловых точек). Ячейки J2:J6 заполняются с помощью формулы массива =ТРАНСП(B2:F2), а в ячейки K1:O1 вводится формула массива =B1:F1.

Вектор коэффициентов для интерполяционного многочлена вычисляется в ячейках P2:P6 с помощью формулы массива =МУМНОЖ(МОБР(K2:O6); ТРАНСП(B3:F3)).

Для вычисления значений интерполяционного многочлена по набору точек заполняем диапазон ячеек A6:A106 значениями переменной x из интервала от -2 до 3. Эти значения послужат аргументом для интерполяционного многочлена и исходной функции, на основании табличных значений которой строится многочлен.

На заметку

Таким образом, хотя интерполирование выполняется на интервале от 0 до 1, полученное выражение для интерполяционного многочлена мы хотим использовать и за пределами интервала интерполирования. В частности, мы посмотрим, насколько поведение многочлена отличается от поведения функции вне пределов интервала интерполирования.

Для заполнения ячеек A6:A106 в ячейку A6 вводим значение -2, в ячейку A7 вводим формулу =A10+1/20 и копируем ее в нижние ячейки.

В ячейках B6:F106 вычисляются (построчно) значения базисных функций (т. е. функции, на основе которых строится интерполяционный многочлен). Для заполнения этого диапазона выполняем такие действия:

- в ячейку B6 вводим формулу =COS(2*B$1*ПИ()*$A6);
- формула из ячейки B6 копируется в ячейки C6 и D6;
- в ячейку E6 вводится формула =SIN(2*(E$1-$D$1)*ПИ()*$A6);
- формула из ячейки E6 копируется в ячейку F6;
- выделяем диапазон ячеек B6:F6 и с помощью маркера автоматического заполнения копируем содержимое этого массива в ячейки снизу.

Значения интерполяционного многочлена вычисляются в ячейках G6:G106. Для заполнения этого массива в ячейку G6 вводим формулу массива =МУМНОЖ(B6:F6;$P$2:$P$6), которой значение интерполяционного многочлена (для аргумента в ячейке A6) вычисляется как скалярное произведение вектора с базисными функциями (ячейки B6:F6) на вектор коэффициентов полинома (ячейки P2:P6). Затем формула из ячейки G6 копируется во все остальные ячейки диапазона G6:G106. В ячейках H6:H106 вычисляются значения функции $f(x) = (3x - 1)^2$ (для значения переменной x из столбца A). Диапазон заполняется копированием из ячейки H6 формулы =(3*A6-1)^2.

На рисунке 13.8, помимо непосредственно вычислительных данных, представлена еще и диаграмма с графиком функции $f(x) = (3x - 1)^2$, графиком интерполяционного многочлена и выделенными узловыми точками. Если характеризовать ситуацию в целом, т. е. несколько моментов, на которые хочется обратить внимание.

Во-первых, интерполяционный многочлен на интервале интерполирования (от 0 до 1) достаточно неплохо приближает исходную функцию — за исключением правой границы интервала (вблизи значения 1). Главная причина связана с тем, что априори мы выбрали функцию, которая не удовлетворяет условию периодичности на интервале интерполирования: при значениях $x = 0$ и $x = 1$ функция $f(x)$ принимает разные значения, т. е. условие $f(0) = f(1)$ не выполнено. Интерполяционный же полином $T_n(x)$ однозначно периодичный.

Во-вторых, вне пределов интерполирования функция и интерполяционный многочлен ведут себя совершенно по-разному и об интерполировании вне пределов интервала (0, 1) говорить не приходится. Если бы нам необходимо было интерполировать функцию на интервале, отличном от интервала (0, 1), пришлось бы строить другой интерполяционный многочлен. Поэтому важно не только уметь построить интерполяционную зависимость, но и четко представлять, когда и где она применима.

ИНТЕРПОЛЯЦИЯ СПЛАЙНАМИ

Фигуры, может, и нет,
а характер — налицо.

Из к/ф «Девчата»

Выше мы строили интерполяционные зависимости (в основном полиномиальные), которые описывались одним общим выражением для всего интервала интерполирования. Но задача может быть сформулирована и несколько иначе. А именно, вместо того, чтобы строить, например, один интерполяционный полином для всего интервала интерполирования, можем для каждого интервала между соседними узловыми точками использовать различные функции. Чтобы построенная таким образом интерполяционная зависимость соответствовала непрерывной и достаточно гладкой функции, в узловых точках накладывается условие непрерывности (вместе с непрерывностью производных нескольких первых порядков). Если еще более конкретно, то можем в качестве функций, которые используются для построения интерполяционной зависимости между соседними узловыми точками, выбрать полиномы небольшой степени (*сплайны*) — на практике обычно речь идет о полиномах третьей степени. В этом случае говорят об *интерполяции сплайнами*.

Как и ранее, рассмотрим задачу об интерполировании некоторой функции, которая задана значениями $y_1, y_2, \ldots, y_n$ в узловых точках $x_1, x_2, \ldots, x_n$ соответственно. Для решения задачи мы рассматриваем систему полиномов $S_1(x), S_2(x), \ldots, S_{n-1}(x)$ (третьей степени каждый) таких, что если для аргу-

мента x имеет место соотношение $x_i \le x < x_{i+1}$, то значение интерполяционной зависимости определяется выражением $S_i(x)$. Задача сводится к тому, чтобы определить коэффициенты полиномов $S_1(x)$, $S_2(x)$, ..., $S_{n-1}(x)$. Поскольку каждый полином, как отмечалось, третей степени, то в каждом полиноме необходимо определить четыре коэффициента. Всего на основе информации об интерполируемой функции необходимо определить $4(n-1)$ коэффициентов (при условии, что узловых точек n — тогда полиномов $n-1$, и в каждом по 4 коэффициента). Ограничения, накладываемые на значения интерполяционной зависимости в узловых точках, дают n условий. Ограничение на непрерывность интерполяционной зависимости на внутренних узловых точках — это еще $n-2$ условия. Для однозначного определения коэффициентов полиномов необходимы еще $2(n-1)$ условий. Если наложить условие непрерывности первой и второй производных на внутренних узловых точках — это еще $2(n-2)$ условия. Недостающие 2 условия можно выбирать по-разному. Но обычно они накладываются на вторые производные на границах интервала интерполирования: в граничных узловых точках интервала интерполирования вторые производные должны равняться нулю (так называемое условие свободной границы). Именно такой подход мы и используем. Тогда задача формализуется следующим образом.

Имеются узловые точки x_1, x_2, ..., x_n, и известны значения некоторой функции в этих точках y_1, y_2, ..., y_n. Необходимо определить набор полиномов $S_1(x)$, $S_2(x)$, ..., $S_{n-1}(x)$ третьей степени по аргументу x таких, чтобы выполнялись следующие условия:

- $S_k(x_k) = y_k$ (для индексов $k = 1, 2, \ldots, n-1$) и $S_{n-1}(x_n) = y_n$ (значения в узловых точках);
- $S_k(x_k) = S_{k+1}(x_k)$ для индексов $k = 2, 3, \ldots, n-1$ (непрерывность в узловых точках);
- $S'_k(x_k) = S'_{k+1}(x_k)$ для индексов $k = 2, 3, \ldots, n-1$ (непрерывность первой производной в узловых точках);
- $S''_k(x_k) = S''_{k+1}(x_k)$ для индексов $k = 2, 3, \ldots, n-1$ (непрерывность второй производной в узловых точках);
- $S''_1(x_1) = S''_{n-1}(x_n) = 0$ (условие свободной границы).

Приведенные выше условия позволяют однозначно определить сплайн-полиномы. Если сплайн-полиномы $S_1(x)$, $S_2(x)$, ..., $S_{n-1}(x)$ определены, то общую функциональную зависимость можно представить в виде $S(x) = \sum_{k=1}^{n-1} \theta(x - x_k)\theta(x_{k+1} - x)S_k(x)$, где через $\theta(z)$ обозначена функция Хевисайда — она равна нулю при отрицательном аргументе и равна единице, если аргумент больше или равен нулю (т. е. $\theta(z) = 0$ при $z < 0$ и $\theta(z) = 1$ при $z \ge 0$).

На заметку

В силу того определения, которое мы привели для функции Хевисайда, результатом выражения $\theta(x - x_k)\theta(x_{k+1} - x)$ является 1, если $x_k \le x < x_{k+1}$; 0 если $x < x_k$ или $x \ge x_{k+1}$.

Поскольку условий достаточно много и они качественно разные, важно удачно определить структуру сплайн-полиномов. Другим словами, далеко

не последнюю роль в решении задачи об интерполяции сплайнами играет система обозначений, которую мы используем. Поэтому мы постараемся максимально облегчить себе задачу, выбрав «правильную» структуру интерполяционных сплайн-полиномов. Предварительно введем некоторые обозначения, которые позволят упростить некоторые выражения. В частности, обозначим $\Delta x_k = x_{k+1} - x_k$ и $\varphi_k(x) = \dfrac{x - x_k}{\Delta x_k} \equiv \dfrac{x - x_k}{x_{k+1} - x_k}$ (индекс $k = 1, 2, \ldots, n-1$). Особенность функций $\varphi_k(x)$, ради которой мы их используем, состоит в том, что $\varphi_k(x_k) = 0$ и $\varphi_k(x_{k+1}) = 1$. Кроме того, обозначим $\psi_k(x) \equiv 1 - \varphi_k(x) = \dfrac{x_{k+1} - x}{x_{k+1} - x_k} \equiv \dfrac{x_{k+1} - x}{\Delta x_k}$. Для функции $\psi_k(x_k) = 1$ и $\psi_k(x_{k+1}) = 0$. Легко понять, что $\varphi_k(x_k)\psi_k(x_k) = \varphi_k(x_{k+1})\psi_k(x_{k+1}) = 0$. Также имеют место следующие соотношения для производных: $\varphi'_k(x) = \dfrac{1}{\Delta x_k}$, $\psi'_k(x) = -\dfrac{1}{\Delta x_k}$ и $\varphi''_k(x) = \psi''_k(x) = 0$. Будем искать интерполяционные сплайн-полиномы в следующем виде:

$$S_k(x) = y_k\psi_k(x) + y_{k+1}\varphi_k(x) - \varphi_k(x)\psi_k(x)\Delta x_k^2\left(a_k(1+\psi_k(x)) + a_{k+1}(1+\varphi_k(x))\right)$$

(индекс $k = 1, 2, \ldots, n-1$). В этом случае в узловых точках они удовлетворяют соотношениям $S_k(x_k) = y_k$ и $S_k(x_{k+1}) = y_{k+1}$, что обеспечивает нужные значения и непрерывность интерполяционной кривой в узловых точках интервала интерполирования.

Для первой и второй производных от сплайн-полинома имеем следующее: $S'_k(x) = \dfrac{\Delta y_k}{\Delta x_k} + \Delta x_k\left(a_{k+1}(3\varphi_k^2(x) - 1) - a_k(3\psi_k^2(x) - 1)\right)$ и $S''_k(x) = 6\left(a_{k+1}\varphi_k(x) + a_k\psi_k(x)\right)$, где мы обозначили $\Delta y_k = y_{k+1} - y_k$. Значения производных в узловых точках определяются соотношениями $S'_k(x_k) = \dfrac{\Delta y_k}{\Delta x_k} - \Delta x_k\left(2a_k + a_{k+1}\right)$, $S'_k(x_{k+1}) = \dfrac{\Delta y_k}{\Delta x_k} + \Delta x_k\left(2a_{k+1} + a_k\right)$, $S''_k(x_k) = 6a_k$ и $S''_k(x_{k+1}) = 6a_{k+1}$.

Условие равенства на внутренних узловых точках (индекс $k = 2, 3, \ldots, n-1$) вторых производных $S''_k(x_{k+1}) = S''_{k+1}(x_{k+1})$ выполняется автоматически, поскольку $S''_{k+1}(x_{k+1}) = 6a_{k+1} = S''_k(x_{k+1})$. Два дополнительных условия (равенство нулю вторых производных в граничных точках интервала интерполирования) записываются как $S''_1(x_1) = 6a_1 = 0$ и $S''_{n-1}(x_n) = 6a_n = 0$, что дает «граничные условия» $a_1 = a_n = 0$ для коэффициентов a_k, через которые определяются сплайн-полиномы $S_k(x)$.

На заметку

Напомним, что в исходной постановке задача содержала $4(n-1)$ неизвестных параметров. Выбирая эти параметры, необходимо было удовлетворить достаточно большому количеству ограничений, накладываемых на интерполяционную зависимость. За счет «удачного» выбора структуры сплайн-полиномов нам удалось удовлетворить большинство условий («правильные» значения в узловых точках, а также непрерывность на внутренних узловых точках вторых производных и равенство нулю вторых производных на граничных узловых точках) и свести задачу к определению n коэффициентов a_k (причем a_1 и a_n вычисляются сразу). Таким образом, нужно вычислить коэффициенты $a_2, a_3, \ldots, a_{n-1}$, исходя из условия непрерывности первых производных в узловых точках.

Условие непрерывности первых производных на внутренних узловых точках $S'_k(x_{k+1}) = S'_{k+1}(x_{k+1})$ (индекс $k = 2, 3, \ldots, n - 1$) дает уравнения $\frac{\Delta y_k}{\Delta x_k} + \Delta x_k (2a_{k+1} + a_k) = \frac{\Delta y_{k+1}}{\Delta x_{k+1}} - \Delta x_{k+1}(2a_{k+1} + a_{k+2})$, или после преобразования $\Delta x_k a_k + 2(\Delta x_k + \Delta x_{k+1})a_{k+1} + \Delta x_{k+1} a_{k+2} = \frac{\Delta y_{k+1}}{\Delta x_{k+1}} - \frac{\Delta y_k}{\Delta x_k}$. Таким образом, имеем дело с системой линейных уравнений относительно коэффициентов $a_2, a_3, \ldots, a_{n-1}$ (напомним, что $a_1 = a_n = 0$).

На заметку

Если для решения используется приложение вроде Excel и интерес представляет конечный результат, то на этом можно теоретическую часть завершать и переходить к вычислениям в рабочем документе. Схема достаточно простая: вычисляются элементы матрицы коэффициентов линейной системы, вектор правых частей, вычисляется обратная матрица, после чего находим коэффициенты для сплайн-полиномов. Тем не менее такой подход не всегда применим — особенно если узловых точек много. Поэтому для решения системы алгебраических линейных уравнений нередко используют специальные методы — благо, матрица коэффициентов линейной системы в данном случае имеет достаточно простую структуру.

Матрица коэффициентов линейной системы, из которой вычисляются параметры сплайн-полиномов, имеет *трехдиагональную структуру*: у нее отличные от нуля элементы только на главной диагонали и смежных с главной. Таким образом, каждое уравнение системы (за исключением первого и последнего) содержит три неизвестных параметра (первое и последнее уравнения — по два). Для решения системы линейных уравнений с трехдиагональной матрицей коэффициентов используется *метод прогонки*. Алгоритм реализации метода прогонки поясним на примере нашей задачи.

Итак, первое уравнение системы содержит параметры a_2 и a_3; второе уравнение системы содержит параметры a_2, a_3 и a_4; третье — a_3, a_4 и a_5, и так до $n - 3$-го уравнения, которое содержит параметры a_{n-3}, a_{n-2} и a_{n-1}. Последнее, $n - 2$-е уравнение содержит параметры a_{n-2} и a_{n-1}. Тогда мы можем из первого уравнения выразить параметр a_2 через a_3 и подставить это выражение во второе уравнение. Это дает возможность выразить параметр a_3 через параметр a_4, подставить это выражение в третье уравнение, выразить a_4 через a_5, и так далее до $n - 3$-го уравнения, на основе которого мы выразим параметр a_{n-2} через a_{n-1}. Подставив это выражение в последнее, $n - 2$-е уравнение, в которое входят только два параметра a_{n-2} и a_{n-1}, находим параметр a_{n-1}. Это *прямая прогонка*. Затем начинается *обратная прогонка* — на основе полученных линейных соотношений находим на основе значения параметра a_{n-1} значение параметра a_{n-2}, на его основе находится параметр a_{n-3}, и т. д., пока не вычислим параметр a_2.

Описанная процедура вычисления параметров $a_2, a_3, \ldots, a_{n-1}$ может быть конкретизирована. Для этого стоит заметить, что если мы выражаем параметр a_k через a_{k+1}, то каждый раз, в силу линейности исходных уравнений, соотношение между этими параметрами также будет линейным, т. е. имеет вид $a_k = \alpha_k a_{k+1} + \beta_k$, где α_k и β_k — некоторые параметры. В силу принятых

обозначений можем записать $a_{k+1} = \alpha_{k+1}a_{k+2} + \beta_{k+1}$. С другой стороны, имеет место соотношение $\Delta x_k a_k + 2(\Delta x_k + \Delta x_{k+1})a_{k+1} + \Delta x_{k+1}a_{k+2} = \frac{\Delta y_{k+1}}{\Delta x_{k+1}} - \frac{\Delta y_k}{\Delta x_k}$. Тогда должно выполняться следующее равенство: $a_{k+1} = -\frac{\Delta x_{k+1}}{(2+\alpha_k)\Delta x_k + 2\Delta x_{k+1}}a_{k+2} + \frac{\frac{\Delta y_{k+1}}{\Delta x_{k+1}} - \frac{\Delta y_k}{\Delta x_k} - \beta_k \Delta x_k}{(2+\alpha_k)\Delta x_k + 2\Delta x_{k+1}}$. и $\beta_{k+1} = \frac{\frac{\Delta y_{k+1}}{\Delta x_{k+1}} - \frac{\Delta y_k}{\Delta x_k} - \beta_k \Delta x_k}{(2+\alpha_k)\Delta x_k + 2\Delta x_{k+1}}$. Эти рекуррентные соотношения позволяют последовательно вычислять параметры α_k и β_k. Учитывая в явном виде первое уравнение $2(\Delta x_1 + \Delta x_2)a_2 + \Delta x_2 a_3 = \frac{\Delta y_2}{\Delta x_2} - \frac{\Delta y_1}{\Delta x_1}$, легко находим $a_2 = -\frac{\Delta x_2}{2(\Delta x_1 + \Delta x_2)}a_3 + \frac{\frac{\Delta y_2}{\Delta x_2} - \frac{\Delta y_1}{\Delta x_1}}{2(\Delta x_1 + \Delta x_2)}$, что дает $\alpha_2 = -\frac{\Delta x_2}{2(\Delta x_1 + \Delta x_2)}$ и $\beta_2 = \frac{\frac{\Delta y_2}{\Delta x_2} - \frac{\Delta y_1}{\Delta x_1}}{2(\Delta x_1 + \Delta x_2)}$. Таким образом, на основе этих «начальных значений», используя приведенные выше рекуррентные соотношения, можем вычислить параметры α_3, β_3, α_4, β_4, …, α_{n-2}, β_{n-2}. Тогда из последнего $n-2$-го уравнения $\Delta x_{n-2}a_{n-2} + 2(\Delta x_{n-2} + \Delta x_{n-1})a_{n-1} = \frac{\Delta y_{n-1}}{\Delta x_{n-1}} - \frac{\Delta y_{n-2}}{\Delta x_{n-2}}$ с учетом соотношения $a_{n-2} = \alpha_{n-2}a_{n-1} + \beta_{n-1}$ находим $a_{n-1} = \frac{\frac{\Delta y_{n-1}}{\Delta x_{n-1}} - \frac{\Delta y_{n-2}}{\Delta x_{n-2}} - \beta_{n-2}\Delta x_{n-2}}{(2+\alpha_{n-2})\Delta x_{n-2} + 2\Delta x_{n-1}}$. После этого на основе соотношений вида $a_k = \alpha_k a_{k+1} + \beta_k$ вычисляем прочие параметры.

Собственно, это весь алгоритм, позволяющий выполнить сплайн-интерполяцию. Теперь рассмотрим, как все это реализуется на практике.

На заметку

При введении формулы для вычисления интерполяционных сплайн-полиномов, вместо громоздкого выражения

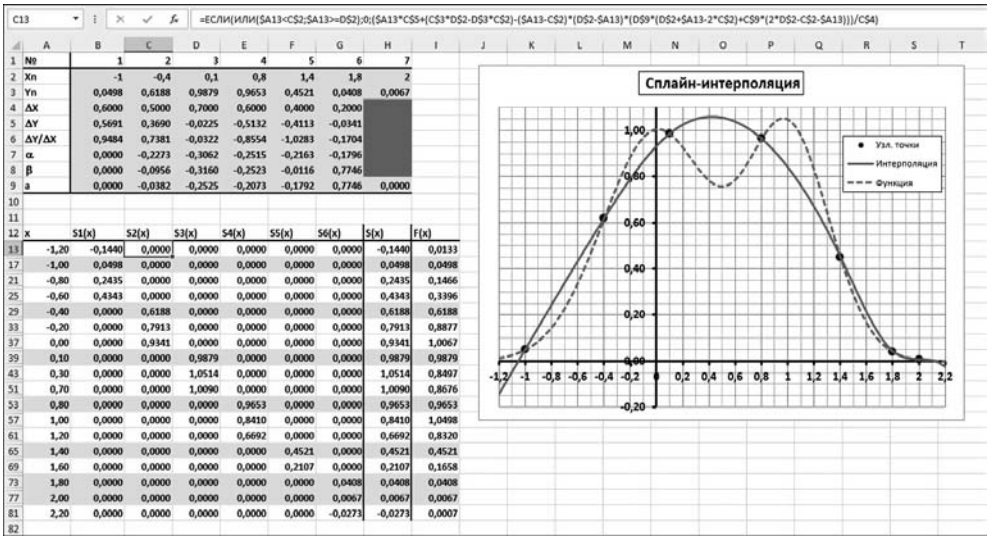
$$S_k(x) = y_k\psi_k(x) + y_{k+1}\varphi_k(x) - \\ - \varphi_k(x)\psi_k(x)\Delta x_k^2(a_k(1+\psi_k(x)) + a_{k+1}(1+\varphi_k(x)))$$

используем не менее громоздкое, но более удобное для практического использования выражение

$$S_k(x) = \frac{1}{\Delta x_k}(x\Delta y_k + y_k x_{k+1} - y_{k+1}x_k - \\ - (x-x_k)(x_{k+1}-x)(a_{k+1}(x_{k+1}+x-2x_{k+1}) + a_k(2x_{k+1}-x_k-x))).$$

На рисунке 13.9 представлен рабочий документ, в котором производятся вычисления для сплайн-интерполяции табулированной функции.

Рабочий документ состоит из двух блоков ячеек с расчетными данными и диаграммы, на которой отображены узловые точки и кривая для функции, на



№		1	2	3	4	5	6	7
Xn		-1	-0,4	0,1	0,8	1,4	1,8	2
Yn		0,0498	0,6188	0,9879	0,9653	0,4521	0,0408	0,0067
ΔX		0,6000	0,5000	0,7000	0,6000	0,4000	0,2000	
ΔY		0,5691	0,3690	-0,0225	-0,5132	-0,4113	-0,0341	
ΔY/ΔX		0,9484	0,7381	-0,0322	-0,8554	-1,0283	-0,1704	
α		0,0000	-0,2273	-0,3062	-0,2515	-0,2163	-0,1796	
β		0,0000	-0,0956	-0,3160	-0,2523	-0,0116	0,7746	
a		0,0000	-0,0382	-0,2525	-0,2073	-0,1792	0,7746	0,0000

x	S1(x)	S2(x)	S3(x)	S4(x)	S5(x)	S6(x)	S(x)	F(x)
-1,20	-0,1440	0,0000	0,0000	0,0000	0,0000	0,0000	-0,1440	0,0133
-1,00	0,0498	0,0000	0,0000	0,0000	0,0000	0,0000	0,0498	0,0498
-0,80	0,2435	0,0000	0,0000	0,0000	0,0000	0,0000	0,2435	0,1466
-0,60	0,4343	0,0000	0,0000	0,0000	0,0000	0,0000	0,4343	0,3396
-0,40	0,6188	0,0000	0,0000	0,0000	0,0000	0,0000	0,6188	0,6188
-0,20	0,7913	0,0000	0,0000	0,0000	0,0000	0,0000	0,7913	0,8877
0,00	0,9341	0,0000	0,0000	0,0000	0,0000	0,0000	0,9341	1,0067
0,10	0,0000	0,0000	0,9879	0,0000	0,0000	0,0000	0,9879	0,9879
0,30	0,0000	0,0000	1,0514	0,0000	0,0000	0,0000	1,0514	0,8497
0,70	0,0000	0,0000	1,0090	0,0000	0,0000	0,0000	1,0090	0,8676
0,80	0,0000	0,0000	0,0000	0,9653	0,0000	0,0000	0,9653	0,9653
1,00	0,0000	0,0000	0,0000	0,8410	0,0000	0,0000	0,8410	1,0498
1,20	0,0000	0,0000	0,0000	0,6692	0,0000	0,0000	0,6692	0,8320
1,40	0,0000	0,0000	0,0000	0,0000	0,4521	0,0000	0,4521	0,4521
1,60	0,0000	0,0000	0,0000	0,0000	0,2107	0,0000	0,2107	0,1658
1,80	0,0000	0,0000	0,0000	0,0000	0,0000	0,0408	0,0408	0,0408
2,00	0,0000	0,0000	0,0000	0,0000	0,0000	0,0067	0,0067	0,0067
2,20	0,0000	0,0000	0,0000	0,0000	0,0000	-0,0273	-0,0273	0,0007

Рис. 13.9
Интерполяция сплайнами

основе которой вычислялись табличные значения, а также кривой, представляющей интерполяционную сплайн-функцию. Проанализируем этот документ.

Ячейки B1:H7 содержат номера (индексы) узловых точек и играют скорее вспомогательную роль (заполняются копированием формулы =B1+1 из ячейки C1 в ячейки справа, при этом в ячейку B1 вводится значение 1). Диапазон ячеек B2:H2 содержит значения узловых точек. В данном случае точки распределены на интервале интерполирования неравномерно, так что каждое значение для узловой точки вводится «индивидуально».

В ячейках диапазона B3:H3 вычисляются значения функции $f(x) = \exp(-3x^2) + \exp(-5(x-1)^2)$ в узловых точках (ячейки B2:H2). Для заполнения ячеек вводим формулу =EXP(-3*B2^2)+EXP(-5*(B2-1)^2) в ячейку B3 и затем копируем эту формулу в ячейки диапазона справа.

В ячейках диапазона B4:G6 вычисляются некоторые вспомогательные величины (при этом ячейки H4:H6 остаются незаполненными). Так, в ячейку B4 вводится формула = C2-B2 и копируется в ячейки той же строки справа — в результате диапазон ячеек B4:G4 содержит значения параметров Δx_k. Диапазон ячеек B5:G5 заполняется копированием из ячейки B5 формулы =C3-B3 (получаем набор значений Δy_k). В ячейки B6:G6 заносятся значения параметров $\dfrac{\Delta y_k}{\Delta x_k}$ (из ячейки B6 копируется формула =B5/B4).

Ячейки B7:G8 содержат результаты вычислений параметров α_k и β_k, необходимых, в свою очередь, для вычисления коэффициентов a_k (через которые выражаются сплайн-полиномы). Вычисления мы проводим по следующей схеме. В ячейку B7 вводится нулевое значение (поскольку $\alpha_1 = 0$). В ячейку C7 вводится формула =-C4/((2+B7)*B4+2*C4). После этого формула

копируется в ячейки D7:G7. В результате параметры α_k вычислены. Для вычисления параметров β_k в ячейку B8 вводится нулевое значение (поскольку $\beta_1 = 1$). В ячейку C8 вводится формула =(C6-B6-B8*B4)/((2+B7)*B4+2*C4). Формулу копируем в ячейки D8:G8. После этого можно приступить к вычислению параметров a_k — вычисления проводятся в ячейках B9:H9. В ячейку H9 вводится нулевое значение (так как $a_n = 0$), а в ячейку G9 вводится формула =G7*H9+G8. Эта формула из ячейки G9 копируется (справа налево, с помощью маркера автоматического заполнения) в ячейки B9:F9. На этом все основные расчеты выполнены. У нас есть вся необходимая информация для вычисления сплайн-полиномов. Осталось одна небольшая проблема: сплайн-полиномов несколько, а для вычисления значения интерполяционной зависимости при заданном значении аргумента нужно выбрать только один из полиномов. У этой задачи могут быть разные решения. Мы воспользуемся следующим подходом: вычисляя сплайн-полиномы для внутренних интервалов, будем полагать их тождественно равными нулю за пределами интервала (левая граничная точка включается в область ненулевого значения сплайн-полинома, а правая — нет). Для сплайн-полиномов, соответствующих граничным интервалам, мы не будем обнулять полином за пределами области интерполирования. При этом интерполяционная зависимость реализуется в виде суммы всех сплайн-полиномов.

На заметку

Фактически, мы используем представление

$$S(x) = \begin{cases} S_1(x) & \text{при} \quad x < x_2, \\ S_2(x) & \text{при} \quad x_2 \le x < x_3, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ S_{n-2}(x) & \text{при} \quad x_{n-2} \le x < x_{n-1}, \\ S_{n-1}(x) & \text{при} \quad x \ge x_{n-1}. \end{cases}$$

Для вычисления значений интерполяционной сплайновой зависимости и построения соответствующей кривой заполняем ячейки A13:A77:

- в ячейку A13 вводим начальное значение -1,2;
- в ячейку A14 вводим формулу =A16+0,05;
- копируем формулу в нижние ячейки.

На заметку

На рисунке 13.9 часть строк в диапазоне с 13-й по 77-ю скрыта.

Что касается ячеек B13:G77, то здесь вычисляются значения сплайн-полиномов (одна строка соответствует определенному аргументу, а каждый столбец соответствует отдельному сплайн-полиному). В ячейку B13 вводится формула =ЕСЛИ($A13>=C$2;0;($A13*B$5+(B$3*C$2-C$3*B$2)-($A13-B$2)*(C$2-$A13)*(C$9*(C$2+$A13-2*B$2)+B$9*(2*C$2-B$2-$A13)))/B$4). В ячейку C13 вводим формулу и копируем ее в ячейки D13, E13 и F13. В ячейку G13 вводится формула =ЕСЛИ($A13<G$2;0;($A13*G$5+(G$3*H$2-H$3*G$2)-($A13-G$2)*(H$2-$A13)*(H$9*(H$2+$A13-2*G$2)+G$9*(2*H$2-G$2-$A13)))/G$4). Затем выделяется строка B13:G13 и копируется (перетягиванием маркера

автоматического заполнения) построчно, вплоть до заполнения ячеек диапазона B13:G77.

На заметку

Выражения для первого (столбец B) и последнего (столбец G) сплайн-полиномов несколько отличаются от выражений для прочих полиномов условием в функции ЕСЛИ(). Поэтому формулы для первого и последнего сплайн-полиномов вводятся отдельно, в то время как для прочих полиномов получаются копированием из одной ячейки. На практике можно поступить проще: заполнить все ячейки диапазона B13:G13 на основе одной формулы (копированием), а затем отредактировать формулы в ячейках B13 и G13.

В ячейку H13 вводится формула =СУММ(B13:G13), которой вычисляется сумма ячеек слева — тех ячеек, в которых записаны значения всех интерполяционных полиномов для данного значения аргумента. Поскольку только один из них может иметь ненулевое значение, то отображаемое в ячейке H13 значение — это как раз и есть значение интерполяционного полинома в данной точке. На основе формулы из ячейки H13 копированием заполняем весь диапазон H13:H77. На основании значений в этих ячейках строится интерполяционная кривая. Кривая для исходной функциональной зависимости строится на основе данных в ячейках I13:I77. Диапазон заполняется копированием из ячейки I13 формулы =EXP(-3*A13^2)+EXP(-5*(A13-1)^2).

Само собой разумеется, что для вычисления интерполяционной зависимости, построенной на основе сплайн-полиномов, вполне допустимо (и даже целесообразно) написать функцию, которую можно было бы вызывать в рабочем листе. Аргументами ей можно было бы передавать диапазоны ячеек с узловыми точками, значениями табулируемой функции в узловых точках, а также значение (точку) для аргумента сплайн-функции — точно такие же аргументы передавались функциям, которые вычисляли интерполяционный полином (Ньютона или Лагранжа).

Программный код такой функции не очень сложен. Как он может выглядеть, показано в листинге 13.4.

Листинг 13.4. Функция для выполнения интерполяции сплайнами

```
Function СПЛАЙН(Px As Range, Py As Range, z As Variant) As Double
 ‘ Переменная для определения количества узловых точек
 Dim n As Integer
 ‘ Количество узловых точек
 n = Px.Count
 ‘ Переменная для аргумента сплайн-полинома
 Dim t As Double
 ‘ Значение аргумента для сплайн-полинома
 t = z
 ‘ Переменная для записи значения сплайн-полинома
 Dim S As Double
 ‘ Переменная для запоминания индекса полинома
 Dim k As Integer
```

```
' Массив для записи узловых точек
ReDim X(1 To n) As Double
' Массив для записи значений функции в узловых точках
ReDim Y(1 To n) As Double
' Массив для коэффициентов сплайн-полинома
ReDim a(1 To n) As Double
' Массив для параметров в методе прогонки
ReDim alpha(1 To n - 1) As Double
' Массив для параметров в методе прогонки
ReDim beta(1 To n - 1) As Double
' Массив приращений аргумента в узловых точках
ReDim dX(1 To n - 1) As Double
' Массив приращений функции в узловых точках
ReDim dY(1 To n - 1) As Double
' Индексная переменная для оператора цикла
Dim i As Integer
' Заполнение массивов
For i = 1 To n
  ' Узловые точки
  X(i) = Px.Cells(i).Value
  ' Значения функции в узловых точках
  Y(i) = Py.Cells(i).Value
Next i
' Заполнение массивов
For i = 1 To n - 1
  ' Приращение аргумента в узловых точках
  dX(i) = X(i + 1) - X(i)
  ' Приращение функции в узловых точках
  dY(i) = Y(i + 1) - Y(i)
Next i
' Начальный параметр в методе прогонки
alpha(1) = 0
' Начальный параметр в методе прогонки
beta(1) = 0
' Вычисление параметров для метода прогонки
For i = 2 To n - 1
  ' Параметр для метода прогонки
  alpha(i) = (-1) * dX(i) / _
  ((2 + alpha(i - 1)) * dX(i - 1) + 2 * dX(i))
  ' Параметр для метода прогонки
  beta(i) = (dY(i) / _
  dX(i) - dY(i - 1) / dX(i - 1) - beta(i - 1) * dX(i - 1)) / _
  ((2 + alpha(i - 1)) * dX(i - 1) + 2 * dX(i))
Next i
' Последний коэффициент для сплайн-полиномов
a(n) = 0
```

```
    ' Обратная прогонка
    For i = n - 1 To 1 Step -1
      ' Вычисление коэффициентов для сплайн-полиномов
      a(i) = alpha(i) * a(i + 1) + beta(i)
    Next i
    ' Начальное значение для индекса сплайн-полинома
    k = n - 1
    ' Определение индекса для сплайн-полинома
    For i = 2 To n - 1
      ' Если аргумент слева от узловой точки
      If (t < X(i)) Then
        ' Новое значение для индекса сплайн-полинома
        k = i - 1
        ' Завершение оператора цикла
        Exit For
      End If
    Next i
    ' Вычисление значения сплайн-полинома
    S = (t * dY(k) + Y(k) * X(k + 1) - Y(k + 1) * X(k) - _
    (t - X(k)) * (X(k + 1) - t) * (a(k + 1) * (X(k + 1) + t - 2 * X(k)) _
    + a(k) * (2 * X(k + 1) - X(k) - t))) / dX(k)
    ' Результат функции
    СПЛАЙН = S
End Function
```

Мы описываем функцию СПЛАЙН(), у которой три аргумента: диапазон ячеек со значениями узловых точек Px, диапазон ячеек со значениями в узловых точках табулированной функции Py, а также аргумент z, определяющий точку, для которой вычисляется значение интерполяционной функции.

Переменная n со значением Px.Count определяет количество узловых точек. Переменная t нужна для запоминания аргумента сплайн-полинома. Значение этой переменной определяется третьим аргументом функции СПЛАЙН(). Значение сплайн-полинома будет записываться в действительную числовую переменную S. Кроме этой переменной, нам понадобится еще одна переменная, в которую будет записано значение индекса полинома. Для этих целей используется целочисленная переменная k.

Также в программном коде объявляется и создается несколько массивов. Три массива содержат по n элементов типа Double: массив узловых точек X, массив значений функции в узловых точках Y, а также массив коэффициентов для сплайн-полинома a. Четыре массива содержат по n-1 элементов типа Double: массивы alpha и beta для вычисления и записи параметров, используемых в методе прогонки, а также массивы dX и dY приращений аргумента в узловых точках и приращений значения функции в узловых точках. Заполнение массивов выполняется с помощью нескольких последовательно выполняемых операторов цикла. Думается, не требуют особых комментариев команды заполнения массивов с узловыми точками, значениями функции

в узловых точках, а также массивы с приращениями этих параметров. Несколько более сложным образом заполняются массивы с параметрами для метода прогонки (массивы alpha и beta). Начальные (с индексом 1) элементы этих массивов получают нулевые значения. Прочие элементы заполняются на основе громоздких формул, являющихся следствием рекуррентных соотношений $\alpha_i = -\frac{\Delta x_i}{(2+\alpha_{i-1})\Delta x_{i-1} + 2\Delta x_i}$ и $\beta_i = \frac{\frac{\Delta y_i}{\Delta x_i} - \frac{\Delta y_{i-1}}{\Delta x_{i-1}} - \beta_{i-1}\Delta x_{i-1}}{(2+\alpha_{i-1})\Delta x_{i-1} + 2\Delta x_i}$ соответственно (здесь мы переобозначили индексы в соответствии с тем, как они используются в операторе цикла).

После того как элементы массивов alpha и beta вычислены, приступаем к вычислению коэффициентов, используемых в выражениях для сплайн-полиномов. Эти коэффициенты записываются в массив a. Заполняем этот массив в обратном порядке, начиная с последнего элемента. Этот элемент получает нулевое значение (команда a(n) = 0). Весь процесс по заполнению массива a представляет собой *обратную прогонку* и базируется на рекуррентном соотношении между коэффициентами $a_i = \alpha_i a_{i+1} + \beta_i$. После этого фактически все, что осталось вычислить — это индекс сплайн-полинома, значение которого и является значением интерполяционной функции. Значение индекса вычисляется на основе значения аргумента t и базируется на следующем алгоритме: переменной k присваивается значение n-1, после чего запускается оператор цикла, и в этом операторе цикла проверяется условие t<X(i) для каждого очередного значения индексной переменной i. Если условие выполнено, переменной k присваивается значение i-1, и командой Exit For завершается работа оператора цикла. Поскольку индексная переменная i принимает значения «по возрастающей», начиная со значения 1, то «выключение» оператора цикла происходит при первом выполнении условия t<X(i). А это означает, что аргумент t находится слева от узловой точки X(i), но не левее чем узловая точка X(i-1). Такой ситуации соответствует полином с индексом i-1.

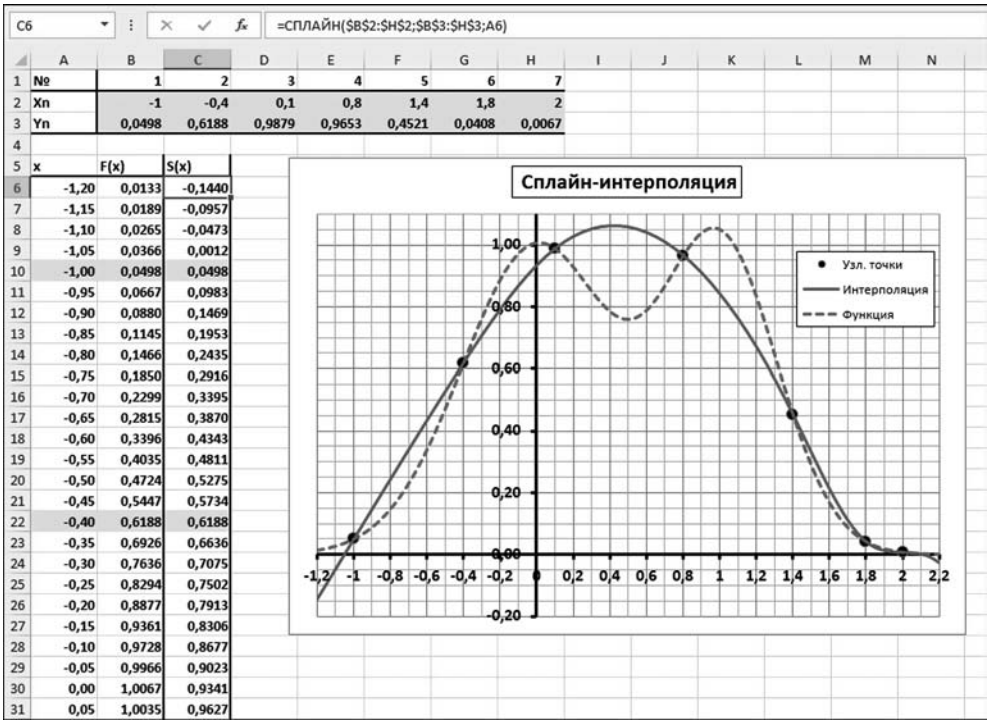
Когда индекс сплайн-полинома найден, значение полинома вычисляется в соответствии с формулой

$$S_k(x) = \frac{1}{\Delta x_k}(x\Delta y_k + y_k x_{k+1} - y_{k+1}x_k - (x - x_k)(x_{k+1} - x)\times$$
$$\times (a_{k+1}(x_{k+1} + x - 2x_{k+1}) + a_k(2x_{k+1} - x_k - x)).$$

Результат вычислений записывается в переменную S. Это и есть значение функции СПЛАЙН().

Пример использования функции СПЛАЙН() для вычисления значения интерполяционного сплайн-полинома приведен в документе на рисунке 13.10.

В качестве исходных данных (узловые точки и значения функции в этих точках) мы использовали тот же массив значений, что и в предыдущем случае — при вычислении сплайн-функции непосредственно в рабочем листе (рис. 13.9). Очевидно, что в данном случае вычислений в ячейках документа производится намного меньше. По большому счету, все, что необходимо сделать — это ввести в ячейки B2:H7 значения узловых точек и функции в этих узловых точках, в ячейки A6:A74 вводятся значения аргументов, для которых



	A	B	C	D	E	F	G	H
1	№	1	2	3	4	5	6	7
2	Xn	-1	-0,4	0,1	0,8	1,4	1,8	2
3	Yn	0,0498	0,6188	0,9879	0,9653	0,4521	0,0408	0,0067

	x	F(x)	S(x)
6	-1,20	0,0133	-0,1440
7	-1,15	0,0189	-0,0957
8	-1,10	0,0265	-0,0473
9	-1,05	0,0366	0,0012
10	-1,00	0,0498	0,0498
11	-0,95	0,0667	0,0983
12	-0,90	0,0880	0,1469
13	-0,85	0,1145	0,1953
14	-0,80	0,1466	0,2435
15	-0,75	0,1850	0,2916
16	-0,70	0,2299	0,3395
17	-0,65	0,2815	0,3870
18	-0,60	0,3396	0,4343
19	-0,55	0,4035	0,4811
20	-0,50	0,4724	0,5275
21	-0,45	0,5447	0,5734
22	-0,40	0,6188	0,6188
23	-0,35	0,6926	0,6636
24	-0,30	0,7636	0,7075
25	-0,25	0,8294	0,7502
26	-0,20	0,8877	0,7913
27	-0,15	0,9361	0,8306
28	-0,10	0,9728	0,8677
29	-0,05	0,9966	0,9023
30	0,00	1,0067	0,9341
31	0,05	1,0035	0,9627

Рис. 13.10
Вычисление интерполяционного сплайн-полинома с помощью функции пользователя

вычисляется сплайн-функция, в ячейках B6:B74 вычисляются значения исходной табулированной функции, а в ячейках C6:C74 — значения интерполяционной зависимости. Последний диапазон заполняется копированием из ячейки C6 формулы =СПЛАЙН($B$2:$H$2;$B$3:$H$3;A6). На основе полученных данных строится диаграмма с кривыми для табулированной функции, сплайн-функции и значениями табулированной функции в узловых точках (в виде отдельных точек). Как можно видеть из сравнения рисунков 13.10 и 13.9, результаты, полученные в обоих случаях, абсолютно аналогичны.

МЕТОДЫ АППРОКСИМАЦИИ

Ведь это же настоящая тайна!
Ты потом никогда себе не простишь!

Из к/ф «Гостья из будущего»

Задача аппроксимирования достаточно близка по своей постановке к задаче интерполирования. В обоих случаях по набору «экспериментальных» данных необходимо восстановить «наилучшее» функциональное выражение между зависимой и независимой переменными. Принципиальное отличие между этими задачами состоит в том, что если в задаче интерполирования

накладывается условие совпадения в узловых точках табличных значений и значений интерполяционной функции, то при аппроксимировании параметров для варьирования недостаточно для того, чтобы аппроксимирующая функциональная кривая в каждой узловой точке принимала предопределенное значение. Поэтому при аппроксимировании на аппроксимирующую функцию накладывается ограничение «наименьшего отклонения» от значений в узловых точках. При этом необходимо задать «метрику», через которую определяется «цена» отклонения от значения в узловой точке (в принципе, для каждой узловой точки она может быть своя). На практике, как правило, в качестве метода оптимизации используют *метод наименьших квадратов*.

Предположим, имеется некоторое функциональное выражение $F(x, a_1, a_2, \ldots, a_m)$, на основе которого мы хотим аппроксимировать зависимость, заданную значениями $y_1, y_2, \ldots, y_n$ в узловых точках $x_1, x_2, \ldots, x_n$. Задача, фактически, состоит в том, чтобы на основе некоторого критерия вычислить параметры оптимизации $a_1, a_2, \ldots, a_m$, которые входят в выражение для функции $F(x, a_1, a_2, \ldots, a_m)$.

На заметку

В данном случае важно то, что количество m варьируемых параметров в аппроксимирующей функции не превышает количество n узловых точек, т. е. $m \le n$. Причем, если имеет место равенство $m = n$, то речь идет о задаче интерполирования.

Принципиально важен вопрос о том, какой критерий используется для определения «оптимальности» функциональной зависимости. Как отмечалось выше, мы будем использовать метод наименьших квадратов. Для этого построим функцию, определяемую как сумма квадратов разности «экспериментальных» и «теоретических» значений в узловых точках: $\Phi(a_1, a_2, \ldots, a_m) = \sum_{k=1}^{n} (y_k - F(x_k, a_1, a_2, \ldots, a_m))^2$. Суть метода состоит в том, чтобы подобрать такие параметры $a_1, a_2, \ldots, a_m$, при которых значение выражения $\Phi(a_1, a_2, \ldots, a_m)$ будет минимальным.

На заметку

Поскольку функция $\Phi(a_1, a_2, \ldots, a_m)$ представляет собой сумму квадратов, то отрицательным это выражение быть не может. Теоретически минимально возможное значение — нулевое. Причем нулю функция $\Phi(a_1, a_2, \ldots, a_m)$ равняется лишь в том случае, когда каждое слагаемое равно нулю. А каждое слагаемое равно нулю, если «теоретическое» значение $F(x_k, a_1, a_2, \ldots, a_m)$ в узловой точке x_k равняется экспериментальному значению y_k в той же узловой точке. А это, в свою очередь, достигается при интерполировании.

Таким образом, нам необходимо минимизировать функцию $\Phi(a_1, a_2, \ldots, a_m)$. Необходимые условия локального экстремума имеют вид $\frac{\partial \Phi(a_1, a_2, \ldots, a_m)}{\partial a_p} = 0$ (индекс $p = 1, 2, \ldots, m$). В результате получаем систему из m алгебраических трансцендентных уравнений, решая которую, находим параметры $a_1, a_2, \ldots, a_m$.

На заметку

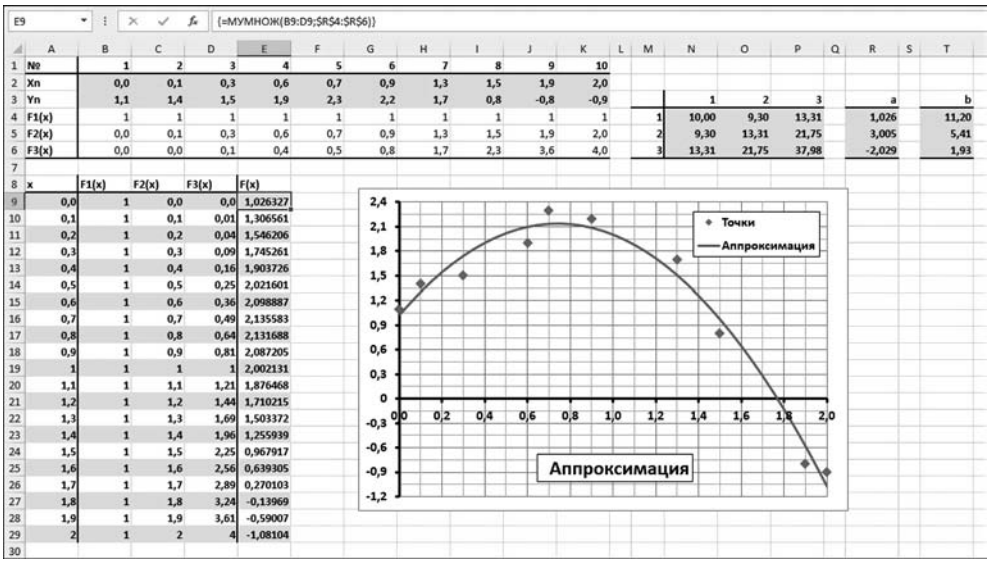
В результате решения системы алгебраических уравнений мы находим значения оптимизационных параметров, которые удовлетворяют необходимому условию наличия минимума. Но в общем случае это еще не означает, что минимум там действительно есть — просто он там может быть. Поэтому теоретически нужно еще и проверить, действительно ли вычисленные параметры минимизируют функцию $\Phi(a_1, a_2, \ldots, a_m)$. Вместе с тем, на практике обычно структура функции $\Phi(a_1, a_2, \ldots, a_m)$ такова, что если решение системы найдено, то это однозначно минимум.

Если учесть явное выражение для функции $\Phi(a_1, a_2, \ldots, a_m)$, уравнения системы для определения оптимизационных параметров могут быть записаны как $\sum_{k=1}^{n}(y_k - F(x_k, a_1, a_2, \ldots, a_m))\frac{\partial F(x_k, a_1, a_2, \ldots, a_m)}{\partial a_p} = 0$ (индекс $p = 1, 2, \ldots, m$). Для решения такой системы могут, кроме прочего, использоваться методы, описанные в предыдущей главе. Тем не менее нередко аппроксимирующая функция представима в виде линейной комбинации базисных функций, и оптимизация выполняется по коэффициентам этой линейной комбинации — имеется в виду, что функция $F(x, a_1, a_2, \ldots, a_m) = a_1 f_1(x) + a_2 f_2(x) + \ldots + a_m f_m(x)$, где базисные функции $f_1(x)$, $f_2(x)$, $\ldots$, $f_m(x)$ известны и определены однозначно.

На заметку

Нередко задачу с нелинейной по параметрам оптимизации аппроксимирующей функцией удается свести к линейному случаю. Например, если по точкам (x_k, y_k) (индекс $k = 1, 2, \ldots, n$) мы пытаемся вычислить оптимальные параметры для функции $F(x, A, b) = A\exp(bx)$, то можем рассмотреть набор точек $(x_k, \ln(y_k))$ и аппроксимировать эту зависимость линейной функцией $\ln(F(x, A, b)) = \ln A + bx$.

В этом случае мы можем значительно продвинуться в вопросе выбора оптимальных параметров $a_1, a_2, \ldots, a_m$. В частности, несложно понять, что $\frac{\partial F(x, a_1, a_2, \ldots, a_m)}{\partial a_p} = f_p(x)$. Тогда система для определения оптимизационных параметров записывается как $\sum_{k=1}^{n}\left(y_k - \sum_{s=1}^{m} a_s f_s(x_k)\right) f_p(x_k) = 0$ для индексов $p = 1, 2, \ldots, m$. После несложных преобразований, эти уравнения можем записать в виде $\sum_{s=1}^{m}\left(\sum_{k=1}^{n} f_p(x_k) f_s(x_k)\right) a_s = \sum_{k=1}^{n} f_p(x_k) y_k$. Относительно параметров a_s (индекс $s = 1, 2, \ldots, m$) — это линейная неоднородная система алгебраических уравнений, которая достаточно легко решается, в том числе и в Excel. Ее удобно записать в матричном виде. Для этого введем ряд обозначений: вектор значений табулированной функции в узловых точках $\vec{y} = \left[y_1, y_2, \ldots, y_n\right]^T$, а также матрица $\hat{\mathbf{F}}$ размерами $m \times n$ (m строк и n столбцов) значений базисных функций в узловых точках с элементами $F_{ij} = f_i(x_j)$ (индексы $i = 1, 2, \ldots, m$ и $j = 1, 2, \ldots, n$). Тогда систему уравнений для определения оптимизационных параметров можно в матричном виде записать как $\hat{\mathbf{F}} \cdot \hat{\mathbf{F}}^T \vec{a} = \vec{\mathbf{F}} \cdot \vec{y}$, где через



	A	B	C	D	E	F	G	H	I	J	K
1	№	1	2	3	4	5	6	7	8	9	10
2	Xn	0,0	0,1	0,3	0,6	0,7	0,9	1,3	1,5	1,9	2,0
3	Yn	1,1	1,4	1,5	1,9	2,3	2,2	1,7	0,8	-0,8	-0,9
4	F1(x)	1	1	1	1	1	1	1	1	1	1
5	F2(x)	0,0	0,1	0,3	0,6	0,7	0,9	1,3	1,5	1,9	2,0
6	F3(x)	0,0	0,0	0,1	0,4	0,5	0,8	1,7	2,3	3,6	4,0

	M	N	O	P	Q	R	S	T
		1	2	3		a		b
	1	10,00	9,30	13,31		1,026		11,20
	2	9,30	13,31	21,75		3,005		5,41
	3	13,31	21,75	37,98		-2,029		1,93

	A	B	C	D	E
8	x	F1(x)	F2(x)	F3(x)	F(x)
9	0,0	1	0,0	0,0	1,026327
10	0,1	1	0,1	0,01	1,306561
11	0,2	1	0,2	0,04	1,546206
12	0,3	1	0,3	0,09	1,745261
13	0,4	1	0,4	0,16	1,903726
14	0,5	1	0,5	0,25	2,021601
15	0,6	1	0,6	0,36	2,098887
16	0,7	1	0,7	0,49	2,135583
17	0,8	1	0,8	0,64	2,131688
18	0,9	1	0,9	0,81	2,087205
19	1	1	1	1	2,002131
20	1,1	1	1,1	1,21	1,876468
21	1,2	1	1,2	1,44	1,710215
22	1,3	1	1,3	1,69	1,503372
23	1,4	1	1,4	1,96	1,255939
24	1,5	1	1,5	2,25	0,967917
25	1,6	1	1,6	2,56	0,639305
26	1,7	1	1,7	2,89	0,270103
27	1,8	1	1,8	3,24	-0,13969
28	1,9	1	1,9	3,61	-0,59007
29	2	1	2	4	-1,08104
30					

Рис. 13.11
Построение аппроксимирующей зависимости

Таблица 13.1

Данные для выполнения аппроксимации

№	1	2	3	4	5	6	7	8	9	10
x	0,0	0,1	0,3	0,6	0,7	0,9	1,3	1,5	1,9	2,0
y	1,1	1,4	1,5	1,9	2,3	2,2	1,7	0,8	-0,8	-0,9

$\vec{a} = [a_1, a_2, \ldots, a_m]^T$ обозначен вектор, составленный из параметров оптимизации. Решение этой системы можем записать в виде $\vec{a} = (\hat{\mathbf{F}} \cdot \hat{\mathbf{F}}^T)^{-1} \vec{\mathbf{F}} \cdot \vec{y}$, где индекс -1 обозначает обратную матрицу. Именно эту схему реализуем в рабочем документе Excel. На рисунке 13.11 представлен документ, в котором с использованием «экспериментальных данных» строится аппроксимирующая зависимость на основе полиномиального выражения второй степени (т. е. аппроксимирующая функция имеет вид $F(x) = a_1 + a_2 x + a_3 x^2$, и задача состоит в определении по набору табличных значений параметров a_1, a_2 и a_3).

Документ достаточно простой. В ячейках B2:K3 содержатся данные, которые дают представление о значениях некоторой функции (ячейки B3:K3) в узловых точках (ячейки B2:K2). Это те исходные данные, на основе которых строится аппроксимирующая функция. Поскольку все значения числовые (т. е. каждая ячейка заполняется отдельно, без использования автоматического заполнения), для удобства восприятия материала они представлены в таблице 13.1.

Кроме непосредственно исходных данных, нам понадобится матрица, составленная из значений базисных функций в узловых точках. Учитывая

явный вид выражения для аппроксимирующей функции $F(x) = a_1 + a_2x + a_3x^2$, несложно догадаться, что базисными функциями в данном случае являются $f_1(x) = 1$, $f_2(x) = x$ и $f_2(x) = x^2$. Значения именно этих функций необходимо вычислить в узловых точках (диапазон ячеек B2:K2). Для этого в ячейку B4 вводим формулу =1, в ячейку B5 вводим формулу =B2, а в ячейку B6 вводим формулу =B2^2 — и затем копируем формулы построчно, заполняя диапазон ячеек B4:K6. В результате эти ячейки содержат значения $f_i(x_j)$, т. е. это та матрица, на основе которой вычисляются параметры аппроксимирующей функции. Кроме этого, в явном виде мы выполним некоторые промежуточные расчеты:

- выделяем диапазон ячеек N4:P6 и вводим туда формулу массива =МУМНОЖ (B4:K6;ТРАНСП(B4:K6)). Результатом вычислений является матрица $\hat{\mathbf{F}} \cdot \hat{\mathbf{F}}^T$ — матрица коэффициентов линейной системы уравнений;
- выделяем диапазон ячеек T4:T6 и вводим в этот диапазон формулу массива =МУМНОЖ(B4:K6;ТРАНСП(B3:K3)). Результатом является произведение $\hat{\mathbf{F}} \cdot \vec{y}$, представляющее собой вектор правой части линейной системы уравнений.

Для вычисления параметров аппроксимирующей функции в ячейки диапазона R4:R6 вводим формулу массива =МУМНОЖ(МОБР(N4:P6);T4:T6). Этой формулой реализуется выражение $\vec{a} = (\hat{\mathbf{F}} \cdot \hat{\mathbf{F}}^T)^{-1} \cdot \vec{\mathbf{F}} \cdot \vec{y}$. Получаем вектор-столбик с коэффициентами $\vec{a}$. Теперь, чтобы вычислить значение аппроксимирующей функции $F(x) = a_1 f_1(x) + a_2 f_2(x) + a_3 f_3(x) \equiv a_1 + a_2x + a_3x^2$ в точке x, достаточно вычислить скалярное произведение векторов $\vec{f}(x) = [f_1(x),\ f_2(x),\ f_3(x)] \equiv [1,\ x,\ x^2]$ и $\vec{a} = \left[a_1, a_2, a_3\right]^T$, т. е. аппроксимирующую функцию можно представить в виде $F(x) = \vec{f}(x)\vec{a}$. Именно такое соотношение мы используем для вычисления аппроксимирующей функции.

Для построения аппроксимирующей кривой в ячейки диапазона A9:A29 вводятся значения аргумента функции: в ячейку A9 вводится формула =B2, а прочие ячейки диапазона заполняются копированием из ячейки A10 формулы =A9+($K$2-$B$2)/20.

Ячейки B9:D29 заполняются значениями базисных функций (по столбцам — значения одной и той же функции для разных аргументов, а в строке — значения разных функций для одного и того же аргумента), а в ячейках диапазона E9:E29 вычисляются значения аппроксимирующей функции:

- в ячейку B9 вводится формула =1;
- в ячейку C9 вводим формулу =A9;
- в ячейку D9 вводится формула =A9^2;
- в ячейку E9 вводится формула массива =МУМНОЖ(B9:D9;$R$4:$R$6);
- формулы из этих ячеек копируются в ячейки снизу.

Результат вычислений лучше всего анализировать с помощью графика. На рисунке 13.11 представлена, кроме всего прочего, диаграмма с кривой для аппроксимирующей функции и исходными «экспериментальными» точками. Кривая строится на основе значений ячеек E9:E29 (функция) и A9:A29 (аргумент). Точки определяются на основе значений в ячейках B2:K3.

В данном случае четко прослеживается разница между аппроксимирующей и интерполирующей кривыми: аппроксимирующая кривая проходит «близко» от базисных точек, но при этом может не проходить ни через одну из них. В этом смысле аппроксимирующая кривая в общем случае не дает совпадения со значением табулированной функции в узловых точках.

Главная ценность моделей аппроксимации обычно связана не с предсказанием поведения исследуемой функции между узловыми точками, но, скорее, с возможностью вычислить параметры аппроксимирующей зависимости. Обычно такие параметры имеют конкретный физический (или иной другой — в зависимости от принадлежности модели к той или иной области естествознания) смысл, и именно их вычисление представляет наибольший интерес. Кроме того, весьма важно априори предложить адекватное выражение для аппроксимирующей функции, а для этого необходимо иметь хотя бы общее представление о функциональной зависимости, которая реализована в виде табличных значений и для которой выполняется аппроксимация.

ГЛАВА 14 ДИФФЕРЕНЦИРОВАНИЕ И ИНТЕГРИРОВАНИЕ

Если мы допустим беспорядок в документации,
потомки нам этого не простят.

Из к/ф «Гостья из будущего»

Вычисление производной — аналитическая процедура, которая по определенному правилу одной функции ставит в соответствие другую функцию.

На заметку

Напомним, что формальное определение производной $f'(x) \equiv \frac{df(x)}{dx}$ от функции $f(x)$ формулируется как предел отношения приращения функции к приращению аргумента при стремлении последнего к нулю, т. е. как $f'(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$.

Задача вычисления производной в числовом виде обычно возникает в двух случаях:

- если исходная функциональная зависимость задана в табличном виде (табулированная функция);
- если аналитическое выражение для функции известно, но слишком сложно и громоздко для вычисления производной в аналитическом виде.

Нас в первую очередь будет интересовать первый случай, т. е. ситуация, когда имеется набор узловых точек, и заданы значения некоторой функции в этих узловых точках. Задача может формулироваться как вычисление производной (первого или более высоких порядков) в узловых точках или на всей области определения функциональной зависимости. Вместе с тем, здесь возникает одна весьма важная проблема. Связана она с тем, что производная от функции в точке (при заданном значении аргумента) — локальная характеристика, поскольку значение производной определяется поведением дифференцируемой функции в окрестности той точки, в которой вычисляется производная. Но если функция задана в виде таблицы, через значения в узловых точках, то вопрос о локальном поведении обсуждать крайне проблематично, поскольку информация практически отсутствует — разве только относительно мал интервал между соседними узловыми точками. В любом случае общепринятой является точка зрения, что вычисление в числовом виде производной — задача «неблагодарная», поскольку в силу объективных причин не всегда имеется возможность обеспечить надежность и точность результатов.

На заметку

Поясним сказанное на простом примере. Предположим, что нам нужно выполнить интерполирование для некоторой функции $y(x)$. Если по набору значений этой функции в узловых точках нам удалось построить интерполяционный многочлен $q(x)$, то «погрешность», связанная с заменой исходной функции на ее интерполяционный многочлен, определяется как разность функции и многочлена, т. е. $r(x) = y(x) - q(x)$. Если интерполяция проведена удачно, то величина $|r(x)|$ должна быть малой (в пределах области интерполирования). В силу используемого подхода, для k-й производной $y^{(k)}(x) \equiv \frac{d^k y(x)}{dx^k}$ интерполяционный многочлен — это $q^{(k)}(x)$. Таким образом, «ошибка» интерполирования производной определяется соотношением $y^{(k)}(x) - q^{(k)}(x) = r^{(k)}(x)$. Проблема в том, что даже при малых значениях $|r(x)|$ малость величины $|r^{(k)}(x)|$ не гарантирована. Например, функция $f(x) = \sqrt{x(1-x)}$ ограничена на интервале от 0 до 1 и по абсолютной величине не превышает значение $1/2$, т. е. $0 \le |f(x)| \le 1/2$. Вместе с тем, производная $f'(x) = \frac{1/2 - x}{\sqrt{x(1-x)}}$ на границах интервала не ограничена: $|f'(x)| \to \infty$ при $x \to 0$ или $x \to 1$.

Тем не менее в ряде случаев удается добиться приемлемого результата. Во всяком случае, задача о вычислении производной для заданной в табличном виде функции имеет право на существование. Здесь мы рассмотрим некоторые вычислительные аспекты такой задачи.

Интегрирование представляет собой задачу, обратную по отношению к задаче дифференцирования. Обычно в контексте проблемы интегрирования рассматривают задачу о вычислении определенного или неопределенного интеграла.

На заметку

Неопределенным интегралом от функции $f(x)$ называется такая функция $F(x) = \int f(x)dx$, производная от которой равняется интегрируемой функции, т. е. имеет место соотношение $\frac{dF(x)}{dx} = f(x)$.

С определенным интегралом ситуация более сложная. Здесь мы будем исходить из следующего упрощенного определения определенного интеграла: интегралом $\int_a^b f(x)dx$ от функции $f(x)$ на интервале значений от a до b является площадь под кривой, определяемой функцией $f(x)$. Между определенным и неопределенным интегралами существует связь: если $F(x) = \int f(x)dx$, то $\int_a^b f(x)dx = F(b) - F(a)$.

Более конкретно, в рамках курса *вычислительной математики* решается задача о вычислении определенного интеграла. В этой главе мы расширим познания в области применения приложения Excel для вычисления определенных интегралов.

На заметку

Ранее в книге вопрос о вычислении интегралов уже кратко рассматривался. Здесь читателю предлагается более общий подход к проблеме.

ВЫЧИСЛЕНИЕ ПРОИЗВОДНОЙ В УЗЛОВЫХ ТОЧКАХ

Это великая победа
дедуктивного метода.

Из к/ф «Гостья из будущего»

Задача о вычислении производной в числовом виде может формулироваться по-разному, однако обычно в основе лежит процедура интерполирования функциональной зависимости, заданной в виде таблицы значений функции в узловых точках, с последующим вычислением производных от интерполяционной функции. В этом отношении нет ничего принципиально нового по сравнению с теми приемами, которые мы рассматривали в предыдущей главе. Однако иногда задача сформулирована так, что вычислять (в явном виде) интерполяционную функцию не обязательно. В этом разделе мы рассмотрим задачу о вычислении значений производной в узловых точках для функции, заданной в виде таблицы.

В частности, допустим, имеется набор узловых точек $x_0, x_1, \ldots, x_n$, и заданы значения некоторой функции в узловых точках $y_0, y_1, \ldots, y_n$. Задача состоит в том, чтобы определить значения производных порядка k (заданное фиксированное значение) в узловых точках $x_0, x_1, \ldots, x_n$.

На заметку

Один из вариантов решения такой задачи — создание интерполяционного полинома. Затем на основании выражения для интерполяционного полинома вычисляются производные — вплоть до требуемого порядка. Причем производные можно вычислять не только в узловых точках. Поскольку производная от полинома вычисляется достаточно просто, а методы построения интерполяционного полинома мы уже рассматривали, то можно полагать, что в такой постановке задача сводится к общеизвестным процедурам. Вместе с тем, если речь идет о вычислении производной именно в узловых точках, мы, используя Excel, можем значительно упростить процесс вычислений.

Для решения этой задачи без построения в явном виде интерполяционного полинома будем исходить из следующего.

Рассмотрим интерполяционный полином Лагранжа, который может быть представлен в данном случае в виде $L_n(x) = y_0\varphi_0(x) + y_1\varphi_1(x) + \ldots + y_n\varphi_n(x)$, где функции $\varphi_j = \dfrac{(x-x_0)(x-x_1)\ldots(x-x_{j-1})(x-x_{j+1})\ldots(x-x_n)}{(x_j-x_0)(x_j-x_1)\ldots(x_j-x_{j-1})(x_j-x_{j+1})\ldots(x_j-x_n)}$ являются полиномами степени n каждая (индекс $j = 0, 1, \ldots, n$) и, что самое главное, зависят только от размещения узловых точек, но не зависят от значения интерполируемой функции в этих точках. Это означает, что, если мы построили интерполяционный полином Лагранжа по одному набору значений некоторой функции, а затем задача ставится вычислить интерполяционный полином Лагранжа для другой функции (но с табличными значениями, заданными в тех же самых узловых точках), то функции $\varphi_j(x)$ остаются теми же — изменяются только коэффициенты линейной комбинации, с которыми данные функции входят в выражение для интерполяционного

полинома. Если речь идет о производной k-го порядка, то очевидно, что $L_n^{(k)}(x) = y_0\varphi_0^{(k)}(x) + y_1\varphi_1^{(k)}(x) + \ldots + y_n\varphi_n^{(k)}(x)$, т. е. такая производная выражается через линейную комбинацию k-х производных $\varphi_j^{(k)}(x)$ от функций $\varphi_j(x)$, причем коэффициентами в этих линейных комбинациях являются значения табулированной функции в узловых точках.

Если нас интересует производная k-го порядка в узловой точке x_i, то соответствующая оценка будет иметь вид $y^{(k)}(x_i) = y_0\varphi_0^{(k)}(x_i) + y_1\varphi_1^{(k)}(x_i) + \ldots + y_n\varphi_n^{(k)}(x_i)$. Примечательным является то обстоятельство, что, если мы возьмем какую-то другую функцию (например, $z(x)$) и будем по значениям этой функции (например, z_0, z_1, …, z_n) в тех же самых узловых точках (т. е. x_0, x_1, …, x_n) оценивать производную порядка k в точке x_i, то справедливым будет соотношение $z^{(k)}(x_i) = z_0\varphi_0^{(k)}(x_i) + z_1\varphi_1^{(k)}(x_i) + \ldots + z_n\varphi_n^{(k)}(x_i)$. Другими словами, какова бы ни была функция, ее k-я производная в точке x_i представляется в виде линейной комбинации значений этой функции в узловых точках и элементов $\varphi_j^{(k)}(x_i)$. Параметры $\varphi_j^{(k)}(x_i)$ зависят и определяются только набором узловых точек x_0, x_1, …, x_n. Это важное обстоятельство наводит нас на простую мысль, что для вычисления параметров $\varphi_j^{(k)}(x_i)$ совсем необязательно брать интерполируемую функцию — можно воспользоваться какой-то другой.

На заметку

Есть еще одно свойство интерполяционных выражений, на которое стоит обратить внимание и которое может быть полезным в дальнейшем: если речь идет о сумме двух функций, то интерполяционный полином суммы табулированных функций равен сумме интерполяционных полиномов. То есть, если $y(x) = u(x) + w(x)$ и в узловых точках x_0, x_1, …, x_n значения функции $y_i = u_i + w_i$ ($i = 0, 1, \ldots, n$), то интерполяционный полином $P_y(x)$ для функции $y(x)$ есть сумма $P_y(x) = P_u(x) + P_w(x)$, где $P_u(x)$ есть интерполяционный полином для функции $u(x)$, а $P_w(x)$ есть интерполяционный полином для функции $w(x)$.

Самыми удобными функциями для полиномиального интерполирования являются, как несложно догадаться, полиномы. В частности, рассмотрим последовательность функций $\xi_m(x) = x^m$ (индекс $m = 0, 1, \ldots, n$), т. е. рассмотрим функции $1, x, \ldots, x^n$. Систему коэффициентов $s_{k,ij} \equiv \varphi_j^{(k)}(x_i)$ будем определять на основе значений $\xi_m^{(k)}(x_i)$. Несложно вычислить, что при $k > m$ имеет место соотношение $\xi_m^{(k)}(x_i) = 0$. При $k \le m$ имеет место соотношение $\xi_m^{(k)}(x_i) = P_m^k x_i^{m-k}$, где коэффициенты $P_m^k = \dfrac{m!}{(m-k)!}$. При этом в узловых точках $\xi_m(x_i) = x_i^m$ (для всех индексов i, $m = 0, 1, \ldots, n$). Таким образом, относительно параметров $s_{k,ij} \equiv \varphi_j^{(k)}(x_i)$ можем записать систему уравнений следующего типа: $\sum_{j=0}^{n} s_{k,ij} x_j^m = 0$ при $0 \le m < k$ и $\sum_{j=0}^{n} s_{k,ij} x_j^m = P_m^k x_i^{m-k}$ при $k \le m \le n$ для всех индексов $i = 0, 1, \ldots, n$. Таким образом, относительно параметров $s_{k,\,ij}$ (при фиксированных индексах k, i и значениях индекса $j = 0, 1, \ldots, n$) имеем систему из k уравнений вида $\sum_{j=0}^{n} s_{k,ij} x_j^m = 0$ (при $m = 0, 1, \ldots, k-1$) с нулевой правой частью и $n - k + 1$ уравнений вида $\sum_{j=0}^{n} s_{k,ij} x_j^m = P_m^k x_i^{m-k}$ (при $m = k, k+1, \ldots, n$) с ненулевой правой частью. Таким образом, всего $n + 1$

уравнений для определения $n+1$ параметра $s_{k,\,ij}$ (индексы k, i заданы, а индекс $j = 0, 1, \ldots, n$). Чтобы ее решить, необходимо определить матрицу коэффициентов уравнения и вектор правых частей.

На заметку

Определить матрицу коэффициентов и вектор правых частей легче, если записать систему уравнений в явном виде. Опять же, для удобства восприятия информации обозначим $c_m \equiv s_{k,\,im}$. Тогда уравнения имеют такой вид:

$$
\begin{aligned}
&c_0 + c_1 + \ldots + c_n = 0,\\
&c_0x_0 + c_1x_1 + \ldots + c_nx_n = 0,\\
&c_0x_0^2 + c_1x_1^2 + \ldots + c_nx_n^2 = 0,\\
&\ldots,\\
&c_0x_0^{k-1} + c_1x_1^{k-1} + \ldots + c_nx_n^{k-1} = 0,\\
&c_0x_0^{k} + c_1x_1^{k} + \ldots + c_nx_n^{k} = k!,\\
&c_0x_0^{k+1} + c_1x_1^{k+1} + \ldots + c_nx_n^{k+1} = (k+1)!x_i,\\
&c_0x_0^{k+2} + c_1x_1^{k+2} + \ldots + c_nx_n^{k+2} = \frac{(k+2)!}{2!}x_i^2,\\
&\ldots,\\
&c_0x_0^{k+p} + c_1x_1^{k+p} + \ldots + c_nx_n^{k+p} = \frac{(k+p)!}{p!}x_i^p,\\
&\ldots,\\
&c_0x_0^{n} + c_1x_1^{n} + \ldots + c_nx_n^{n} = \frac{n!}{(n-k)!}x_i^{n-k}.
\end{aligned}
$$

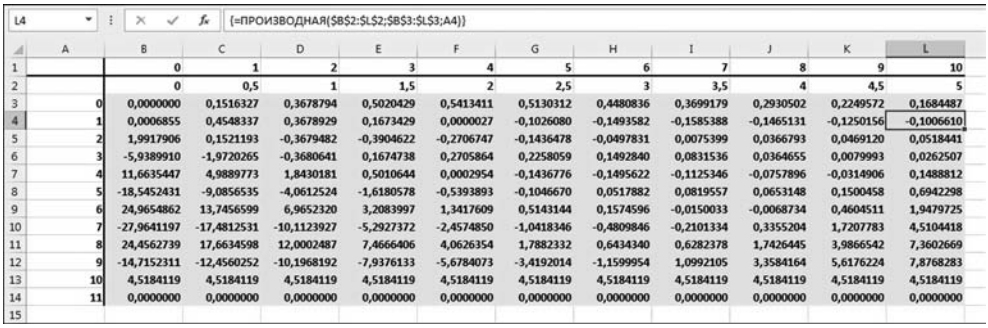
Данная система решается относительно коэффициентов c_m ($m = 0, 1, \ldots, n$), при этом порядок производной и индекс точки, в которой она вычисляется, фиксированы (другими словами, для каждой узловой точки и каждой производной коэффициенты свои и определяются отдельно).

Таким образом, матрица коэффициентов системы — это транспонированная матрица Вандермонда. Элементы a_{mp} матрицы $\hat{\mathbf{A}}$ коэффициентов системы могут быть выражены формулой $a_{mp} = x_p^m$ (индексы $m, p = 0, 1, \ldots, n$). Вектор правой части $\vec{B}$ также достаточно прост: его элементы b_m принимают значения $b_m = 0$ при $0 \le m < k$ и $b_m = \frac{m!}{(m-k)!}x_i^{m-k}$ при $k \le m \le n$. Для решения поставленной задачи достаточно вычислить вектор $\vec{C}$ с элементами c_m по формуле $\vec{C} = \hat{\mathbf{A}}^{-1}\vec{B}$, а затем по формуле $y^{(k)}(x_i) = \sum_{m=0}^{n} c_m y_m$ вычисляется значение производной в точке x_i.

Решая стандартными методами систему линейных уравнений, находим значение производных (нужного порядка) в узловых точках.

На рисунке 14.1 представлен документ, в котором производится вычисление производных нескольких порядков на основе значения функции в узловых точках.

Причем для вычисления производной произвольного порядка мы создали функцию пользователя с названием ПРОИЗВОДНАЯ. У функции три аргумента: диапазон ячеек со значениями узловых точек, диапазон ячеек со значениями табулированной функции в узловых точках, а также значение,



L4 {=ПРОИЗВОДНАЯ($B$2:$L$2;$B$3:$L$3;A4)}

	A	B	C	D	E	F	G	H	I	J	K	L
1		0	1	2	3	4	5	6	7	8	9	10
2		0	0,5	1	1,5	2	2,5	3	3,5	4	4,5	5
3	0	0,0000000	0,1516327	0,3678794	0,5020429	0,5413411	0,5130312	0,4480836	0,3699179	0,2930502	0,2249572	0,1684487
4	1	0,0006855	0,4548337	0,3678929	0,1673429	0,0000027	-0,1026080	-0,1493582	-0,1585388	-0,1465131	-0,1250156	-0,1006610
5	2	1,9917906	0,1521193	-0,3679482	-0,3904622	-0,2706747	-0,1436478	-0,0497831	0,0075399	0,0366793	0,0469120	0,0518441
6	3	-5,9389910	-1,9720265	-0,3680641	0,1674738	0,2705864	0,2258059	0,1492840	0,0831536	0,0364655	0,0079993	0,0262507
7	4	11,6635447	4,9889773	1,8430181	0,5010644	0,0002954	-0,1436776	-0,1495622	-0,1125346	-0,0757896	-0,0314906	0,1488812
8	5	-18,5452431	-9,0856535	-4,0612524	-1,6180578	-0,5393893	-0,1046670	0,0517882	0,0819557	0,0653148	0,1500458	0,6942298
9	6	24,9654862	13,7456599	6,9652320	3,2083997	1,3417609	0,5143144	0,1574596	-0,0150033	-0,0068734	0,4604511	1,9479725
10	7	-27,9641197	-17,4812531	-10,1123927	-5,2927372	-2,4574850	-1,0418346	-0,4809846	-0,2101334	0,3355204	1,7207783	4,5104418
11	8	24,4562739	17,6634598	12,0002487	7,4666406	4,0626354	1,7882332	0,6434340	0,6282378	1,7426445	3,9866542	7,3602669
12	9	-14,7152311	-12,4560252	-10,1968192	-7,9376133	-5,6784073	-3,4192014	-1,1599954	1,0992105	3,3584164	5,6176224	7,8768283
13	10	4,5184119	4,5184119	4,5184119	4,5184119	4,5184119	4,5184119	4,5184119	4,5184119	4,5184119	4,5184119	4,5184119
14	11	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000	0,0000000
15												

Рис. 14.1
Вычисление производных в узловых точках на основе интерполяционного полинома с помощью функции пользователя

определяющее порядок производной. В качестве результата функция возвращает массив ячеек со значениями производной в узловых точках. Другими словами, функция вычисляет значения производной сразу во всех узловых точках.

Структура документа такая, что в ячейках B1:L1 содержатся индексы узловых точек, а в ячейках A3:A14 указан порядок производной (производная нулевого порядка соответствует самой функции). В ячейках B2:L2 отображаются значения узловых точек (от 0 до 5 с интервалом между соседними узловыми точками 0.5). Табличные значения функции вычисляются в ячейках B3:L3: в ячейку B3 вводится формула =B2^2*EXP(-B2) и затем копируется во все остальные ячейки этого диапазона. Таким образом, мы имеем дело с функцией $f(x) = x^2\exp(-x)$.

Для вычисления первой производной выделяем ячейки B4:L4 и вводим в эти ячейки формулу массива =ПРОИЗВОДНАЯ($B$2:$L$2;$B$3:$L$3;A4) (вводится комбинацией клавиш <Ctrl>+<Shift>+<Enter>). Формула представляет собой вызов функции ПРОИЗВОДНАЯ() с такими аргументами: абсолютная ссылка $B$2:$L$2 на диапазон ячеек со значениями узловых точек, абсолютная ссылка $B$3:$L$3 на диапазон ячеек со значениями табулированной функции в узловых точках, а также относительная ссылка A4 на ячейку со значением порядка производной. В результате ввода формулы значения для первой производной вычисляются во всех узловых точках (диапазон ячеек B4:L4). Для вычисления производных других порядков выделяем диапазон ячеек B4:L4 и, перетаскивая маркер заполнения, копируем формулу массива во все прочие ячейки диапазона B4:L12. Последняя строка этого диапазона (ячейки B12:L12) содержит значения производной 11-го порядка и, как несложно заметить, во всех узловых точках эта производная равняется нулю. Так и должно быть.

На заметку

Более того, 11-я производная равняется нулю не только в узловых точках — она тождественно равна нулю. Объяснение очень простое: достаточно вспомнить, что значения производных вычисляются на основе интерполяционного полинома для исходной табулированной функции. Если функция интерполируется по значениям в $n + 1$ узловой точке, то полином имеет степень n. Производные от полинома — тоже полиномы. Но каждая производная уменьшает

степень полинома на единицу. Поэтому производная порядка n от полинома степени n — это полином нулевой степени, т. е. число. А $(n+1)$-я производная от полинома степени n — это производная от константы, т. е. тождественно равна нулю. Поэтому для полинома степени n все производные, начиная с $(n+1)$-го порядка тождественно равны нулю.

Что касается «точности» вычисления производной для данного примера, то желающие могут самостоятельно проделать соответствующие вычисления и сравнить полученные нами числовые значения для производных разных порядков в узловых точках и значения производных в этих же точках от исходной функции $f(x) = x^2\exp(-x)$. Здесь лишь отметим, что чем ниже порядок производной, тем больше совпадение между значениями производной, вычисленными на основе интерполяционного выражения, и аналитического выражения для производной от интерполируемой функции. Также следует учесть, что функция $f(x) = x^2\exp(-x)$ может быть представлена в виде ряда $f(x) = x^2 - x^3 + \frac{x^4}{2!} - \frac{x^5}{3!} + \ldots + (-1)^n \frac{x^{n+2}}{n!} + \ldots$ Поэтому интерполяционный полином достаточно неплохо описывает эту функциональную зависимость. Однако по мере вычисления производной, с одной стороны, степень полинома понижается, а производная от функции $f(x)$ — это все равно бесконечный ряд. В какой-то момент слагаемых в продифференцированном интерполяционном выражении становится недостаточно для качественного приближения производной от функции $f(x)$.

Далее рассмотрим программный код функции ПРОИЗВОДНАЯ(), которая использовалась нами для вычисления производных в числовом виде в узловых точках. Обратимся к листингу 14.1.

Листинг 14.1. Функция для вычисления значения производной от интерполяционного полинома в узловых точках

```
Function ПРОИЗВОДНАЯ(Px As Range, Py As Range, d As Variant)
 ‘ Переменная для определения порядка производной
 Dim k As Integer
 ‘ Порядок производной
  k = d
 ‘ Переменная для степени интерполяционного полинома
 Dim n As Integer
 ‘ Степень интерполяционного полинома
 n = Px.Count - 1
 ‘ Массив для записи узловых точек
 ReDim x(0 To n) As Double
 ‘ Массив для записи значений функции в узловых точках
 ReDim y(0 To n) As Double
 ‘ Массив для записи значений векторов правых частей системы
 ReDim b(0 To n, 0 To n) As Double
 ‘ Массив для записи матрицы коэффициентов
 ReDim A(0 To n, 0 To n) As Double
 ‘ Определение матрицы коэффициентов
 A = Fa(Px)
 ‘ Целочисленные индексные переменные для операторов цикла
```

```
    Dim i As Integer, j As Integer
    ' Внешний оператор цикла
    For i = 0 To n
      ' Узловая точка
      x(i) = Px.Cells(i + 1).Value
      ' Значение функции в узловой точке
      y(i) = Py.Cells(i + 1).Value
    Next i
    ' Внешний оператор цикла
    For i = 0 To n
      ' Внутренний оператор цикла
      For j = 0 To n
        ' Элемент матрицы правых частей системы уравнений
        b(i, j) = Fb(k, i, x(j))
      Next j
    Next i
    ' Вызов функций рабочего листа
    With Application.WorksheetFunction
      ' Значение функции
      ПРОИЗВОДНАЯ = .MMult(y, .MMult(.MInverse(A), b))
    End With
  End Function
```

Назначение аргументов этой функции мы уже обсуждали: диапазон узловых точек Px, диапазон значений функции в узловых точках Py и порядок производной d. В теле функции объявляется целочисленная переменная k, в которую командой `k = d` записывается порядок вычисляемой производной. В целочисленную переменную n командой `n = Px.Count-1` записывается значение для степени интерполяционного полинома (на единицу меньше количества узловых точек). Эта переменная используется для определения размеров нескольких массивов. Так, массив для записи узловых точек x индексируется значениями от 0 до n (т. е. всего в массиве n+1 элемент). Такие же параметры имеет массив y, предназначенный для записи значений функции в узловых точках. Еще два массива используются для реализации квадратных матриц:

- массив b, составленный из векторов правых частей систем уравнений, решаемых для вычисления производных в узловых точках;
- массив A с коэффициентами матрицы коэффициентов линейной системы уравнений.

Оба массива состоят из n+1 строк и n+1 столбцов, а индексация по обеим размерностям массивов начинается с нуля.

На заметку

Нелишним будет напомнить, что мы решаем систему из уравнений вида $\sum_{j=0}^{n} c_j x_j^m = 0$ при $m = 0, 1, \ldots, k-1$ и $\sum_{j=0}^{n} c_j x_j^{k+p} = \frac{m!}{(m-k)!} x_i^{m-k}$ при $m = k, k+1, \ldots, n$. Решая эту систему, получаем значение производной порядка k в одной узло-

вой точке — точке x_i. Следовательно, для каждой узловой точки решается своя система уравнений. Матрица коэффициентов при этом не меняется, а меняются только выражения в правой части: если меняется индекс i, то меняется и значение выражения $\frac{m!}{(m-k)!}x_i^{m-k}$. Вместо того чтобы для каждой узловой точки отдельно решать систему уравнений вида $\hat{\mathbf{A}}\cdot\vec{C}=\vec{B}$, где вектор $\vec{B}$ для каждой узловой точки разный, можем рассмотреть матрицу $\hat{\mathbf{B}}$, составленную из векторов для разных узловых точек. В результате получим матрицу, у которой по столбцам сгруппированы коэффициенты c_m ($m = 0, 1, \ldots, n$) для вычисления производной в соответствующей узловой точке. Если обозначить через вектор-строку со значениями функции в узловых точках, то результатом произведения будет вектор-строка со значениями производной в узловых точках. Именно это выражение вычисляется функцией ПРОИЗВОДНАЯ().

Матрица коэффициентов A определяется командой A = Fa(Px). В данном случае мы вызываем еще одну функцию пользователя Fa(), аргументом которой передается диапазон ячеек Px со значениями узловых точек, а результатом является транспонированная матрица Вандермонда, построенная на основе этих значений. Код функции Fa() обсуждается позже.

При вызове вложенных операторов цикла во внешнем цикле (индексная переменная i пробегает значения от 0 до n) командами x(i) = Px.Cells(i+1).Value и y(i) = Py.Cells(i+1).Value определяются и заносятся в массивы значения узловых точек и значения функции в узловых точках. Во внутреннем цикле (индексная переменная j пробегает значения от 0 до n при каждом фиксированном значении i) командой b(i,j) = Fb(k,i,x(j)) вычисляются элементы для матрицы правых частей системы уравнений. Здесь мы сталкиваемся еще с одной функцией пользователя Fb(), у которой три аргумента (порядок производной, индекс уравнения в системе и значение узловой точки) и которая в качестве результата возвращает правую часть уравнения в решаемой системе уравнений (код функции обсуждается далее).

Для вычисления конечного результата (значение функции) мы предполагаем использовать функции рабочего листа (умножение матриц MMult() и вычисление обратной матрицы MInverse()), поэтому команда вычисления результата помещена в With-блок с инструкцией Application.WorksheetFunction (это нас избавляет от необходимости указывать эту ссылку каждый раз при вызове функции рабочего листа). Значение функции вычисляем командой ПРОИЗВОДНАЯ = .MMult(y,.MMult(.MInverse(A),b)): вектор-строка со значениями функции в узловых точках умножается на матрицу, которая, в свою очередь, получается перемножением обратной матрицы коэффициентов системы и матрицы, составленной из векторов правых частей уравнений.

Нам осталось только проанализировать программный код двух пользовательских функций, которые мы использовали в качестве вспомогательных при вызове функции ПРОИЗВОДНАЯ().

Программный код функции Fa(), предназначенной для вычисления матрицы коэффициентов линейной системы, приведен в листинге 14.2.

Листинг 14.2. Функция для вычисления матрицы коэффициентов системы линейных уравнений

```
Private Function Fa(Px As Range)
 ' Переменная для определения степени интерполяционного полинома
 Dim n As Integer
 ' Степень интерполяционного полинома
 n = Px.Count - 1
 ' Массив для записи элементов матрицы коэффициентов линейной системы
 ReDim A(0 To n, 0 To n) As Double
 ' Целочисленные индексные переменные для операторов цикла
 Dim i As Integer, j As Integer
 ' Внешний цикл
 For i = 0 To n
  ' Внутренний цикл
  For j = 0 To n
   ' Элемент матрицы коэффициентов системы
   A(i, j) = Px.Cells(j + 1).Value^i
  Next j
 Next i
 ' Результат функции
 Fa = A
End Function
```

У функции один аргумент — это диапазон Px со значениями узловых точек. В теле функции командой n = Px.Count-1 на основе количества узловых точек определяется степень интерполяционного полинома, которая совпадает в нашей системе обозначений с максимальным индексом узловой точки (первый индекс нулевой). После этого объявляется двумерный массив A, в который будет записываться результат, и запускаются вложенные операторы цикла (индексные переменные i и j пробегают значения от 0 до n). При этом выполняется всего одна команда A(i,j) = Px.Cells(j+1).Value^i, которой и вычисляются элементы матрицы коэффициентов системы уравнений. Массив A возвращается как результат функции (команда Fa = A).

На заметку

Функция Fa(), так же, как и описываемая далее функция Fb(), объявлена с ключевым словом Private. Сделано это для того, чтобы функции не были доступны в рабочем документе Excel, а только в модуле проекта. Разумеется, ничего страшного не произойдет, если функции можно будет вызвать из рабочего листа, но смысла в этом нет: во-первых, в рабочем листе пользы от них немного, а, во-вторых, «лишние» функции в списке доступных в документе функций могут серьезно усложнять работу.

Программный код функции Fb() приведен в листинге 14.3.

Листинг 14.3. Функция для вычисления правых частей уравнений системы

```
Private Function Fb(k As Integer, i As Integer, z As Double) As Double
 ' Если показатель степени меньше порядка производной
 If (i < k) Then
  ' Нулевая правая часть
   Fb = 0
  ' Если показатель степени не меньше порядка производной
 Else
  ' Вызов функций рабочего листа
  With Application.WorksheetFunction
   ' Ненулевая правая часть
    Fb = .Fact(i) / .Fact(i - k) * (z ^ (i - k))
  End With
 End If
End Function
```

Аргументами функции мы передаем: порядок производной k, индекс i уравнения в системе и значение z узловой точки, для которой записывается/решается система уравнений.

В теле функции проверяется условие i<k (показатель степени аргумента меньше порядка производной), которое означает, что правая часть уравнения должна быть равна нулю. В этом случае выполняется команда Fb = 0. Если условие i<k не выполнено, то результат функции вычисляется командой Fb = .Fact(i)/.Fact(i-k)*(z^(i-k)). Поскольку в этой команде мы использовали вызов функции рабочего листа Fact() для вычисления факториала числа, то команда помещается в With-блок с инструкцией Application.WorksheetFunction.

На заметку

Если мы попытаемся в рабочем документе Excel вычислить результат выражения =0^0, получим ошибку #ЧИСЛО!. Если вычислять выражение 0^0 в программном коде VBA, получим 1.

ОБЩИЕ ПОДХОДЫ К ЧИСЛОВОМУ ИНТЕГРИРОВАНИЮ

Как говорит наш дорогой шеф,
в нашем деле главное — этот самый реализм.

Из к/ф «Бриллиантовая рука»

Здесь мы рассмотрим некоторые приемы, которые позволяют вычислять интегралы вида $\int_a^b f(x)dx$ в числовом виде с использованием возможностей приложения Excel. Ситуации, предполагающие решение подобной задачи, могут быть разными. Например, если первообразная $\int f(x)dx$ не существует в классе элементарных функций или это выражение слишком громоздко для практического использования. Может статься, что подынтегральная функция

задана в виде таблицы. Возможны и другие варианты. Но все они в той или иной степени сводятся, в общем, к одним и тем же алгоритмам вычисления интеграла вида $\int_a^b f(x)dx$. Здесь мы их и рассмотрим. Обычно формулы, по которым вычисляют в числовом виде значения интегралов, называются квадратурными. Существует несколько «классических» квадратурных формул, каждая из которых основывается на интерполировании подынтегральной функции тем или иным способом.

На заметку

Если исходить из формального определения определенного интеграла $\int_a^b f(x)dx$, то это предел интегральной суммы $S_n = \sum_{i=1}^{n} f(x_i)\Delta x_i$ при $n \to \infty$, где n — количество узловых точек x_i ($i = 1, 2, \ldots, n$) на интервале от a до b, а $\Delta x_i = x_i - x_{i-1}$ (при том, что $x_0 \equiv a$ и $x_n \equiv b$). Поэтому если интервал (a, b) конечный, а подынтегральная функция на этом интервале регулярна, то вопрос сводится к характеру сходимости интегральной суммы. Надо отметить, что, хотя интегральная сумма и сходится к интегралу, такая сходимость может быть достаточно медленной. Поэтому вопрос о выборе подходящего алгоритма вычисления интеграла решается в контексте конкретной задачи.

Достаточно часто на практике используют квадратурные *формулы Ньютона — Котеса*. Схема вычисления интеграла $\int_a^b f(x)dx$ в этом случае базируется на интерполировании подынтегральной функции полиномом Лагранжа при условии, что узловые точки равномерно распределены на интервале интегрирования (интервал интегрирования разбивается на равные части). Для удобства будем полагать, что узловые точки выбираются так: $x_i = a + i\Delta x$, где $\Delta x = \frac{b-a}{n}$, а $i = 0, 1, 2, \ldots, n$. В этом случае для интеграла $I = \int_a^b f(x)dx$ имеем оценку $I \approx \int_a^b L_n(x)dx = \int_a^b \sum_{i=0}^{n} y_i \varphi_i(x)dx = \sum_{i=0}^{n} \omega_i y_i$, где мы ввели обозначение $\omega_i = \int_a^b \varphi_i(x)dx$ и $\varphi_i(x) = \prod_{\substack{j=0,\\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}$.

Замена подынтегральной функции на интерполяционный полином позволяет записать интеграл через сумму слагаемых, содержащих интегралы от простых функций, которые в принципе можно вычислить аналитически. Тем не менее совершенно очевидно, что если степень интерполяционного полинома высокая, то задача по вычислению интеграла становится достаточно громоздкой, даже с учетом того, что интегрировать приходится полиномиальные выражения. Поэтому на практике используют интерполяционные зависимости невысоких порядков. При этом исходный интеграл разбивается на сумму интегралов — так, чтобы каждый из подинтервалов содержал небольшое количество узловых точек и интерполяция полиномами небольших степеней (обычно не выше второй) была применима.

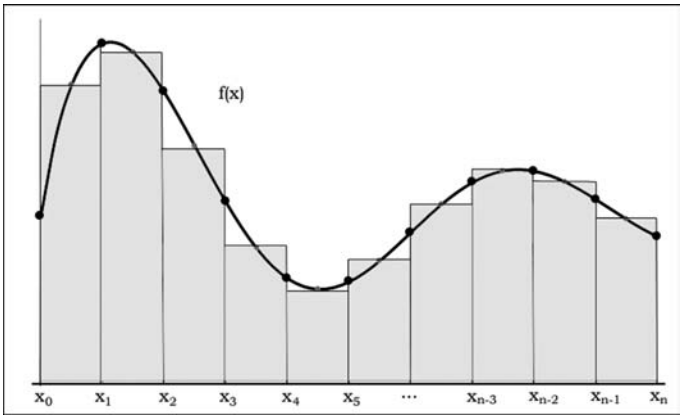


Рис. 14.2
Вычисление интеграла методом прямоугольников: серым цветом выделена область, площадь которой вычисляется в результате подсчета интегральной суммы

Например, при $n = 0$ на каждом из интервалов между соседними узловыми точками подынтегральная функция приближается полиномами нулевой степени, т. е. константой. В этом случае получаем *формулу прямоугольников*: $\int_a^b f(x)dx \approx \sum_{i=1}^{n} f(z_i)\Delta x_i$, где $x_{i-1} \le z_i \le x_i$ — точка на интервале от x_{i-1} до x_i определяет константу (значение $f(z_i)$), которой аппроксимируется функция на указанном интервале. Обычно берут $z_i = \frac{x_{i-1} + x_i}{2}$, т. е. центральную точку интервала. Суть метода прямоугольников иллюстрирует рисунок 14.2.

Значению $n = 1$ соответствует интерполяция линейными полиномами: соседние узловые точки соединяются прямыми линиями. Интеграл вычисляется по формуле

$$\int_a^b f(x)dx \approx \frac{1}{2}\sum_{i=1}^{n}(f(x_i) + f(x_{i-1}))\Delta x_i = \frac{(b-a)}{n}\left(\frac{f(x_0)+f(x_n)}{2} + \sum_{i=2}^{n-1} f(x_i)\right).$$

В этом случае говорят о вычислении интеграла по *методу трапеций*. Методика расчета интегральной суммы по методу трапеций иллюстрируется на рисунке 14.3.

Достаточно популярен также *метод Симпсона* или *метод параболических трапеций*. В этом случае интерполирование выполняется полиномом второго порядка (парабола) по трем соседним узловым точкам.

На заметку

Простоты ради будем полагать, что интервал интегрирования разбит узловыми точками на парное число подынтервалов, т. е. что n — число четное. В противном случае можем изменить пределы интегрирования, изменить количество узловых точек или вычислять интеграл на разных интервалах разными методами. Как и ранее, мы предполагаем, что интервал интегрирования делится на равные части.

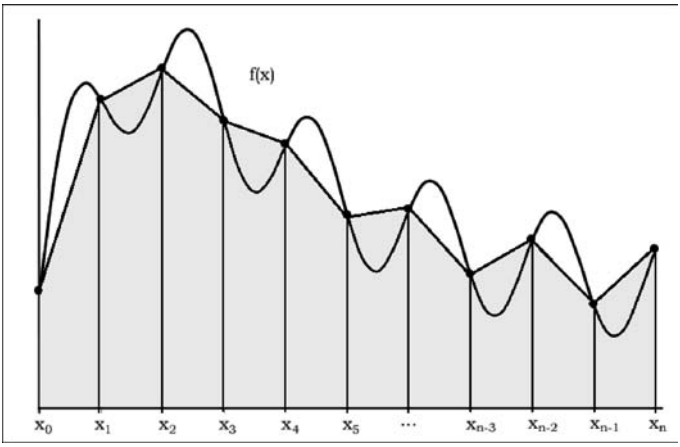


Рис. 14.3
Вычисление интеграла методом трапеций: серым цветом выделена область, площадь которой вычисляется в результате подсчета интегральной суммы

Если рассмотреть три узловые точки x_{i-1}, x_i и x_{i+1}, то парабола, проходящая через эти точки и принимающая в этих точках соответственно значения $f(x_{i-1})$, $f(x_i)$ и $f(x_{i+1})$, задается уравнением

$$y_i(x) = f(x_{i-1})\frac{(x-x_i)(x-x_{i+1})}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})} + f(x_i)\frac{(x-x_{i-1})(x-x_{i+1})}{(x_i-x_{i-1})(x_i-x_{i+1})} + \\ + f(x_{i+1})\frac{(x-x_{i-1})(x-x_i)}{(x_{i+1}-x_{i-1})(x_{i+1}-x_i)}.$$

На заметку

Фактически, в данном случае речь идет о полиноме Лагранжа, построенном по трем точкам.

Несложно вычислить интеграл $\int_{x_{i-1}}^{x_{i+1}} f(x)dx$, который для случая равномерно распределенных узловых точек равен

$$\int_{x_{i-1}}^{x_{i+1}} f(x)dx = \frac{x_{i+1}-x_{i-1}}{6}\left(f(x_{i-1})+4f(x_i)+f(x_{i+1})\right) = \frac{b-a}{6m}\left(f(x_{i-1})+4f(x_i)+f(x_{i+1})\right),$$

где мы обозначили $n = 2m$. Тогда для интеграла по всему интервалу от a до b мы можем записать $\int_a^b f(x)dx = \sum_{i=1}^{m}\int_{x_{2i-2}}^{x_{2i}} f(x)dx$, причем здесь, как и ранее, мы полагали $n = 2m$, $x_0 = a$ и $x_{2m} = x_n = b$. Учитывая явное выражение для интерполяционного параболического полинома, можем получить такую формулу для вычисления интеграла по методу Симпсона:

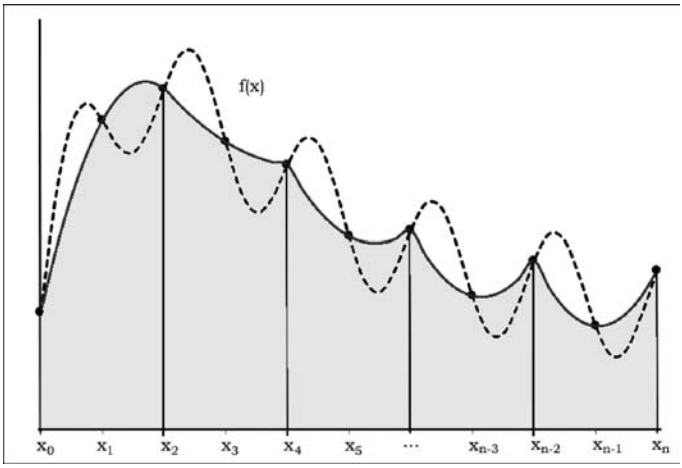


Рис. 14.4
Вычисление интеграла по методу Симпсона:

пунктирной кривой показан график подынтегральной функции, сплошная кривая дает представление о параболических трапециях, серым цветом выделена область, площадь которой вычисляется как результат интегральной суммы.

$$\int_a^b f(x)dx = \frac{b-a}{6m}\left(f(x_0)+4f(x_1)+2f(x_2)+4f(x_3)+2f(x_5)+\ldots+4f(x_{2m-1})+f(x_{2m})\right) =$$

$$= \frac{b-a}{6m}\left(f(x_0)+f(x_{2m})+4\sum_{i=1}^{m} f(x_{2i-1})+2\sum_{i=1}^{m-1} f(x_{2i})\right).$$

Схема интегрирования по Симпсону проиллюстрирована на рисунке 14.4.

Далее рассмотрим примеры, которые позволяют составить некоторое представление о том, как описанные алгоритмы могут быть реализованы средствами приложения Excel.

Сначала рассмотрим пример, в котором для вычисления интеграла методом трапеций создается функция пользователя. Аргументами функции передаются границы области интегрирования, а также ссылка на подынтегральную функцию. Последняя реализуется как метод класса. Программный код этой функции, которая называется ИНТЕГРАЛ(), представлен в листинге 14.4.

Листинг 14.4. Функция для вычисления интегралов

```
Function ИНТЕГРАЛ(f, a, b, Optional n = 100) As Double
 ' Переменная для интегральной суммы
 Dim s As Double
 ' Начальное значение интегральной суммы
  s = 0
 ' Переменная для запоминания длины интервала между узловыми точками
  Dim dx As Double
```

```
    ' Значение длины интервала между узловыми точками
    dx = (b - a) / n
    ' Целочисленная индексная переменная для оператора цикла
    Dim i As Integer
    ' Создаем объект для вызова из него метода,
    ' определяющего подынтегральную функцию
    Set obj = New MyFunctions
    ' Вычисление интегральной суммы
    For i = 0 To n - 1
      ' Добавление очередного слагаемого к интегральной сумме
      s = s + dx * (CallByName(obj, f, VbMethod, a + i * dx) + _
      CallByName(obj, f, VbMethod, a + (i + 1) * dx)) / 2
    Next i
    ' Результат функции
    ИНТЕГРАЛ = s
End Function
```

Помимо трех обязательных аргументов (имя метода f, определяющего подынтегральную функцию, а также границы a и b области интегрирования), у функции есть еще один опционный (необязательный) аргумент — количество частей n, на которые разбивается область интегрирования. Если этот аргумент не указан, по умолчанию область интегрирования разбивается на 100 одинаковых частей.

На заметку

О том, что аргумент опционный и при вызове функции может явно не указываться, свидетельствует ключевое слово Optional в списке аргументов функции ИНТЕГРАЛ(). Значение аргумента по умолчанию (используется, только если при вызове функции аргумент отсутствует) указывается после имени аргумента через знак равенства (в принципе, перед знаком равенства можно было бы еще указать тип аргумента).

Для запоминания значения интегральной суммы вводим переменную s с нулевым начальным значением. В переменную dx командой dx = (b-a)/n записывается значение длины интервала между соседними узловыми точками. Для вычисления значения интегральной суммы запускается оператор цикла, в котором индексная переменная i пробегает значения от 0 до n-1. За каждый цикл к переменной s прибавляется величина dx*(CallByName(obj,f, VbMethod,a+i*dx)+CallByName(obj,f,VbMethod,a+(i+1)*dx))/2. Здесь мы, фактически, для вычисления интегральной суммы используем формулу $\frac{dx}{2}\sum_{i=0}^{n-1}(f(x_i)+f(x_{i+1}))$. Для вычисления значения подынтегральной функции в точке вызывается встроенная функция CallByName(), аргументами которой передаются:

- объект obj, из которого вызывается метод, определяющий подынтегральную функцию;
- имя f вызываемого метода;

- ключевое слово VbMethod, означающее, что вызывается именно метод;
- непосредственно аргумент (a+i*dx или a+(i+1)*dx — значение узловой точки) для вызываемого метода f.

Предварительно (до запуска оператора цикла) объект obj создается командой Set obj = New MyFunctions (создается объект класса MyFunctions). Это означает, что в дальнейшем при вызове функции ИНТЕГРАЛ() первым аргументом этой функции должно передаваться имя метода, описанного в модуле класса с названием MyFunctions.

После завершения оператора цикла переменная s содержит значение интегральной суммы. Командой ИНТЕГРАЛ = s это значение присваивается функции и будет возвращаться функцией как результат.

Для проверки возможностей созданной нами функции вычислим интеграл $\int_0^x z \cdot \exp(-z)dz = 1 - \exp(-x)(1+x)$. Для этого в проекте создаем модуль класса с названием MyFunctions и описываем в этом модуле функцию (метод) с названием Fn и программным кодом таким, как в листинге 14.5.

Листинг 14.5. Подынтегральная функция

```
Function Fn(x) As Double
 ' Значение функции
  Fn = x * Exp(-x)
End Function
```

Очевидно, что в данном случае функция Fn() определена так, что в результате вычисления выражения Fn(x) возвращается значение $x\exp(-x)$, т. е. подынтегральная функция.

На заметку

Напомним, что для создания модуля класса в меню **Insert** редактора VBA необходимо выбрать команду **Class Module**. В результате в проект добавляется новый модуль для описания классов с названием по умолчанию Class1. Чтобы изменить название класса, в окне проекта **Project** в папке **Class Modules** выбираем элемент класса Class1 и затем в окне свойств Properties в поле **Name** редактируем название класса. На рисунке 14.5 показано окно редактора VBA с программным кодом класса MyFunctions.

Программный код функции ИНТЕГРАЛ() вводится в обычном модуле: для этого в меню **Insert** выбираем команду **Module**.

На рисунке 14.6 приведен документ, в котором функция ИНТЕГРАЛ() используется для вычисления означенного выше интеграла для разных значений верхней границы области интегрирования и различного количества интервалов, на которые разбивается область интегрирования.

Документ не очень большой (в смысле количества ячеек, в которых проводятся вычисления), но довольно показательный. Стоит отметить несколько моментов. Во-первых, мы вычисляем интеграл $\int_0^x z \cdot \exp(-z)dz$ для трех значений верхней границы области интегрирования: $x = 1$, $x = 10$ и $x = 100$.

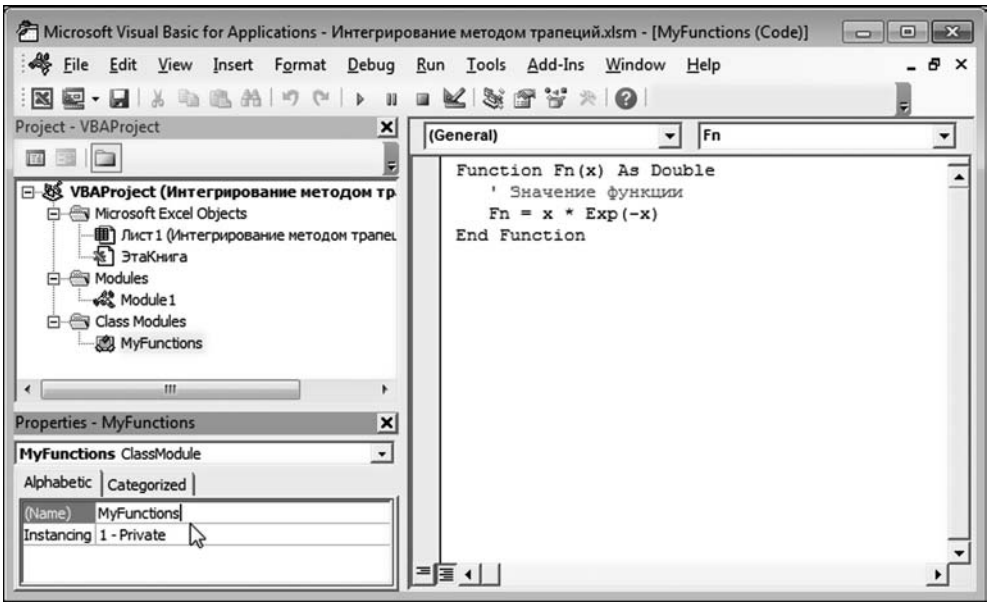
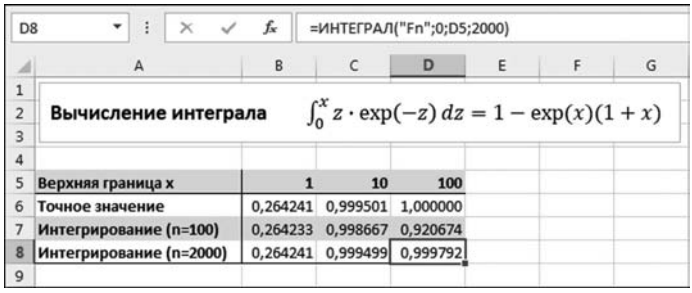
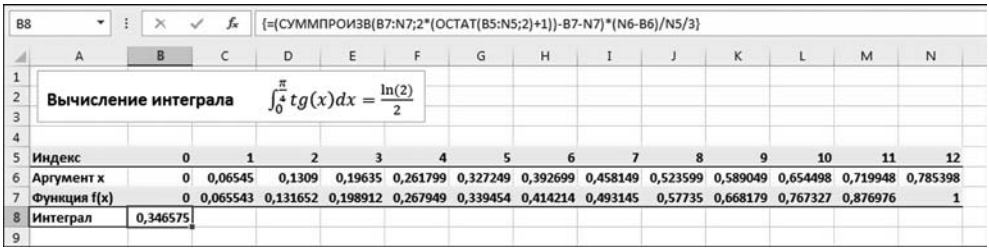


Рис. 14.5
В модуле класса MyFunctions описана функция (метод), определяющая подынтегральное выражение



	A	B	C	D	E	F	G
5	Верхняя граница x	1	10	100			
6	Точное значение	0,264241	0,999501	1,000000			
7	Интегрирование (n=100)	0,264233	0,998667	0,920674			
8	Интегрирование (n=2000)	0,264241	0,999499	0,999792			

Рис. 14.6
Вычисление интегралов с помощью функции пользователя



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
5	Индекс	0	1	2	3	4	5	6	7	8	9	10	11	12
6	Аргумент x	0	0,06545	0,1309	0,19635	0,261799	0,327249	0,392699	0,458149	0,523599	0,589049	0,654498	0,719948	0,785398
7	Функция f(x)	0	0,065543	0,131652	0,198912	0,267949	0,339454	0,414214	0,493145	0,57735	0,668179	0,767327	0,876976	1
8	Интеграл	0,346575												

Рис. 14.7
Вычисление в рабочем документе интеграла по методу Симпсона

Значения верхней границы интегрирования отображаются в ячейках B5:D5. Для сравнения результатов, полученных на основе разработанной нами функции, с тем, что должно быть на самом деле, мы используем точное аналитическое выражение для интеграла и вычисляем соответствующие «точные» значения в ячейках B6:D6. Для этого мы в ячейку B6 вводим формулу =1-EXP(-B5)*(B5+1) (точное значение для интеграла дается выражением $1 - \exp(-x)(1 + x)$, реализацией которого и является данная формула), и затем копируем ее в ячейки C6 и D6.

В ячейках B7:D7 значения для интеграла вычисляются на основе функции ИНТЕГРАЛ(): в ячейку B7 вводится формула =ИНТЕГРАЛ("Fn";0;B5) и затем последовательно копируется в ячейки C7 и D7. Таким образом, поскольку четвертый необязательный аргумент не указан, в этом случае интеграл вычисляется путем разбивки области интегрирования на 100 одинаковых областей (значение четвертого аргумента по умолчанию). И если для относительно небольших значений верхней границы области интегрирования совпадение с «точным» значением неплохое, то по мере увеличения области интегрирования вычисленное на основе функции пользователя значение для интеграла все больше отклоняется от «нормы». Объяснение очевидное: поскольку количество узловых точек одно и то же, а область интегрирования расширяется, длина интервала между узловыми точками увеличивается, что приводит к возрастанию погрешности вычислений. Чтобы улучшить результат, необходимо увеличить количество узловых точек (уменьшив тем самым длину интервала между узловыми точками). В ячейках B8:D8 интеграл вычисляется при условии, что область интегрирования разбивается на 2000 частей: в ячейку B8 вводится формула =ИНТЕГРАЛ("Fn";0;B5;2000) и копируется в остальные ячейки диапазона. Как видим, точность вычислений в этом случае существенно выше.

В качестве еще одного примера полезности приложения Excel при вычислении определенных интегралов рассмотрим документ, в котором средствами рабочего листа Excel по методу Симпсона вычисляется интеграл $\int_0^{\pi/4} \mathrm{tg}(x)dx = \frac{\ln(2)}{2} \approx 0{,}346574$. Документ, в котором производятся соответствующие вычисления, представлен на рисунке 14.7.

Интеграл вычисляется на интервале от 0 до $\pi/4$, который делится на 12 равных по величине (длине) частей. Узловые точки в этом случае вычисляются соотношением $x_k = \frac{k\pi}{48}$ при $k = 0, 1, 2, \ldots, 12$. Значения индекса k вводятся в ячейки B5:N5. Диапазон ячеек B6:N6 заполняется значениями узловых точек. Для этого в ячейку B6 вводится формула =B5/$N$5*ПИ()/4 и копируется в ячейки справа, вплоть до заполнения массива B6:N6. Ячейки B7:N7 заполняются значениями подынтегральной функции в узловых точках. Эти ячейки заполняются копированием формулы =TAN(B6) из ячейки B7.

Для вычисления результата (значения интеграла) в ячейку B8 вводится формула массива =(СУММПРОИЗВ(B7:N7;2*(ОСТАТ(B5:N5;2)+1))-B7-N7)*(N6-B6)/N5/3 (т. е. формула вводится нажатием комбинации клавиш <Ctrl>+<Shift>+<Enter>).

На заметку

Пожалуй, имеет смысл прокомментировать формулу =(СУММПРОИЗВ(B7:N7;2*(ОСТАТ(B5:N5;2)+1))-B7-N7)*(N6-B6)/N5/3, которой вычисляется результат (интеграл). Сразу отметим, что здесь мы воспользовались следующей формулой для вычисления интеграла по методу Симпсона: $\int_{x_0}^{x_n} f(x)dx = \frac{x_n - x_0}{3n}\left(2\sum_{k=0}^{n} c_k f(x_k) - f(x_0) - f(x_n)\right)$, где коэффициенты $c_k = 1$ для четных индексов k, и $c_k = 2$ для нечетных индексов k. Если через mod обозначить оператор вычисления остатка от целочисленного деления, то можем записать $c_k = (k \bmod 2) + 1$, т. е. это остаток от деления на 2 плюс 1. В Excel для вычисления остатка от деления используется функция ОСТАТ(). Первым аргументом передается делимое, а вторым — делитель. Поэтому результатом выражения 2*(ОСТАТ(B5:N5;2)+1) (с учетом того, что формула вводится как формула массива) является массив значений из единиц и двоек, умноженных на два, т. е. последовательно сменяющиеся числа 2 и 4. Этот массив поэлементно перемножается со значениями из диапазона B7:N7 (первый аргумент функции СУММПРОИЗВ()), от полученного выражения отнимаются значения подынтегральной функции в начальной и конечной узловой точках (ячейки B7 и N7), и полученное таким образом значение умножается на длину интервала интегрирования (разница конечных узловых точек N6-B6), делится на максимальный индекс узловых точек (ячейки N5) и делится на 3 (в соответствии с исходной формулой для интегральной суммы).

Если сравнить вычисленное нами значение для интеграла с точным результатом, легко заметить, что точность вычислений вполне приемлемая.

На заметку

Разумеется, существуют и более точные способы определения погрешности вычислений интегралов. Однако эта тема может послужить предметом отдельного исследования и однозначно выходит за рамки данной книги. Заинтересованный читатель, без сомнения, в случае необходимости может обратиться к специальной литературе, посвященной этому вопросу.

МЕТОД ЧЕБЫШЕВА

Это в твоих руках все горит,
а в его руках все работает!

Из к/ф «Покровские ворота»

Выше мы рассматривали случаи, когда подынтегральная функция известна, а узловые точки в области интегрирования распределены равномерно. Что касается равноотстоящих узловых точек, то даже без специального исследования вполне очевидно, что такой способ выбора узловых точек не всегда является оптимальным. Скажем, в той области, где подынтегральная функция достаточно плавная, интервал между узловыми точками может быть больше, чем в области, где функция быстро осциллирующая. Поэтому задачу о вычислении интегральной суммы можно рассматривать и в контексте выбора оптимального расположения узловых точек. Кроме того, ранее мы отмечали: совсем необязательно, чтобы подынтегральная функция была за-

дана в аналитическом виде (т. е. известна в каждой точке области интегрирования) — она может быть задана в виде таблицы, так, как это было при построении интерполяционного полинома. Подобная ситуация может сложиться, если значения функции определяются (в том или ином виде) из эксперимента. При этом мы исходим из того, что в наших силах выбирать узловые точки, а подынтегральная функция для каждой узловой точки может быть определена. Причем, если отталкиваться от «гипотезы экспериментального определения подынтегральной функции», то разумно предположить, что значения функции в узловых точках содержат некоторую случайную составляющую (т. е. содержат некоторую погрешность). Задача сводится к тому, чтобы выбрать «наилучшее» расположение узловых точек для вычисления интеграла от функции по некоторой области. Это преамбула. Теперь приступим к более конкретной постановке задачи и опишем алгоритм ее решения.

В первую очередь отметим, что интеграл вида $\int_a^b f(x)dx$ по конечной области мы можем свести к интегралу по симметричному интервалу от -1 до 1. Для этого достаточно сделать замену переменной $z = 2\cdot\frac{x-a}{b-a} - 1$, т. е. положить $x = a + \frac{(b-a)(z+1)}{2} = \frac{a+b}{2} + z\frac{b-a}{2}$. Тогда $\int_a^b f(x)dx = \frac{(b-a)}{2}\int_{-1}^1 f\left(\frac{a+b}{2} + z\frac{(b-a)}{2}\right)dz \equiv \int_{-1}^1 F(z)dz$, где мы ввели обозначение $F(t) = \frac{(b-a)}{2} f\left(\frac{a+b}{2} + t\frac{b-a}{2}\right)$. Поэтому, не ограничивая общности, можем далее рассматривать интеграл вида $\int_{-1}^1 F(z)dz$.

На заметку

На практике алгоритм действий при вычислении интеграла может быть таким:

- для вычисления интеграла $\int_a^b f(x)dx$ на основе подынтегральной функции $f(x)$ «создаем» функцию $F(x) = \frac{(b-a)}{2} f\left(\frac{a+b}{2} + x\frac{b-a}{2}\right)$;
- вычисляем интеграл $I = \int_{-1}^1 F(x)dx$;
- это и есть значение искомого интеграла, т. е. $\int_a^b f(x)dx = I$.

Если количество узловых точек n фиксировано (но априори не задано их положение — мы можем выбирать его сами), то задача состоит в определении весовых коэффициентов ω_k ($k = 1, 2, \ldots, n$ — индексация начинается с 1!) интегральной суммы $\sum_{k=1}^n \omega_k F(x_k)$, которой оценивается значение интеграла $\int_{-1}^1 F(x)dx$. Из условия, что при $F(x) \equiv 1$ интеграл $\int_{-1}^1 F(x)dx = \int_{-1}^1 dx = 2$, получаем, что $\sum_{k=1}^n \omega_k = 2$. Еще одно условие, которое накладывается на узловые

точки, состоит в том, что среднеквадратичная ошибка интеграла, вычисленного по формуле $\int_{-1}^{1} F(x)dx = \sum_{k=1}^{n} \omega_k F(x_k)$, была минимальной (при условии постоянства суммы весовых коэффициентов). Несложно показать, что в этом случае все весовые коэффициенты должны быть одинаковыми, т. е. $\omega_k = \omega = \text{const}$. С учетом соотношения $\omega_1 + \omega_2 + \ldots + \omega_n = 2$ легко получить $\omega_k = 2/n$ для всех $k = 1, 2, \ldots, n$.

Таким образом, задача свелась к следующей: на интервале от -1 до 1 необходимо определить узловые точки $x_1, x_2, \ldots, x_n$ так, чтобы интеграл $\int_{-1}^{1} F(x)dx = \frac{2}{n}\sum_{k=1}^{n} F(x_k)$. Метод (алгоритм), который позволяет решить эту задачу, носит имя *Чебышева* и состоит в следующем: будем выбирать узловые точки так, чтобы формула $\int_{-1}^{1} F(x)dx = \frac{2}{n}\sum_{k=1}^{n} F(x_k)$ была точной для полиномиальных функций $x, x^2, \ldots, x^n$. Чтобы найти узловые точки, необходимо каждую из этих функций подставить в интеграл, вычислить его и приравнять к соответствующей интегральной сумме. В результате получим систему из n уравнений для определения узловых точек $x_1, x_2, \ldots, x_n$. В частности, несложно вычислить $\int_{-1}^{1} x^{2m+1}dx = 0$ и $\int_{-1}^{1} x^{2m}dx = \frac{2}{2m+1}$, поэтому получаем такие уравнения: $\frac{2}{n}\sum_{k=1}^{n} x_k^{2m+1} = 0$ и $\frac{2}{n}\sum_{k=1}^{n} x_k^{2m} = \frac{2}{2m+1}$, что в конечном итоге дает $\sum_{k=1}^{n} x_k^{2m+1} = 0$ и $\sum_{k=1}^{n} x_k^{2m} = \frac{n}{2m+1}$ соответственно. Другими словами, имеем дело со следующей системой нелинейных алгебраических уравнений:

$$x_1 + x_2 + \ldots + x_n = 0,$$
$$x_1^2 + x_2^2 + \ldots + x_n^2 = \frac{n}{3},$$
$$x_1^3 + x_2^3 + \ldots + x_n^3 = 0,$$
$$x_1^4 + x_2^4 + \ldots + x_n^4 = \frac{n}{5},$$

и т. д. — последнее уравнение в левой части содержит сумму $x_1^n + x_2^n + \ldots + x_n^n$, которая равняется нулю для нечетных n и равна $n/(n+1)$ для четных n. Следовательно, имеем систему из n уравнений относительно n неизвестных величин $x_1, x_2, \ldots, x_n$. Проблема в том, что эта система нелинейная.

На заметку

Стоит обратить внимание на одно немаловажное обстоятельство: выбор узловых точек $x_1, x_2, \ldots, x_n$ не зависит от подынтегральной функции. Более того, поскольку фиксирован и интервал интегрирования, то вычисляются эти точки всего один раз и, как говорится, на все случаи жизни. Более того, при $n > 9$ и при $n = 8$ среди решений приведенной выше системы имеются комплексные. Это означает, что для таких значений n метод Чебышева использован быть не может. Поэтому «рабочими» являются следующие значения n: 1 (на практике малоинтересный случай), 2, 3, 4, 5, 6, 7 и 9. Естественно, для всех

этих случаев узловые точки уже вычислены, т. е. мы можем воспользоваться готовыми табличными значениями. Тем не менее используя Excel, все эти узловые точки можно легко вычислить.

Хотя узловые точки, вычисляемые в методе Чебышева, собственно вычислены до нас, мы все же проиллюстрируем возможности Excel, рассчитав несколько интегралов средствами рабочего листа Excel, включая и вычисление узловых точек. В последнем случае, учитывая нелинейный характер решаемой при этом алгебраической системы, воспользуемся надстройкой **Поиск решения**.

На рисунке 14.8 представлена уже финальная версия документа, в котором вычислены узловые точки для $n = 9$ (т. е. девять узловых точек) и на основе полученных значений рассчитаны несколько интегралов.

В частности, мы в числовом виде вычисляем такие интегралы:

$$\int_a^b x(1-x)dx = \left(\frac{b^2}{2} - \frac{b^3}{3}\right) - \left(\frac{a^2}{2} - \frac{a^3}{3}\right),$$

$$\int_a^b x\cos(\pi x)dx = \frac{(\cos(\pi b) - \cos(\pi a)) + (b\sin(\pi b) - a\sin(\pi a))}{\pi^2},$$

$$\frac{2}{\sqrt{\pi}}\int_a^b \exp(-x^2)dx = erf(b) - erf(a)$$

(через $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x \exp(-t^2)dt$ обозначена «функция ошибок») и $\int_a^b x^2\exp(-x)dx =$ $= \exp(-a)(a^2 + 2a + 2) - \exp(-b)(b^2 + 2b + 2)$. Документ организован так, что границы a и b области интегрирования вводятся в ячейки рабочего документа и при их изменении выполняется автоматический пересчет интегралов. Кроме

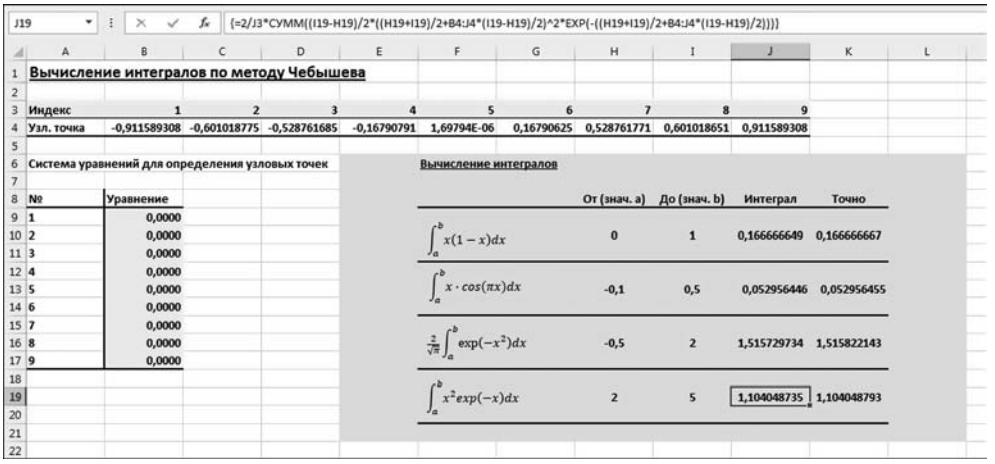


Рис. 14.8
Вычисление интегралов по методу Чебышева

этого, для сравнения также приводятся значения, вычисленные на основе аналитических выражений для приведенных выше интегралов.

На заметку

Собственно, поэтому для каждого из интегралов указано аналитическое выражение, зависящее от границ области интегрирования, как от параметров — эти выражения используются при вводе формул в рабочий документ Excel для вычисления «точного» значения интеграла.

Узловые точки вычислены в ячейках B4:J4. Это числовые значения, которые заносятся в указанные ячейки после запуска надстройки **Поиск решения**.

На заметку

Как вычисляются узловые точки, будет рассказано несколько позже. Кроме того, в документе в ячейках B9:B17 содержатся нулевые значения. Так и должно быть. В этих ячейках вычисляются функции, определяющие уравнения, через которые определяются узловые точки. Если узловые точки (диапазон ячеек B4:J4) вычислены правильно, то все функции уравнений должны обращаться в ноль. Какие формулы вводятся в ячейки B9:B17 и какова их роль при запуске надстройки **Поиск решения**, также будет рассказано.

Рассмотрим более детально методику вычисления интегралов. Здесь важно понимать, что залог успеха кроется в том, какие формулы и как используются для вычисления интегральной суммы. Надо признать, что формулы эти достаточно громоздкие. Более того, каждый интеграл требует индивидуального подхода. Индивидуальность интеграла, как несложно догадаться, «спрятана» в его подынтегральной функции.

На заметку

Прежде, чем приступить к анализу конкретных случаев, отметим одну общую особенность, которая была использована при вычислении интегралов. Дело в том, что формула Чебышева для интегральной суммы подразумевает вычисление суммы значений в узловых точках некоторой функции (подынтегральной). Более конкретно, значения узловых точек у нас записаны в ячейках B4:J4. Если бы нас интересовала одна узловая точка, то ссылка на соответствующую ячейку играла бы роль аргумента подынтегральной функции. Если нам нужны значения функции в нескольких точках, то аргументом функции можно передать сразу весь массив значений аргумента (диапазон ячеек), а формулу вычислять как формулу массива. В этом случае получаем массив значений функции. Если необходимо просуммировать эти значения, используем встроенную функцию СУММ(). Таким образом, все формулы для вычисления интегральных сумм вводятся как формула массива (комбинация клавиш <Ctrl>+<Shift>+<Enter>), основу формулы составляет встроенная функция СУММ(), а роль переменной интегрирования играет ссылка на диапазон B4:J4.

Если бы речь шла о вычислении интеграла от функции $F(z)$ на интервале от -1 до 1, то достаточно было бы аргументом функции СУММ() ввести выражение, определяющее функцию $F(x)$, где роль аргумента играла бы ссылка на диапазон ячеек с узловыми точками. Все это выражение умножается на 2, делится на количество узловых точек и вводится как формула массива. Проблема, однако, в том, что функция $F(x)$ — это не та функция, что интегри-

руется, но определяется через нее соотношением $F(x)=\frac{(b-a)}{2}f\left(\frac{a+b}{2}+x\frac{b-a}{2}\right)$. Параметры a и b реализуются через ссылки на ячейки рабочего листа. Но даже если выражение для «реальной» подынтегральной функции $f(x)$ простое, формулу вводить нужно для вычисления выражения $\frac{(b-a)}{2}f\left(\frac{a+b}{2}+x\frac{b-a}{2}\right)$. Отсюда и громоздкость выражений. Например, для подынтегральной функции $f(x)=x+\frac{1}{x}$ нужно ввести формулу для вычисления выражения $\left(\frac{b-a}{2}\right)\left(\left(\frac{a+b}{2}+x\frac{b-a}{2}\right)+\frac{1}{\left(\frac{a+b}{2}+x\frac{b-a}{2}\right)}\right)$. Пожалуй, это самый неприятный момент в том подходе, который мы здесь используем.

Для вычисления интеграла $\int_a^b x(1-x)dx$ в ячейку J10 вводится формула массива =2/J3*СУММ((I10-H10)/2*((H10+I10)/2+B4:J4*(I10-H10)/2)*(1-((H10+I10)/2+B4:J4*(I10-H10)/2))). При этом нижняя граница области интегрирования записана в ячейку H10, а верхняя граница области интегрирования задается значением в ячейке I10. Ячейка J3 содержит значение количества узловых точек. Для проверки в ячейке K10 по формуле =(I10^2/2-I10^3/3)-(H10^2/2-H10^3/3) вычисляется «точное» значение для интеграла.

Интеграл $\int_a^b x\cos(\pi x)dx$ вычисляем в ячейке J13 по формуле массива =2/J3*СУММ((I13-H13)/2*((H13+I13)/2+B4:J4*(I13-H13)/2)*COS(ПИ()*((H13+I13)/2+B4:J4*(I13-H13)/2))). Значение этого же интеграла, вычисленное на основе аналитической формулы, получаем в ячейке K13 благодаря формуле =(COS(ПИ()*I13)-COS(ПИ()*H13)+ПИ()*(I13*SIN(ПИ()*I13)-H13*SIN(ПИ()*H13)))/ПИ()^2. Значения границ области интегрирования содержатся в этом случае в ячейках H13 и I13. Для вычисления значения числа π используем встроенную функцию ПИ().

Для вычисления интеграла $\frac{2}{\sqrt{\pi}}\int_a^b \exp(-x^2)dx$ в ячейку J16 вводим формулу массива =2/КОРЕНЬПИ(1)*2/J3*СУММ((I16-H16)/2*EXP((-1)*((H16+I16)/2+B4:J4*(I16-H16)/2)^2)). В ячейках H16 и I16 содержатся значения границ области интегрирования. Примечательно также то, что для вычисления значения $\sqrt{\pi}$ мы использовали встроенную функцию КОРЕНЬПИ() с единичным аргументом (коэффициент, на который умножается число π). «Проверочное» значение для интеграла вычисляется в ячейке K16 по формуле =ФОШ(H16;I16). Здесь мы воспользовались встроенной функцией ФОШ(), предназначенной для вычисления «функции ошибок» (или функции Лапласа). Аргументы функции ФОШ() определяют границы области интегрирования: т. е. значением выражения ФОШ(a,b) является разность $\mathrm{erf}(b)-\mathrm{erf}(a)$.

Еще один интеграл, а именно, $\int_a^b x^2\exp(-x)dx$, вычисляется в ячейке J19 по формуле массива =2/J3*СУММ((I19-H19)/2*((H19+I19)/2+B4:J4*(I19-H19)/

2)^2*EXP(-((H19+I19)/2+B4:J4*(I19-H19)/2))). Формула, кроме всего прочего, содержит ссылки на ячейки H19 и I19 со значениями границ области интегрирования. Для проверки результатов вычислений в ячейку K19 вводится формула =EXP(-H19)*(H19^2+2*H19+2)-EXP(-I19)*(I19^2+2*I19+2).

Как видим, в основном результаты вычислений вполне приемлемые — во всяком случае, для тех областей интегрирования, которые заданы априори.

На заметку

Область интегрирования для каждого из интегральных выражений определяется значениями в ячейках рабочего документа. Изменение значения этих ячеек приводит к автоматическому пересчету интеграла, причем как того, что получен по методу Чебышева, так и вычисленного на основе аналитического выражения.

Теперь кратко остановимся на том, как же все-таки вычисляются узловые точки. Для этого обратимся к рисунку 14.9, на котором показан рабочий документ в своем почти исходном состоянии: интегралы еще не вычислялись, и узловые точки еще не вычислены.

В ячейки B4:J4 введены начальные числовые значения для узловых точек, более-менее равномерно распределенные на интервале от −1 до 1. Ячейки B9:B17 заполняются не так прозаично: в ячейку B9 вводится формула массива =СУММ($B$4:$J$4^A9)-(1-ОСТАТ(A9;2))*$A$17/(A9+1), после чего она копируется во все остальные ячейки диапазона B9:B17. Соответствующая формула представляет собой реализацию уравнений вида $x_1^m + x_2^m + \ldots + x_n^m - c_m \frac{n}{m+1} = 0$, которые мы уже обсуждали выше (правда, записаны они были в несколько иной форме). Коэффициенты c_m в данном случае могут принимать два значения: $c_m = 0$ для нечетных m и $c_m = 1$ для четных m. Значение параметра m определяется значениями в ячейках A9:A17.

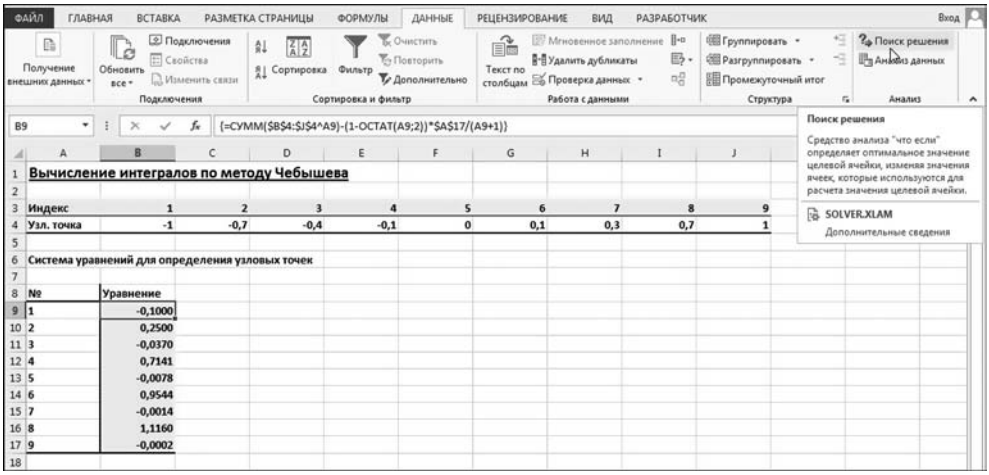


Рис. 14.9
Запуск надстройки **Поиск решения** для вычисления узловых точек в методе Чебышева

На заметку

Значения в ячейках A9:A17 вычисляются на основе значений ячеек B3:J3 через формулу массива =ТРАНСП(B3:J3).

Коэффициент c_m в ячейке B9 вычисляется на основе значения параметра m (ячейка A9) как разность единицы и остатка от деления параметра m на 2, что в итоге дает выражение (1-ОСТАТ(A9;2)). Выражение для вычисления слагаемого $c_m \frac{n}{m+1}$ выглядит как (1-ОСТАТ(A9;2))*$A$17/(A9+1). Здесь мы учли, что значение n записано в ячейку A17, и при копировании формулы ссылка на эту ячейку меняться не должна (поэтому мы использовали абсолютную ссылку).

Для вычисления суммы $x_1^m + x_2^m + \ldots + x_n^m$ использовано выражение СУММ ($B$4:$J$4^A9). Именно ради обработки этого выражения вся формула вводится как формула массива. Значение выражения вычисляется так: значения ячеек диапазона $B$4:$J$4 возводятся в степень A9 каждое, и затем вычисляется сумма всех элементов полученного таким образом массива.

Итак, для запуска надстройки **Поиск решения** нужно щелкнуть на одноименной пиктограмме в группе **Анализ** вкладки **Данные** ленты (рис. 14.9).

На заметку

Напомним, что прежде, чем использовать надстройку, ее необходимо подключить. Подключена или нет надстройка, понять легко: если на вкладке **Данные** в группе **Анализ** имеется пиктограмма **Поиск решения** — надстройка подключена. Иначе надстройку нужно подключать. Для этого открываем окно **Параметры Excel**, в котором выбираем раздел **Надстройки**. В раскрывающемся списке **Управление** выбираем пункт **Надстройки Excel** и щелкаем кнопку **Перейти**. Открывается окно **Надстройки** со списком доступных надстроек. Возле позиции, соответствующей надстройке **Поиск решения**, щелкаем кнопку **ОК**.

После запуска надстройки **Поиск решения** открывается окно **Параметры поиска решения**, в котором выполняются настройки для поиска узловых точек (см. рис. 14.10).

Сразу отметим, что решать задачу (и, соответственно, выполнять настройки) можно по-разному. Здесь мы пойдем наиболее простым путем:

- в поле **Оптимизировать целевую ячейку** указываем ячейку B9;
- переключатель **До** устанавливаем в положение **Значения** и в поле справа вводим 0;
- в поле **Изменяя ячейки переменных** указываем диапазон B4:J4;
- добавляем ограничение B10:B17=0 (нулевые значения для функций, определяющих уравнения системы);
- добавляем ограничение B4:J4<=1 (значения узловых точек не превышают 1);
- добавляем ограничение B4:J4>=–1 (значения узловых точек не меньше –1).

После щелчка кнопки **Найти решение** (рис. 14.10) и подтверждения для полученного решения рабочий документ может выглядеть так, как показано на рисунке 14.11.

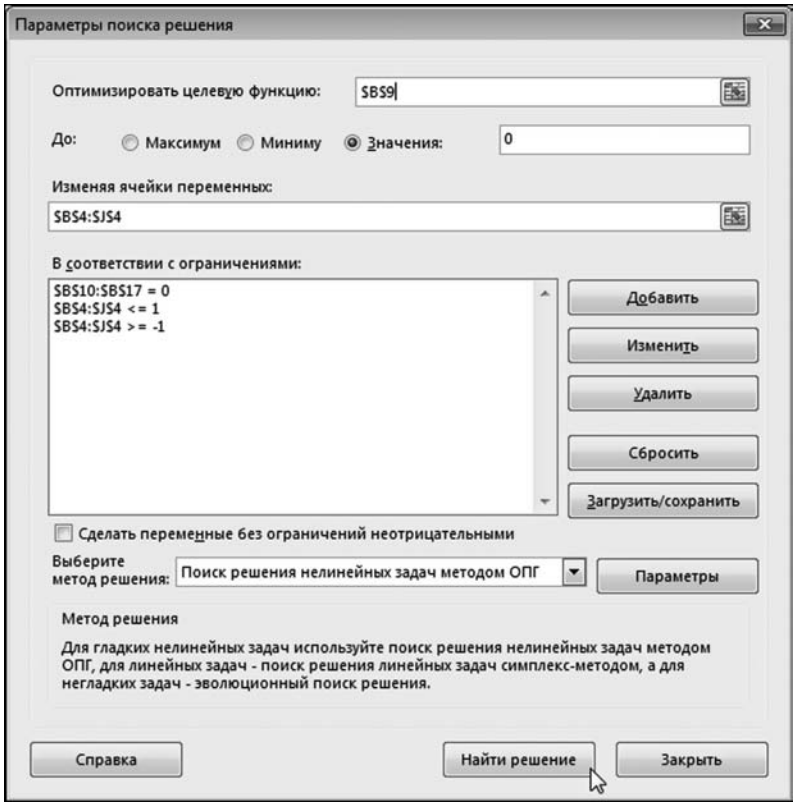


Рис. 14.10
Окно надстройки **Поиск решения** с выполненными настройками для вычисления узловых точек

B9 {=СУММ($B$4:$J$4^A9)-(1-ОСТАТ(A9;2))*$A$17/(A9+1)}

	A	B	C	D	E	F	G	H	I	J
1	**Вычисление интегралов по методу Чебышева**									
2										
3	**Индекс**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
4	**Узл. точка**	**-0,911589306**	**-0,601017391**	**-0,52876302**	**-0,167906237**	**-1,64439E-12**	**0,167906237**	**0,52876302**	**0,601017392**	**0,911589306**
5										
6	**Система уравнений для определения узловых точек**									
7										
8	**№**	**Уравнение**								
9	**1**	**0,0000**								
10	**2**	**0,0000**								
11	**3**	**0,0000**								
12	**4**	**0,0000**								
13	**5**	**0,0000**								
14	**6**	**0,0000**								
15	**7**	**0,0000**								
16	**8**	**0,0000**								
17	**9**	**0,0000**								
18										

Рис. 14.11
Результат вычисления узловых точек с помощью надстройки **Поиск решения**

Главным критерием корректности полученного результата является равенство нулю значений в ячейках B9:B17. Значения узловых точек отображаются, напомним, в ячейках B4:J4.

После этого можно приступить к вычислению интегралов. Как это делается, было описано выше. Вместе с тем, совершенно очевидно, что процесс вычисления интегралов по методу Чебышева можно существенно оптимизировать и автоматизировать. Этим и займемся далее.

Создадим несколько функций, которые позволят быстро и легко вычислять интегралы по методу Чебышева. В частности, нам понадобятся, по меньшей мере, такие функции:

- функция для вычисления узловых точек;
- функция для вычисления интегральной суммы по методу Чебышева;
- функция для вычисления интеграла на заданном интервале от заданной функции.

Такие функции могут быть реализованы по принципу «матрешки»: функция для вычисления узловых точек вызывается в функции для вычисления интегральной суммы по методу Чебышева, а эта, в свою очередь, вызывается в функции для вычисления интеграла от заданной функции по заданной области.

В листинге 14.6 приведен программный код функции, предназначенной для вычисления узловых точек.

Листинг 14.6. Функция для вычисления узловых точек

```
Function УЗЛТОЧКИ(n, Optional i)
 ' Переменная для записи массива узловых точек
 Dim x As Variant
 ' Перебираем варианты
 Select Case n
  ' Две узловые точки
  Case 2
    x = Array(-0.57735, 0.57735)
   ' Три узловые точки
  Case 3
    x = Array(-0.707107, 0, 0.707107)
   ' Четыре узловые точки
  Case 4
    x = Array(-0.794654, -0.187592, 0.187592, 0.794654)
   ' Пять узловых точек
  Case 5
    x = Array(-0.832498, -0.374541, 0, 0.374541, 0.832498)
   ' Шесть узловых точек
  Case 6
    x = Array(-0.866247, -0.422519, -0.266635, 0.266635, _
   0.422519, 0.866247)
  ' Семь узловых точек
  Case 7
```

```
    x = Array(-0.883862, -0.529657, -0.323912, 0, 0.323912, _
   0.529657, 0.883862)
  ' Девять узловых точек
  Case 9
    x = Array(-0.911589, -0.601019, -0.528762, -0.167906, 0, _
   0.167906, 0.528762, 0.601019, 0.911589)
  ' Все прочие случаи
  Case Else
    ' Ошибка #ПУСТО!
    УЗЛТОЧКИ = CVErr(xlErrNull)
    ' Завершение выполнения кода функции
    Exit Function
 End Select
 ' Если второй аргумент функции не указан
 If (IsMissing(i)) Then
  ' Функция возвращает массив узловых точек
  УЗЛТОЧКИ = x
 Else
  ' Второй аргумент указан, но выходит за допустимые пределы
  If (i < 1 Or i > n) Then
    ' Ошибка #Н/Д
    УЗЛТОЧКИ = CVErr(xlErrNA)
  Else
    ' Функция возвращает одну узловую точку
    УЗЛТОЧКИ = x(i - 1)
  End If
 End If
End Function
```

У функции УЗЛТОЧКИ() два аргумента, причем второй — необязательный. В качестве результата функция возвращает одну узловую точку (если ей переданы два аргумента) или весь массив узловых точек (если функции при вызове передан только один аргумент). Первый обязательный аргумент функции n определяет количество узловых точек. Второй необязательный аргумент i определяет индекс узловой точки, возвращаемой функцией в качестве результата.

На заметку

Необязательные аргументы описываются с ключевым словом Optional. Через знак равенства для таких аргументов можно указать значение по умолчанию.

В теле функции мы объявляем переменную x типа Variant, в которую затем запишем массив узловых точек. С помощью оператора Select начинается перебор вариантов: мы проверяем значение первого аргумента функции n, и в зависимости от него создаются различные массивы с узловыми точками. В любом случае результат записывается в переменную x.

На заметку

Обратите внимание, что в данном случае мы массив явно не создаем. Мы вместо этого объявляем переменную типа Variant. В эту переменную можно записать «все, что угодно», в том числе и массив. Сам массив далее будет создаваться с помощью функции Array() с передачей аргументами этой функции значений элементов массива.

Мы рассматриваем как допустимые случаи, когда переменная n принимает значения 2, 3, 4, 5, 6, 7 и 9. Для каждого из этих значений переменной n с помощью функции Array() создается массив и записывается в переменную x. В каждом случае количество элементов массива и их значения различны. Сами значения мы не вычисляли, а воспользовались справочной информацией (т. е. прибегли к помощи уже готовых и рассчитанных таблиц). На тот случай, когда значение переменной n не совпадает ни с одним из перечисленных выше значений, предназначен блок Case Else оператора выбора Select. В этом случае функцией возвращается значение ошибки #ПУСТО! (команда УЗЛТОЧКИ = CVErr(xlErrNull)), и выполнение кода функции завершается инструкцией Exit Function.

После завершения выполнения оператора выбора, начинается обработка второго аргумента (если он имеется). Наличие этого аргумента проверяется в условном операторе, в котором в качестве проверяемого условия указано выражение IsMissing(i). Выражением IsMissing(i) возвращается значение True, если при вызове функции УЗЛТОЧКИ() ее второй (необязательный) аргумент указан (обозначается переменной i), и False — в противном случае. Если второй аргумент функции УЗЛТОЧКИ() не передавался, то командой УЗЛТОЧКИ = x в Then-блоке условного оператора в качестве значения функции присваивается весь массив узловых точек. Если второй аргумент явно указан, то на сцену выходит Else-блок условного оператора. В этом блоке с помощью еще одного условного оператора проверяется условие выхода переменной i за допустимые границы индексирования элементов массива. При некорректном значении переменной i возвращается значение ошибки #Н/Д (команда УЗЛТОЧКИ = CVErr(xlErrNA)). Иначе результат функции определяется выражением УЗЛТОЧКИ = x(i-1).

На заметку

В массиве, который создается функцией Array(), индексация элементов начинается с нуля. Поэтому при обращении к i-му элементу через переменную x указывается индекс i-1.

Созданная нами функция для вычисления узловых точек представляет определенный интерес не только в контексте решения той задачи, которую мы здесь рассматриваем. Она вполне может использоваться и самостоятельно. Например, можно было бы модифицировать рассмотренный выше пример и вместо вычисления узловых точек с помощью надстройки **Поиск решения** вызвать функцию УЗЛТОЧКИ(). Желающие могут проделать такую процедуру самостоятельно.

Следующая функция, которая называется ЧЕБИНТЕГРАЛ() предназначена для вычисления интеграла от некоторой функции в интервале значений аргумента от –1 до 1. Программный код этой функции представлен в листинге 14.7.

Листинг 14.7. Функция для вычисления базового интеграла

```
Function ЧЕБИНТЕГРАЛ(F, Optional n = 9, Optional A = 0, _
 Optional B = 1, Optional C = 1)
 ' Переменная для вычисления интегральной суммы
 Dim s As Double
 ' Начальное значение интегральной суммы
  s = 0
 ' Объект для вызова метода, определяющего подынтегральную функцию
 Set obj = New MyFunctions
 ' Целочисленная индексная переменная
 Dim i As Integer
 ' Оператор цикла для вычисления интегральной суммы
 For i = 1 To n
  ' Добавляется очередное слагаемое - значение подынтегральной
  ' функции в узловой точке (с учетом сдвигов и масштабирования)
  s = s + C * CallByName(obj, F, VbMethod, A + B * УЗЛТОЧКИ(n, i))
 Next i
 ' Значение функции
 ЧЕБИНТЕГРАЛ = 2 * s / n
End Function
```

У функции довольно много аргументов (точнее, пять), но только один из них обязательный. Первый (обязательный) аргумент F является названием метода, через который реализована подынтегральная функция. Второй (необязательный) аргумент n определяет количество узловых точек, по которым вычисляется интегральная сумма. Еще три необязательных аргумента введены «с дальним прицелом» — они входят в выражение для интегральной суммы и предназначены для выполнения масштабирования подынтегральной функции и линейного преобразования ее аргумента. Для всех опционных аргументов указаны значения по умолчанию.

На заметку

Фактически, функцией ЧЕБИНТЕГРАЛ() вычисляется интегральная сумма для интеграла вида $C\int_{-1}^{1} F(A+Bx)dx$, где сама функция (ее имя) F, а также параметры A, B и C передаются аргументами функции ЧЕБИНТЕГРАЛ(). Обязательным является только аргумент, определяющий функцию F. Значения по умолчанию: $A = 0$, $B = 1$ и $C = 1$. Поэтому, если эти параметры при вызове функции явно не указать, то будет вычислен интеграл $\int_{-1}^{1} F(x)dx$.

Для вычисления интегральной суммы действительной переменной s присваивается начальное нулевое значение, после чего запускается оператор

цикла с индексной переменной i, пробегающей значения от 1 до n. За каждый цикл значение переменной s увеличивается на величину C*CallByName (obj,F,VbMethod,A+B*УЗЛТОЧКИ(n,i)). В данном случае метод с именем, определяемым переменной F, вызывается из объекта obj (создается командой Set obj = New MyFunctions, поэтому подынтегральные функции должны быть описаны как методы класса MyFunctions — такой подход мы уже рассматривали ранее). Аргументом методу передается выражение A+B*УЗЛТОЧКИ(n,i), в котором использован вызов функции вычисления узловой точки. Все это умножается на переменную C. После завершения выполнения оператора цикла командой ЧЕБИНТЕГРАЛ = 2*s/n определяется окончательное значение, возвращаемое функцией ЧЕБИНТЕГРАЛ().

Еще одна функция играет роль своеобразной «оболочки» для вызова функции ЧЕБИНТЕГРАЛ(), поскольку ее программный код состоит фактически из одной лишь команды, в которой вызывается функция ЧЕБИНТЕГРАЛ() с «правильно» указанными аргументами. Речь идет о функции НАЙТИИНТЕГРАЛ(), программный код которой представлен в листинге 14.8.

Листинг 14.8. Функция для вычисления интеграла в указанных пределах

```
Function НАЙТИИНТЕГРАЛ(F, a, b, Optional n = 9)
 ' Значение функции вычисляется вызовом функции ЧЕБИНТЕГРАЛ()
 ' с явно указанными значениями для аргументов,
 ' определяющих сдвиги и масштабирование при вычислении интеграла
 НАЙТИИНТЕГРАЛ = ЧЕБИНТЕГРАЛ(F, n, (a + b) / 2, _
 (b - a) / 2, (b - a) / 2)
End Function
```

У функции четыре аргумента:

- переменная F определяет имя метода, через который реализуется подынтегральная функция;
- переменная a определяет нижнюю границу области интегрирования;
- переменная b определяет верхнюю границу области интегрирования;
- переменная n определяет количество узловых точек, по которым вычисляется интегральная сумма (необязательный аргумент).

Значение функции вычисляется командой НАЙТИИНТЕГРАЛ = ЧЕБИНТЕГРАЛ (F,n,(a+b)/2,(b-a)/2,(b-a)/2). Этой командой, по большому счету, реализуется преобразование вида $\int_a^b f(x)dx = \frac{b-a}{2}\int_{-1}^{1} f\left(\frac{a+b}{2} + z\frac{b-a}{2}\right)dz$, которое мы обсуждали в начале раздела.

После этого осталось описать методы, определяющие подынтегральные функции, и весь созданный нами арсенал можно будет использовать в рабочем листе Excel.

На заметку

Все методы описываются в модуле класса с названием MyFunctions. Как в проект добавляется модуль класса, задается название класса и добавляется

программный код функции/метода, ранее уже описывалось, поэтому здесь детально останавливаться на этом вопросе не будем.

В листинге 14.9 приведен программный код, определяющий подынтегральную функцию $f(x) = x(1 - x)$.

Листинг 14.9. Первая подынтегральная функция

```
Function Fn1(x)
 ' Значение функции
  Fn1 = x * (1 - x)
End Function
```

В листинге 14.10 приведен программный код, определяющий подынтегральную функцию $f(x) = x\cos(\pi x)$.

Листинг 14.10. Вторая подынтегральная функция

```
Function Fn2(x)
 ' Значение функции
  Fn2 = x * Cos(WorksheetFunction.Pi * x)
End Function
```

В листинге 14.11 приведен программный код, определяющий подынтегральную функцию $f(x) = \frac{2}{\sqrt{\pi}}\exp(-x^2)$.

Листинг 14.11. Третья подынтегральная функция

```
Function Fn3(x)
 ' Значение функции
  Fn3 = 2 / Sqr(WorksheetFunction.Pi) * Exp(-x * x)
End Function
```

В листинге 14.12 приведен программный код, определяющий подынтегральную функцию $f(x) = x^2\exp(-x)$.

Листинг 14.12. Четвертая подынтегральная функция

```
Function Fn4(x)
 ' Значение функции
  Fn4 = x ^ 2 * Exp(-x)
End Function
```

Еще раз обращаем внимание, что все эти четыре функции должны быть описаны в модуле класса MyFunctions.

Пример использования функции НАЙТИИНТЕГРАЛ() для вычисления некоторых интегралов (собственно, тех же самых интегралов, что вычислялись ранее без привлечения программного кода) приведен в документе на рисунке 14.12.

Рис. 14.12
Вычисление интегралов по методу Чебышева
с использованием пользовательских функций

Значения интегралов вычисляются в ячейках F5, F8, F11 и F13 соответственно по формулам =НАЙТИИНТЕГРАЛ("Fn1";D5;E5), =НАЙТИИНТЕГРАЛ ("Fn2";D8;E8), =НАЙТИИНТЕГРАЛ("Fn3";D11;E11) и =НАЙТИИНТЕГРАЛ("Fn4"; D14;E14). Все эти формулы однотипны. Первым аргументом передается текстовое значение с именем подынтегральной функции, а второй и третий аргументы — ссылки на ячейки, содержащие значения границ области интегрирования. Поскольку четвертый аргумент явно не указан, то во всех случаях интегральная сумма вычисляется по девяти узловым точкам.

МЕТОД ГАУССА

Независимые умы
никогда не боялись банальностей.

Из к/ф «Покровские ворота»

В методе Гаусса для вычисления определенного интеграла используется интегральная сумма со значениями подынтегральной функции $f(x)$ в узловых точках x_1, x_2, ..., x_n с весовыми коэффициентами ω_1, ω_2, ..., ω_n, причем и узловые точки, и весовые коэффициенты подлежат определению путем «оптимизации» интегральной суммы. Под «оптимизацией» подразумевается некоторый критерий, который позволяет однозначно идентифицировать как узловые точки, так и весовые множители. Теперь конкретизируем постановку задачи.

В первую очередь отметим, что, не ограничивая общности, можем рассматривать интегралы в пределах от −1 до 1. Соображения в данном случае могут быть совершенно такими же, как и в случае, когда использовался метод Чебышева: если необходимо вычислить интеграл $\int_a^b f(x)dx$, то, сделав замену

переменной интегрирования $x = \frac{a+b}{2} + z\frac{b-a}{2}$ и введя новую функцию $F(z) =$ $= \frac{b-a}{2} f\left(\frac{a+b}{2} + z\frac{b-a}{2}\right)$, получим равенство $\int_a^b f(x)dx = \int_{-1}^{1} F(z)dz$.

Как и ранее, для вычисления интеграла в числовом виде используем соотношение вида $\int_{-1}^{1} f(x)dx = \sum_{k=1}^{n} \omega_k f(x_k)$, где, как отмечалось выше, узловые точки x_k и весовые множители ω_k нужно определить. При этом количество узловых точек n полагаем известным и фиксированным. Поэтому в общей сложности имеем $2n$ параметров (n узловых точек и n весовых множителей). Этих параметров достаточно для того, чтобы интеграл от любого полинома степени не выше чем $2n - 1$, вычисленный по квадратурной формуле $\int_{-1}^{1} f(x)dx = \sum_{k=1}^{n} \omega_k f(x_k)$, давал точное значение. Это и есть «отправная точка» для метода Гаусса.

Решение этой задачи может быть получено, если рассмотреть в качестве подынтегральной функции полином $T_{2n-1}(x)$ степени $2n - 1$, причем, не ограничивая общности, можем рассматривать его в виде $T_{2n-1}(x) = P_n(x)Q_{n-1}(x) + L_{n-1}(x)$, где $P_n(x) = A_n(x - x_1)(x - x_2)\ldots(x - x_n)$, A_n — коэффициент, а $Q_{n-1}(x)$ и $L_{n-1}(x)$ — полиномы степени не выше, чем $n - 1$. Задача состоит в том, чтобы найти такие узловые точки x_k и весовые коэффициенты ω_k, при которых равенство $\int_{-1}^{1} T_{2n-1}(x)dx = \sum_{k=1}^{n} \omega_k T_{2n-1}(x_k)$ было точным вне зависимости от того, как мы выбираем полиномы $Q_{n-1}(x)$ и $L_{n-1}(x)$.

Поскольку в силу определения $P_n(x_k) = 0$ для любого индекса $k = 1, 2, \ldots, n$, то $T_{2n-1}(x_k) = L_{n-1}(x_k)$. Тогда можем записать

$$\int_{-1}^{1} T_{2n-1}(x)dx = \int_{-1}^{1} P_n(x)Q_{n-1}(x)dx + \int_{-1}^{1} L_{n-1}(x)dx = \sum_{k=1}^{n} \omega_k L_{n-1}(x_k).$$

С другой стороны, в силу того, что $L_{n-1}(x)$ — полином, должно выполняться точное соотношение $\int_{-1}^{1} L_{n-1}(x)dx = \sum_{k=1}^{n} \omega_k L_{n-1}(x_k)$. Отсюда получаем соотношение $\int_{-1}^{1} P_n(x)Q_{n-1}(x)dx = 0$. Это соотношение должно выполняться для любого полинома степени не выше чем $n - 1$. Очевидно, что если оно выполняется для каждой из функций $1, x, x^2, \ldots, x^{n-1}$, то оно будет выполняться и для любого полинома степени $n - 1$. В результате мы получаем следующие соотношения для определения узловых точек: $\int_{-1}^{1} (x - x_1)(x - x_2)\ldots(x - x_n)x^k dx = 0$ для индексов $k = 0, 1, \ldots, n - 1$. Из этих соотношений узловые точки вычисляются однозначно. После того как найдены узловые точки, можем найти весовые коэффициенты. Исходным для поиска весовых коэффициентов мо-

жет стать соотношение $\int_{-1}^{1} L_{n-1}(x)dx = \sum_{k=1}^{n} \omega_k L_{n-1}(x_k)$. Если представить полином $L_{n-1}(x)$ в форме полинома Лагранжа:

$$L_{n-1}(x) = \sum_{k=1}^{n} L_{n-1}(x_k)\varphi_k(x) = \sum_{k=1}^{n} T_{2n-1}(x_k)\varphi_k(x)$$

с функциями

$$\varphi_k(x) = \frac{(x-x_1)(x-x_2)\ldots(x-x_{k-1})(x-x_{k+1})\ldots(x-x_n)}{(x_k-x_1)(x_k-x_2)\ldots(x_k-x_{k-1})(x_k-x_{k+1})\ldots(x_k-x_n)},$$

то легко получим $\omega_k = \int_{-1}^{1} \varphi_k(x)dx$ (индекс $k = 1, 2, \ldots, n$).

Приведенные соотношения для определения узловых точек и весовых коэффициентов хотя и выглядят внешне просто и элегантно, для использования на практике достаточно сложны. Поэтому обычно идут несколько иным путем.

На заметку

После того как узловые точки определены, вычисление весовых коэффициентов представляет собой задачу трудоемкую, но понятную. С вычислением же узловых точек дело обстоит несколько хуже, поскольку определяются они как корни нелинейной системы алгебраических уравнений.

Важно отметить одно обстоятельство, касающееся полинома $P_n(x)$: хотя мы знаем, что это полином степени n, пока не найдены узловые точки x_1, x_2, …, x_n, входящие в выражение для этого полинома, полином собственно неизвестен. Другими словами, вычисляя узловые точки, определяем структуру полинома. После несложных преобразований, суть которых для нас в данном случае малоинтересна, можно показать, что полином $P_n(x)$ на самом деле является полиномом Лежандра. Для этих полиномов справедливо представление $P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}(x^2-1)^n$. У полиномов много интересных свойств. Для нас же важно то, что узловые точки — это нули полиномов Лежандра. Вычислять их от этого становится ничуть не легче, зато появляется надежда (надо сказать, небезосновательная) воспользоваться результатами исследования этих полиномов. По крайней мере, для не очень больших значений n составлены таблицы корней полиномов. Если узловые точки x_k (индекс $k = 1, 2, \ldots, n$), в которых $P_n(x_k) = 0$, вычислены, то весовые коэффициенты могут быть вычислены по формуле $\omega_k = \frac{2}{(1-x_k^2)(P_n'(x_k))^2}$. Вместо использования производной от полинома Лежандра можем воспользоваться рекуррентным соотношением $(1-x_k^2)P_n'(x_k) = -(n+1)P_{n+1}(x_k)$, что в результате позволяет записать $\omega_k = \frac{2(1-x_k^2)}{(n+1)^2 P_{n+1}(x_k)^2}$. Это же выражение с учетом еще одного рекуррентного соотношения $(n+1)P_{n+1}(x_k) + nP_{n-1}(x_k)$ может быть записано несколько иначе: $\omega_k = \frac{2(1-x_k^2)}{n^2 P_{n-1}(x_k)^2}$. Последнее соотношение более предпочтительно по

сравнению с предыдущим, поскольку не требует вычисления полинома более высокого порядка, чем n.

Понятно, что даже с учетом приведенных соотношений процедура вычисления узловых точек и весовых коэффициентов достаточно сложна. Поэтому обычно самым разумным и простым подходом является использование таблиц: например, в таблице 14.1 представлены значения для узловых точек и весовых коэффициентов для некоторых начальных значений параметра n (в пределах первого десятка).

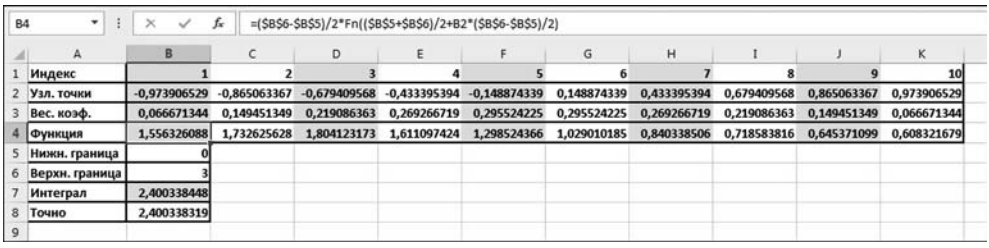
На заметку

Несложно заметить, что узловые точки и весовые коэффициенты обладают некоторой симметрией. Так, если среди узловых точек имеется точка x_k, то среди узловых точек должна быть точка $-x_k$. Весовой коэффициент для двух этих точек один и тот же.

В некоторых случаях также может быть полезным асимптотическое представление для полиномов Лежандра $P_n(\cos(\theta)) = \sqrt{\frac{2}{\pi n}} \frac{\cos\left(\left(n+\frac{1}{2}\right)\theta - \frac{\pi}{4}\right)}{\sqrt{\sin(\theta)}}$, используемое при больших n. Оно позволяет оценить корни полинома и значение полинома — правда, с не очень высокой точностью (порядок точности дается выражением $n^{-3/2}$). Так, оценку для узловых точек можем получить из равенства $\cos\left(\left(n+\frac{1}{2}\right)\theta - \frac{\pi}{4}\right) = 0$, что дает $\left(n+\frac{1}{2}\right)\theta - \frac{\pi}{4} = \frac{\pi}{2} + \pi k$ и, соответственно, $\theta_k = \frac{(3+4k)\pi}{2(2n+1)}$ ($k = 0, \pm1, \pm2, \ldots, \pm n$). Узловые точки вычисляются как $x_k = \cos(\theta_k)$. Для примера: приведенные соотношения позволяют вычислить узловые точки при $n = 10$ с точностью до двух знаков после запятой, что весьма скромно.

Проиллюстрируем некоторые возможности приложения Excel при вычислении интегралов с помощью квадратурных формул Гаусса. Обратимся к документу, представленному на рисунке 14.13.

Здесь мы с помощью квадратурных формул Гаусса вычисляем интеграл $\int_a^b \frac{(x+1)dx}{1+x^2}$. Для этого в ячейки B2:K2 вводятся значения десяти узловых точек, а в ячейки B3:K3 вводятся весовые коэффициенты квадратурной формулы Гаусса. Что касается подынтегральной функции $f(x) = \frac{x+1}{1+x^2}$, то мы создаем функцию пользователя, которая вычисляет соответствующее значение. Программный код этой функции представлен в листинге 14.13.



B4 | =($B$6-$B$5)/2*Fn(($B$5+$B$6)/2+B2*($B$6-$B$5)/2)

	A	B	C	D	E	F	G	H	I	J	K
1	Индекс	1	2	3	4	5	6	7	8	9	10
2	Узл. точки	-0,973906529	-0,865063367	-0,679409568	-0,433395394	-0,148874339	0,148874339	0,433395394	0,679409568	0,865063367	0,973906529
3	Вес. коэф.	0,066671344	0,149451349	0,219086363	0,269266719	0,295524225	0,295524225	0,269266719	0,219086363	0,149451349	0,066671344
4	Функция	1,556326088	1,732625628	1,804123173	1,611097424	1,298524366	1,029010185	0,840338506	0,718583816	0,645371099	0,608321679
5	Нижн. граница	0									
6	Верхн. граница	3									
7	Интеграл	2,400338448									
8	Точно	2,400338319									
9											

Рис. 14.13
Вычисление интеграла с помощью квадратурных формул Гаусса

Таблица 14.1

Параметры для квадратурных формул Гаусса

Количество узловых точек	Узловые точки	Весовые коэффициенты
2	-0,5773502692 0,5773502692	1 1
3	-0,7745966692 0 0,7745966692	0,5555555555 0,8888888888 0,5555555555
4	-0,8611363116 -0,3399810436 0,3399810436 0,8611363116	0,3478548451 0,6521451549 0,6521451549 0,3478548451
5	-0,9061798459 -0,5384693101 0 0,5384693101 0,9061798459	0,2362688506 0,4786286705 0,5688888888 0,4786286705 0,2362688506
6	-0,9324695142 -0,6612093865 -0,2386191861 0,2386191861 0,6612093865 0,9324695142	0,1713244924 0,3607615731 0,4679139346 0,4679139346 0,3607615731 0,1713244924
7	-0,9491079123 -0,7415311856 -0,4058451514 0 0,4058451514 0,7415311856 0,9491079123	0,1294849662 0,2797053915 0,3818300505 0,4179591837 0,3818300505 0,2797053915 0,1294849662
8	-0,9602898565 -0,7966664774 -0,5255324099 -0,1834346422 0,1834346422 0,5255324099 0,7966664774 0,9602898565	0,1012285363 0,222381035 0,3137066459 0,3626837834 0,3626837834 0,3137066459 0,222381035 0,1012285363
9	-0,9681602395 -0,8360311073 -0,6133714327 -0,3242534234 0 0,3242534234 0,6133714327 0,8360311073 0,9681602395	0,1806481607 0,2606106964 0,3123470770 0,3302393550 0,3302393550 0,3123470770 0,2606106964 0,1806481607 0,08127438836
10	-0,9739065285 -0,8650633667 -0,6794095683 -0,4333953941 -0,1488743390 0,1488743390 0,4333953941 0,6794095683 0,8650633667 0,9739065285	0,0666713443 0,1494513491 0,2190863625 0,2692667193 0,2955242247 0,2955242247 0,2692667193 0,2190863625 0,1494513491 0,0666713443

Листинг 14.13. Подынтегральная функция

```
Function Fn(x) As Double
  ' Значение функции
  Fn = (x + 1) / (1 + x ^ 2)
End Function
```

На заметку

Программный код функции вводится в обычном модуле (не в модуле класса). Обычный модуль добавляется в проект так: в редакторе VBA выбираем команду **Insert ▷ Module**.

Ячейки B4:K4 заполняются значениями подынтегральной функции $F(x)=\frac{b-a}{2}f\left(\frac{a+b}{2}+x\frac{b-a}{2}\right)$ в узловых точках. Для этого в ячейку B4 вводится формула =($B$6-$B$5)/2*Fn(($B$5+$B$6)/2+B2*($B$6-$B$5)/2) и затем копируется в прочие ячейки диапазона B4:K4. Копируемая формула содержит абсолютные ссылки на ячейки B5 и B6 со значениями границ (нижней и верхней соответственно) области интегрирования, а также относительную ссылку B2 на узловую точку, для которой вычисляется значение подынтегральной функции.

Для вычисления интеграла по методу квадратурных формул Гаусса в ячейку B7 вводим формулу =СУММПРОИЗВ(B4:K4;B3:K3). Этой формулой вычисляется сумма поэлементных произведений диапазонов B4:K4 и B3:K3. Результатом является значение интегральной суммы. Точное значение для интеграла вычисляется в ячейке B8 по формуле =1/2*LN((1+B6^2)/(1+B5^2))+ATAN(B6)-ATAN(B5). Здесь мы воспользовались тем, что $\int\frac{(x+1)dx}{1+x^2}=\frac{1}{2}\ln(1+x^2)+\operatorname{arctg}(x)$, поэтому по формуле Ньютона — Лейбница имеем $\int_a^b\frac{(x+1)dx}{1+x^2}=\frac{1}{2}\ln\left(\frac{1+b^2}{1+a^2}\right)+$ $+\operatorname{arctg}(b)-\operatorname{arctg}(a)$. Этим соотношением мы воспользовались для вычисления «точного» значения для интеграла.

На заметку

При изменении значений в ячейках B5 и B6 для границ области интегрирования значение интеграла (приближенное и «точное») пересчитывается автоматически.

ВЫЧИСЛЕНИЕ НЕСОБСТВЕННЫХ ИНТЕГРАЛОВ

Чего не надо, того не сделают.

Из к/ф «Покровские ворота»

Задача о вычислении несобственных интегралов возникает в том случае, если в пределах области интегрирования подынтегральная функция имеет особенности, или, например, если интервал интегрирования бесконечный (полубесконечный). Общий подход состоит во введении весового множителя

в подынтегральное выражение: вместо интеграла $\int_a^b F(x)dx$ рассматривают интеграл $\int_a^b p(x)f(x)dx$, где весовая функция $p(x) > 0$ на всей области интегрирования, а функция $f(x)$ ограничена на области интегрирования и обладает достаточным количеством производных.

Для вычисления интегралов вида $\int_a^b p(x)f(x)dx$ используются квадратурные формулы вида $\int_a^b p(x)f(x)dx = \sum_{k=1}^{n} \omega_k f(x_k)$. Особенность ситуации в том, что весовые коэффициенты ω_k не зависят от функции $f(x)$ и определяются областью интегрирования и весовой функцией $p(x)$.

На заметку

Как и в предыдущих случаях, интеграл по конечной области путем сдвигов и масштабирования может быть сведен к интегралу в пределах от -1 до 1. Если речь идет о полубесконечном интервале интегрирования, то без ограничения общности можем рассматривать интегралы с пределами интегрирования от 0 до бесконечности. Этими случаями мы и ограничимся.

Далее мы рассмотрим интегралы трех видов: $\int_{-1}^{1} \frac{f(x)}{\sqrt{1-x^2}}dx$, $\int_0^{\infty} \exp(-x)f(x)dx$ и $\int_{-\infty}^{\infty} \exp(-x^2)f(x)dx$, соответственно с весовыми функциями $p(x) = \frac{1}{\sqrt{1-x^2}}$, $p(x) = \exp(-x)$ и $p(x) = \exp(-x^2)$. Ограничимся констатацией тех правил и алгоритмов, по которым вычисляются квадратурные формулы.

При вычислении интегралов вида $\int_{-1}^{1} \frac{f(x)}{\sqrt{1-x^2}}dx = \sum_{k=1}^{n} \omega_k f(x_k)$ весовые коэффициенты $\omega_k = \pi/n$, а узловые точки вычисляются по формулам $x_k = \cos\left(\frac{2k-1}{2n}\pi\right)$ (индекс $k = 1, 2, \ldots, n$). Соответствующие формулы называются *формулами численного интегрирования Эрмита*.

При вычислении интегралов вида $\int_0^{\infty} \exp(-x)f(x)dx = \sum_{k=1}^{n} \omega_k f(x_k)$ узловые точки x_k вычисляются как корни полинома Лагерра $L_n(x_k) = 0$. Полиномы Лагерра вычисляются как $L_n(x) = (-1)^n \exp(x)\frac{d^n}{dx^n}\left(x^n \exp(-x)\right)$. Весовые коэффициенты квадратурной формулы при этом вычисляются как $\omega_k = \frac{(n!)^2}{x_k (L_n'(x_k))^2}$. В таблице 14.2 приведены узловые точки (нули полиномов Лагерра) и весовые коэффициенты для *квадратурных формул Чебышева — Лагерра*.

При вычислении интегралов по бесконечному интервалу нередко сталкиваются с интегралами вида $\int_{-\infty}^{\infty} \exp(-x^2)f(x)dx$. Для вычисления такого

Таблица 14.2

Квадратурные формулы Чебышева — Лагерра

Количество узловых точек	Узловые точки	Весовые коэффициенты
1	1	1
2	0,5857864376 3,4142135624	0,8535533906 0,1464466094
3	0,4157745567 2,2942803603 6,2899450829	0,7110930099 0,2785177336 0,0103892565
4	0,3225476896 1,7457611011 4,5366202969 9,3950709123	0,6031541043 0,3574186924 0,03888790851 0,0005392947056
5	0,2635603197 1,4134030591 3,5964257710 7,0858100059 12,6408008443	0,5217556106 0,3986668111 0,07594244968 0,003611758679 0,00002336997239
6	0,2228466042 1,1889321017 2,9927363261 5,7751435691 9,8374674184 15,9828739806	0,4589646740 0,4170008308 0,1133733821 0,01039919745 0,0002610172028 0,0000008985479064
7	0,1930436766 1,0266648953 2,5678767450 4,90003530845 8,1821534446 12,7341802918 19,3957278623	0,4093189517 0,4218312779 0,1471263487 0,02063351447 0,001074010143 0,00001586546435 0,00000003170315479
8	0,1702796323 0,9037017768 2,2510866299 4,2667001703 7,0459054024 10,7585160102 15,7406786413 22,8631317369	0,3691885893 0,4187867808 0,1757949866 0,03334349226 0,002794536235 0,00009076508773 0,0000008485746716 0,000000001048001175
9	0,1523222277 0,8072200227 2,0051351556 3,7834739733 6,2049567778 9,3729852517 13,4662369110 18,8335977889 26,3740718909	0,3361264218 0,4112139804 0,1992875254 0,04746056277 0,005599626611 0,0003052497671 0,000006592123026 0,00000004110769330 0,0000000003290874030
10	0,1377934705 0,7294545495 1,8083429017 3,4014336979 5,5524961400 8,3301527468 11,8437858379 16,2792578314 21,9965858120 29,9206970122	0,3084411158 0,4011199292 0,2180682876 0,06208745610 0,009501516975 0,0007530083886 0,00002825923350 0,0000004249313985 0,00000001839564824 0,000000000009911827220

Таблица 14.3

Квадратурные формулы Чебышева — Эрмита

Количество узловых точек	Узловые точки	Весовые коэффициенты
1	0	1,772453851
2	-0,7071067812 0,7071067812	8,8862269255 8,8862269255
3	-1,2247448714 0 1,2247448714	0,2954089752 1,181635901 0,2954089752
4	-1,6506801239 -0,5246476233 0,5246476233 1,6506801239	0,08131283545 0,8049140900 0,8049140900 0,08131283545
5	-2,0201828705 -0,9585724646 0 0,9585724646 2,0201828705	0,01995324206 0,3936193232 0,9453087205 0,3936193232 0,01995324206
6	-2,3506049737 -1,3358490740 -0,4360774119 0,4360774119 1,3358490740 2,3506049737	0,004530009906 0,1570673203 0,7246295952 0,7246295952 0,1570673203 0,004530009906
7	-2,6519613568 -1,6735516288 -0,8162878829 0 0,8162878829 1,6735516288 2,6519613568	0,0009717812451 0,05451558282 0,4256072526 0,8102646176 0,4256072526 0,05451558282 0,0009717812451
8	-2,9306374203 -1,9816567567 -1,1571937124 -0,3811869902 0,3811869902 1,1571937124 1,9816567567 2,9306374203	0,0001996040722 0,01707798301 0,2078023258 0,6611470126 0,6611470126 0,2078023258 0,01707798301 0,0001996040722
9	-3,1909932018 -2,2665805845 -1,4685532892 -0,7235510188 0 0,7235510188 1,4685532892 2,2665805845 3,1909932018	0,00003960697726 0,004943624276 0,08847452739 0,4326515590 0,7202352156 0,4326515590 0,08847452739 0,004943624276 0,00003960697726
10	-3,4361591188 -2,5327316742 -1,7566836493 -1,0366108298 -0,3429013272 0,3429013272 1,0366108298 1,7566836493 2,5327316742 3,4361591188	0,000007640432855 0,001343645747 0,03387439446 0,2401386111 0,6108626337 0,6108626337 0,2401386111 0,03387439446 0,001343645747 0,000007640432855

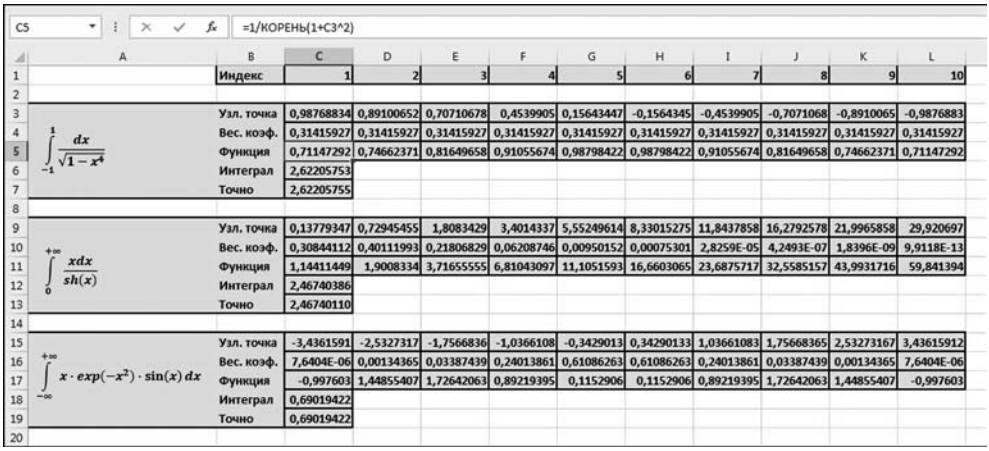
интеграла используют квадратурную формулу $\int_{-\infty}^{\infty} \exp(-x^2) f(x) dx = \sum_{k=1}^{n} \omega_k f(x_k)$, узловые точки x_k в которой определяются как нули полиномов Эрмита $H_n(x_k) = 0$. Полиномы Эрмита, в свою очередь, могут быть определены как $H_n(x) = (-1)^n \exp(x^2) \frac{d^n}{dx^n}\left(\exp(-x^2)\right)$. Весовые коэффициенты определяются соотношением $\omega_k = \frac{2^{n+1} n! \sqrt{\pi}}{(H_n'(x_k))^2}$. В таблице 14.3 приведены узловые точки и весовые коэффициенты для *квадратурных формул Чебышева — Эрмита*.

На заметку

Узловые точки квадратурных формул Чебышева — Эрмита расположены симметрично относительно нулевой точки: если среди узловых точек есть x_k, то имеется и точка $-x_k$. Весовые коэффициенты для обеих точек одинаковы.

Далее рассмотрим небольшие примеры по вычислению несобственных интегралов средствами Excel. Соответствующий документ показан на рисунке 14.14.

В качестве примеров мы рассмотрели интеграл $\int_{-1}^{1} \frac{dx}{\sqrt{1-x^4}} = \frac{2\sqrt{\pi}\Gamma\left(\frac{5}{4}\right)}{\Gamma\left(\frac{3}{4}\right)} \approx$ $\approx 2{,}622057554$, интеграл $\int_{-\infty}^{+\infty} \frac{x}{\mathrm{sh}(x)} dx = \frac{\pi^2}{4} \approx 2{,}467401100$, а также интеграл $\int_{-\infty}^{+\infty} x \exp(-x^2) \sin(x) dx = \frac{\sqrt{\pi}}{2} \exp\left(-\frac{1}{4}\right) \approx 0{,}6901942235$. Во всех трех случаях вычисления производились по десяти узловым точкам. Для первого интеграла эти точки вычислялись, для двух других — вводились на основе табличных значений. Это же относится и к весовым коэффициентам квадратурных фор-



	A	B	C	D	E	F	G	H	I	J	K	L
1		Индекс	1	2	3	4	5	6	7	8	9	10
2												
3	$\int_{-1}^{1} \frac{dx}{\sqrt{1-x^4}}$	Узл. точка	0,98768834	0,89100652	0,70710678	0,4539905	0,15643447	-0,1564345	-0,4539905	-0,7071068	-0,8910065	-0,9876883
4		Вес. коэф.	0,31415927	0,31415927	0,31415927	0,31415927	0,31415927	0,31415927	0,31415927	0,31415927	0,31415927	0,31415927
5		Функция	0,71147292	0,74662371	0,81649658	0,91055674	0,98798422	0,98798422	0,91055674	0,81649658	0,74662371	0,71147292
6		Интеграл	2,62205753									
7		Точно	2,62205755									
8												
9	$\int_{0}^{+\infty} \frac{xdx}{sh(x)}$	Узл. точка	0,13779347	0,72945455	1,8083429	3,4014337	5,55249614	8,33015275	11,8437858	16,2792578	21,9965858	29,920697
10		Вес. коэф.	0,30844112	0,40111993	0,21806829	0,06208746	0,00950152	0,00075301	2,8259E-05	4,2493E-07	1,8396E-09	9,9118E-13
11		Функция	1,14411449	1,9008334	3,71655555	6,81043097	11,1051593	16,6603065	23,6875717	32,5585157	43,9931716	59,841394
12		Интеграл	2,46740386									
13		Точно	2,46740110									
14												
15	$\int_{-\infty}^{+\infty} x \cdot exp(-x^2) \cdot \sin(x)\, dx$	Узл. точка	-3,4361591	-2,5327317	-1,7566836	-1,0366108	-0,3429013	0,34290133	1,03661083	1,75668365	2,53273167	3,43615912
16		Вес. коэф.	7,6404E-06	0,00134365	0,03387439	0,24013861	0,61086263	0,61086263	0,24013861	0,03387439	0,00134365	7,6404E-06
17		Функция	-0,997603	1,44855407	1,72642063	0,89219395	0,1152906	0,1152906	0,89219395	1,72642063	1,44855407	-0,997603
18		Интеграл	0,69019422									
19		Точно	0,69019422									
20												

Рис. 14.14
Вычисление несобственных интегралов

мул. Далее, для каждой узловой точки вычислялись значения подынтегральной функции, и затем вычислялась интегральная сумма.

Для вычисления интеграла $\int_{-1}^{1}\frac{dx}{\sqrt{1-x^4}}$ следует учесть, что $\frac{1}{\sqrt{1-x^4}}=$ $=\frac{1}{\sqrt{1-x^2}}\cdot\frac{1}{\sqrt{1+x^2}}$, т. е. речь идет о вычислении интеграла вида $\int_{-1}^{1}\frac{f(x)}{\sqrt{1-x^2}}dx$ с функцией $f(x)=\frac{1}{\sqrt{1+x^2}}$. Значение этой функции вычисляется в узловых точках. Сами узловые точки вычисляются в диапазоне ячеек C3:L3. Для этого в ячейку C3 вводится формула =COS((2*C1-1)/2/$L$1*ПИ()) и копируется во все остальные ячейки диапазона C3:L3.

Весовые коэффициенты вычисляются в диапазоне C4:L4 копированием формулы =ПИ()/$L$1 из ячейки C4. Наконец, для вычисления значения подынтегральной функции в узловых точках в ячейку C5 вводится формула =1/КОРЕНЬ(1+C3^2), на основе которой заполняется диапазон ячеек C5:L5. После этого для вычисления интеграла в ячейку C6 вводится формула =СУММПРОИЗВ(C4:L4;C5:L5). Формулой вычисляется сумма произведений весовых коэффициентов и значений подынтегральной функции в соответствующих узловых точках. Для сравнения в ячейке C7 по формуле =КОРЕНЬПИ(4)*EXP(ГАММАНЛОГ(5/4))/EXP(ГАММАНЛОГ(3/4)) вычисляется значение интеграла, получаемое из аналитического выражения для интеграла.

На заметку

В формуле, описанной выше, коэффициент $2\sqrt{\pi}$ вычисляется инструкцией КОРЕНЬПИ(4) (значение $\sqrt{4\pi}=2\sqrt{\pi}$). Гамма-функции $\Gamma\left(\frac{5}{4}\right)$ и $\Gamma\left(\frac{3}{4}\right)$ вычисляются инструкциями EXP(ГАММАНЛОГ(5/4)) и EXP(ГАММАНЛОГ(3/4)) соответственно. Здесь мы воспользовались встроенной функцией ГАММАНЛОГ(), которая в качестве значения возвращает натуральный логарифм от гамма-функции с соответствующим аргументом.

Интеграл $\int_{-\infty}^{+\infty}\frac{x}{\mathrm{sh}(x)}dx=\int_{-\infty}^{+\infty}\exp(-x)\frac{x\cdot\exp(x)}{\mathrm{sh}(x)}dx$ (т. е. функция $f(x)=\frac{x\cdot\exp(x)}{\mathrm{sh}(x)}$) вычисляются следующим образом. В ячейки C9:L9 вводятся значения узловых точек, а в ячейки C10:L10 — значения весовых коэффициентов. Это значения, представленные в таблице 14.2. Значения подынтегральной функции в узловых точках вычисляется копированием формулы =C9*EXP(C9)/SINH(C9) из ячейки C11 в ячейки диапазона C11:L11. Интеграл вычисляется в ячейке C12 по формуле =СУММПРОИЗВ(C10:L10;C11:L11). «Контрольное» значение интеграла вычисляется в ячейке C13 по формуле =ПИ()^2/4.

Аналогично вычисляется и интеграл $\int_{-\infty}^{+\infty}x\exp(-x^2)\sin(x)dx$ (функция $f(x)=x\sin x$): в ячейки C15:L15 вводятся узловые точки, в ячейки C16:L16 вводятся значения весовых коэффициентов (табл. 14.3), а значения подынтегральной функции в узловых точках вычисляем в диапазоне C16:L16 копированием из ячейки C16 формулы =C15*SIN(C15).

Интеграл вычисляем в ячейке C17 с помощью формулы =СУММПРОИЗВ(C16:L16;C17:L17), а в ячейке C18 значение вычисляется по формуле =КОРЕНЬПИ(1)/2*EXP(-1/4) (значение для сравнения).

Как несложно заметить, для всех трех случаев вычисленные на основе квадратурных формул значения для интегралов достаточно неплохо согласуются со значениями, вычисленными на основе аналитических выражений для интегралов.

ВЫЧИСЛЕНИЕ ПОВТОРНЫХ ИНТЕГРАЛОВ

История, леденящая кровь.
Под маской овцы скрывался лев!

Из к/ф «Покровские ворота»

Важной с практической точки зрения является задача вычисления повторного интеграла. Задача может быть сразу сформулирована как вычисление повторного интеграла, или может сводиться к вычислению повторного интеграла — например, при вычислении кратного (двойного или тройного) интеграла. В этом разделе мы кратко рассмотрим подходы, которые могут быть применены для вычисления повторных интегралов средствами приложения Excel. А именно, остановимся на вычислении повторных интегралов вида $\int_a^b dx \int_{c(x)}^{d(x)} f(x, y)dy$. Пример подобной задачи — вычисление массы тонкой пластинки, плотность которой задается функцией $f(x, y)$, а форма (область пластинки) определяется соотношениями $a \le x \le b$ и $c(x) \le y \le d(x)$, где $c(x)$ и $d(x)$ — известные функции.

Если ввести в рассмотрение функцию $F(x) = \int_{c(x)}^{d(x)} f(x,y)dy$, то задача сводится к вычислению интеграла $\int_a^b F(x)dx$. Воспользуемся для вычисления такого интеграла методом трапеций. Это позволяет записать:

$$\int_a^b F(x)dx = \Delta x\left(\frac{F(x_0)+F(x_n)}{2} + \sum_{k=1}^{n-1} F(x_k)\right),$$

где введены обозначения $\Delta x = \frac{b-a}{n}$ и $x_k = a + k\Delta x$ ($k = 0, 1, \ldots, n$). Вместе с тем, можем записать $F(x_k) = \int_{c(x_k)}^{d(x_k)} f(x_k,y)dy$. Воспользовавшись для таких интегралов формулой трапеций, получаем:

$$F(x_k) = \int_{c(x_k)}^{d(x_k)} f(x_k,y)dy = \Delta y_k\left(\frac{f(x_k,y_{k,0})+f(x_k,y_{k,m_k})}{2} + \sum_{p=1}^{m_k-1} f(x_k,y_{k,p})\right),$$

где $\Delta y_k = \frac{d(x_k) - c(x_k)}{m_k}$ и $y_{k,p} = c(x_k) + p\Delta y_k$. Эти соотношения позволяют нам свести вычисление повторного интеграла к вычислению двойной суммы. Для вычисления такой суммы удобно воспользоваться программными возможностями VBA. В листинге 14.14 представлен программный код функции Integrate(), предназначенной для вычисления интеграла по одной переменной от функции, зависящей от двух переменных.

Листинг 14.14. Функция для вычисления внутреннего интеграла

```
Private Function Integrate(F As String, c As String, d As String, _
 x As Double, dx As Double) As Double
 ' Целочисленная индексная переменная
 Dim i As Integer
 ' Создание объекта для вызова метода,
 ' определяющего подынтегральную функцию
 Set obj = New MultIntegrals
 ' Переменные для записи границ интегрирования
 Dim a As Double, b As Double
 ' Значение нижней границы интервала интегрирования
 a = CallByName(obj, c, VbMethod, x)
 ' Значение верхней границы интервала интегрирования
 b = CallByName(obj, d, VbMethod, x)
 ' Переменная для количества частей,
 ' на которые разбивается область интегрирования
 Dim m As Integer
 ' Количество частей, на которые разбивается область
 ' интегрирования
 m = Fix((b - a) / dx) + 1
 ' Переменная для шага приращения по переменной интегрирования
 Dim dy As Double
 ' Значение шага приращения по переменной интегрирования
 dy = (b - a) / m
 ' Переменная для вычисления интегральной суммы
 Dim s As Double
 ' Начальное значение для интегральной суммы
 s = (CallByName(obj, F, VbMethod, x, a) + _
 CallByName(obj, F, VbMethod, x, b)) / 2
 ' Оператор цикла для вычисления интегральной суммы
 For i = 1 To m - 1
  ' Добавляем очередное слагаемое к сумме
  s = s + CallByName(obj, F, VbMethod, x, a + dy * i)
 Next i
 ' Значение функции
 Integrate = s * dy
End Function
```

У функции пять аргументов:

- текстовый аргумент F определяет название метода, через который реализуется подынтегральная функция (предполагается, что эта функция зависит от двух аргументов x и y);
- текстовый аргумент c определяет имя функции, задающей нижнюю границу интегрирования по второй переменной y, зависящее от первой переменной x;
- текстовый аргумент d определяет имя функции, задающей верхнюю границу интегрирования по второй переменной y, зависящее от первой переменной x;
- действительная числовая переменная x, определяющая фиксированное значение первой переменной x, при которой вычисляется интеграл по второй переменной y;
- действительная числовая переменная dx определяет максимальный шаг приращения для переменной интегрирования.

Предполагается, что функция, определяющая подынтегральное выражение, и функции, определяющие границы интегрирования по переменной y, описываются в модуле класса с названием MultIntegrals. Командой Set obj = New MultIntegrals создается объект соответствующего класса. Для удобства мы вводим две действительные числовые переменные a и b, которым командами a = CallByName(obj,c,VbMethod,x) и b = CallByName(obj,d,VbMethod,x) присваиваем значения. Эти значения — границы интегрирования по переменной y при фиксированном значении переменной x. Далее интервал интегрирования следует разбить на части. Количество частей, на которые разбивается интервал интегрирования, вычисляется на основе значения пятого аргумента dx функции Integrate().

На заметку

Некоторая проблема связана с выбором количества частей, на которые разбивается интервал интегрирования (по переменной y). Если зафиксировать количество узловых точек (количество частей), то, в силу того, что с интервал «плавающий» (при изменении значения переменной x меняется интервал интегрирования), фиксация количества узловых точек приводит к тому, что изменяется шаг приращения по переменной y, что не всегда разумно. Поэтому мы отталкиваемся от того, какой шаг интегрирования будет по переменной x. Предполагается, что при интегрировании по переменной x шаг приращения будет определяться значением аргумента dx. Вместе с тем не факт, что на интервале интегрирования поместится целое число частей «длиной» dx. Поэтому количество частей, на которые разбивается интервал интегрирования по переменной y, определяется из того условия, чтобы шаг приращения по переменной y был близок к шагу приращения по переменной x.

Количество частей, на которые разбивается область интегрирования, определяется командой m = Fix((b-a)/dx)+1. Суть ее в следующем: частное (b-a)/dx, определяющее количество интервалов «длиной» dx, помещающихся в интервале интегрирования «длиной» (b-a). Это значение округляется до целочисленного отбрасыванием дробной части (за эту «работу» ответственна функция Fix()), и к полученному результату добавляется единица.

На заметку

При таком подходе, если случайно оказалось, что отношение (b-a)/dx есть число целое, то увеличивать это значение на единицу, в принципе, не надо. При желании читатель может обработать эту ситуацию более корректно, чем предлагается здесь.

Значение переменной dy для шага приращения по переменной интегрирования *y* определяется командой dy = (b-a)/m.

Интегральную сумму (с точностью до множителя) будем записывать в переменную s, начальное значение которой определяется как s = (CallByName (obj,F,VbMethod,x,a)+CallByName(obj,F,VbMethod,x,b))/2, после чего запускается оператор цикла, и при изменении индексной переменной i в пределах от 1 до m-1 сумма вычисляется с помощью команды s = s+CallByName (obj,F,VbMethod,x,a+dy*i) за каждый цикл.

На заметку

В инструкции CallByName(obj,F,VbMethod,x,a+dy*i), которая определяет значение добавки к интегральной сумме, четвертым и пятым аргументами передаются аргументы x и a+dy*i для функции F (второй аргумент).

По завершении оператора цикла переменная s умножается на переменную dy, и это значение возвращается как результат функции (команда Integrate = s*dy).

Описанная выше функция Integrate() описана с ключевым словом Private, поэтому в рабочем листе она недоступна. Мы воспользуемся этой функцией в другой функции, которая называется ПОВТИНТЕГРАЛ(). Программный код этой функции представлен в листинге 14.15.

Листинг 14.15. Функция для вычисления повторного интеграла

```
Function ПОВТИНТЕГРАЛ(F As String, c As String, d As String, _
 a1 As Variant, b1 As Variant, Optional n1 As Variant = 500) As Double
 ‘ Переменная для записи нижней границы интервала интегрирования
 Dim a As Double
 ‘ Значение нижней границы интервала интегрирования
  a = a1
 ‘ Переменная для записи верхней границы интервала интегрирования
 Dim b As Double
 ‘ Значение верхней границы интервала интегрирования
  b = b1
 ‘ Переменная для записи количества частей,
 ‘ на которые разбивается интервал интегрирования
 Dim n As Integer
 ‘ Количество частей, на которые разбивается интервал интегрирования
  n = n1
 ‘ Целочисленная индексная переменная
 Dim i As Integer
  ‘ Переменная для записи шага приращения по переменной интегрирования
```

```
    Dim dx As Double
    ' Значение шага приращения по переменной интегрирования
    dx = (b - a) / n
    ' Переменная для записи интегральной суммы
    Dim s As Double
    ' Начальное значение для интегральной суммы
    s = (Integrate(F, c, d, a, dx) + Integrate(F, c, d, b, dx)) / 2
    ' Оператор цикла для вычисления интегральной суммы
    For i = 1 To n - 1
      ' Добавляем очередное слагаемое к интегральной сумме
      s = s + Integrate(F, c, d, a + i * dx, dx)
    Next i
    ' Значение функции
    ПОВТИНТЕГРАЛ = s * dx
End Function
```

У функции шесть аргументов, причем последний аргумент необязательный. В частности, текстовые аргументы F, c и d есть имена методов, описанных в модуле класса MultIntegrals, определяющих соответственно подынтегральную функцию (от двух аргументов), функцию (от одного аргумента) для нижней границы для внутреннего интеграла и функцию для верхней границы для внутреннего интеграла. Аргументы a1 и b1 определяют границы области интегрирования по первой переменной, причем предполагается, что они могут быть как числами, так и адресами ячеек. Необязательный аргумент n1 имеет по умолчанию значение 500, может быть как числом, так и ссылкой на ячейку и определяет количество частей, на которые разбивается область (интервал) интегрирования по переменной x. Значение нижней границы интервала интегрирования записывается в переменную a, значение верхней границы интервала интегрирования записывается в переменную b. Количество частей, на которые разбивается интервал интегрирования, записывается в переменную n.

Шаг приращения по переменной интегрирования определяется командой dx = (b-a)/n. Начальное значение для интегральной суммы вычисляем командой s = (Integrate(F,c,d,a,dx)+Integrate(F,c,d,b,dx))/2, а затем в операторе цикла эта сумма вычисляется прибавлением за каждую итерацию добавки Integrate(F,c,d,a+i*dx,dx). Именно здесь мы используем описанную выше функцию Integrate() для снятия (вычисления) внутреннего интеграла. С точностью до множителя dx значение переменной s (после завершения оператора цикла) совпадает со значением повторного интеграла.

Для применения разработанных нами утилит на практике осталось описать еще три функции: определяющие подынтегральную функцию, а также функции, определяющие границы области интегрирования во внутреннем интеграле. Здесь нужно определиться с тем, какой именно интеграл мы будем вычислять. Рассмотрим интеграл $\int_0^3 dx \int_x^{x(4-x)} (2x-y)^2 dy = \frac{729}{140} \approx 5{,}20714286$. Это означает, что подынтегральная функция дается выражением $F(x, y) = (2x - y)^2$. Код соответствующей функции приведен в листинге 14.16.

Листинг 14.16. Подынтегральная функция

```
Function Fn(x As Double, y As Double) As Double
 ' Значение подынтегральной функции
  Fn = (2 * x - y) ^ 2
End Function
```

Функция для определения нижней границы области интегрирования внутреннего интеграла описана в листинге 14.17.

Листинг 14.17. Нижняя граница интервала интегрирования

```
Function Fc(x As Double) As Double
 ' Значение функции, определяющей нижнюю границу интегрирования
  Fc = x
End Function
```

Функция для определения верхней границы области интегрирования внутреннего интеграла описана в листинге 14.18.

Листинг 14.18. Верхняя граница интервала интегрирования

```
Function Fd(x As Double) As Double
 ' Значение функции, определяющей верхнюю границу интегрирования
  Fd = x * (4 - x)
End Function
```

На заметку

Еще раз напоминаем, что функции Fn(), Fc() и Fd() описываются в модуле класса MultIntegrals: в проект нужно добавить модуль класса, изменить его имя и добавить в модуль программный код означенных трех функций.

Рабочий документ, в котором разработанные нами функции используются (явно или неявно) для вычисления повторного интеграла $\int_0^3 dx \int_x^{x(4-x)} (2x-y)^2 dy$, показан на рисунке 14.15.

Интеграл вычисляется с помощью формулы =ПОВТИНТЕГРАЛ("Fn";"Fc";"Fd";0;3). Если сравнить с точным значением, то можно заметить, что точность вычислений ограничивается третьей цифрой после запятой, что, в принципе, не так уж и плохо.

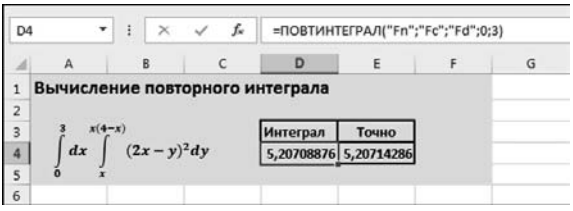


Рис. 14.15
Вычисление повторного интеграла с помощью пользовательской функции

ГЛАВА 15 РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНЫХ И ИНТЕГРАЛЬНЫХ УРАВНЕНИЙ

Работа секретная.
Думаю, с космосом связанная.
Так что читайте газеты!

Из к/ф «Усатый нянь»

Эта глава посвящена обыкновенным дифференциальным уравнениям и системам обыкновенных дифференциальных уравнений. Некоторое внимание также уделяется и интегральным уравнениям. Понятно, что полностью осветить тематику решения дифференциальных уравнений и их систем в рамках одной главы не удастся — да такая задача и не стоит. Здесь мы рассмотрим лишь ряд вопросов, связанных с решением (в числовом виде) дифференциальных и интегральных уравнений — разумеется, в контексте того, как соответствующие вычисления могут проводиться в Excel. Но и этого будет вполне достаточно для того, чтобы с успехом использовать Excel для решения задач, которые, на первый взгляд, в «сферу интересов» этого приложения не входят.

На заметку

Стоит напомнить, что дифференциальным уравнением называется соотношение, содержащее неизвестную функцию и производные от этой функции. Если функция зависит только от одного аргумента и, следовательно, производные вычисляются только по этому аргументу, то соответствующее дифференциальное уравнение называется обыкновенным дифференциальным уравнением.

О системе обыкновенных дифференциальных уравнений говорят, если решается система из нескольких уравнений, содержащих неизвестные функции от одного аргумента и их производные (первого, второго и т. д. порядков). Обычно количество уравнений совпадает с количеством неизвестных функций.

Если функция зависит от нескольких переменных и дифференциальное уравнение содержит производные по разным аргументам, то такое уравнение называется дифференциальным уравнением в частных производных.

Если уравнение содержит неизвестную функцию и интегралы от этой функции, то уравнение называется интегральным.

Порядком дифференциального уравнения называется порядок старшей производной, входящей в это уравнение.

Предметом нашего исследования станут явные дифференциальные уравнения первого порядка, а также системы дифференциальных уравнений пер-

вого порядка. Здесь мы исходим из того, что уравнения и системы более высоких порядков можно свести к системе уравнений первого порядка (читатель, интересующийся техникой таких преобразований, может обратиться к специальной литературе). Поэтому с точки зрения практической уравнения и системы первого порядка представляют достаточный интерес.

Для решения дифференциальных уравнений и систем в числовом виде исходная задача обычно формулируется в форме *задачи Коши*: помимо самого уравнения (системы уравнений) задается еще и значение функции (или функций) в начальной точке (т. е. при некотором фиксированном значении аргумента). Далее приступим к рассмотрению конкретных подходов и примеров.

На заметку

Вообще методы поиска приближенных решений дифференциальных уравнений обычно разделяют на *аналитические* и *числовые*. Если мы получаем аналитическое выражение для функции-решения дифференциального уравнения, которое не является точным решением, а лишь приближением этого решения, то речь идет об аналитических методах. Если же решение дифференциального уравнения ищется в виде таблицы (значений искомой функции в некоторых узловых точках), то речь идет о числовых методах. В этой главе мы рассмотрим как аналитические, так и числовые методы. Однако сразу следует отметить, что возможности реализации аналитических методов в Excel достаточно ограничены. Поэтому здесь рассмотрим лишь частные случаи. В частности, мы проиллюстрируем возможности по применению приложения Excel для вычисления приближенных аналитических решений дифференциальных уравнений первого порядка методом Пикара и методом разложения в степенной ряд.

МЕТОД ПОСЛЕДОВАТЕЛЬНЫХ ПРИБЛИЖЕНИЙ

И мы ловили рыбу вчера.
Ни одной не поймали.

Из к/ф «Усатый нянь»

Предположим, что необходимо решить уравнение $y'(x) = f(x, y(x))$ при начальном условии $y(x_0) = y_0$. Здесь $y(x)$ — неизвестная функция, которую следует найти, через $y'(x) \equiv \frac{dy(x)}{dx}$ обозначена производная от этой функции по независимому аргументу x, функция от двух аргументов $f(x, y)$ известна и фактически определяет дифференциальное уравнение, а x_0 и y_0 — заданные числовые значения.

Метод последовательных приближений Пикара базируется на итерационной процедуре, в рамках которой очередное приближение для решения уравнения $y_{n+1}(x)$ вычисляется на основе текущего приближения $y_n(x)$ по формуле $y_{n+1}(x) = y_0 + \int_{x_0}^{x} f(t, y_n(t))dt$. В качестве начального (нулевого) приближения для искомой функции $y(x)$ используется значение функции в точке x_0: т. е. по определению $y(x_0) = y_0$.

На заметку

Следовательно, чтобы найти в аналитическом виде приближенное решение уравнения $y'(x) = f(x, y(x))$, удовлетворяющее начальному условию $y(x_0) = y_0$, поступаем следующим образом. Сначала вместо $y(x)$ подставляем в функцию $f(x, y(x))$ значение y_0 и находим первообразную, в результате чего получаем первое приближение $y_1(x) = y_0 + \int_{x_0}^{x} f(t, y_0)dt$ для решения уравнения. Выражение для $y_1(x)$ подставляем в $f(x, y(x))$, интегрируем и получаем второе приближение $y_2(x) = y_0 + \int_{x_0}^{x} f(t, y_1(t))dt$ для решения уравнения и т. д.

Этот итерационный процесс, при некоторых дополнительных условиях (непрерывности функции $f(x, y)$ и ее производных будет достаточно), сходится к точному решению задачи Коши. Поэтому при невозможности найти точное решение он позволяет получить достаточно неплохие результаты. Вместе с тем, несмотря на кажущуюся простоту, практическое применение означенного алгоритма может столкнуться со значительными трудностями. Ну а если речь идет о реализации такой процедуры в Excel, то сразу становится очевидным, что далеко не во всех случаях приложение может быть полезным. Тем не менее существует класс уравнений, для которых применение приложения Excel представляется целесообразным. Один из таких частных случаев имеет место, когда функция $f(x, y)$ представляет собой полиномиальное выражение по обоим своим аргументам. В этом случае при интегрировании полиномиального выражения получаем полиномиальное выражение, и т. д. Кроме того, полиномы обладают той чудесной особенностью, что для их определения достаточно указать набор коэффициентов, что само по себе удобно с точки зрения реализации в Excel. Именно такую ситуацию рассмотрим далее.

Будем решать уравнение $y'(x) = (2x + y(x))^2$ с некоторым начальным условием $y(0) = y_0$.

На заметку

Здесь мы в качестве значения параметра x_0 рассматриваем нулевое значение (т. е. $x_0 = 0$). Общности это не ограничивает, поскольку, введя новую переменную $z = x - x_0$, мы всегда можем свести задачу именно к такому случаю.

Что касается уравнения $y'(x) = (2x + y(x))^2$, то оно имеет точное решение $y(x) = \frac{\sqrt{2}(y_0 - 2x)\cos(\sqrt{2}x) + 2(1 + y_0 x)\sin(\sqrt{2}x)}{\sqrt{2}\cos(\sqrt{2}x) - y_0 \sin(\sqrt{2}x)}$, которое удовлетворяет начальному условию $y(0) = y_0$. Учитывая сингулярность знаменателя, рассматривать такое решение имеет смысл только на интервале $0 \le x < \frac{\operatorname{arctg}\left(\frac{\sqrt{2}}{y_0}\right)}{\sqrt{2}}$. Это соотношение определяет область применимости того решения, которое мы получим.

Для поиска решения и реализации всего итерационного процесса нам понадобится несколько пользовательских функций. В листинге 15.1 приведен программный код функции, которая предназначена для интегрирования полиномиального выражения.

Листинг 15.1. Функция для интегрирования полиномиального выражения

```
Private Function PolyIntegrate(A() As Double, y0 As Double) As Double()
 ' Переменная для определения степени полинома
 Dim n As Integer
 ' Вычисляем значение для степени полинома
 n = UBound(A) - LBound(A)
 ' Создаем массив для записи коэффициентов полинома
 ReDim B(0 To n + 1) As Double
 ' Начальный (с нулевым индексом) коэффициент полинома
 B(0) = y0
 ' Индексная переменная для оператора цикла
 Dim i As Integer
 ' Оператор цикла для заполнения элементов массива
 For i = 1 To n + 1
  ' Вычисляем коэффициенты полинома
  B(i) = A(LBound(A) + i - 1) / i
 Next i
 ' Значение функции - массив с коэффициентами полинома
 PolyIntegrate = B
End Function
```

Функция PolyIntegrate() предназначена для «внутреннего» использования, поэтому описана она с ключевым словом Private. Полином мы отождествляем с массивом коэффициентов, поэтому в качестве аргумента функция PolyIntegrate() принимает числовой массив и результатом возвращает числовой массив. Еще один аргумент функции — значение первообразной для нулевого значения аргумента.

На заметку

Результатом интегрирования полинома $A(x) = a_0 + a_1x + \ldots + a_nx^n$ является полином $B(x) = b_0 + b_1x + b_2x^2 + \ldots + b_{n+1}x^{n+1}$, где $b_1 = a_0$, $b_2 = \frac{a_1}{2}$, $b_3 = \frac{a_2}{3}$, …, $b_{n+1} = \frac{a_n}{n+1}$. Параметр b_0 определяется из начального условия. Условие $B(0) = y_0$, то $b_0 = y_0$ и тогда полином-результат интегрирования может быть записан как $B(x) = y_0 + \frac{a_0}{1}x + \frac{a_1}{2}x^2 + \frac{a_2}{3}x^3 + \ldots + \frac{a_n}{n+1}x^{n+1}$. На «входе» разработанная нами функция получает коэффициенты a_0, a_1, …, a_n (в виде массива) и значение y_0, а на «выходе» получаем массив с элементами y_0, $\frac{a_0}{1}$, $\frac{a_1}{2}$, $\frac{a_2}{3}$, …, $\frac{a_n}{n+1}$. Если интерпретировать эти параметры и массивы в терминах разрабатываемой итерационной процедуры, то полином, который передается аргументом функции PolyIntegrate(), соответствует представлению функции $f(x, y(x))$ (для текущего приближения $y(x)$), а полином, который получаем на выходе — следующее приближение для $y(x)$).

В теле функции вводится и определяется целочисленная переменная n, определяющая значение исходного (интегрируемого) полинома. Значение

переменной определяем командой n = UBound(A)-LBound(A), в которой из наибольшего индекса в массиве A (инструкция UBound(A)) вычитается значение наименьшего индекса в массиве A (инструкция LBound(A)). Затем объявляется числовой массив B, индексация в котором выполняется в пределах от 0 до n+1. Первый (с нулевым индексом) элемент этого массива определяется командой B(0) = y0. Для вычисления прочих элементов массива B запускается оператор цикла с индексной переменной i, пробегающей значения от 1 до n+1. Элементы массива B определяются на основе коэффициентов массива A командой B(i) = A(LBound(A)+i-1)/i. Здесь мы учитываем, что индексация элементов массива A начинается со значения LBound(A) (которое, на самом деле, по умолчанию равно нулю).

После того как массив B заполнен, командой PolyIntegrate = B он присваивается значением функции.

Еще одна функция, которая нам понадобится для вычисления значения полиномиального выражения в точке (т. е. значения полинома, а не его коэффициентов), называется PolyValue(). Ее программный код приведен в листинге 15.2.

Листинг 15.2. Функция для вычисления значения полинома в точке

```
Private Function PolyValue(A() As Double, x As Double) As Double
  ' Переменная для записи значения полинома
  Dim P As Double
  ' Начальное значение полиномиальной суммы
   P = 0
  ' Переменная для запоминания степенного множителя
  ' в полиномиальной сумме
  Dim Q As Double
  ' Начальное значение для запоминания степенного множителя
  ' в полиномиальной сумме
   Q = 1
  ' Индексная переменная для оператора цикла
  Dim i As Integer
  ' Оператор цикла для вычисления полиномиальной суммы
  For i = LBound(A) To UBound(A)
    ' К сумме прибавляется очередное слагаемое
     P = P + A(i) * Q
   ' Вычисляем слагаемое в полиномиальной сумме
   ' для следующей итерации
    Q = Q * x
  Next i
  ' Значение функции
   PolyValue = P
End Function
```

У функции два аргумента: массив A с коэффициентами полинома и переменная x, играющая роль аргумента полинома.

На заметку

При вычислении значения полинома с коэффициентами $a_0, a_1, \ldots, a_n$ в точке x нам необходимо вычислить сумму $a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$. Эту сумму мы вычисляем, последовательно прибавляя слагаемые вида a_kx^k. Коэффициенты a_k «считываются» из массива, а для вычисления степенных множителей x^k вводим специальную переменную с начальным единичным значением и затем последовательно ее умножаем на x.

Для записи полиномиальной суммы вводится переменная P, начальное значение которой равно нулю. Для записи степенного множителя вводится переменная Q с начальным единичным значением. После этого запускается оператор цикла, в котором индексная переменная i перебирает все значения (диапазон значений от LBound(A) — нижний индекс массива A, — до UBound(A) — верхний индекс массива A). За каждый цикл выполняется две команды: сначала к полиномиальной сумме прибавляется очередное слагаемое (команда P = P+A(i)*Q), а затем командой Q = Q*x вычисляется степенной множитель для следующей итерации. Вычисленное значение переменной P и есть значение функции (команда PolyValue = P).

Достаточно нетривиальной является процедура вычисления подынтегрального полиномиального выражения. Дело в том, что даже если мы на какой-то итерации вычислили полиномиальное выражение для функции $y(x)$, то для вычисления следующей итерации необходимо проинтегрировать выражение $f(x, y(x))$, а его, в свою очередь, следует предварительно вычислить. Именно этой цели, т. е. определению коэффициентов полиномиального выражения $f(x, y(x)) = (2x + y(x))^2$ при условии, что известны коэффициенты полиномиального выражения $y(x)$, служит функция, которая называется getEqn(), и ее программный код представлен в листинге 15.3.

Листинг 15.3. Функция для вычисления полинома в подынтегральном выражении

```
Private Function getEqn(P() As Double) As Double()
 ‘ Переменная для запоминания степени исходного полинома
 ‘ (полинома, переданного аргументом функции через
 ‘ массив коэффицентов)
 Dim n As Integer
 ‘ Объявляем динамический массив для записи
 ‘ коэффициентов “технического” полинома,
 ‘ вычисляемого на промежуточном этапе
 Dim Q() As Double
 ‘ Объявляем динамический массив для записи
 ‘ коэффициентов полинома, который получается
 ‘ в результате вычисления подынтегрального выражения
 Dim Pn() As Double
 ‘ Степень исходного полинома
 n = UBound(P) - LBound(P)
 ‘ Размер полинома для подынтегрального выражения
 ReDim Q(0 To 2 * n)
```

```
    ' Размер "технического" промежуточного массива
    ReDim Pn(0 To n)
    ' Индексные переменные для операторов цикла
    Dim i As Integer, j As Integer
    ' Начальный (с нулевым индексом) коэффициент "технического" полинома
    Pn(0) = P(LBound(P))
    ' Второй (с единичным индексом) коэффициент "технического" полинома
    Pn(1) = P(LBound(P) + 1) + 2
    ' Оператор цикла для заполнения прочих элементов массива
    ' для "технического" полинома
    For i = 2 To n
      ' Определяем элементы массива промежуточного полинома
      Pn(i) = P(LBound(P) + i)
    Next i
    ' Оператор цикла для обнуления элементов массива,
    ' в который будет записываться результат функции
    ' (коэффициенты подынтегрального полинома)
    For i = 0 To 2 * n
      ' Вначале все элементы равны нулю
      Q(i) = 0
    Next i
    ' Внешний оператор цикла для вычисления коэффициентов
    ' подынтегрального полинома
    For i = 0 To n
      ' Внутренний оператор цикла для вычисления коэффициентов
      ' подынтегрального полинома
      For j = 0 To n
        ' Уточняем значение коэффициентов
        Q(i + j) = Q(i + j) + Pn(i) * Pn(j)
      Next j
    Next i
    ' Значение функции - массив с коэффициентами
    ' подынтегрального полинома
    getEqn = Q
  End Function
```

Аргументом функции передается массив P с коэффициентами полинома, представляющего функцию $y(x)$, а на выходе получаем массив с коэффициентами полинома, определяющего выражение $f(x, y(x))$.

В программном коде функции определяется переменная n = UBound(P)-LBound(P), которая определяет степень полинома «на входе». Также мы объявляем два массива:

- один «технический» массив называется Pn и предназначен для записи коэффициентов полинома, получающегося на промежуточном этапе вычислений, когда к полиному $y(x)$ прибавляется значение $2x$. Степень этого полинома, очевидно, такая же, как и степень полинома $y(x)$ (при условии, что речь не идет о нулевой итерации);

- другой массив называется Q и предназначен для записи коэффициентов результирующего полинома. Элементы полинома Q индексируются в пределах от 0 до 2*n, где n, напомним, обозначает степень полинома, переданного аргументом функции getEqn().

На заметку

Если $y(x)$ представляет собой полином степени n, то $2x + y(x)$ — тоже полином степени n, а $(2x + y(x))^2$ — полином степени $2n$.

Для вычисления результата (коэффициентов полинома, определяющего выражение $(2x + y(x))^2$), сначала вычисляем коэффициенты полинома $2x + y(x)$. Сделать это довольно просто — у этого полинома коэффициенты такие же, как и у полинома $y(x)$, за исключением лишь второго коэффициента (с индексом 1, соответствующим слагаемому с первой степенью переменной x), к которому следует прибавить значение 2. Именно поэтому командами Pn(0) = P(LBound(P)) и Pn(1) = P(LBound(P)+1)+2 сначала определяются два начальных элемента, а остальные определяются «дублированием» элементов массива P (команда Pn(i) = P(LBound(P)+i) в операторе цикла, в котором индексная переменная пробегает значения от 2 до n). Что касается массива Q, то сначала все его элементы получают нулевые значения. Добиваемся этого, выполнив оператор цикла с командой Q(i) = 0 в теле цикла (переменная i пробегает значения от 0 до 2*n). После этого запускаются вложенные операторы цикла, в котором каждая из индексных переменных i и j независимо пробегает значения от 0 до n. В теле внутреннего цикла выполняется команда Q(i+j) = Q(i+j)+Pn(i)*Pn(j). Здесь мы не стали использовать аналитическое выражение для коэффициентов полинома, являющегося квадратом другого полинома, а произвели «прямые» вычисления. Идея, которая нами реализована, достаточно проста: произведение двух полиномов (точнее, квадрат полинома, т. е. произведение его на себя же) представляет собой линейную комбинацию всевозможных попарных произведений слагаемых полиномов. При этом произведение с коэффициентом вида Pn(i)*Pn(j) в результирующем полиноме появится возле степенного слагаемого с показателем степени i+j.

Вычисленный таким образом массив Q является значением функции getEqn() (команда getEqn = Q).

Функция getDESoln() предназначена для выполнения заданного количества итераций при вычислении решения дифференциального уравнения. Аргументом функции передаются начальное значение y_0 искомой функции $y(x)$ в нулевой точке, а также количество итераций, выполняемых для поиска решения. Результатом является массив с коэффициентами полинома, определяющего искомую функцию. Программный код функции представлен в листинге 15.4.

Листинг 15.4. Функция для вычисления коэффициентов полиномиальных приближений для искомой функции

```
Private Function getDESoln(y0 As Double, n As Integer) As Double()
 ' Номер итерации для поиска приближенного решения
 ' не больше единицы
 If (n < 2) Then
```

```
    ' Массив из трех элементов для записи коэффициентов
    ' подынтегрального полинома
    Dim P(0 To 2) As Double
    ' Значение первого (индекс 0) элемента массива
    P(0) = y0 * y0
    ' Значение второго (индекс 1) элемента массива
    P(1) = 2 * y0
    ' Значение третьего (индекс 2) элемента массива
    P(2) = 4
    ' Приближенное решение для первой итерации получается
    ' интегрированием подынтегрального полинома
    getDESoln = PolyIntegrate(P, y0)
    ' Номер итерации не меньше, чем два
  Else
    ' Рекурсивное вычисление значения функции
    getDESoln = PolyIntegrate(getEqn(getDESoln(y0, n - 1)), y0)
  End If
End Function
```

В функции использован рекурсивный вызов. В теле функции в условном операторе проверяется условие n<2, которое означает, что необходимо выполнить меньше двух итераций (т. е. одну, хотя в силу того, как записано условие, соответствующая ветка кода будет выполняться при любом значении аргумента n, меньшем двух). Если это условие выполнено (мы в этом случае полагаем, что вычисляется первая итерация), то далее создается массив P из трех элементов (с индексацией элементов от 0 до 2), и в явном виде выписываются значения элементов этого массива (команды P(0) = y0*y0, P(1) = 2*y0 и P(2) = 4). Здесь мы исходим из того, что в нулевом приближении $y_0(x) = y_0$, поэтому подынтегральная функция для вычисления первого приближения дается выражением $f(x, y_0(x)) = (2x + y_0)^2 = y_0^2 + 4y_0x + 4x^2$ — отсюда и значения коэффициентов полинома.

Значение функции, после того, как определен массив P, вычисляется командой getDESoln = PolyIntegrate(P,y0), в которой мы использовали функцию PolyIntegrate(), описанную ранее и предназначенную для интегрирования полиномов.

Для итераций более высокого порядка, чем первый, результат вычисляется командой getDESoln = PolyIntegrate(getEqn(getDESoln(y0,n-1)),y0), в которой использован рекурсивный вызов функции getDESoln(). Мы воспользовались тем, что для n-й итерации результат может быть вычислен, если на основе n-1-й итерации сформировать подынтегральное выражение (с помощью функции getEqn()) и затем его проинтегрировать (с помощью функции PolyIntegrate()).

Все перечисленные выше функции, включая и функцию getDESoln(), предназначены исключительно для выполнения «черновых» расчетов. Нужна функция, которая станет «посредником» между рабочим листом и той вычислительной мощью, которую представляют собой означенные выше функции. Такая функция может иметь программный код, примерно такой, как в листинге 15.5.

Листинг 15.5. Функция вычисления приближенного решения дифференциального уравнения для использования в рабочем листе

```
Function getY(y0, n, Optional x)
 ' Переменная для запоминания значения функции в начальной точке
 Dim z0 As Double
 ' Начальное значение искомой функции
 z0 = y0
 ' Переменная для запоминания номера итерации приближенного
 ' решения для искомой функции
 Dim m As Integer
 ' Номер итерации для приближенного решения
 m = n
 ' Если третий аргумент функции отсутствует
 If (IsMissing(x)) Then
  ' Значение функции - массив с коэффициентами полиномиального
  ' выражения-приближенного решения дифференциального уравнения
  getY = getDESoln(z0, m)
  ' третий аргумент функции указан
 Else
  ' Переменная для запоминания значения аргумента, для которого
  ' вычисляется значение функции - приближенного решения
  ' дифференциального уравнения
  Dim z As Double
  ' Значение аргумента
  z = x
  ' Значение функции - значение полинома, который соответствует
  ' приближенному решению дифференциального уравнения в точке,
  ' определяемой третьим аргументом функции
  getY = PolyValue(getDESoln(z0, m), z)
 End If
End Function
```

У функции getY() всего три аргумента, причем третий аргумент необязательный. В качестве результата функция может возвращать массив с коэффициентами полинома, определяющего приближенное решение дифференциального уравнения, или значение полинома в одной точке — все зависит от того, передан третий аргумент функции или нет. Если третий аргумент при вызове функции ей не передавался, функция возвращает массив коэффициентов полинома. Если третий аргумент (переменная x) указан, то значением функции является значение полинома в точке, определяемой этим третьим аргументом.

Что касается первых двух аргументов функции getY(), то первый аргумент (переменная y0) соответствует значению функции-решения дифференциального уравнения в нулевой точке, второй аргумент (переменная n) определяет количество итераций, на основе которых вычисляется приближенное решение дифференциального уравнения.

Аргументами функции getY() могут передаваться как числовые значения, так и ссылки на ячейки. В последнем случае могла бы возникнуть проблема, поскольку вызываемые в теле функции getY() «вспомогательные» функции оперируют с числовыми значениями. Для надежности вводим числовые переменные и присваиваем им значения на основе значений аргументов: командой z0 = y0 значение первого аргумента записывается в переменную z0, командой m = n значение второго аргумента записывается в переменную m, а позднее, при наличии третьего аргумента, командой z = x значение третьего аргумента записывается в переменную z.

С помощью условного оператора мы проверяем условие IsMissing(x). Истинность условия означает, что третий аргумент при вызове функции не указан. В этом случае результат функции вычисляется командой getY = getDESoln(z0,m). Если третий аргумент все же указан, то результат вычисляется командой getY = PolyValue(getDESoln(z0,m),z).

Таким образом, для вычисления решения (приближенного) уравнения $y'(x) = (2x + y(x))^2$ нам в рабочем листе понадобится всего одна функция — это функция getY(). Обязательные аргументы этой функции — значение в нулевой точке функции-решения дифференциального уравнения и количество итераций, которые следует выполнить. В качестве примера использования этой функции на рисунке 15.1 приведен фрагмент документа, содержащий соответствующие вычисления.

Решение дифференциального уравнения вычисляется последовательно по десяти итерациям, начиная с первой. В ячейку B7 вводится значение искомой функции в нулевой точке (на рисунке 15.1 это значение 2). В ячейке B8 указано значение аргумента x, для которого вычисляется функция (значение 0,2). Точное решение вычисляется по формуле =(КОРЕНЬ(2)*(B7-2*B8)*COS(КОРЕНЬ(2)*B8)+2*(1+B7*B8)*SIN(КОРЕНЬ(2)*B8))/(КОРЕНЬ(2)*COS(КОРЕНЬ(2)*B8)-B7*SIN(КОРЕНЬ(2)*B8)) в ячейке B9. Граничное значение для аргумента (точка сингулярности решения) вычисляется в ячейке B10 по формуле =ATAN(КОРЕНЬ(2)/B7)/КОРЕНЬ(2).

В ячейках D4:D13 содержатся номера итераций, по которым строятся решения уравнения. Диапазон ячеек E4:E13 содержит вычисленные значения приближенных решений в указанной точке (ячейка B8). Диапазон E4:E13

E4 =getY($B$7;D4;$B$8)

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	**Решение дифференциального уравнения методом Пикара**												
2													
3	Уравнение $y'(x) = (2x + y(x))^2$			Итерация	Решение	Коэффициенты							
4				1	2,8906667	2,00000	4,00000	2,00000	1,33333				
5	Начальное условие $y(0) = y_0$			2	3,4104068	2,00000	4,00000	12,00000	14,66667	7,33333	4,00000	0,88889	0,25397
6				3	3,6132129	2,00000	4,00000	12,00000	28,00000	50,66667	69,86667	76,00000	63,23810
7	Начальное значение функции (параметр y0)	2		4	3,6749913	2,00000	4,00000	12,00000	28,00000	64,00000	136,53333	259,91111	448,91429
8	Аргумент для функции (переменная x)	0,2		5	3,6899735	2,00000	4,00000	12,00000	28,00000	64,00000	147,20000	331,02222	714,00635
9	Точное решение (значение в точке)	3,6935466		6	3,6929586	2,00000	4,00000	12,00000	28,00000	64,00000	147,20000	338,13333	772,92698
10	Максимальное значение аргумента	0,435		7	3,6934621	2,00000	4,00000	12,00000	28,00000	64,00000	147,20000	338,13333	776,99048
11				8	3,6935358	2,00000	4,00000	12,00000	28,00000	64,00000	147,20000	338,13333	776,99048
12				9	3,6935453	2,00000	4,00000	12,00000	28,00000	64,00000	147,20000	338,13333	776,99048
13				10	3,6935464	2,00000	4,00000	12,00000	28,00000	64,00000	147,20000	338,13333	776,99048
14													

Рис. 15.1
Вычисление приближенного решения дифференциального уравнения с помощью пользовательской функции, реализующей метод Пикара

заполняется копированием формулы =getY($B$7;D4;$B$8) из ячейки E4. Ячейки справа содержат значения коэффициентов соответствующего полинома: в ячейках F4:I4 содержатся коэффициенты (всего четыре) полинома, получающегося на первой итерации (вычисляются формулой массива =getY($B$7;D4)). Для прочих итераций вычисляется и отображается по восемь первых коэффициентов (это не все коэффициенты). Диапазон ячеек F5:M13 можно заполнить, скопировав построчно формулу массива =getY($B$7;D5) из диапазона ячеек F5:M5.

Что касается результатов вычислений, то, сравнив приближенное значение с «точным» значением, можем заключить, что точность довольно неплохая.

На заметку

Вместе с тем, особо обольщаться не стоит. С помощью предложенной выше процедуры удастся вычислить не очень большое количество итераций. Дело в том, что на каждой новой итерации количество коэффициентов полинома удваивается. Поэтому полином для n-й итерации содержит порядка 2^n коэффициентов. То есть коэффициенты размножаются, как кролики, и это не очень хорошо. Оправданием для применения означенного подхода может служить то обстоятельство, что уже начальные итерации обычно дают довольно неплохой результат.

МЕТОД СТЕПЕННЫХ РЯДОВ

А когда человек теряет чувство юмора,
невозможно предугадать,
что он натворит.

Из к/ф «Старики-разбойники»

Еще один «аналитический» метод состоит в том, что решение уравнения вида $y'(x) = f(x, y(x))$, удовлетворяющее начальному условию $y(x_0) = y_0$, ищется в виде степенного ряда $y(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)^2 + \ldots + a_n(x - x_0)^n + \ldots$ В принципе, этот ряд бесконечный, и если удается найти все его коэффициенты, то фактически будет найдено точное решение. Но это скорее исключение из правил. Обычно ряд ограничивают на каком-то слагаемом и вычисляют коэффициенты полиномиального выражения. Полученное таким образом решение будет приближенным.

На заметку

Применение этого метода подразумевает, что искомая функция $y(x)$ может быть разложена в ряд Тейлора в окрестности точки $x = x_0$. Это, разумеется, не всегда так.

Таким образом, задача состоит в вычислении коэффициентов следующего представления для искомой функции $y(x) \approx a_0 + a_1(x - x_0) + a_2(x - x_0)^2 + \ldots + a_n(x - x_0)^n$. Один вывод можно сделать сразу: из условия $y(x_0) = y_0$ получаем $a_0 = y_0$. Отсюда можно записать $y(x) - y_0 = a_1(x - x_0) + a_2(x - x_0)^2 + \ldots + a_n(x - x_0)^n$. Это выражение для функции $y(x)$ необходимо подставить в

исходное дифференциальное уравнение. При этом $y'(x) = a_1 + 2a_2(x - x_0) + 3a_3(x - x_0)^2 + \ldots + na2(x - x_0)^{n-1}$. В правой части получаем выражение $f\left(x, \sum_{k=1}^{n} a_k(x - x_0)^k\right)$. Это выражение нужно разложить в ряд и приравнять коэффициенты возле однотипных слагаемых в правой и левой части. Здесь, фактически, кроется самая сложная часть процедуры по вычислению приближенного решения методом разложения в ряд. Поэтому, как правило, ограничиваются вычислением лишь нескольких слагаемых. Более того, процедура разложения в ряд сложного выражения является по сути своей аналитической и явно не коррелирует с назначением и возможностями приложения Excel. Учитывая все эти обстоятельства, мы воспользуемся некоторой «гибридной» моделью, в рамках которой функцию $f(x, y)$ будем раскладывать в ряд по обоим аргументам в окрестности точки (x_0, y_0) и использовать приближенное представление $f(x, y) \approx \sum_{k=0}^{n}\sum_{m=0}^{n} f_{km}(x - x_0)^k(y - y_0)^m$ при условии, что решение следует искать в виде ряда до слагаемых степени не выше, чем n. В результате получим соотношение $\sum_{k=0}^{n-1}(k+1)a_{k+1}(x - x_0)^k = \sum_{k=0}^{n}\sum_{m=0}^{n} f_{km}(x - x_0)^k\left(\sum_{p=1}^{n} a_p(x - x_0)^p\right)^m$, из которого последовательно находятся коэффициенты a_k $(k = 1, 2, \ldots, n)$. Для этого в правой части соотношения нужно раскрыть все скобки, привести выражение к каноническому полиномиальному виду и приравнять коэффициенты при одинаковых степенных одночленах справа и слева. Аналитическое выражение в данном случае получить довольно сложно, да это и не нужно. Зато мы можем «алгоритмизировать» процесс поиска коэффициентов разложения в ряд. Для этого представим, что нам каким-то образом удалось вычислить несколько коэффициентов — до коэффициента a_s включительно. Проанализируем процесс вычисления следующего коэффициента, т. е. коэффициента a_{s+1}. В левой части уравнения этот коэффициент встречается только со слагаемым $(s + 1)a_{s+1}(x - x_0)^s$. Поэтому выражение $(s + 1)a_{s+1}$ необходимо приравнять к коэффициенту возле множителя $(x - x_0)^s$ в правой части уравнения. Как отмечалось выше, хотя мы не планируем в явном виде вычислять этот коэффициент, для нас важно другое: этот коэффициент определяется значением коэффициентов с индексом не больше, чем s, т. е. это некоторая функция $\psi(a_0, a_1, \ldots, a_s)$ от коэффициентов $a_0, a_1, \ldots, a_s$. Поэтому если эти коэффициенты известны, то мы в принципе можем вычислить следующий коэффициент $a_{s+1} = \dfrac{\psi(a_0, a_1, \ldots, a_s)}{s+1}$. И хотя явное выражение для функции $\psi(a_0, a_1, \ldots, a_s)$ нам неизвестно, мы можем вычислить ее значение в числовом виде. Далее, поскольку вычислен коэффициент a_0, мы можем вычислить коэффициент a_1. Затем на основе значений коэффициентов a_0 и a_1 вычисляется значение коэффициента a_2 и т. д.

На заметку

В принципе, если мы используем разложение в ряд до слагаемых степени n, то вычислить мы сможем не только все коэффициенты разложения, но еще и коэффициент a_{n+1}.

Задачу можно формально упростить, если ввести в рассмотрение новую независимую переменную $t = x - x_0$ и новую функцию $z(t) = y(x) - y_0 = y(t + x_0) - y_0$. При этом исходное уравнение трансформируется в $z'(t) = f(t + x_0, y(t + x_0)) = f(t + x_0, z(t) + y_0) \equiv F(t, z(t))$, а начальное условие будет иметь вид z(0) = 0. Решение этого уравнения, таким образом, ищем в виде $z(t) = a_0 + a_1 t + a_2 t^2 + \ldots + a_n t^n$. Из начального условия $z(0) = 0$ автоматически следует, что $a_0 = 0$. Далее, если мы «обрываем» ряд для искомой функции на слагаемом с показателем степени n, то нам понадобятся коэффициенты разложения функции $F(t, z)$ в ряд Тейлора (в окрестности нулевых значений) до слагаемых не ниже суммарно степени n, т. е. перед применением приложения Excel для поиска решения уравнения придется представить функцию в правой части дифференциального уравнения в виде $F(t, z) \approx \sum_{k=0}^{n} \sum_{m=0}^{n-k} F_{km} t^k z^m$. Коэффициенты F_{km} удобно представить в виде матрицы. Причем с практической точки зрения удобнее во внутренней сумме рассматривать верхнюю границу суммирования, равную n. Чтобы компенсировать увеличение объема «ручной» вычислительной работы, можем «лишние» коэффициенты не вычислять, а полагать нулевыми.

На заметку

Следует заметить, что «лишними» являются коэффициенты F_{km}, для которых $k + m > n$.

Для реализации метода поиска приближенного решения дифференциального уравнения в виде степенного ряда разработаем несколько пользовательских функций. Важную роль при этом будет играть функция, предназначенная для вычисления по специальным правилам произведения двух полиномов. Как и в предыдущем примере, полином мы отождествляем с массивом из коэффициентов этого полинома. Таким образом, функция должна на основе двух массивов вычислять новый массив. Упомянутые выше «специальные правила» вычисления полиномиального произведения касаются «отбрасывания хвоста полинома» — мы перемножаем полиномы, но учитываем только слагаемые не выше определенной степени. Эта «определенная степень» задается количеством элементов в массиве: степень полинома на единицу меньше, чем количество элементов в массиве коэффициентов этого полинома. Следовательно, хотя в результате вычисления произведения двух полиномов степени n каждый раз мы получаем полином степени $2n$, в этом полиноме мы оставляем слагаемые степени не выше, чем n. Это и есть результат произведения полиномов, полученный «по специальной процедуре». Для выполнения подобных вычислений предназначена функция PMult(), описанная в листинге 15.6.

Листинг 15.6. Функция для вычисления полиномиального произведения (с отбрасыванием остаточных членов)

```
Private Function PMult(P1() As Double, P2() As Double) As Double()
 ' Переменная для определения степени полиномиального выражения
 Dim n As Integer
```

```
' Степень полиномиального выражения
n = UBound(P1) - LBound(P1)
' Массив для записи коэффициентов полиномиального произведения
' (вычисляется с отбрасыванием остаточных членов)
ReDim P(0 To n) As Double
' Индексные переменные для операторов цикла
Dim i As Integer, j As Integer
' Оператор цикла для обнуления коэффициентов
' полинома - элементов массива
For i = 0 To n
  ' Нулевое значение элемента массива
  P(i) = 0
Next i
' Внешний оператор цикла для вычисления коэффициентов
' полиномиального выражения
For i = 0 To n
  ' Внутренний цикл для вычисления коэффициентов
  ' полиномиального выражения
  For j = 0 To n - i
    ' Уточняем текущие значения коэффициентов полинома
    P(i + j) = P(i + j) + P1(LBound(P1) + i) * P2(LBound(P2) + j)
  Next j
Next i
' Значение функции - массив с коэффициентами
' полиномиального выражения
PMult = P
End Function
```

У функции два аргумента — это массивы P1 и P2 с коэффициентами перемножаемых полиномов. Мы предполагаем, что эти массивы одинаковой длины. Степень полиномов, которые реализуются через эти массивы, определяется командой n = UBound(P1)-LBound(P1) как разность последнего (вычисляется функцией UBound()) и начального (вычисляется функцией LBound()) индексов в первом массиве. Для записи результата произведения мы объявляем массив P, элементы которого индексируются от 0 до n. Вначале элементы у этого массива нулевые. Затем запускаются вложенные операторы цикла, и выполняется команда P(i+j) = P(i+j)+P1(LBound(P1)+i)*P2(LBound(P2)+j), которой уточняем текущие значения коэффициентов полинома. При этом мы учли, что у массивов P1 и P2 индексация элементов начинается со значений LBound(P1) и LBound(P2)+j соответственно. После проведения всех вычислений массив P возвращается в качестве значения функции.

Разработанная таким образом функция нужна для вычисления выражений, в которых полином возводится в степень (по тем же правилам, по которым вычисляется произведение полиномов). Такие операции придется выполнять каждый раз при вычислении ограниченного ряда для функции $F(t, z)$ в правой части уравнения, когда полиномиальное выражение для искомой

функции $z(t)$ возводится в целочисленную степень. Функция, которой вычисляется результат возведения полинома в степень (с отбрасыванием лишних слагаемых), называется PPower(). У нее два аргумента (массив, определяющий полином, и целое число, определяющее показатель степени). Функция описана в листинге 15.7.

Листинг 15.7. Функция для возведения полинома в степень (с отбрасыванием остаточных членов)

```
Private Function PPower(P() As Double, m As Integer) As Double()
 ' Проверяем значение показателя степени с помощью
 ' оператора выбора
 Select Case m
  ' Полином в нулевой степени
  Case 0
   ' Переменная для определения степени полинома-аргумента
   Dim n As Integer
   ' Степень полинома, переданного аргументом функции
   ' (через массив коэффициентов)
    n = UBound(P) - LBound(P)
   ' Создаем массив для записи результата функции
   ReDim R(0 To n) As Double
   ' Индексная переменная для оператора цикла
   Dim i As Integer
   ' Элемент с нулевым индексом равен единице
    R(0) = 1
   ' Заполняем прочие элементы массива
    For i = 1 To n
     ' Все остальные элементы равны нулю
      R(i) = 0
   Next i
   ' Значение функции - массив с единицей
   ' на первой позиции и остальными нулями
    PPower = R
   ' Полином в первой степени
  Case 1
   ' Результат функции - массив, переданный функции аргументом
    PPower = P
   ' Степень полинома выше первой
  Case Else
   ' Рекурсивный вызов функции возведения полинома в степень
   PPower = PMult(P, PPower(P, m - 1))
 End Select
End Function
```

В теле функции выполняется оператор Select, в котором проверяется значение второго аргумента (переменная m). Это, напомним, показатель степени,

в которую возводится полином — его коэффициенты передаются функции через первый аргумент-массив P. Если полином возводится в нулевую степень, то на выходе должна быть единица, но не просто единица, а полином. В том представлении полиномов, которое мы используем, такой «полиномиальной» единице соответствует массив, у которого первый элемент равен единице, а все остальные элементы нулевые. Количество элементов такого массива определяется количеством элементов в массиве P (первый аргумент функции PPower()). Поэтому при нулевом значении m командой n = UBound(P)-LBound(P) определяется степень полинома P, после чего создается массив R с индексацией элементов от 0 до n. Первый элемент у этого массива равен единице, а остальные обнуляются. Массив R возвращается как результат функции PPower().

Если второй аргумент функции PPower() равен единице, в качестве значения функции возвращается массив P. Для иных значений m результат функции вычисляется командой PPower = PMult(P,PPower(P,m-1)) через рекурсивный вызов функции Power(). Здесь мы воспользовались тем, что $P^m = P \cdot P^{m-1}$, и для вычисления произведения полиномов использовалась функция PMult().

Специальную функцию описываем для вычисления произведений вида $t^m P(t)$, где $P(t)$ — полином. Результатом выражения $t^m P(t)$, разумеется, является полином. Причем этот результат можно получить на основе полинома $P(t)$, сместив в соответствующем массиве все коэффициенты на m позиций вправо, с отбрасыванием лишних коэффициентов справа и нулевыми недостающими коэффициентами слева. Этот принцип реализуется в функции PShift(), программный код которой приведен в листинге 15.8.

Листинг 15.8. Функция для вычисления результата произведения полинома на множитель степенного типа (с отбрасыванием «хвоста»)

```
Private Function PShift(P() As Double, m As Integer) As Double()
 ' Переменная для определения степени полинома "на входе"
 Dim n As Integer
 ' Степень исходного полинома
 n = UBound(P) - LBound(P)
 ' Полином для записи результата
 ReDim Q(0 To n) As Double
 ' Индексная переменная для оператора цикла
 Dim i As Integer
 ' Оператор цикла для заполнения массива-результата
 For i = 0 To n
  ' Начальные элементы нулевые
  If (i < m) Then
    Q(i) = 0
   ' Ненулевые элементы
  Else
   ' "Сдвиг" элементов массива
    Q(i) = P(LBound(P) + i - m)
  End If
```

```
  Next i
  ' Значение функции
  PShift = Q
End Function
```

У функции два аргумента: массив P для представления полинома и целочисленная переменная m, через которую задается степень множителя (параметр сдвига коэффициентов полинома). В теле функции командой n = UBound(P)-LBound(P) мы определяем степень исходного полинома. Массив Q содержит n+1 элемент с индексацией от 0 до n. Элементы массива заполняются так: если индекс элемента меньше m, то значение элемента нулевое. В противном случае значение элемента массива Q вычисляется «сдвигом» элемента из массива P по формуле Q(i) = P(LBound(P)+i-m). Массив Q определяет значение функции PShift().

Еще две небольшие функции играют непринципиальную вспомогательную роль. Так, функция getLast() с программным кодом, представленным в листинге 15.9, возвращает в качестве значения последний элемент массива, переданного аргументом функции.

Листинг 15.9. Функция для считывания последнего элемента массива

```
Private Function getLast(P() As Double) As Double
  ' Значение функции
  getLast = P(UBound(P))
End Function
```

Функция getPoly() в качестве значения возвращает массив, представляющий полином, который является результатом двух последовательных операций: сначала некоторый исходный полином возводится в целочисленную степень, а затем полученный результат умножается на степенной одночлен. Несложно догадаться, что означенные операции выполняются при вычислении слагаемых вида $t^k z^m(t)$. Благодаря разработанным ранее функциям, программный код функции getPoly() достаточно прост, в чем несложно убедиться, если обратиться к листингу 15.10.

Листинг 15.10. Функция для вычисления полинома после возведения в степень и умножения на одночлен

```
Private Function getPoly(P() As Double, i As Integer, j As Integer) As Double()
  ' Значение функции
  getPoly = PShift(PPower(P, j), i)
End Function
```

Все основные вычисления выполняются в функции, предназначенной для вычисления нового коэффициента в разложении на основе массива с уже вычисленными коэффициентами и массива разложения функции $F(t, z)$ от двух переменных в правой части уравнения в степенной ряд. Проанализируем программный код функции getNextA(), представленный в листинге 15.11.

Листинг 15.11. Функция для вычисления коэффициента разложения

```
Private Function getNextA(F() As Double, P() As Double) As Double
 ' Переменная для записи результата
 Dim R As Double
 ' Начальное значение переменной-результата
  R = 0
 ' Индексные переменные для операторов цикла
 Dim i As Integer, j As Integer
 ' Внешний оператор цикла
 For i = LBound(F, 1) To UBound(F, 1)
  ' Внутренний оператор цикла
   For j = LBound(F, 2) To UBound(F, 2)
    ' Уточнение результата
    R = R + F(i, j) * getLast(getPoly(P, i - LBound(F, 1), _
    j - LBound(F, 2)))
  Next j
 Next i
 ' Значение функции
 getNextA = R / (UBound(P) - LBound(P) + 1)
End Function
```

У функции getNextA() два аргумента: двумерный массив F с коэффициентами F_{km} разложения функции двух $F(t, z)$ аргументов в правой части уравнения и одномерный массив P с коэффициентами a_0, a_1, ..., a_s разложения функции-решения дифференциального уравнения (теми коэффициентами, что вычислены на данный момент). В теле функции запускаются вложенные операторы цикла, в которых индексные переменные i и j перебирают все индексы двумерного массива F. При этом второй аргумент в функциях LBound() и UBound() определяет размерность (номер индекса), по которой вычисляются соответственно нижняя и верхняя границы. Результат вычислений записывается в переменную R, которой сначала присваивается нулевое значение, а затем при каждом фиксированном значении индексных переменных к этой переменной добавляется значение выражения F(i,j)*getLast(getPoly(P,i-LBound(F,1),j-LBound(F,2))). Этой командой коэффициент F(i,j) умножается на последний элемент массива getPoly(P,i-LBound(F,1),j-LBound(F,2)), который получается возведением в степень j-LBound(F,2) массива P и последующим умножением на степенной одночлен с показателем степени i-LBound(F,1).

Полученное значение для переменной R делится на значение выражения (UBound(P)-LBound(P)+1) (степень исходного полинома P плюс единица). Это и есть значение функции getNextA().

На заметку

Здесь мы воспользовались тем обстоятельством, что, если на данный момент вычислены коэффициенты до индекса s включительно, то чтобы вычислить следующий коэффициент (с индексом $s + 1$) нужно в правой части уравнения рассчитать коэффициент возле аргумента t с показателем степени s и поделить полученное значение на $s + 1$.

Для формирования массива из коэффициентов разложения функции-решения уравнения в ряд создаем еще одну функцию. Она называется getSoln(). У этой функции два аргумента: двумерный массив F с коэффициентами, определяющими функцию в правой части уравнения, а также целое число n, определяющее порядок разложения (степень последнего слагаемого в разложении). Программный код функции представлен в листинге 15.12.

Листинг 15.12. Функция для вычисления массива с коэффициентами полинома-результата

```
Private Function getSoln(F() AsDouble, n As Integer) As Double()
 ' Объявляем динамический массив для записи результата
 Dim P() As Double
 ' Создаем массив из двух элементов
 ReDim P(0 To 1)
 ' Первый (с нулевым индексом) элемент массива
 P(0) = 0
 ' Второй (с единичным индексом) элемент массива
 P(1) = F(LBound(F, 1), LBound(F, 2))
 ' Если полином первой степени
 If (n = 1) Then
  ' Значение функции
  getSoln = P
  ' Если степень полинома выше, чем первая
 Else
  ' Переменная для записи значения очередного коэффициента
  Dim A As Double
  ' Индексная переменная для оператора цикла
  Dim i As Integer
  ' Вычисляем коэффициенты полинома
  For i = 2 To n
   ' Новый коэффициент
   A = getNextA(F, P)
   ' Расширяем массив
   ReDim Preserve P(0 To UBound(P) + 1)
   ' Добавляем новый коэффициент
   P(UBound(P)) = A
  Next i
  ' Значение функции
  getSoln = P
 End If
End Function
```

Программный код функции содержит команду явного объявления динамического массива Dim P() As Double. Нам нужен именно динамический массив, поскольку в дальнейшем его размер будет меняться. В самом начале командой ReDim P(0 To 1) мы устанавливаем размер массива всего на два

элемента. Первый (с нулевым индексом) элемент массива равен нулю (команда P(0) = 0), а второй (с единичным индексом) элемент массива вычисляется командой P(1) = F(LBound(F,1),LBound(F,2)) и равен, очевидно, элементу в левом верхнем углу матрицы, определяемой двумерным массивом F.

На заметку

Если исходить из представлений $z(t) = a_0 + a_1 t + a_2 t^2 + \ldots + a_n t^n$, $F(t, z) = \sum_{k=0}^{n}\sum_{m=0}^{n} F_{km} t^k z^m$, а также учесть начальное условие $z(0) = 0$, то получим, во-первых, $a_0 = 0$. Во-вторых, поскольку $z'(t) = a_1 + 2a_2 t^2 + \ldots + na_n t^{n-1}$, то как следствие $z'(0) = a_1$. С другой стороны, $z'(t) = F(t, z(t))$, что дает $z'(0) = F(0, z(0)) = F_{00}$. В итоге получаем $a_1 = F(0, z(0)) = F_{00}$.

Если нам нужно ограничиться линейным приближением в разложении функции-решения в ряд, то командой getSoln = P значение полинома P определяется как значение функции getSoln(). В противном случае запускается оператор цикла, в котором индексная переменная i пробегает значения от 2 до n. За каждый цикл командой A = getNextA(F,P) в переменную A записывается новый вычисленный коэффициент. Затем командой ReDim Preserve P(0 To UBound(P)+1) размер массива P увеличивается на один элемент, причем уже записанные в массив значения сохраняются (инструкция Preserve). Наконец, командой P(UBound(P)) = A новый коэффициент добавляется последним элементом в массив P. Теперь на основе этого массива можно вычислять новый коэффициент, что и делается на следующем цикле. Массив P возвращается как значение функции getSoln().

Функция ДРЕШЕНИЕ() в некотором смысле дублирует функцию getSoln(), но, в отличие от последней, предназначена для использования в рабочем документе. Программный код функции ДРЕШЕНИЕ() приведен в листинге 15.13.

Листинг 15.13. Функция для использования в рабочем документе, предназначенная для вычисления коэффициентов полинома-решения уравнения

```
Function ДРЕШЕНИЕ(R As Range, num)
 ' Целочисленные переменные для запоминания количества строк
 ' и столбцов в диапазоне ячеек, переданном аргументом функции
 Dim m As Integer, n As Integer
 ' Количество строк в диапазоне
 m = R.Rows.Count
 ' Количество столбцов в диапазоне
 n = R.Columns.Count
 ' Массив для записи коэффициентов полинома
 ReDim F(0 To m - 1, 0 To n - 1) As Double
 ' Индексные переменные для операторов цикла
 Dim i As Integer, j As Integer
 ' Внешний цикл (перебираем строки матрицы)
 For i = 0 To m - 1
  ' Внутренний цикл (перебираем столбцы матрицы)
  For j = 0 To n - 1
```

```
    ' Заполняем двумерный массив
     F(i, j) = R.Cells(i + 1, j + 1).Value
   Next j
 Next i
 ' Переменная для запоминания второго аргумента функции,
 ' определяющего степень полинома
 Dim k As Integer
 ' Степень полинома
  k = num
 ' Значение функции
 ДРЕШЕНИЕ = getSoln(F, k)
End Function
```

У функции два аргумента: массив с коэффициентами разложения в ряд функции в правой части уравнения, а также аргумент, определяющий порядок разложения. В теле функции из диапазона с коэффициентами разложения функции выполняется поэлементное копирование во внутренний двумерный массив, после чего вызывается функция getSoln(), и вычисляются коэффициенты разложения.

Для иллюстрации работы описанных выше утилит решим уже знакомое нам уравнение $y'(x) = (2x + y(x))^2$ с начальным условием $y(0) = y_0$. Для удобства вводим новую функцию $z(x) = y(x) - y_0$. Уравнение в этом случае перепишется как $z'(x) = (2x + z(x) + y_0)^2$, а граничное условие станет нулевым, т. е. $z(0) = 0$. Точное решение для такой задачи дается выражением

$$z(x) = \frac{(2 + 2xy_0 + y_0^2)\sin(\sqrt{2}x) - 2\sqrt{2}x\cos(\sqrt{2}x)}{\sqrt{2}\cos(\sqrt{2}x) - y_0\sin(\sqrt{2}x)}.$$

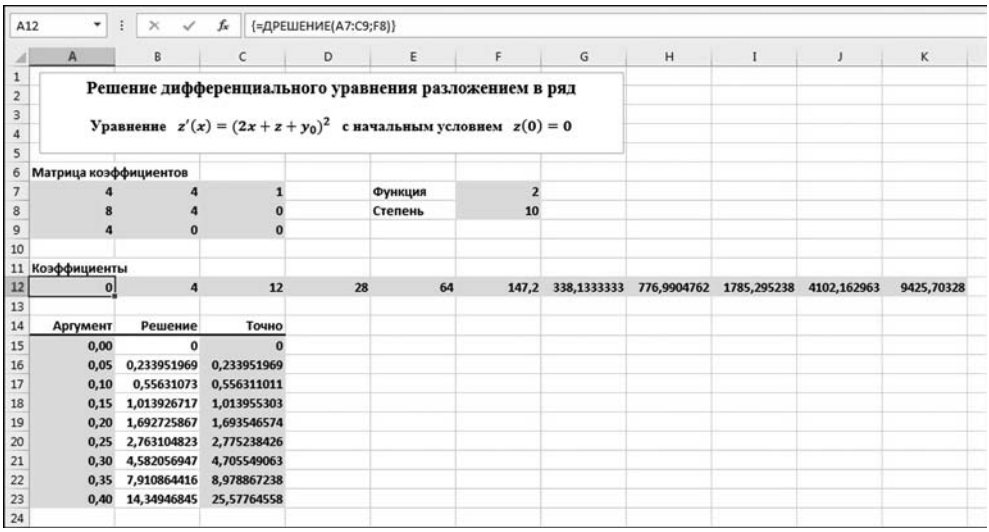
Поскольку функция в правой части уравнения является полиномом, то матрицу коэффициентов легко вычисляем по выражению

$$(2x + z(x) + y_0)^2 = y_0^2 + 2y_0z(x) + 4y_0x + 4x^2 + 4xz(x) + z(x)^2.$$

Рабочий документ с вычислениями представлен на рисунке 15.2.

В ячейку F7 вводится число 2 как значение параметра y_0. В ячейку F8 вводится число 10, определяющее порядок разложения (значение показателя степени слагаемых ряда, до которого выполняется разложение). В ячейках A7:C9 вычисляются коэффициенты разложения в ряд функции в правой части уравнения. Значения, которые вводятся в эти ячейки, описаны в таблице 15.1.

Мы выделяем ячейки A12:K12 и вводим в эти ячейки формулу массива =ДРЕШЕНИЕ(A7:C9;F8). В результате в этих ячейках вычисляются коэффициенты разложения в ряд Тейлора функции-решения дифференциального уравнения. Для вычисления значения этой функции заполняем ячейки A15:A23 значениями аргумента для функции (в ячейку A15 вводим значение 0, в ячейку A16 вводим формулу =A15+0,05 и копируем ее во все остальные ячейки). Для вычисления приближенного решения в ячейку B15 вводится



A12 | {=ДРЕШЕНИЕ(A7:C9;F8)}

Решение дифференциального уравнения разложением в ряд

Уравнение $z'(x) = (2x + z + y_0)^2$ с начальным условием $z(0) = 0$

	A	B	C	D	E	F	G	H	I	J	K
6	Матрица коэффициентов										
7	4	4	1		Функция	2					
8	8	4	0		Степень	10					
9	4	0	0								
10											
11	Коэффициенты										
12	0	4	12	28	64	147,2	338,1333333	776,9904762	1785,295238	4102,162963	9425,70328
13											
14	Аргумент	Решение	Точно								
15	0,00	0	0								
16	0,05	0,233951969	0,233951969								
17	0,10	0,55631073	0,556311011								
18	0,15	1,013926717	1,013955303								
19	0,20	1,692725867	1,693546574								
20	0,25	2,763104823	2,775238426								
21	0,30	4,582056947	4,705549063								
22	0,35	7,910864416	8,978867238								
23	0,40	14,34946845	25,57764558								
24											

Рис. 15.2
Решение дифференциального уравнения методом разложения в степенной ряд

Таблица 15.1

Коэффициенты разложения функции в ряд

Ячейка	Значение	Комментарий
A7	=F7^2	Слагаемое y_0^2
B7	=2*F7	Коэффициент $2y_0$ в слагаемом $2y_0z(x)$
C7	1	Коэффициент 1 в слагаемом $z(x)^2$
A8	=4*F7	Коэффициент $4y_0$ в слагаемом $4y_0x$
B8	4	Коэффициент 4 в слагаемом $4xz(x)$
C8	0	Коэффициент возле слагаемого с произведением $xz(x)^2$ (такого слагаемого нет, поэтому коэффициент равен 0)
A9	4	Коэффициент 4 в слагаемом $4x^2$
B9	0	Коэффициент возле слагаемого с произведением $x^2z(x)$ (такого слагаемого нет, поэтому коэффициент равен 0)
C9	0	Коэффициент возле слагаемого с произведением $x^2z(x)^2$ (такого слагаемого нет, поэтому коэффициент равен 0)

формула =$A$12+РЯД.СУММ(A15;1;1;$B$12:$K$12), которой вычисляется полиномиальная сумма на основе коэффициентов ряда $B$12:$K$12, с начальным показателем степени 1 и шагом дискретности в приращении степени 1. Значение ячейки $A$12, соответствующее нулевой степени, прибавляем явно, дабы избежать ошибки возведения нуля в нулевую степень. Затем эта формула копируется в нижние ячейки, вплоть до заполнения диапазона B15:B23.

Значения на основе точного решения вычисляются в ячейках C15:C23. В ячейку C15 вводится довольно громоздкая формула =((2+2*A15*$F$7+$F$7^2)*SIN(КОРЕНЬ(2)*A15)-2*КОРЕНЬ(2)*A15*COS(КОРЕНЬ(2)*A15))/(КОРЕНЬ(2)*

COS(КОРЕНЬ(2)*A15)-$F$7*SIN(КОРЕНЬ(2)*A15)). Диапазон C15:C23 заполняется копированием этой формулы.

Сравнивая точное и приближенное решения приходим к выводу, что чем ближе значение аргумента к нулевой точке (точке начального условия), тем выше точность приближенного решения. Ситуация усугубляется еще и тем, что решение уравнения имеет особенность. Если характеризовать использованный подход в общем, то для увеличения точности решения можно увеличить количество слагаемых в разложении.

МЕТОД РУНГЕ — КУТТЫ

Посторонние разговоры прекратить.
Операция началась.
За мной!

Из к/ф «Старики-разбойники»

Далее мы приступаем к рассмотрению числовых методов решения дифференциальных уравнений, когда решение для искомой функции получается в виде таблицы. А именно, мы будем решать задачу Коши вида $y'(x) = f(x, y(x))$ с начальным условием $y(x_0) = y_0$. Задача состоит в том, чтобы вычислить значения y_n функции $y(x)$ в узловых точках x_n. Рассматривать будем равноотстоящие узлы с расстоянием h между соседними узлами. Поэтому можно полагать $x_n = x_0 + nh$, где индекс $n = 0, 1, 2, \ldots$

Существуют различные алгоритмы решения подобной задачи — в частности, *метод Эйлера*, который мы уже использовали в одной из предыдущих глав. Здесь рассмотрим подход, который носит название метода Рунге — Кутты.

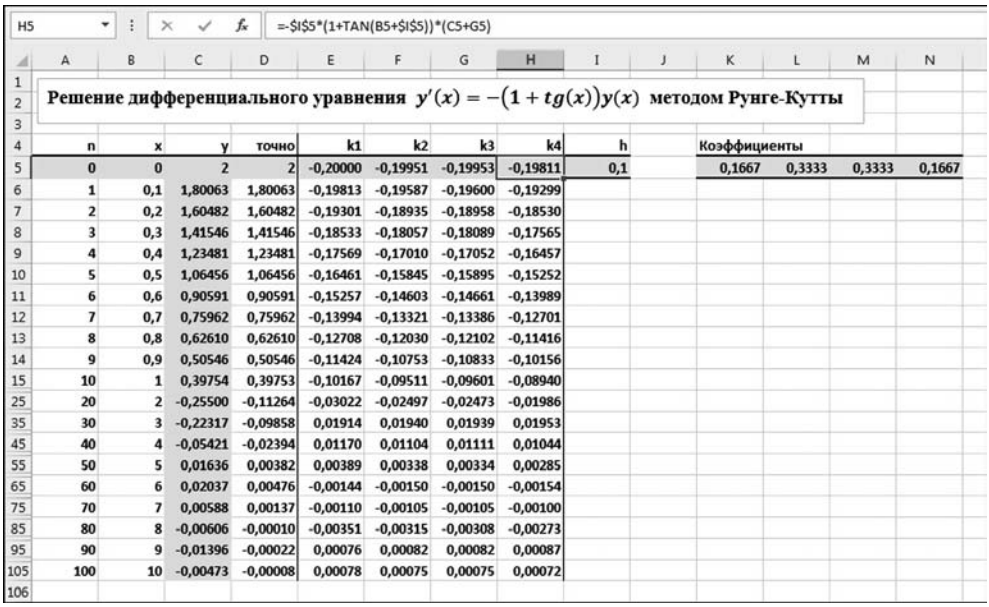
На заметку

Существуют различные варианты метода Рунге — Кутты. Все они относятся к так называемым одношаговым методам, поскольку в них значение функции y_{n+1} в точке x_{n+1} вычисляется на основе значения y_n в точке x_n, и другие узловые точки (значения функции в других узловых точках) явно в расчет не принимаются.

В методе Рунге — Кутты для вычисления значения y_{n+1} на основе значения y_n используется формула $y_{n+1} = y_n + \sum_{i=1}^{m} p_i k_i(h)$, где $k_1(h) = hf(x_n, y_n)$, $k_2(h) = hf(x_n + \alpha_2 h, y_n + \beta_{21} k_1(h))$, $k_3(h) = hf(x_n + \alpha_3 h, y_n + \beta_{31} k_1(h) + \beta_{32} k_2(h))$, и т. д., вплоть до $k_m(h) = hf(x_n + \alpha_m h, y_n + \beta_{m1} k_1(h) + \beta_{m2} k_2(h) + \ldots + \beta_{m\,m-1} k_{m-1}(h))$. Константы α_i, β_{ij} и p_i являются оптимизационными и выбираются на основе критериев точности для приближенного решения. Здесь мы рассмотрим метод Рунге — Кутты четвертого порядка, для которого в очередной узловой точке значение для функции-решения определяется соотношением y_{n+1}

$y_{n+1} = y_n + \frac{1}{6}\left(k_1(h) + 2k_2(h) + 2k_3(h) + k_4(h)\right)$ и при этом $k_1(h) = hf(x_n, y_n)$, $k_2(h) =$

$= hf\left(x_n + \frac{h}{2}, y_n + \frac{k_1(h)}{2}\right)$, $k_3(h) = hf\left(x_n + \frac{h}{2}, y_n + \frac{k_2(h)}{2}\right)$ и $k_4(h) = hf(x_n + h, y_n + k_3(h))$.



H5 | =-$I$5*(1+TAN(B5+$I$5))*(C5+G5)

Решение дифференциального уравнения $y'(x) = -(1 + tg(x))y(x)$ методом Рунге-Кутты

n	x	y	точно	k1	k2	k3	k4	h	Коэффициенты			
0	0	2	2	-0,20000	-0,19951	-0,19953	-0,19811	0,1	0,1667	0,3333	0,3333	0,1667
1	0,1	1,80063	1,80063	-0,19813	-0,19587	-0,19600	-0,19299					
2	0,2	1,60482	1,60482	-0,19301	-0,18935	-0,18958	-0,18530					
3	0,3	1,41546	1,41546	-0,18533	-0,18057	-0,18089	-0,17565					
4	0,4	1,23481	1,23481	-0,17569	-0,17010	-0,17052	-0,16457					
5	0,5	1,06456	1,06456	-0,16461	-0,15845	-0,15895	-0,15252					
6	0,6	0,90591	0,90591	-0,15257	-0,14603	-0,14661	-0,13989					
7	0,7	0,75962	0,75962	-0,13994	-0,13321	-0,13386	-0,12701					
8	0,8	0,62610	0,62610	-0,12708	-0,12030	-0,12102	-0,11416					
9	0,9	0,50546	0,50546	-0,11424	-0,10753	-0,10833	-0,10156					
10	1	0,39754	0,39753	-0,10167	-0,09511	-0,09601	-0,08940					
20	2	-0,25500	-0,11264	-0,03022	-0,02497	-0,02473	-0,01986					
30	3	-0,22317	-0,09858	0,01914	0,01940	0,01939	0,01953					
40	4	-0,05421	-0,02394	0,01170	0,01104	0,01111	0,01044					
50	5	0,01636	0,00382	0,00389	0,00338	0,00334	0,00285					
60	6	0,02037	0,00476	-0,00144	-0,00150	-0,00150	-0,00154					
70	7	0,00588	0,00137	-0,00110	-0,00105	-0,00105	-0,00100					
80	8	-0,00606	-0,00010	-0,00351	-0,00315	-0,00308	-0,00273					
90	9	-0,01396	-0,00022	0,00076	0,00082	0,00082	0,00087					
100	10	-0,00473	-0,00008	0,00078	0,00075	0,00075	0,00072					

Рис. 15.3
Решение дифференциального уравнения методом Рунге — Кутты

Процесс вычисления решения дифференциального уравнения методом Рунге — Кутты достаточно легко организовать в рабочем документе Excel. Покажем это на примере решения уравнения $y'(x) = -(1 + \mathrm{tg}(x))y(x)$ с начальным условием $y(0) = y_0$. Сразу отметим, что точное решение этой задачи имеет вид $y(x) = y_0\exp(-x)\cos(x)$, и именно с ним мы будем сверять результаты числовых расчетов. Документ, в котором выполняются эти расчеты, представлен в документе на рисунке 15.3.

В ячейках A5:A105 отображается индекс узловой точки: в ячейку A5 вводится нулевое значение, в ячейку A6 вводится формула =A5+1, и эта формула копируется во все остальные ячейки диапазона. В ячейках B5:B105 отображаются значения узловых точек. Этот диапазон заполняем так: в ячейку B5 вводим значение аргумента, для которого задано начальное условие — в данном случае значение 0. В ячейку B6 вводим формулу =$B$5+A6*$I$5, на основе которой копированием заполняются остальные ячейки в диапазоне B5:B105. В формуле =$B$5+A6*$I$5 использована ссылка на индекс узловой точки A6, абсолютная ссылка $B$5 на ячейку с начальным значением аргумента, а также абсолютная ссылка $I$5 на ячейку, содержащая значение шага по аргументу (значение переменной h). Таким образом, данная формула является реализацией соотношения $x_n = x_0 + nh$.

В ячейках E5:H105 вычисляются значения выражений $k_i(h)$ $(i = 1, 2, 3, 4)$ для каждого итерационного шага. Заполняем диапазон так. В ячейку E5 вводится формула =-$I$5*(1+TAN(B5))*C5, в ячейку F5 вводится формула =-$I$5*(1+TAN(B5+$I$5/2))*(C5+E5/2), в ячейку G5 вводится формула =-$I$5*(1+TAN(B5+$I$5/2))*(C5+F5/2), а в ячейку H5 вводится формула =-$I$5*

(1+TAN(B5+$I$5))*(C5+G5). Затем диапазон ячеек E5:H5 выделяется, и с помощью маркера заполнения содержимое копируется в остальные ячейки диапазона E5:H105.

Для вычисления значения функции в узловых точках, в ячейку C5 вводится значение функции, определяемое начальным условием (значение 2). В ячейку C6 вводится формула =C5+СУММПРОИЗВ(E5:H5;$K$5:$N$5). Этой формулой вычисляется значение функции для новой узловой точки. Это реализация соотношения $y_{n+1} = y_n + \frac{1}{6}\left(k_1(h) + 2k_2(h) + 2k_3(h) + k_4(h)\right)$, причем коэффициенты соответствующей линейной комбинации вычисляются в ячейках диапазона K5:N5, на который в формуле =C5+СУММПРОИЗВ(E5:H5;$K$5:$N$5) имеется абсолютная ссылка.

На заметку

Значения в диапазоне вычисляются так: в ячейки K5 и N5 вводится формула =1/6, а в ячейки L5 и M5 вводится формула =1/3.

Ячейки в столбце C заполняются (вплоть до 105-й строки) копированием формулы из ячейки C6. Точные значения в узловых точках вычисляем копированием формулы =$C$5*EXP(-B5)*COS(B5) из ячейки D5 в ячейки диапазона D5:D105.

Сравнивая приближенное и точное решения, легко заметить, что точность приближенного решения вполне приемлемая, даже принимая во внимание значительный шаг приращения по аргументу. Очевидно также, что точность вычислений тем выше, чем меньше индекс узловой точки.

На заметку

На рисунке 15.3 часть ячеек ради удобства восприятия скрыта.

В случае необходимости, вместо того, чтобы явно вычислять все вспомогательные параметры в рабочем листе Excel, можем написать функцию пользователя, которой бы вычислялось приближенное решение дифференциального уравнения методом Рунге — Кутты. Программный код такой функции представлен в листинге 15.14.

Листинг 15.14. Функция для вычисления решения уравнения методом Рунге — Кутты

```
Function РУНГКУТ(f, x0, y0, t, Optional n = 100)
 ' Шаг по аргументу
 Dim h As Double
 h = (t - x0) / n
 ' Массив для записи коэффициентов формулы Рунге–Кутты
 Dim p(1 To 4) As Double
 p(1) = 1 / 6
 p(2) = 1 / 3
 p(3) = 1 / 3
 p(4) = 1 / 6
 ' Значение функции-решения уравнения
```

```
    Dim y As Double
    y = y0
    ' Аргумент функции-решения уравнения
    Dim x As Double
    x = x0
    ' Массив для записи параметров формулы Рунге–Кутты
    Dim k(1 To 4) As Double
    ' Объект для вызова метода, определяющего уравнение
    Set obj = New DEquations
    ' Индексные переменные
    Dim i As Integer, j As Integer
    ' Вычисление результата
    For i = 1 To n
      ' Параметры для формулы Рунге–Кутты
      k(1) = h * CallByName(obj, f, VbMethod, x, y)
      k(2) = h * CallByName(obj, f, VbMethod, x + h / 2, y + k(1) / 2)
      k(3) = h * CallByName(obj, f, VbMethod, x + h / 2, y + k(2) / 2)
      k(4) = h * CallByName(obj, f, VbMethod, x + h, y + k(3))
      ' Реализация формулы Рунге–Кутты
      For j = 1 To 4
        y = y + p(j) * k(j)
      Next j
      ' Аргумент для следующего цикла для функции-решения уравнения
      x = x + h
    Next i
    ' Результат
    РУНГКУТ = y
End Function
```

Функция называется РУНГКУТ() и у нее пять аргументов (последний необязательный):

- переменная f является ссылкой (текстовым именем) на метод, определяющий правую часть дифференциального уравнения $y'(x) = f(x, y(x))$; предполагается, что это функция $f(x, y)$ двух переменных;
- значение аргумента x0, для которого задается начальное условие $y(x_0) = y_0$;
- значение функции y0 в начальной точке (значение y_0 функции в начальном условии $y(x_0) = y_0$);
- значение аргумента t, для которого необходимо найти значение функции-решения уравнения;
- необязательный аргумент n, определяющий количество интервалов, на которые разбивается диапазон значений аргумента от x0 до t — по умолчанию значение аргумента равно 100.

Командой h = (t-x0)/n вычисляется шаг приращения по аргументу. Для записи параметров, используемых в формуле Рунге — Кутты, создаются два статических массива по четыре элемента в каждом: это массивы p и k. Элементам массива сразу присваиваются числовые значения. Массив k только

объявляется. Значения его элементов будут меняться в процессе выполнения оператора цикла.

Для записи значения искомой функции объявляется переменная y (с начальным значением y0). Для записи значения аргумента функции объявляется переменная x (с начальным значением x0).

Также для того, чтобы вызывать метод, определяющий правую часть дифференциального уравнения, мы создаем объект obj класса DEquations.

На заметку

В проект добавляется модуль класса, которому задается имя DEquations. В этом модуле класса описывается функция (как увидим далее, она называется DEqn()), которая определяет правую часть решаемого уравнения.

Для вычисления решения в нужной точке необходимо последовательно вычислить решения во всех предыдущих точках. Поэтому запускается оператор цикла с индексной переменной i, изменяющейся в пределах от 1 до n, и за каждый цикл выполняются следующие команды и операции:

- последовательно для данной узловой точки (определяется значением индексной переменной оператора цикла) в явном виде вычисляются значения параметров $k_1(h)$, $k_2(h)$, $k_3(h)$ и $k_4(h)$ — в вычислениях используется функция CallByName(), двумя последними аргументами которой передаются значения аргументов функции $f(x, y)$. Ссылка на имя метода, определяющего функцию $f(x, y)$, передается вторым аргументом функции CallByName();
- запускается внутренний оператор цикла с индексной переменной j, благодаря чему по формуле Рунге — Кутты вычисляется функция для данной узловой точки;
- командой x = x+h вычисляется значение аргумента для следующей узловой точки.

По окончании вычислений переменная y возвращается как значение функции РУНГКУТ().

Для иллюстрации работы разработанной нами функции решим дифференциальное уравнение $y'(x) = x^2 - y(x)$ с начальным условием $y(x_0) = y_0$. Точное решение этой задачи дается соотношением $y(x) = x_2 - 2x + 2 + \exp(x_0 - x) \times$ $\times (y_0 - x_0^2 + 2x_0 - 2)$. Мы попытаемся для этого уравнения реализовать процедуру вычисления приближенного решения по методу Рунге — Кутты, для чего, как отмечалось, используем описанную выше функцию РУНГКУТ(). Кроме этого, в модуле класса DEquations описываем функцию DEqn(), которая определяет функцию в правой части дифференциального уравнения, т. е. функцию $f(x, y) = x^2 - y$. Программный код функции DEqn() приведен в листинге 15.15 и, думается, особых комментариев не требует.

Листинг 15.15. Функция в правой части дифференциального уравнения

```
Function DEqn(x As Double, y As Double) As Double
  DEqn = x ^ 2 - y
End Function
```

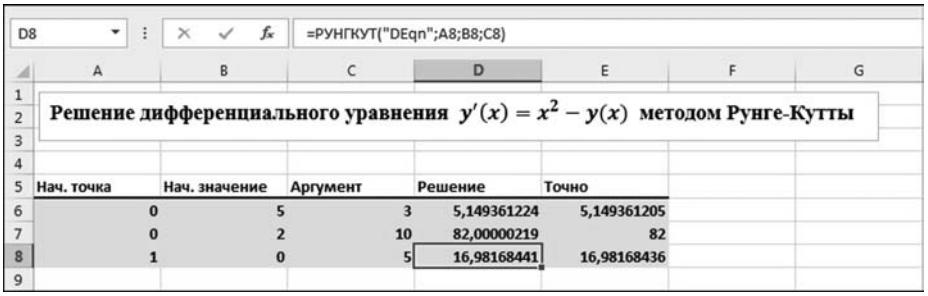


Рис. 15.4
Решение дифференциального уравнения методом Рунге — Кутты с помощью пользовательской функции

Рабочий документ, в котором реализованы все вычисления, представлен на рисунке 15.4.

В ячейках столбца A вводятся значения для аргумента в начальной точке (значение параметра x_0), в ячейках столбца B вводятся значения функции в начальной точке (параметр y_0), в ячейках столбца C указываются значения аргумента x для вычисления в них значения функции $y(x)$. Приближенное решение вычисляется в ячейках столбца D, а точное решение — в ячейках столбца E. В частности, в ячейку D6 вводится формула =РУНГКУТ("DEqn";A6;B6;C6), а в ячейку E6 вводится формула =C6^2-2*C6+2+(B6-2+2*A6-A6^2)*EXP(A6-C6). Затем эти формулы копируются в нижние ячейки.

На заметку

Желающие могут поэкспериментировать с начальными условиями и другими параметрами вычислений, однако в целом можно заключить, что точность вычислений вполне приемлемая.

МЕТОД АДАМСА

Я мечтал об этом всю свою сознательную жизнь.

Из к/ф «Ирония судьбы, или С легким паром!»

Метод Адамса относится к *конечно-разностным методам*. В отличие от одношаговых методов (таких, как метод Эйлера или Рунге — Кутты), в которых для расчета значения функции в узловой точке используется значение этой функции в соседней точке, в конечно-разностных схемах для вычисления значения в узловой точке используют значения в нескольких соседних узловых точках. Например, в методе Адамса третьего порядка вычисление значения функции y_{n+1} в узловой точке x_{n+1} выполняется по формуле

$$y_{n+1} = y_n + \frac{h}{12}\left(23f(x_n, y_n) - 16f(x_{n-1}, y_{n-1}) + 5f(x_{n-2}, y_{n-2})\right).$$

Как и ранее, здесь речь идет о решении уравнения $y'(x) = f(x, y(x))$ с начальным условием $y(x_0) = y_0$, и использованы обозначения $h = x_{k+1} - x_k$, $x_n = x_0 + nh$, а через y_k обозначается значение (вычисляемое или вычисленное) функции $y(x)$ в точке x_k.

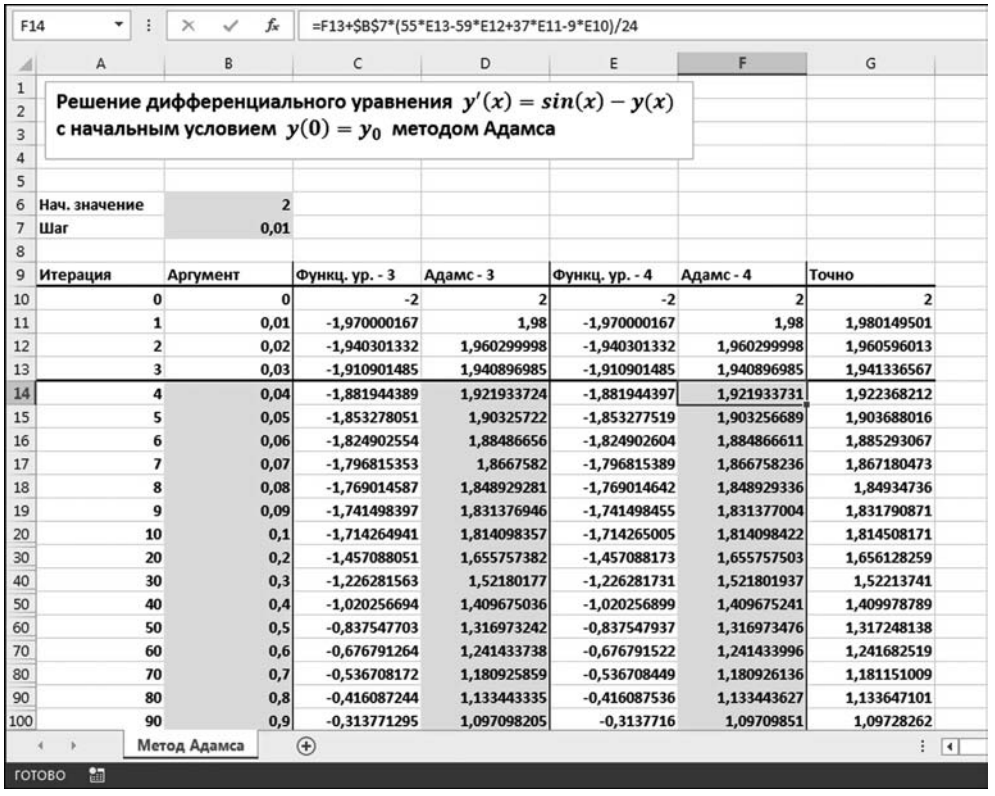
В методе Адамса четвертого порядка рекуррентное соотношение для определения значения искомой функции в новой узловой точке имеет вид

$$y_{n+1} = y_n + \frac{h}{24}\left(55f(x_n,\ y_n) - 59f(x_{n-1},\ y_{n-1}) + \right.$$
$$\left. + 37f(x_{n-2},\ y_{n-2}) - 9f(x_{n-3},\ y_{n-3})\right).$$

На заметку

Очевидно, что приведенные выше соотношения не могут применяться для вычисления значения функции в нескольких начальных точках. Для вычисления функции в начальных узловых точках используют одношаговые методы: например, в точках с индексами 1 и 2 значения функции вычисляются одношаговым методом (скажем, Эйлера или Рунге — Кутты), а во всех остальных — по методу Адамса третьего порядка. Если мы хотим использовать метод Адамса четвертого порядка, то одношаговыми методами придется вычислять значения в узловых точках с индексами от 1 по 3 включительно.

Рабочий документ, в котором методом Адамса решается уравнение $y'(x) = \sin x - y(x)$ с начальным условием $y(0) = y_0$, представлен на рисунке 15.5.



F14 | =F13+$B$7*(55*E13-59*E12+37*E11-9*E10)/24

Решение дифференциального уравнения $y'(x) = sin(x) - y(x)$ с начальным условием $y(0) = y_0$ методом Адамса

	A	B	C	D	E	F	G
6	Нач. значение	2					
7	Шаг	0,01					
9	Итерация	Аргумент	Функц. ур. - 3	Адамс - 3	Функц. ур. - 4	Адамс - 4	Точно
10	0	0	-2	2	-2	2	2
11	1	0,01	-1,970000167	1,98	-1,970000167	1,98	1,980149501
12	2	0,02	-1,940301332	1,960299998	-1,940301332	1,960299998	1,960596013
13	3	0,03	-1,910901485	1,940896985	-1,910901485	1,940896985	1,941336567
14	4	0,04	-1,881944389	1,921933724	-1,881944397	1,921933731	1,922368212
15	5	0,05	-1,853278051	1,90325722	-1,853277519	1,903256689	1,903688016
16	6	0,06	-1,824902554	1,88486656	-1,824902604	1,884866611	1,885293067
17	7	0,07	-1,796815353	1,8667582	-1,796815389	1,866758236	1,867180473
18	8	0,08	-1,769014587	1,848929281	-1,769014642	1,848929336	1,84934736
19	9	0,09	-1,741498397	1,831376946	-1,741498455	1,831377004	1,831790871
20	10	0,1	-1,714264941	1,814098357	-1,714265005	1,814098422	1,814508171
30	20	0,2	-1,457088051	1,655757382	-1,457088173	1,655757503	1,656128259
40	30	0,3	-1,226281563	1,52180177	-1,226281731	1,521801937	1,52213741
50	40	0,4	-1,020256694	1,409675036	-1,020256899	1,409675241	1,409978789
60	50	0,5	-0,837547703	1,316973242	-0,837547937	1,316973476	1,317248138
70	60	0,6	-0,676791264	1,241433738	-0,676791522	1,241433996	1,241682519
80	70	0,7	-0,536708172	1,180925859	-0,536708449	1,180926136	1,181151009
90	80	0,8	-0,416087244	1,133443335	-0,416087536	1,133443627	1,133647101
100	90	0,9	-0,313771295	1,097098205	-0,3137716	1,09709851	1,09728262

Метод Адамса

ГОТОВО

Рис. 15.5
Решение дифференциального уравнения методом Адамса

Мы для сравнения вычисляли решение методом Адамса как третьего, так и четвертого порядка. Кроме того, мы учли, что точное решение уравнения (с соответствующим граничным условием) имеет вид $y(x) = \frac{1}{2}(\sin(x) - \cos(x) + \exp(-x)(1 + 2y_0))$, поэтому в рабочем документе также приводятся и «точные» значения в узловых точках искомой функции, вычисленные по аналитической формуле.

В столбце A, начиная с ячейки A10 (значение 0) отображаются индексы узловых точек. В столбце B вычисляются значения аргумента в узловых точках: в ячейку B10 вводится формула =A10*$B$7, и затем она копируется в нижние ячейки. Эта формула содержит абсолютную ссылку на ячейку B7, в которую вводятся значения для шага приращения аргумента.

Значения в ячейках столбца C вычисляются так: в ячейку C10 вводим формулу =SIN(B10)-D10 и копируем ее в ячейки внизу. Данная формула является реализацией выражения для функции $f(x, y) = \sin x - y$ (функция в правой части дифференциального уравнения). Формула содержит ссылку на ячейку B10 со значением аргумента x, и ссылку на ячейку D10 со значением аргумента y.

Столбец D, в котором вычисляются приближенные значения решения уравнения по методу Адамса третьего порядка, заполняется следующим образом:

- в ячейку D10 вводим формулу =B6 со ссылкой на ячейку B6, содержащую значение функции в начальной точке;
- в ячейку D11 вводится формула =D10+$B$7*(SIN(B10)-D10), которой по методу Эйлера (соотношение $y_{n+1} = y_n + hf(x_n, y_n)$) вычисляется решение для узловой точки с индексом 1. Эта формула копируется в ячейки D12 и D13;
- в ячейку D14 вводится формула =D13+$B$7*(23*C13-16*C12+5*C11)/12, представляющая собой реализацию формулы для метода Адамса третьего порядка (соотношение $y_{n+1} = y_n + \frac{h}{12}(23f(x_n, y_n) - 16f(x_{n-1}, y_{n-1}) + 5f(x_{n-2}, y_{n-2}))$). Эта формула содержит ссылки на значения функции уравнения в трех предыдущих точках. Формулу копируем в нижние ячейки.

Два соседних столбца (E и F) предназначены для вычисления приближенного решения по методу Адамса четвертого порядка. Для этого в ячейку F10 вводим формулу =SIN(B10)-F10 и копируем эту формулу в нижние ячейки.

На заметку

Формулу =SIN(B10)-F10 можно было бы не вводить в ячейку F10, а скопировать из ячейки C10. Правда, для этого в ячейку C10 нужно вводить не формулу =SIN(B10)-D10, а формулу =SIN($B10)-D10 (т. е. использовать смешанную ссылку на ячейку B10).

Вычисления по методу Адамса четвертого порядка выполняются в столбце F следующим образом:

- в ячейку F10 вводим формулу =B6, и эта формула копируется в нижние ячейки, вплоть до ячейки F13 — мы исходим из того, что нет необходимости дважды выполнять одни и те же вычисления (имеется в виду вы-

численение значения для функции-решения уравнения в начальных узловых точках по методу Эйлера);

- в ячейку F14 вводится формула =F13+$B$7*(55*E13-59*E12+37*E11-9*E10)/24, которой реализуется соотношение $y_{n+1} = y_n + \frac{h}{24}(55f(x_n, y_n) - 59f(x_{n-1}, y_{n-1}) + 37f(x_{n-2}, y_{n-2}) - 9f(x_{n-3}, y_{n-3}))$ для метода Адамса четвертого порядка. Эта формула копируется в нижние ячейки.

Наконец, ячейки в столбце G заполняются копированием из ячейки G10 формулы =(SIN(B10)-COS(B10)+EXP(-B10)*(1+2*$B$6))/2.

Специфика разработанного нами документа такова, что при изменении значения функции в начальной точке (ячейка B6) или шага приращения по аргументу (ячейка B7) весь документ, включая ячейки с точным значением для решения, пересчитывается автоматически.

Создадим функцию пользователя для решения уравнений методом Адамса. Используем такой же подход, как и в случае, когда мы разрабатывали функцию для решения уравнений методом Рунге — Кутты: функция $f(x, y)$ двух аргументов из правой части уравнения $y'(x) = f(x, y(x))$ описывается как метод в модуле класса проекта, и затем имя этого метода передается в качестве текстового параметра функции, в которой собственно и реализуется метод Адамса. Такой подход удобен, поскольку позволяет использовать одну и ту же функцию для решения различных уравнений.

Программный код функции АДАМС(), в которой реализованы все эти амбициозные планы, представлен в листинге 15.16.

Листинг 15.16. Функция для решения дифференциального уравнения методом Адамса

```
Function АДАМС(f, x0, y0, t, Optional method = True, Optional dx = 0.001)
 ' Переменная для записи количества узловых точек (кроме начальной)
 Dim n As Integer
 ' Индекс последней узловой точки
 n = Fix((t - x0) / dx)
 ' Общее количество точек не меньше пяти
 If (n < 4) Then n = 4
 ' Переменная для приращения аргумента
 Dim h As Double
 ' Значение приращения аргумента
 h = (t - x0) / n
 ' Индексная переменная для оператора цикла
 Dim i As Integer
 ' Объект для вызова метода, определяющего функцию уравнения
 Set obj = New DEquations
 ' Массив для записи значений узловых точек
 ReDim x(0 To n) As Double
 ' Начальная узловая точка
 x(0) = x0
 ' Массив для записи значений функции-решения в узловых точках
```

```
ReDim y(0 To n) As Double
' Начальное значение функции-решения
y(0) = y0
' Массив для записи значений функции в правой части уравнения
ReDim Fn(0 To n) As Double
' Начальное значение функции в правой части уравнения
Fn(0) = CallByName(obj, f, VbMethod, x(0), y(0))
' Переменная для определения индекса узловой точки,
' до которой вычисления выполняются по методу Эйлера
Dim k As Integer
' Индекс узловой точки, до которой вычисления проводятся
' по методу Эйлера: 3 для метода Адамса 4-го порядка,
' и 2 для метода Адамса 3-го порядка
If method Then k = 3 Else k = 2
' Оператор цикла для вычисления значений по методу Эйлера
For i = 1 To k
 ' Значение функции-решения уравнения
  y(i) = y(i - 1) + h * Fn(i - 1)
 ' Значение узловой точки
  x(i) = x(i - 1) + h
 ' Значение функции в правой части уравнения
 Fn(i) = CallByName(obj, f, VbMethod, x(i), y(i))
Next i
' Определяем порядок метода Адамса
If method Then
 ' Метод Адамса 4-го порядка
  For i = k + 1 To n
  ' Значение функции-решения в узловой точке
   y(i) = y(i - 1) + h / 24 * (55 * Fn(i - 1) - 59 * _
   Fn(i - 2) + 37 * Fn(i - 3) - 9 * Fn(i - 4))
  ' Значение узловой точки
   x(i) = x(i - 1) + h
  ' Значение функции в правой части уравнения
  Fn(i) = CallByName(obj, f, VbMethod, x(i), y(i))
 Next i
 ' Метод Адамса 3-го порядка
Else
  For i = k + 1 To n
  ' Значение функции-решения в узловых точках
   y(i) = y(i - 1) + h / 12 * (23 * Fn(i - 1) - 16 * _
   Fn(i - 2) + 5 * Fn(i - 3))
  ' Значение узловых точек
   x(i) = x(i - 1) + h
  ' Значение функции в правой части уравнения
  Fn(i) = CallByName(obj, f, VbMethod, x(i), y(i))
 Next i
```

```
    End If
    ' Значение функции
    АДАМС = y(n)
End Function
```

Функция АДАМС() имеет шесть аргументов (два последних необязательные). Через f обозначена ссылка на имя метода, определяющего функцию в правой части дифференциального уравнения. Аргумент x0 обозначает начальную точку (по аргументу), в которой задается начальное условие. Значение функции-решения в начальной точке задается аргументом y0. Через t определяется значение аргумента, для которого вычисляется значение функции-решения уравнения. Это обязательные аргументы. Два необязательных (опционных) аргумента определяют порядок метода Адамса и величину для шага приращения аргумента. Аргумент method имеет по умолчанию значение True, и при таком значении аргумента вычисления производятся по методу Адамса четвертого порядка. Аргумент dx имеет значение по умолчанию 0.001 и определяет (примерный) шаг для приращения аргумента.

Количество частей, на которые разбивается интервал от начальной точки до точки, для которой вычисляется решение уравнения, записывается в переменную n и вычисляется командой n = Fix((t-x0)/dx). В данном случае мы берем целую часть от деления длины интервала t-x0 между начальной и конечной точками на величину шага приращения по аргументу dx. Кроме этого, используемый для вычисления решения метод предполагает наличие нескольких узловых точек. Мы требуем, чтобы узловых точек было не меньше пяти (т. е. количество интервалов — не меньше четырех). Поэтому в условном операторе проверяем условие n<4, и если оно выполнено, то командой n = 4 задаем минимально необходимое значение для переменной n.

Командой h = (t-x0)/n определяем шаг приращения для аргумента (расстояние между соседними узловыми точками). Помимо этого, создаем несколько массивов (в каждом из них индексация элементов выполняется от 0 до n): массив x для записи узловых точек, массив y для записи значений функции-решения уравнения, а также массив Fn для записи значений функции в правой части уравнения. В каждом из этих массивов определяется начальный (с нулевым индексом) элемент: x(0) = x0 (начальное значение для аргумента), y(0) = y0 (начальное значение функции-решения) и Fn(0) = CallByName (obj,f,VbMethod,x(0),y(0)) — данной командой вычисляется значение функции в правой части дифференциального уравнения в начальной точке (при значениях аргумента и искомой функции-решения, определяемых начальным условием). При этом объект obj, из которого вызывается метод, передаваемый по ссылке f, заранее создается командой Set obj = New DEquations.

На заметку

Таким образом, в проект необходимо добавить модуль класса с названием DEquations и описать там функцию двух переменных, которая будет определять выражение в правой части дифференциального уравнения. Имя этой функции (в текстовом формате) будет передаваться функции АДАМС() в качестве аргумента.

Значения в нескольких начальных узловых точках мы будем вычислять по методу Эйлера (формула $y_i = y_{i-1} + hf(x_i, y_i)$). Если используется метод Адамса четвертого порядка, то по методу Эйлера вычисляются значения функции-решения уравнения до узловой точки с индексом 3 включительно. Если используется метод Адамса третьего порядка, вычислять по методу Эйлера значения функции-решения следует до узловой точки с индексом 2. Значение индекса узловой точки, до которой производятся вычисления по методу Эйлера, определяем с помощью условного оператора и записываем в переменную k. Затем запускается оператор цикла с индексной переменной i, пробегающей значения от 1 до k. За каждый цикл выполняется три команды:

- командой y(i) = y(i-1)+h*Fn(i-1) вычисляется значение функции-решения в очередной узловой точке;
- значение узловой точки вычисляется командой x(i) = x(i-1)+h;
- значение функции в правой части уравнения для вычисленных значений x(i) и y(i) вычисляется по формуле Fn(i) = CallByName(obj,f,VbMethod,x(i),y(i)).

После того как по методу Эйлера вычислены значения в начальных точках, во всех последующих точках значения для функции-решения уравнения вычисляются по методу Адамса третьего или четвертого порядков — в зависимости от значения аргумента method. И в том, и в другом случае запускается оператор цикла, с командами, аналогичными рассмотренным выше. Разница только в способе вычисления значения функции-решения дифференциального уравнения в очередной узловой точке. Для метода Адамса четвертого порядка следующее значение функции вычисляется на основе данных по четырем предыдущим точкам по формуле y(i) = y(i-1)+h/24*(55*Fn(i-1)-59*Fn(i-2)+37*Fn(i-3)-9*Fn(i-4)), а для метода Адамса третьего порядка новое значение функции-решения вычисляется по формуле y(i) = y(i-1)+h/12*(23*Fn(i-1)-16*Fn(i-2)+5*Fn(i-3)), в которой используются данные вычислений по трем предыдущим узловым точкам.

Значение y(n) в последней узловой точке возвращается как результат функции АДАМС().

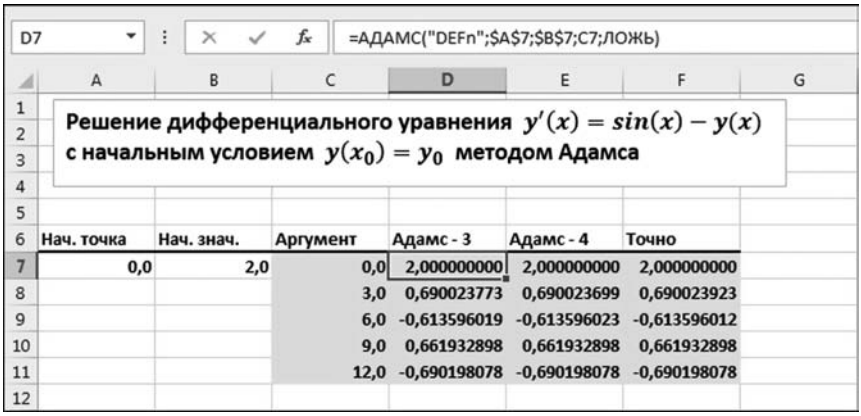
Помимо этой функции, нам нужно описать функцию от двух аргументов, определяющую правую часть решаемого дифференциального уравнения. Для решения уравнения $y'(x) = \sin x - y(x)$ в модуле класса DEquations описываем функцию DEFn(), задающую зависимость $f(x, y) = \sin x - y$ (листинг 15.17).

Листинг 15.17. Функция, определяющая решаемое методом Адамса уравнение

```
Function DEFn(x As Double, y As Double) As Double
  DEFn = Sin(x) - y
End Function
```

Документ, в котором с использованием разработанных и описанных выше пользовательских функций решается уравнение $y'(x) = \sin x - y(x)$ с начальным условием общего вида $y(x_0) = y_0$, представлен на рисунке 15.6.

В ячейку A7 вводится значение для аргумента x_0 (точка, в которой задается начальное условие). В ячейку B4 вводится значение искомой функции в



	A	B	C	D	E	F	G
1							
2							
3							
4							
5							
6	Нач. точка	Нач. знач.	Аргумент	Адамс - 3	Адамс - 4	Точно	
7	0,0	2,0	0,0	2,000000000	2,000000000	2,000000000	
8			3,0	0,690023773	0,690023699	0,690023923	
9			6,0	-0,613596019	-0,613596023	-0,613596012	
10			9,0	0,661932898	0,661932898	0,661932898	
11			12,0	-0,690198078	-0,690198078	-0,690198078	
12							

Рис. 15.6
Решение дифференциального уравнения методом Адамса
с помощью пользовательских функций

начальной точке (параметр y_0). Аргументы, для которых вычисляется решение, заносятся в ячейки C7:C11. При этом в ячейку C7 мы ввели формулу =A7, в ячейку C8 ввели формулу =C7+3, а затем скопировали ее в ячейки C9:C11 — в результате получаем несколько значений для аргумента с шагом дискретности 3. Решения (приближенные) вычисляются в ячейках диапазона D7:E11: а именно, в ячейках D7:D11 представлены значения аргументов, а в ячейках C7:C11 решение вычисляется по методу Адамса третьего порядка. Вычисления по методу Адамса четвертого порядка выполняются в ячейках E7:E11. Ячейки заполняются так: в ячейку D7 вводим формулу =АДАМС (“DEFn”;$A$7;$B$7;C7;ЛОЖЬ) и копируем ее в остальные ячейки диапазона D7:D11, а в ячейку E7 вводится формула =АДАМС(“DEFn”;$A$7;$B$7;C7) и копируется в нижние ячейки до заполнения диапазона E7:E11. Точное решение вычисляется в ячейках F7:F11 копированием формулы =(SIN(C7)-COS(C7)+EXP($A$7-C7)*(2*$B$7+COS ($A$7)-SIN($A$7)))/2 из ячейки F7 в нижние ячейки диапазона. В данном случае мы воспользовались тем, что точным решением для задачи $y'(x) = = \sin x - y(x)$ с начальным условием $y(x_0) = y_0$ является функция

$$y(x) = \frac{1}{2}\big(\sin(x) - \cos(x) + \exp(x_0 - x)(2y_0 + \cos(x_0) - \sin(x_0))\big).$$

МЕТОД МИЛНА

Это же вам не лезгинка, а твист!

Из к/ф «Кавказская пленница»

Метод Милна относится к методам типа *предиктор-корректор*, в которых сначала вычисляется приближенное значение для функции в узловой точке, а потом это значение уточняется. Мы воспользуемся следующей версией метода Милна для решения уравнения $y'(x) = f(x, y(x))$ с начальным

условием $y(x_0) = y_0$: в узловой точке x_{n+1} значение функции y_{n+1} (первое приближение) вычисляется по формуле

$$y_{n+1} = y_{n-3} + \frac{4h}{3}\left(2f(x_n, y_n) - f(x_{n-1}, y_{n-1}) + 2f(x_{n-2}, y_{n-2})\right)$$

— через h обозначено расстояние между соседними узловыми точками (шаг приращения по аргументу). Затем это значение уточняется (второе приближение) на основе формулы

$$y_{n+1} = y_{n-3} + \frac{h}{3}\left(f(x_{n+1}, y_{n+1}) + 4f(x_n, y_n) + f(x_{n-1}, y_{n-1})\right),$$

причем в правой части этого выражения при вычислениях используется значение y_{n+1} в первом приближении, т. е. рассчитанное по первой формуле. Другими словами, сначала мы вычисляем оценочное значение для функции $y(x)$ в узловой точке (первое приближение), а потом на основе этого значения вычисляем второе приближение — более точное значение (для этой же узловой точки).

На заметку

Как и в методе Адамса, в методе Милна несколько начальных значений (а именно, в четырех узловых точках, включая начальную) нужно рассчитать с помощью одношаговых методов.

На рисунке 15.7 представлен рабочий документ Excel, в котором метод Милна применяется для вычисления решения дифференциального уравнения $y'(x) = x\exp(-x) - 2y(x)$ с начальным условием общего вида $y(x_0) = y_0$.

F11 | =F9+$G$4/3*(C11+4*D10+D9)

Решение методом Милна дифференциального уравнения $y'(x) = xe^{-x} - 2y(x)$ с начальным условием $y(x_0) = y_0$

	F	G
Нач. точка		1
Нач. знач.		0,5
Шаг		0,1

Индекс	Узл. точка	Уравнение - 1	Уравнение - 2	1-е прибл.	2-е прибл.	Точно
0	1	-0,632120559	-0,632120559	0,5	0,5	0,5
1	1,1	-0,507417696	-0,519887982	0,436787944	0,443023087	0,442652485
2	1,2	-0,408165237	-0,431807798	0,384799146	0,396620426	0,395398865
3	1,3	-0,328945401	-0,362874202	0,341618366	0,358582766	0,356165356
4	1,4	-0,299106499	-0,302511029	0,322171124	0,323873389	0,323303268
5	1,5	-0,25648068	-0,260510359	0,29558796	0,2976028	0,295504801
6	1,6	-0,218404266	-0,220515235	0,270719347	0,271774832	0,271735017
7	1,7	-0,19712563	-0,19533048	0,25384381	0,252946235	0,251176949
8	1,8	-0,165897729	-0,168162673	0,231717864	0,232850336	0,23318737
9	1,9	-0,155818467	-0,153458785	0,219999422	0,218819581	0,217261201
10	2	-0,131701535	-0,134116815	0,201186051	0,202393691	0,203002925
11	2,1	-0,128440587	-0,125922887	0,192799543	0,191540693	0,19010365
12	2,2	-0,108694649	-0,111253566	0,176230798	0,177510257	0,178322767
13	2,3	-0,109759055	-0,107106299	0,170177198	0,16885082	0,167473286
14	2,4	-0,092449679	-0,09515553	0,155086383	0,156439309	0,157410166
15	2,5	-0,096345718	-0,093550867	0,150779107	0,149381682	0,148021032

Рис. 15.7
Решение дифференциального уравнения методом Милна

Документ содержит достаточно большое количество ячеек с данными:

- в ячейки G2:G4 вводятся параметры, определяющие начальное условие и шаг приращения по аргументу;
- столбец A содержит значения индексов узловых точек n;
- столбец B содержит значения узловых точек x_n;
- столбец C содержит значения для функции уравнения $f(x, y)$, рассчитанные на основе первого приближения для функции-решения уравнения $y(x)$;
- столбец D содержит значения для функции уравнения $f(x, y)$, рассчитанные на основе второго приближения для функции-решения уравнения $y(x)$;
- столбец E содержит значения для функции-решения уравнения $y(x)$, вычисленные в первом приближении;
- столбец F содержит значения для функции-решения уравнения $y(x)$, вычисленные в первом приближении — это и есть конечная цель вычислений;
- столбец G содержит рассчитанные на основе аналитической формулы для решения дифференциального уравнения значения в узловых точках для функции $y(x)$.

На заметку

Решением дифференциального уравнения $y'(x) = x\exp(-x) - 2y(x)$ с начальным условием $y(x_0) = y_0$ является функция $y(x) = \exp(-x)(x - 1) - \exp(x_0 - 2x)(x_0 - 1) + y_0\exp(2(x_0 - x))$.

Для удобства способ заполнения в документе отдельных ячеек и диапазонов ячеек описан в таблице 15.2.

Конечная цель наших калькуляций — значения в ячейках F7:F22. В этих ячейках отображается второе приближение для функции $y(x)$ в узловых точках.

Таблица 15.2

Решение дифференциального уравнения методом Милна

Ячейка или диапазон	Способ заполнения	Комментарий
G2	Значение 1	Точка (по аргументу), в которой задается начальное условие, т. е. параметр x_0
G3	Значение 0,5	Значение функции, которое задается в начальном условии (параметр y_0)
G4	Значение 0,1	Шаг приращения по аргументу (параметр h)
A7:A22	В ячейку A7 вводится значение 0, в ячейку A8 вводится формула = A7+1 и копируется во все прочие ячейки диапазона	В этих ячейках вычисляются и отображаются индексы узловых точек
B7:B22	В ячейку B7 вводится формула = $G$2+A7*$G$4, а затем копируется в нижние ячейки до заполнения всего диапазона	В ячейках вычисляются и отображаются значения узловых точек x_n, вычисляемые по формуле $x_n = x_0 + nh$
C7:C22	В ячейку C7 вводим формулу =$B7*EXP(-$B7)-2*E7 и копируем ее во все остальные ячейки диапазона. Формула содержит ссылку на ячейку E7 — значение для искомой функции-решения уравнения в первом приближении	Значения f_n функции уравнения $f(x, y) = x\exp(-x) - 2y$, которые вычисляются в узловых точках по формуле $f_n = f(x_n, y_n) = x_n\exp(-x_n) - 2y_n$, в которой значения y_n берутся в первом приближении

Продолжение табл. 15.2

Ячейка или диапазон	Способ заполнения	Комментарий
D7:D22	Диапазон заполняется копированием формул из диапазона C7:C22. Можно также скопировать формулу из ячейки C7 в ячейку D7 (получим формулу =$B7*EXP(-$B7)-2*F7), а затем формулу из ячейки D7 скопировать в нижние ячейки. Формула =$B7*EXP(-$B7)-2*F7 содержит ссылку на ячейку F7 со значением функции-решения уравнения во втором приближении	Значения f_n вычисляются в узловых точках по формуле $$f_n = f(x_n, y_n) = x_n\exp(-x_n) - 2y_n,$$ но на этот раз значения y_n берутся во втором приближении
E7:E10	В ячейку E7 вводится формула =$G$3. В ячейку E8 вводим формулу =E7+$G$4*D7 и копируем ее в ячейки E9:E10	Значения функции-решения уравнения в начальных узловых точках вычисляются по методу Эйлера (с уточнением). В первом приближении значение в новой узловой точке вычисляется по формуле $$y_{n+1} = y_n + hf(x_n, y_n)$$
E11:E22	В ячейку E11 вводится формула =F7+4*$G$4/3*(2*D10-D9+2*D8), после чего эта формула копируется во все прочие ячейки диапазона	Вычисление в узловых точках в первом приближении значений функции-решения уравнения в соответствии с формулой $$y_{n+1} = y_{n-3} + \frac{4h}{3}(2f(x_n, y_n) - $$ $$-f(x_{n-1}, y_{n-1}) + 2f(x_{n-2}, y_{n-2}))$$
F7:F10	Ячейки диапазона заполняются следующим образом: в ячейку F7 вводится формула =$G$3, а в ячейку F8 вводим формулу =F7+$G$4/2*(D7+C8) и копируем ее в ячейки F9 и F10	Уточняются значения функции-решения уравнения в начальных узловых точках. Используем метод Эйлера с уточнением. В данном случае вычисляется второе приближение для функции-решения уравнения. Вычисления производим на основе формулы $$y_{n+1} = y_n +$$ $$+\frac{h}{2}(f(x_n, y_n) + f(x_{n+1}, y_{n+1})).$$ В этой формуле в правой части в качестве значения y_{n+1} используем первое приближение для функции, вычисленное по формуле $$y_{n+1} = y_n + hf(x_n, y_n)$$
F11:F22	В ячейку F11 вводим формулу =F9+$G$4/3*(C11+4*D10+D9). Затем формулу из ячейки F11 копируем в ячейки F12:F22	Вычисление результата — значений функции-решения уравнения в узловых точках (второе приближение). Вычисления проводятся в соответствии с формулой $$y_{n+1} = y_{n-3} + \frac{h}{3}(f(x_{n+1}, y_{n+1}) + $$ $$+4f(x_n, y_n) + f(x_{n-1}, y_{n-1}))$$
G7:G22	В ячейку G7 вводится формула =EXP(-B7)*(B7-1)-EXP($G$2-2*B7)*($G$2-1)+$G$3*EXP(2*($G$2-B7)), и затем копируется в нижние ячейки до заполнения всего диапазона	Точное значение для решения дифференциального уравнения, которое вычисляется на основе аналитической формулы $$y(x) = \exp(-x)(x-1) - \exp(x_0 - 2x)(x_0 - -1) + y_0\exp(2(x_0 - x)),$$ учитывающей параметры начального условия

На заметку

Также стоит обратить внимание на одно «техническое» обстоятельство: в начальной точке значение функции определяется начальным условием, а в следующих точках для вычисления значения функции мы использовали метод Эйлера с уточнением. Суть этого метода состоит в том, что сначала значение в новой узловой точке вычисляется по формуле $y_{n+1} = y_n + hf(x_n, y_n)$ (первое приближение для y_{n+1}). Затем вычисленное значение уточняется по формуле $y_{n+1} = y_n + \frac{h}{2}(f(x_n, y_n) + f(x_{n+1}, y_{n+1}))$ (второе приближение). Правая часть приведенного соотношения вычисляется со значением y_{n+1}, рассчитанным в первом приближении по формуле $y_{n+1} = y_n + hf(x_n, y_n)$. В документе на рисунке 15.7 для первого приближения метода Эйлера вычисления выполняются в ячейках E7:E10, а для второго приближения по методу Эйлера вычисления выполняются в ячейках F7:F10.

Далее разработаем функцию пользователя, которая бы позволяла вычислять решение дифференциального уравнения по методу Милна. Такая функция называется МЕТМИЛНА(), и ее программный код представлен в листинге 15.18.

Листинг 15.18. Функция для решения уравнения методом Милна

```
Function МЕТМИЛНА(f, x0, y0, t, Optional dx = 0.001)
 ‘ Переменная для определения количества частей, на которые узловыми
 ‘ точками разбивается интервал от начальной точки до точки,
 ‘ в которой вычисляется решение уравнения
 Dim n As Integer
 n = Fix((t - x0) / dx)
 ‘ Количество частей, на которые разбивается интервал,
 ‘ не должно быть меньше четырех
 If n < 4 Then n = 4
 ‘ Шаг приращения по аргументу
 Dim h As Double
 h = (t - x0) / n
 ‘ Индексные переменные для операторов цикла
 Dim i As Integer, j As Integer
 ‘ Переменная для определения индекса узловой точки,
 ‘ до которой вычисления выполняются одношаговым методом
 Dim k As Integer
 k = 3
 ‘ Объект для вызова метода, определяющего функцию
 ‘ в правой части дифференциального уравнения
 Set obj = New DEquations
 ‘ Массив для запоминания четырех последних узловых точек
 ReDim x(0 To k) As Double
 ‘ Начальная узловая точка
 x(0) = x0
 ‘ Массив для записи в последних четырех узловых точках
 ‘ значения функции-решения уравнения
```

```
ReDim y(0 To k) As Double
' Значение искомой функции в начальной точке
y(0) = y0
' Массив для записи значений функции в правой части
' уравнения в четырех последних узловых точках
ReDim Fn(0 To k) As Double
' Значение в начальной точке
Fn(0) = CallByName(obj, f, VbMethod, x(0), y(0))
' Оператор цикла для вычисления решения в начальных
' узловых точках по методу Эйлера с уточнением
For i = 1 To k
  ' Узловая точка
  x(i) = x(i - 1) + h
  ' Значение функции в узловой точке (первое приближение)
  y(i) = y(i - 1) + h * Fn(i - 1)
  ' Значение функции в правой части уравнения
  ' (вычисляется по первому приближению для функции-решения)
  Fn(i) = CallByName(obj, f, VbMethod, x(i), y(i))
  ' Второе приближение для функции-решения
  y(i) = y(i - 1) + h / 2 * (Fn(i - 1) + Fn(i))
  ' Значение функции в правой части уравнения
  ' (вычисляется по второму приближению для функции-решения)
  Fn(i) = CallByName(obj, f, VbMethod, x(i), y(i))
Next i
' Переменная для записи значения новой узловой точки
Dim Xnew As Double
' Переменная для записи значения функции-решения уравнения
' в новой узловой точке
Dim Ynew As Double
' Переменная для записи значений функции в правой части уравнения
' для новой узловой точки
Dim Fnew As Double
' Оператор цикла для вычисления результата
For i = k + 1 To n
  ' Значение новой узловой точки
  Xnew = x(k) + h
  ' Значение функции-решения уравнения вычисляется
  ' по методу Милна (первое приближение)
  Ynew = y(k - 3) + 4 * h / 3 * (2 * Fn(k) - Fn(k - 1) + _
  2 * Fn(k - 2))
  ' Значение функции в правой части уравнения
  ' (в первом приближении)
  Fnew = CallByName(obj, f, VbMethod, Xnew, Ynew)
  ' Значение функции-решения уравнения вычисляется
  ' по методу Милна (второе приближение)
  Ynew = y(k - 1) + h / 3 * (Fnew + 4 * Fn(k) + Fn(k - 1))
```

```
    ' Значение функции в правой части уравнения
    ' по методу Милна (второе приближение)
    Fnew = CallByName(obj, f, VbMethod, Xnew, Ynew)
    ' Смещение элементов массивов на одну позицию влево
    For j = 0 To k - 1
      ' Смещение узловых точек
      x(j) = x(j + 1)
      ' Смещение значений функции-решения уравнения
      y(j) = y(j + 1)
      ' Смещение значений функции в правой части уравнения
      Fn(j) = Fn(j + 1)
    Next j
    ' Новое значение для узловой точки заносится
    ' последним элементом в массив
    x(k) = Xnew
    ' Новое значение для функции-решения уравнения заносится
    ' последним элементом в массив
    y(k) = Ynew
    ' Новое значение для функции в правой части уравнения заносится
    ' последним элементом в массив
    Fn(k) = Fnew
  Next i
  ' Значение функции
  МЕТМИЛНА = y(k)
End Function
```

У функции МЕТМИЛНА() несколько аргументов:

- переменная f обозначает функцию двух переменных, определяющую правую часть решаемого дифференциального уравнения;
- переменная x0 обозначает начальную точку по аргументу (точка определения начального условия);
- переменная y0 обозначает значение искомой функции в начальной точке;
- переменная t определяет значение аргумента, для которого вычисляется решение дифференциального уравнения;
- необязательный аргумент dx (со значением по умолчанию 0.001) определяет примерный шаг дискретности по аргументу.

В теле функции командой n = Fix((t-x0)/dx) определяется количество частей, на которые узловыми точками разбивается интервал по аргументу функции от начальной точки до точки, в которой вычисляется решение. Условный оператор позволяет удостовериться, что значение переменной n будет не меньше, чем 4. Шаг приращения по аргументу задаем командой h = (t-x0)/n.

При вычислении решения уравнения наша стратегия состоит в том, чтобы запоминать значения четырех последних узловых точек, а также значения в этих точках искомой функции-решения уравнения и функции, определяющей правую часть уравнения. Все эти значения мы будем записывать в массивы из четырех элементов каждый. Для удобства вводим целочисленную

переменную k со значением 3. Эту переменную будем использовать для определения границ массивов и индексации элементов. А именно, мы создаем такие массивы: x — для записи значения узловых точек, y — для записи значения функции-решения уравнения в узловых точках и Fn — для записи значений функции в правой части уравнения. В каждом из этих массивов индексация элементов выполняется от 0 до k.

На основе параметров x0 и y0 для начального условия определяем первые элементы соответственно в массивах x и y (команды x(0) = x0 и y(0) = y0). Первый элемент в массиве Fn вычисляем по формуле Fn(0) = CallByName (obj,f,VbMethod,x(0),y(0)). В данном случае мы из объекта obj (создается командой Set obj = New DEquations) вызываем метод f с аргументами x(0) и y(0). Для вычисления остальных элементов массивов запускается оператор цикла, в котором индексная переменная пробегает значения от 1 до k. В теле оператора цикла выполняются такие команды:

- командой x(i) = x(i-1)+h вычисляется значение узловых точек;
- командой y(i) = y(i-1)+h*Fn(i-1) в первом приближении вычисляется значение в узловой точке для функции-решения уравнения;
- командой Fn(i) = CallByName(obj,f,VbMethod,x(i),y(i)) вычисляем значение в узловой точке функции, определяющей правую часть дифференциального уравнения;
- командой y(i) = y(i-1)+h/2*(Fn(i-1)+Fn(i)) вычисляем второе приближение для функции-решения уравнения в соответствующей узловой точке;
- вычисленное значение для функции y(i) используется в команде Fn(i) = CallByName(obj,f,VbMethod,x(i),y(i)) для вычисления значения функции из правой части уравнения.

На заметку

В данном случае вычисления производятся в соответствии с методом Эйлера с уточнением: сначала по формуле $y_{n+1} = y_n + hf(x_n, y_n)$ вычисляем значение для y_{n+1}, а затем с помощью этого вычисленного значения производим «уточнение» по формуле $y_{n+1} = y_n + \frac{h}{2}(f(x_n,y_n) + f(x_{n+1},y_{n+1}))$. Таким образом, после выполнения описанных выше команд получим расчетные данные для четырех первых узловых точек: в начальной точке все определяется начальным условием, а в остальных трех точках вычисления производятся по методу Эйлера с уточнением.

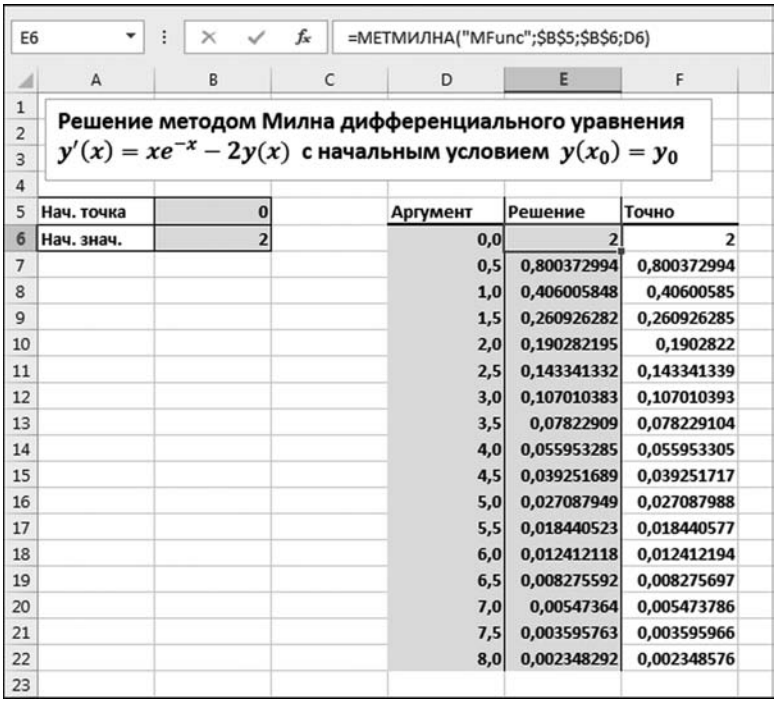
Дальше в дело вступает метод Милна. Запускается оператор цикла (индексная переменная i пробегает значения от k+1 до n), в котором:

- вычисляется значение для очередной узловой точки x_{n+1} (команда Xnew = x(k)+h);
- по методу Милна (формула $y_{n+1} = y_{n-3} + \frac{4h}{3}(2f(x_n,y_n) - f(x_{n-1},y_{n-1}) + 2f(x_{n-2},y_{n-2}))$) вычисляется значение (в первом приближении) функции-решения уравнения в этой точке y_{n+1} (команда Ynew = y(k-3)+4*h/3*(2*Fn(k)-Fn(k-1)+2*Fn(k-2)));
- вычисляется значение функции $f(x, y)$ в правой части уравнения в этой узловой точке, т. е. $f(x_{n+1}, y_{n+1})$, причем значение y_{n+1} берется в первом приближении (команда Fnew = CallByName(obj,f,VbMethod,Xnew,Ynew));

- по методу Милна (формула $y_{n+1} = y_{n-3} + \frac{h}{3}(f(x_{n+1}, y_{n+1}) + 4f(x_n, y_n) + f(x_{n-1}, y_{n-1}))$) вычисляется второе приближение для y_{n+1} (команда Ynew = y(k-1)+h/3*(Fnew+4*Fn(k)+Fn(k-1)));
- на основе вычисленного второго приближения для y_{n+1} вычисляем значение функции в правой части $f(x_{n+1}, y_{n+1})$ (команда Fnew = CallByName (obj,f,VbMethod,Xnew,Ynew)).

На финальном этапе вычисленные значения необходимо занести последними элементами в соответствующие массивы, а текущие элементы в массивах сместить на одну позицию влево. Для этого во внешнем операторе цикла запускаем внутренний оператор с индексной переменной j, пробегающей значения от 0 до k-1. В этом внутреннем цикле на одну позицию смещаются элементы массива. После завершения внутреннего оператора цикла задаются старшие «недостающие» элементы массивов: выполняются команды x(k) = Xnew, y(k) = Ynew и Fn(k) = Fnew. После выполнения внешнего оператора цикла командой МЕТМИЛНА = y(k) определяем значение функции МЕТМИЛНА().

Для решения уравнения $y'(x) = x\exp(-x) - 2y(x)$ в модуле класса DEquations описываем метод MFunc(), который определяет функцию в правой части уравнения. Программный код этой функции приведен в листинге 15.19.



E6 =МЕТМИЛНА("MFunc";$B$5;$B$6;D6)

Решение методом Милна дифференциального уравнения $y'(x) = xe^{-x} - 2y(x)$ с начальным условием $y(x_0) = y_0$

	A	B	C	D	E	F
5	Нач. точка	0		Аргумент	Решение	Точно
6	Нач. знач.	2		0,0	2	2
7				0,5	0,800372994	0,800372994
8				1,0	0,406005848	0,40600585
9				1,5	0,260926282	0,260926285
10				2,0	0,190282195	0,1902822
11				2,5	0,143341332	0,143341339
12				3,0	0,107010383	0,107010393
13				3,5	0,07822909	0,078229104
14				4,0	0,055953285	0,055953305
15				4,5	0,039251689	0,039251717
16				5,0	0,027087949	0,027087988
17				5,5	0,018440523	0,018440577
18				6,0	0,012412118	0,012412194
19				6,5	0,008275592	0,008275697
20				7,0	0,00547364	0,005473786
21				7,5	0,003595763	0,003595966
22				8,0	0,002348292	0,002348576

Рис. 15.8
Использование пользовательской функции для решения дифференциального уравнения методом Милна

Листинг 15.19. Функция, определяющая решаемое методом Милна уравнение

```
Function MFunc(x As Double, y As Double) As Double
  MFunc = x * Exp(-x) - 2 * y
End Function
```

Как разработанные нами функции могут использоваться для решения дифференциального уравнения, иллюстрирует документ, представленный на рисунке 15.8.

В этом документе ячейка B5 содержит значение начальной точки x_0, в ячейку B6 вводится значение y_0 функции в начальной точке.

Ячейки D6:D22 заполняются значениями аргумента для вычисляемой функции: в ячейку D6 вводится формула =$B$5, в ячейку D7 вводится формула =D6+0,5. Формулу из ячейки D7 копируем в нижние ячейки.

В ячейках E6:E22 вычисляются приближенные решения для дифференциального уравнения в соответствующих точках: в ячейку E6 вводится формула =МЕТМИЛНА("MFunc";$B$5;$B$6;D6), после чего эта формула копируется в остальные ячейки диапазона E6:E22.

Для сравнения в ячейках F6:F22 решение уравнения вычисляется точное решение (вычисляется на основе выражения $y(x) = \exp(-x)(x - 1) - \exp(x_0 - 2x)(x_0 - 1) + y_0\exp(2(x_0 - x))$). С этой целью в ячейку F6 вводим формулу =EXP(-D6)*(D6-1)-EXP($B$5-2*D6)*($B$5-1)+$B$6*EXP(2*($B$5-D6)) и копируем ее в нижние ячейки.

СИСТЕМЫ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

Бывают такие случаи,
когда неплохо соврать.

Из к/ф «Ирония судьбы, или С легким паром!»

Системы явных дифференциальных уравнений первого порядка решаются практически теми же методами, что и обыкновенные дифференциальные уравнения. Чтобы понять принципы применения этих методов для решения систем уравнений разумно представить систему уравнений в векторном виде, т. е. записать систему дифференциальных уравнений в виде векторного уравнения.

Для формализации задачи предположим, что нам нужно решить систему уравнений $y_1'(x) = f_1(x, y_1(x), y_2(x), \ldots, y_n(x))$, $y_2'(x) = f_2(x, y_1(x), y_2(x), \ldots, y_n(x))$, $\ldots$, $y_n'(x) = f_n(x, y_1(x), y_2(x), \ldots, y_n(x))$ с начальными условиями $y_1(x_0) = y_{1,0}, y_2(x_0) = y_{2,0}, \ldots, y_n(x_0) = y_{n,0}$ относительно неизвестных функций $y_1(x), y_2(x), \ldots, y_n(x)$. Чтобы представить эту систему в векторном виде, введем в рассмотрение вектор-функцию $\vec{y}(x) = (y_1(x), y_2(x), \ldots, y_n(x))^T$, а также вектор-функцию $\vec{f}(x, \vec{y}(x)) = [f_1(x, y_1(x), \ldots, y_n(x)), \ldots, f_n(x, y_1(x), \ldots, y_n(x))]^T$ правых частей системы уравнений. Также обозначим вектор начальных условий $\vec{y}_0 = [y_{1,0}, y_{2,0}, \ldots, y_{n,0}]^T$. Тогда в векторном виде система урав-

нений запишется как $\frac{d\vec{y}(x)}{dx} = \vec{f}(x, \vec{y}(x))$. Начальные условия в векторной форме принимают вид $\vec{y}(x_0) = \vec{y}_0$.

Например, если мы хотим реализовать метод Эйлера, то в начальной точке берем $\vec{y}(x_0) = \vec{y}_0$, а для прочих узловых точек $x_n = x_0 + nh$ значение вектор-функции решения системы вычисляем в соответствии с соотношением $\vec{y}_{n+1} = \vec{y}_n + h\vec{f}(x_n, \vec{y}_n)$. В методе Эйлера с уточнением второе приближение можем вычислить как $\vec{y}_{n+1} = \vec{y}_n + \frac{h}{2}(\vec{f}(x_n, \vec{y}_n) + \vec{f}(x_{n+1}, \vec{y}_{n+1}))$ (в правой части формулы $\vec{y}_{n+1}$ берем в первом приближении).

Аналогично реализуется метод Рунге — Кутты. В этом случае рекуррентные соотношения имеют вид $\vec{y}_{n+1} = \vec{y}_n + \frac{1}{6}(\vec{k}_1(h) + 2\vec{k}_2(h) + 2\vec{k}_3(h) + \vec{k}_4(h))$, где введены такие обозначения: $\vec{k}_1(h) = h\vec{f}(x_n, \vec{y}_n)$, $\vec{k}_2(h) = h\vec{f}\left(x_n + \frac{h}{2}, \vec{y}_n + \frac{\vec{k}_1}{2}\right)$, $\vec{k}_3(h) = h\vec{f}\left(x_n + \frac{h}{2}, \vec{y}_n + \frac{\vec{k}_2}{2}\right)$ и $\vec{k}_4(h) = h\vec{f}\left(x_n + h, \vec{y}_n + \vec{k}_3\right)$. В качестве примера рассмотрим программный код в листинге 15.20 для функции, предназначенной для решения системы из двух дифференциальных уравнений.

Листинг 15.20. Функция для решения системы дифференциальных уравнений

```
Function РЕШИТЬСДУ(f1, f2, x0, y10, y20, t, Optional dx = 0.001)
 ' Массив из двух элементов для записи названий функций,
 ' определяющих правые части системы уравнений
 Dim f(1 To 2) As String
 ' Функция для правой части первого уравнения
 f(1) = f1
 ' Функция для правой части второго уравнения
 f(2) = f2
 ' Массив для записи решения системы
 ' дифференциальных уравнений
 Dim y(1 To 2) As Double
 ' Начальное значение первой функции
 y(1) = y10
 ' Начальное значение второй функции
 y(2) = y20
 ' Массив для записи параметров для формулы Рунге–Кутты
 Dim k(1 To 4, 1 To 2) As Double
 ' Переменная для записи значения аргумента функций
 Dim x As Double
 ' Начальное значение аргумента
 x = x0
 ' Объект для вызова методов, определяющих правые части
 ' уравнений системы
 Set obj = New DESystem
 ' Количество частей, на которые разбивается
```

```
    ' интервал от начальной точки до точки,
    ' в которой вычисляется решение
    Dim n As Integer
    n = Fix((t - x0) / dx) + 1
    ' Шаг приращения по аргументу
    Dim h As Double
    h = (t - x0) / n
    ' Индексные переменные для операторов цикла
    Dim i As Integer, j As Integer
    ' Перебираем узловые точки
    For i = 1 To n
      ' Перебираем функции
      For j = 1 To 2
        ' Параметры для формулы Рунге-Кутты
        k(1, j) = h * CallByName(obj, f(j), VbMethod, x, y(1), y(2))
        k(2, j) = h * CallByName(obj, f(j), VbMethod, x + h / 2, _
        y(1) + k(1, 1) / 2, y(2) + k(1, 2) / 2)
        k(3, j) = h * CallByName(obj, f(j), VbMethod, x + h / 2, _
        y(1) + k(2, 1) / 2, y(2) + k(2, 2) / 2)
        k(4, j) = h * CallByName(obj, f(j), VbMethod, x + h, _
        y(1) + k(3, 1), y(2) + k(3, 2))
      Next j
      ' Значение функции в новой узловой точке
      For j = 1 To 2
        y(j) = y(j) + (k(1, j) + 2 * k(2, j) + 2 * k(3, j) + k(4, j)) / 6
      Next j
      ' Новая узловая точка
      x = x + h
    Next i
    ' Результат решения системы дифференциальных уравнений
    РЕШИТЬСДУ = y
  End Function
```

Функции РЕШИТЬСДУ() передаются такие аргументы:

- текстовые переменные f1 и f2 предназначены для обозначения названий методов, определяющих правые части уравнений системы;
- аргумент x0 обозначает начальную точку, в которой задается начальное условие;
- через аргументы y10 и y20 обозначаются начальные значения для искомых функций, входящих в систему дифференциальных уравнений;
- переменная t дает значение аргумента, для которого вычисляются значения функций, входящих в систему уравнений;
- опционный аргумент dx со значением по умолчанию 0.001 определяет максимальный шаг приращения.

В теле программного кода функции содержится несколько команд. Их назначение в принципе должно быть понятно читателю. Поэтому остано-

вимся на них лишь вкратце. Так, текстовые (как мы предполагаем) значения для названий методов f1 и f2 заносятся в текстовый массив f из двух элементов. Для записи значений искомых функций (их, напомним, две) вводится массив y из двух элементов. Начальные значения элементов массива определяются аргументами y10 и y20.

Вычисления производятся по методу Рунге — Кутты. Причем коэффициенты (их четыре), входящие в соответствующую формулу, в данном случае представляют собой векторы — по два элемента в каждом. Для этого объявляется двумерный массив k, первый индекс в этом массиве определяет номер коэффициента, а второй индекс — это индекс элемента в векторе.

Переменная x с начальным значением x0 содержит значение узловой точки, для которой вычисляется решение системы уравнений. Шаг приращения по аргументу вычисляется и записывается в переменную h. Также мы создаем объект obj класса DESystem, в проекте должен быть модуль класса с таким названием, в котором описываются функции (от трех аргументов каждая), определяющие правые части уравнений системы.

Далее в соответствии с формулой Рунге — Кутты последовательно вычисляются значения векторных коэффициентов для этой формулы, и, на их основе, значения искомых функций в данной узловой точке. После вычисления нового значения для узловой точки все расчетные процедуры повторяются, и так до тех пор, пока не будет найдено значение для решения уравнения в точке, определяемой аргументом t (или x0+n*h). В качестве результата функцией РЕШИТЬСДУ() возвращается массив y, т. е. два значения. Это и есть решение системы дифференциальных уравнений (в точке).

Разработанную функцию используем для решения системы уравнений $y_1'(x) = 4y_1(x) - 3y_2(x) + \sin(x)$ и $y_2'(x) = 2y_1(x) - y_2(x) - 2\cos(x)$ с начальными условиями $y_1(x_0) = y_{1,0}$ и $y_2(x_0) = y_{2,0}$.

На заметку

Точное решение задачи такое:

$$y_1(x) = \cos x + 3\exp(2(x - x_0))(y_{1,0} - y_{2,0} + \cos x_0) -$$
$$- 2\sin x + \exp(x - x_0)(2\sin x_0 + 3y_{2,0} - 2y_{1,0} - 4\cos x_0)$$

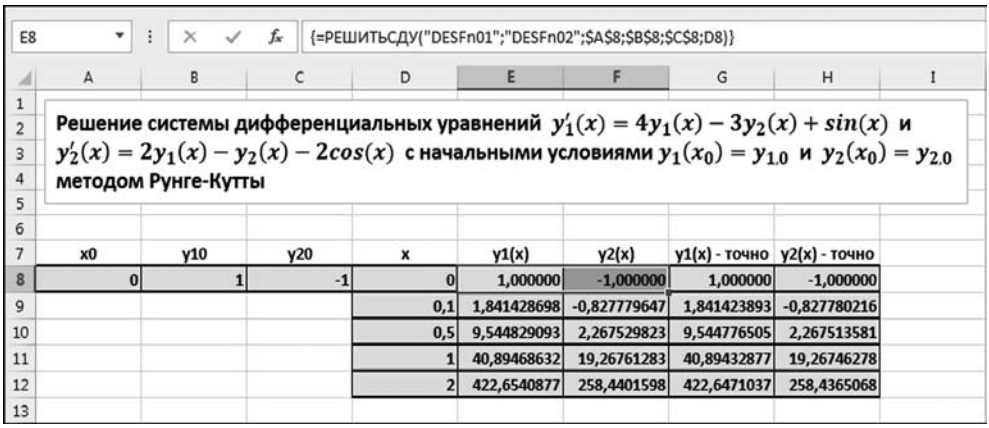
и

$$y_2(x) = 2\cos x + 2\exp(2(x - x_0))(y_{1,0} - y_{2,0} + \cos x_0) -$$
$$- 2\sin x + \exp(x - x_0)(2\sin x_0 + 3y_{2,0} - 2y_{1,0} - 4\cos x_0).$$

Для решения системы в проект добавляется модуль класса DESystem, и в этом модуле описываем две функции. Функция DESFn01() определяет соотношение $f_1(x, y_1, y_2) = 4y_1 - 3y_2 + \sin x$. Программный код функции приведен в листинге 15.21.

Листинг 15.21. Функция для первого уравнения системы

```
Function DESFn01(x As Double, y1 As Double, y2 As Double) As Double
  DESFn01 = 4 * y1 - 3 * y2 + Sin(x)
End Function
```



E8 | {=РЕШИТЬСДУ("DESFn01";"DESFn02";$A$8;$B$8;$C$8;D8)}

Решение системы дифференциальных уравнений $y_1'(x) = 4y_1(x) - 3y_2(x) + sin(x)$ и $y_2'(x) = 2y_1(x) - y_2(x) - 2cos(x)$ с начальными условиями $y_1(x_0) = y_{1,0}$ и $y_2(x_0) = y_{2,0}$ методом Рунге-Кутты

x0	y10	y20	x	y1(x)	y2(x)	y1(x) - точно	y2(x) - точно
0	1	-1	0	1,000000	-1,000000	1,000000	-1,000000
			0,1	1,841428698	-0,827779647	1,841423893	-0,827780216
			0,5	9,544829093	2,267529823	9,544776505	2,267513581
			1	40,89468632	19,26761283	40,89432877	19,26746278
			2	422,6540877	258,4401598	422,6471037	258,4365068

Рис. 15.9
Решение системы дифференциальных уравнений методом Рунге — Кутты

Функция DESFn02() определяет соотношение $f_2(x, y_1, y_2) = 2y_1 - y_2 - 2\cos x$. Программный код функции приведен в листинге 15.22.

Листинг 15.22. Функция для второго уравнения системы

```
Function DESFn02(x As Double, y1 As Double, y2 As Double) As Double
  DESFn02 = 2 * y1 - y2 - 2 * Cos(x)
End Function
```

Документ, в котором решается система уравнений, показан на рисунке 15.9.

В ячейку A8 вводим значение начальной точки (параметр x_0). В ячейки B8 и C8 вводятся начальные значения для искомых функций (параметры $y_{1,0}$ и $y_{2,0}$). Ячейки D8:D12 содержат несколько значений для аргумента функций, для этих значений аргумента вычисляются приближенные и точные решения системы. Приближенные значения вычисляем в ячейках E8:F12. Заполняются ячейки так: выделяем диапазон ячеек E8:F8 и вводим в ячейки формулу массива =РЕШИТЬСДУ("DESFn01";"DESFn02";$A$8;$B$8; $C$8;D8) (формула вводится нажатием комбинации клавиш <Ctrl>+ <Shift>+ <Enter>). Формула из ячеек E8:F8 копируется в нижние ячейки диапазона E8:F12.

Поскольку для системы известно точное решение, то мы для сравнения производим вычисление и «точного» результата. Для этого в ячейку G8 вводим формулу =COS(D8)+3*EXP(2*(D8-$A$8))*($B$8-$C$8+COS($A$8))-2*SIN(D8)+EXP(D8-$A$8)*(2*SIN($A$8)+3*$C$8-2*$B$8-4*COS($A$8)) (первая функция-решение системы), а в ячейку H8 вводим формулу =2*COS(D8)+2*EXP(2*(D8-$A$8))*($B$8-$C$8+COS($A$8))-2*SIN(D8)+EXP(D8-$A$8)*(2*SIN($A$8)+3*$C$8-2*$B$8-4*COS($A$8)) (вторая функция-решение системы уравнений). Затем формулы копируются в нижние ячейки диапазона G8:H12.

ИНТЕГРАЛЬНЫЕ УРАВНЕНИЯ

Ну, а это довесок к кошмару.

Из к/ф «Старики-разбойники»

В этом разделе мы кратко остановимся на *интегральных уравнениях*: в частности, мы покажем, что приложение Excel может быть полезно при решении такого класса задач. Более конкретно, мы рассмотрим *уравнение Фредгольма второго рода* и средствами рабочего листа Excel найдем его приближенное решение.

На заметку

Интегральными называются уравнения, в которые неизвестная функция входит, кроме всего прочего, и под знак интеграла.

Рассмотрим уравнение $y(x) = f(x) + \lambda\int_a^b K(x,t)y(t)dt$. В этом уравнении $f(x)$ и $K(x, t)$ — известные функции, параметр λ задан, а функцию $y(x)$ нужно найти. Функция $K(x, t)$ называется *ядром интегрального уравнения*. При этом предполагается, что аргумент x находится в пределах диапазона значений от a до b (границы области интегрирования в интегральном выражении).

Подход, который мы будем использовать для поиска приближенного решения интегрального уравнения, базируется на вычислении интеграла с помощью квадратурной формулы. В частности, для вычисления интеграла $\int_a^b F(x)dx$ от некоторой функции $F(x)$ мы используем соотношение $\int_a^b F(x)dx = \sum_{k=0}^{n} \omega_k F(x_k)$, где ω_k — коэффициенты квадратурной формулы, которые (и это важно) не зависят от функции $F(x)$, а узловые точки $x_k = a + kh$, где $h = \frac{b-a}{n}$, индекс $k = 0, 1, 2, \ldots, n$.

На заметку

Коэффициенты ω_k квадратурной формулы хотя и не зависят от подынтегральной функции, зависят от границ области интегрирования. Вместе с тем, если сделать замену переменной $z = \frac{x-a}{b-a}$ (в том числе и в интегральном уравнении), можем добиться, чтобы интегрирование выполнялось в пределах от 0 до 1.

Если подставить вместо интеграла в интегральном уравнении квадратурную формулу и вычислять значения в узловых точках, получим следующее соотношение: $y_i = f_i + \lambda\sum_{j=0}^{n} \omega_j K_{ij} y_j$, где мы обозначили $y_i = y(x_i)$, $f_i = f(x_i)$, $K_{ij} = K(x_i, x_j)$, а индексы $i, j = 0, 1, 2, \ldots, n$. Относительно неизвестных параметров y_i это линейная система алгебраических уравнений. Мы ее можем записать как $\sum_{j=0}^{n} (\delta_{ij} - \lambda\omega_j K_{ij}) y_j = f_i$, где символ Кронекера $\delta_{ij} = 1$ при $i = j$ и $\delta_{ij} = 0$

в случае, если $i \neq j$. Эту же систему можно записать в матричном виде $\hat{\mathbf{A}}\vec{y} = \vec{f}$, при этом у матрицы $\hat{\mathbf{A}}$ элементы $a_{ij} = \delta_{ij} - \lambda\omega_j K_{ij}$, а векторы $\vec{y} = [y_0, y_1, \ldots, y_n]^T$ и $\vec{f} = [f_0, f_1, \ldots, f_n]^T$. В этом случае решение системы можем записать как $\vec{y} = \hat{\mathbf{A}}^{-1}\vec{f}$, где $\hat{\mathbf{A}}^{-1}$ — матрица, обратная матрице $\hat{\mathbf{A}}$.

Если вектор $\vec{y}$ найден (т. е. вычислены значения $y_0, y_1, \ldots, y_n$), мы можем записать приближенное решение интегрального уравнения в виде $y(x) = f(x) + \lambda\sum_{i=0}^{n}\omega_i K(x, x_i)y_i$.

Именно такой подход реализуем в рабочем документе Excel для решения интегрального уравнения $y(x) = 1 - x + \lambda\int_0^1 xt^2 y(t)dt$.

На заметку

У этого уравнения есть точное решение $y(x) = 1 - \frac{4x(\lambda - 3)}{3(\lambda - 4)}$.

В качестве квадратурной формулы для вычисления интеграла выберем формулу Симпсона. Документ, в котором выполняются вычисления, представлен на рисунке 15.10.

Наша стратегия состоит в том, чтобы все основные вычисления выполнять в явном виде и отображать результаты промежуточных вычислений в рабочем документе. Для начала в рабочий документ вводим значения узловых точек, весовые коэффициенты квадратурной формулы Симпсона, а также значения функции $f(x) = 1 - x$ в узловых точках. А именно, на начальном этапе мы производим следующие вычисления в рабочем документе:

- в ячейку B5 вводим значение 0, в ячейку B6 вводим формулу =B5+1 и копируем эту формулу в ячейки справа, вплоть до заполнения диапазона B5:L5. В результате этот диапазон содержит индексы узловых точек;
- в ячейки диапазона B6:L6 вводится формула массива =B5:L5/$L$5 (т. е. формула вводится в диапазон комбинацией клавиш <Ctrl>+<Shift>+<Enter>). После этого диапазон B6:L6 содержит значения одиннадцати узловых точек, равномерно распределенных на интервале от 0 до 1;
- диапазон ячеек B7:L7 заполняется весовыми коэффициентами для квадратурной формулы Симпсона. Для этого в ячейки B7 и L7 вводятся одна и та же формула =1/3/$L$5, а для заполнения остальных ячеек поступаем следующим образом: в ячейку C7 вводим формулу =2*(1+ОСТАТ(C5;2))/3/$L$5 и копируем ее в ячейки справа вплоть до ячейки K7;
- значения функции $f(x) = 1 - x$ в узловых точках вычисляются следующим образом: выделяем диапазон ячеек B8:L8 и вводим в этот диапазон формулу массива =1-B6:L6.

Ячейки диапазона B9:L9 предназначены для вычисления значений искомой функции $y(x)$ в узловых точках. Для этого в диапазон вводится формула массива =ТРАНСП(МУМНОЖ(МОБР(B25:L35);ТРАНСП(B8:L8))), но делать это лучше после того, как вычислены значения в диапазоне ячеек B25:L35, на который в этой формуле имеется ссылка.

Матрица с элементами $K(x_i, x_j)$ — значениями функции ядра в узловых точках — вычисляется в ячейках диапазона B12:L22. Указанный диапазон ячеек выделяется, и в него вводится формула массива =ТРАНСП(B6:L6)* B6:L6^2 (рис. 15.11).

Значения в ячейках B12:L22 используются для вычисления значений в ячейках диапазона B25:L35. Здесь мы вычисляем матрицу с элементами $\delta_{ij} - \lambda\omega_j K_{ij}$. Обращением этой матрицы и перемножением с вектором-столбиком значений функции $f(x)$ в узловых точках вычисляем значения в узловых точках искомой функции $y(x)$.

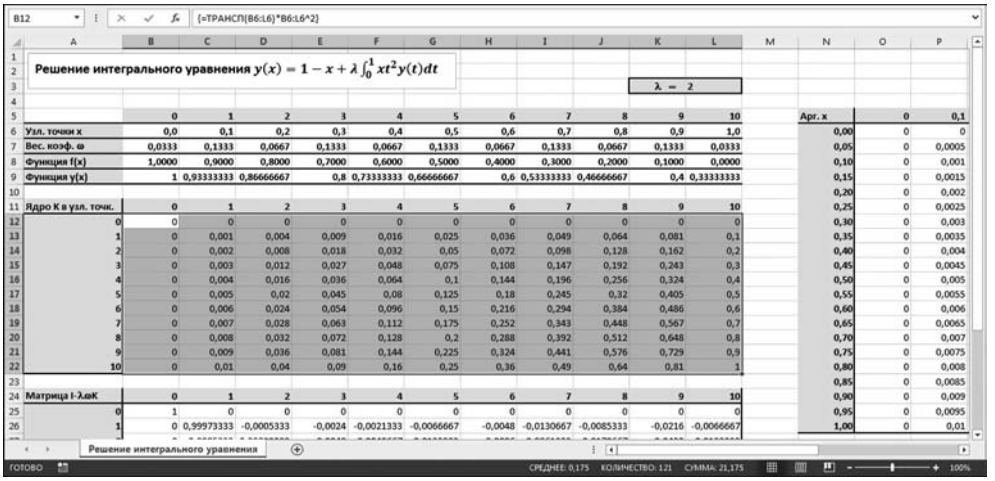


Рис. 15.10
Решение интегрального уравнения: начало вычислений

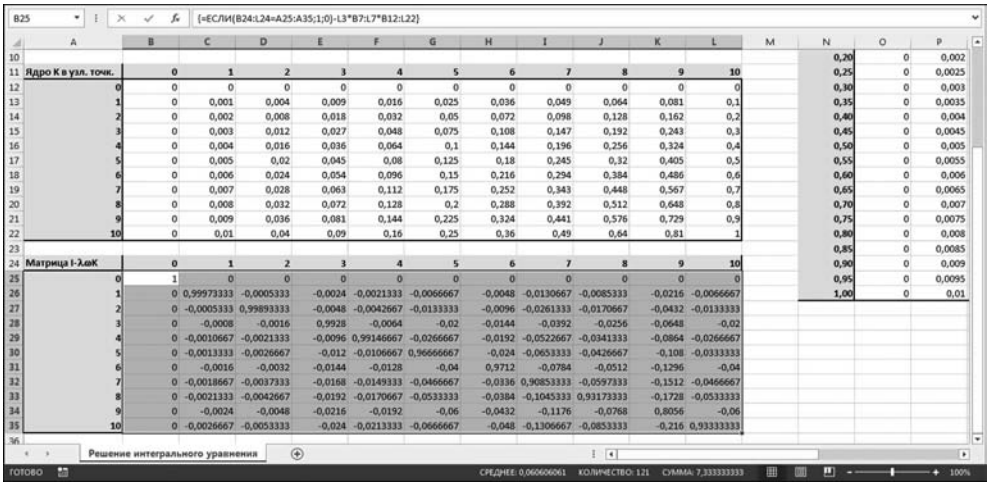
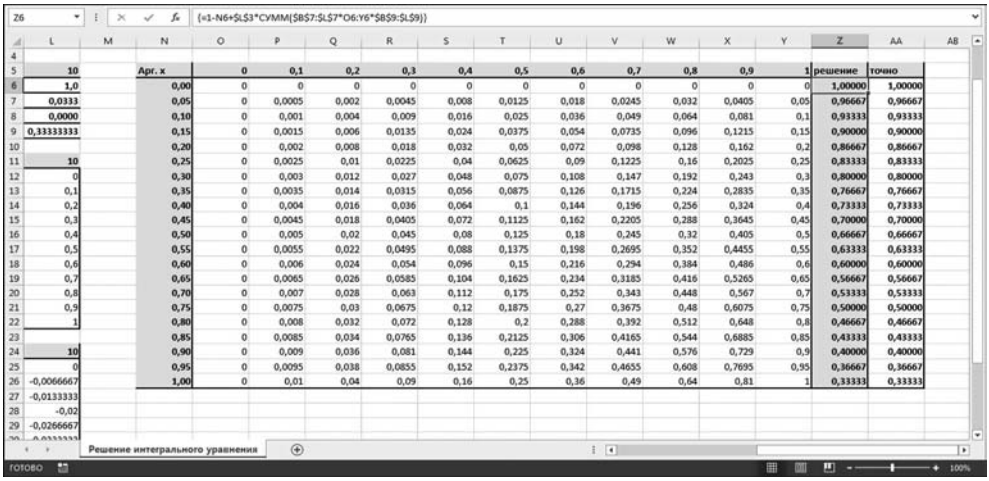


Рис. 15.11
Решение интегрального уравнения: промежуточные вычисления

Диапазон ячеек B25:L35 выделяется, и в него вводится формула массива =ЕСЛИ(B24:L24 = A25:A35;1;0)-L3*B7:L7*B12:L22. Формула содержит ссылки на диапазон B7:L7 со значениями весовых коэффициентов квадратурной формулы Симпсона, а также ссылку на ячейку L3 со значением параметра λ.

На финальной стадии вычислений (рис. 15.12) мы заполняем диапазон ячеек N6:N26 значениями аргумента x, для которых будет вычисляться значение функции $y(x)$ — решения интегрального уравнения.

Диапазон N6:N26 заполняется так: в ячейку N6 вводим нулевое значение, в ячейку N7 вводится формула =N6+0,05, и эта формула копируется в нижние ячейки. В ячейках O5:Y5 для удобства отображаем значения узловых точек: в этот диапазон вводим формулу массива =B6:L6. Далее, выделяем диапазон ячеек O6:Y26 и вводим в этот диапазон формулу массива =N6:N26*O5:Y5^2. В этом случае вычисляется матрица для функции ядра $K(x, x_i)$, второй аргумент которой x_i берется в узловых точках, а значения первого аргумента x определяются диапазоном ячеек N6:N26. Результат (значения функции $y(x)$) вычисляется в ячейках диапазона Z6:Z26: в ячейку Z6 вводится формула массива =1-N6+$L$3*СУММ($B$7:$L$7*O6:Y6*$B$9:$L$9), и эта формула копируется в нижние ячейки, вплоть до ячейки Z26. В данном случае мы поэтапно (для каждого фиксированного аргумента x) вычисляем значение выражения $y(x) = f(x) + \lambda \sum_{i=0}^{n} \omega_i K(x, x_i) y_i$. Для сравнения в ячейках AA6:AA26 приведены значения для функции $y(x)$ в соответствии с формулой $y(x) = 1 - \frac{4x(\lambda - 3)}{3(\lambda - 4)}$. С этой целью мы вводим в ячейку AA6 формулу =1-4*N6*($L$3-3)/3/($L$3-4) и копируем ее в нижние ячейки. Как несложно заметить, совпадение результатов очень даже неплохое (рис. 15.12). Объяснение простое: решением интегрального уравнения является линейная функция, аппроксимация которой параболами в методе Симпсона дает вполне приемлемые результаты.



	L	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA
5	10	Арг. x	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1	решение	точно
6	1,0	0,00	0	0	0	0	0	0	0	0	0	0	0	1,00000	1,00000
7	0,0333	0,05	0	0,0005	0,002	0,0045	0,008	0,0125	0,018	0,0245	0,032	0,0405	0,05	0,96667	0,96667
8	0,0000	0,10	0	0,001	0,004	0,009	0,016	0,025	0,036	0,049	0,064	0,081	0,1	0,93333	0,93333
9	0,33333333	0,15	0	0,0015	0,006	0,0135	0,024	0,0375	0,054	0,0735	0,096	0,1215	0,15	0,90000	0,90000
10		0,20	0	0,002	0,008	0,018	0,032	0,05	0,072	0,098	0,128	0,162	0,2	0,86667	0,86667
11	10	0,25	0	0,0025	0,01	0,0225	0,04	0,0625	0,09	0,1225	0,16	0,2025	0,25	0,83333	0,83333
12	0	0,30	0	0,003	0,012	0,027	0,048	0,075	0,108	0,147	0,192	0,243	0,3	0,80000	0,80000
13	0,1	0,35	0	0,0035	0,014	0,0315	0,056	0,0875	0,126	0,1715	0,224	0,2835	0,35	0,76667	0,76667
14	0,2	0,40	0	0,004	0,016	0,036	0,064	0,1	0,144	0,196	0,256	0,324	0,4	0,73333	0,73333
15	0,3	0,45	0	0,0045	0,018	0,0405	0,072	0,1125	0,162	0,2205	0,288	0,3645	0,45	0,70000	0,70000
16	0,4	0,50	0	0,005	0,02	0,045	0,08	0,125	0,18	0,245	0,32	0,405	0,5	0,66667	0,66667
17	0,5	0,55	0	0,0055	0,022	0,0495	0,088	0,1375	0,198	0,2695	0,352	0,4455	0,55	0,63333	0,63333
18	0,6	0,60	0	0,006	0,024	0,054	0,096	0,15	0,216	0,294	0,384	0,486	0,6	0,60000	0,60000
19	0,7	0,65	0	0,0065	0,026	0,0585	0,104	0,1625	0,234	0,3185	0,416	0,5265	0,65	0,56667	0,56667
20	0,8	0,70	0	0,007	0,028	0,063	0,112	0,175	0,252	0,343	0,448	0,567	0,7	0,53333	0,53333
21	0,9	0,75	0	0,0075	0,03	0,0675	0,12	0,1875	0,27	0,3675	0,48	0,6075	0,75	0,50000	0,50000
22	1	0,80	0	0,008	0,032	0,072	0,128	0,2	0,288	0,392	0,512	0,648	0,8	0,46667	0,46667
23		0,85	0	0,0085	0,034	0,0765	0,136	0,2125	0,306	0,4165	0,544	0,6885	0,85	0,43333	0,43333
24	10	0,90	0	0,009	0,036	0,081	0,144	0,225	0,324	0,441	0,576	0,729	0,9	0,40000	0,40000
25	0	0,95	0	0,0095	0,038	0,0855	0,152	0,2375	0,342	0,4655	0,608	0,7695	0,95	0,36667	0,36667
26	-0,0066667	1,00	0	0,01	0,04	0,09	0,16	0,25	0,36	0,49	0,64	0,81	1	0,33333	0,33333
27	-0,0133333														
28	-0,02														
29	-0,0266667														

Рис. 15.12
Решение интегрального уравнения: окончательные результаты

ГЛАВА 16 ВМЕСТО ЗАКЛЮЧЕНИЯ. СРАВНИТЕЛЬНЫЙ АНАЛИЗ

Только не подумайте удирать!
Я и в Полтаве найду!

Из к/ф «За двумя зайцами»

Заключительную часть всегда писать сложно. Ведь все, что хотел написать в книге, уже вроде бы и написано. А то, что надо было бы написать, и не написал — в *заключение* не втиснешь (да и смысла особого нет). Тем не менее, тема для разговора существует. В начале книги мы вскользь упомянули, что у Excel, как табличного процессора, конкурентов фактически нет. То есть имелось в виду, что конкуренты есть, но им пока до Excel довольно далеко (хотя это, конечно, точка зрения субъективная). Другими словами, неплохо было бы познакомиться и с прочими пунктами «меню», или, на худой конец, с наиболее примечательными его пунктами.

Среди конкурентов Excel наиболее продвинутым (утверждение тоже, кстати, субъективное), пожалуй, является редактор электронных таблиц Calc, который входит в состав пакета Open Office. Далее приведен очень краткий обзор по этому редактору. У кого-то может возникнуть вполне резонный вопрос: зачем это нужно людям, которые целенаправленно изучают/используют Excel? Ответ состоит из нескольких пунктов:

- преимущества и недостатки Excel легче оценить, если есть с чем сравнивать;
- всегда полезно знать, каков выбор на рынке редакторов электронных таблиц. Ведь даже если какое-то приложение уступает по многим параметрам Excel, очень может статься, что для решения какой-то конкретной задачи оно подходит лучше.

Теперь перейдем к конкретике.

ЗНАКОМСТВО С ПРИЛОЖЕНИЕМ OPEN OFFICE CALC

Николай Николаевич, дорогой!
Мне кажется, что Вы, как человек искусства,
излишне драматизируете ситуацию.

Из к/ф «Окно в Париж»

У пакета Open Office есть несколько примечательных свойств, среди которых можно выделить два:

- пакет распространяется бесплатно, т. е. можно вполне официально загрузить установочные файлы и без ощутимых финансовых последствий установить и использовать пакет;

- пакет достаточно профессиональный (в первую очередь в плане редактора электронных таблиц), и позволяет решать широкий спектр задач. С учетом первого обстоятельства, это довольно неплохо.

На заметку

Установочные файлы русифицированного пакета Open Office можно найти на сайте **www.openoffice.org**. Здесь мы описываем версию пакета Open Office 3.3.

Пакет разрабатывался компанией Sun Microsystems, которую поглотила корпорация Oracle. И теперь на «гербе» пакета логотип **ORACLE**. На рисунке 16.1 представлена заставка пакета Open Office, в которой выбирается тип приложения.

Нас, как подчеркивалось, интересует редактор электронных таблиц. Поэтому в окне-заставке щелкаем пиктограмму **Электронная таблица**. В результате открывается редактор электронных таблиц, рабочее окно которого представлено на рисунке 16.2.

Для тех, кто помнит старый добрый интерфейс Excel с меню и панелями инструментов, в этом рабочем окне есть много знакомых мотивов. Проще говоря, рабочее окно редактора электронных таблиц Open Office Calc выполнено в стиле «панелей инструментов». Все остальное очень сильно напоминает приложение Excel (особенно до версии Excel 2003). В первую очередь, это замечание касается рабочей области приложения.

На заметку

Здесь, скорее, проявляется не недостаток фантазии разработчиков, но в большей мере универсальность организации электронной таблицы. Ведь если что-то придумано хорошо, то придумать что-то лучшее крайне сложно.

Рис. 16.1
Заставка пакета Open Office

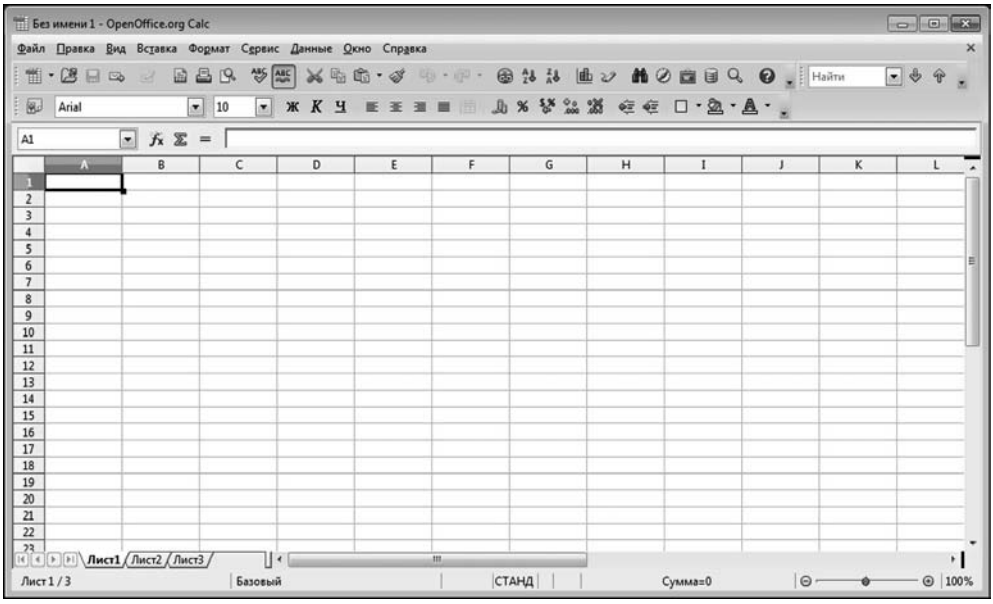


Рис. 16.2
Окно приложения редактора электронных таблиц Open Office Calc

По умолчанию рабочий документ Open Office Calc состоит из трех рабочих листов, корешки которых отображаются в нижней части рабочей области. Сама рабочая область, если не считать некоторых дизайнерских особенностей, до боли напоминает рабочую область приложения Excel. Методы работы с Open Office Calc тоже во многом схожи с теми, что мы наблюдали ранее при обсуждении методов работы с Excel.

В Open Office Calc использована классическая схема именования ячеек: столбцы ячеек именуются буквами, а строки ячеек просто нумеруются. Например, ячейка B3 находится на пересечении второго столбца и третьей строки. Формулы вводятся, начиная со знака равенства (т. е. =). Копирование ячеек, эта визитная карточка электронной таблицы, выполняется с учетом относительности и абсолютности ссылок.

На заметку

Как и в приложении Excel, в Open Office Calc для обозначения абсолютных ссылок используется знак доллара $. Так, ссылка B3 является относительной. Если она входит в формулу, то при копировании формулы ссылка автоматически будет изменяться. Напомним, относительные ссылки меняются так, чтобы относительное положение ячеек не изменилось (имеются в виду ячейка с формулой и ячейка, на которую выполнена ссылка). Чтобы сделать ссылку на ячейку абсолютной, ссылка выполняется как $B$3. При копировании такая ссылка не меняется и все время «указывает» на ячейку B3. Ссылка может быть смешанной — абсолютной/относительной по столбцу и относительной/абсолютной по строке (например, $B3 или B$3). То есть все, как в Excel.

Ввод значений в ячейки также выполняется без особой оригинальности. Данные в ячейку (выделенную) можно вводить непосредственно, или в строку формул (которая в Open Office Calc расположена в том же месте рабочей области, что и в Excel). Идеологически близки у Open Office Calc и Excel и другие редакционные операции и процедуры. Например, в Open Office Calc можно выделить диапазон ячеек и переместить его в другое место рабочего документа (перетаскиванием с помощью мыши).

На заметку

Разумеется, не все так одинаково в Open Office Calc и Excel. Даже на поверхностном уровне. Например, такая банальная процедура, как выделение диапазона ячеек, в Open Office Calc внешне проявляется иначе, чем в Excel. На рисунке 16.3 показано, как будет выглядеть в Open Office Calc диапазон ячеек. Рамкой выделяется правая нижняя ячейка выбранного диапазона. В Excel в таком случае рамка вокруг всего диапазона. Да и курсор в рабочей области приложения Open Office Calc выглядит иначе, чем в Excel. Однако, разумеется, не эти мелочи являются определяющими при сравнении приложений.

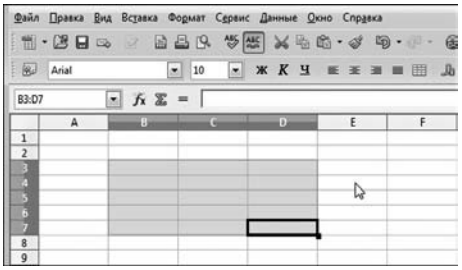


Рис. 16.3
В окне приложения Open Office Calc выделен диапазон ячеек

Есть в Open Office Calc и встроенные функции, которые, правда, даже в русифицированной версии приложения имеют английские названия. Также в Open Office Calc легко и просто создаются диаграммы, и записываются макросы.

ПРИМЕР ВЫЧИСЛЕНИЙ В OPEN OFFICE CALC

Никаких ученых!
Вдвоем все поделим!

Из к/ф «Окно в Париж»

Мы не ставим перед собой цель дать полный обзор по приложению Open Office Calc. Ограничимся лишь иллюстрацией некоторых возможностей. Возможности имеет смысл демонстрировать на конкретных примерах. Поэтому здесь рассмотрим процесс табулирования функции и создания на основе этой таблицы диаграммы.

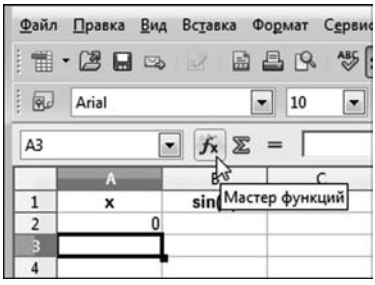


Рис. 16.4
Щелчок на пиктограмме **Мастер функций** для ввода формулы

На рисунке 16.4 представлен документ Open Office Calc на самой начальной стадии табулирования функции.

В столбец A мы будем заносить значения узловых точек, а в столбец B будем заносить

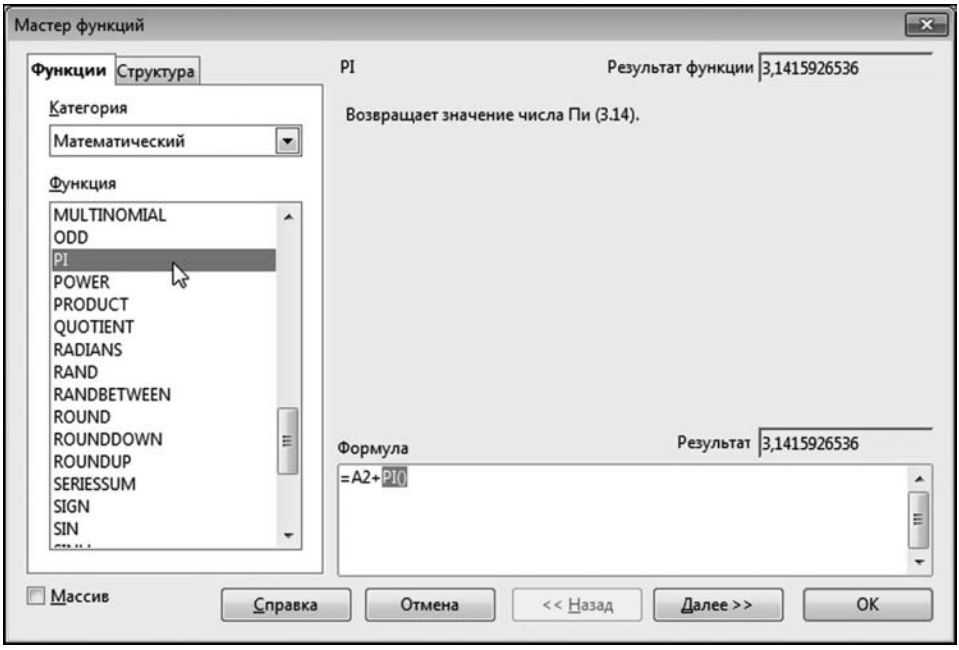


Рис. 16.5
Добавление функции PI() (вычисление числа π) в формулу

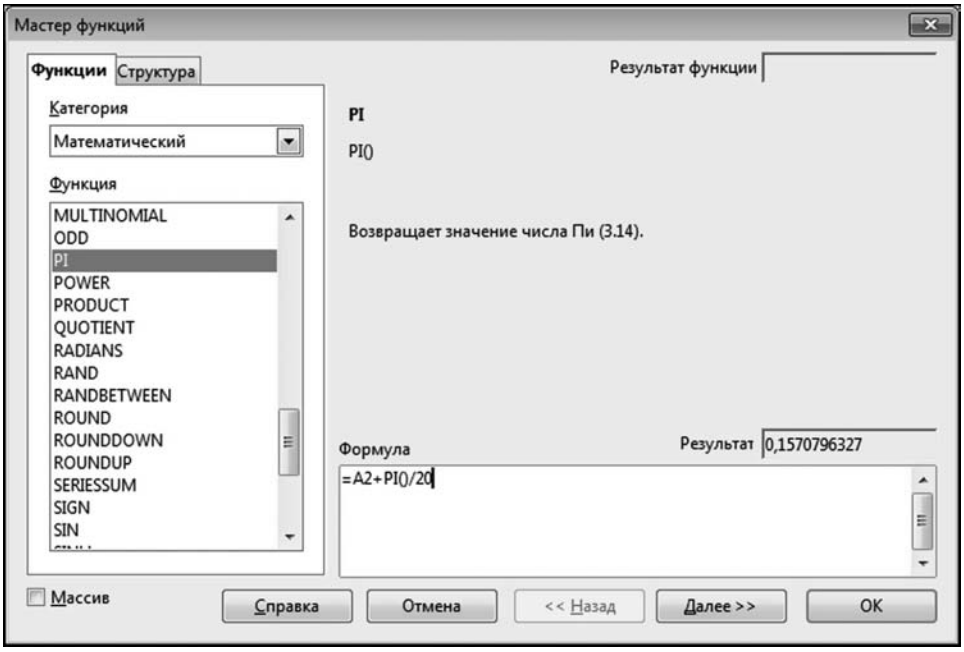


Рис. 16.6
Завершение ввода формулы в диалоговом окне **Мастер функций**

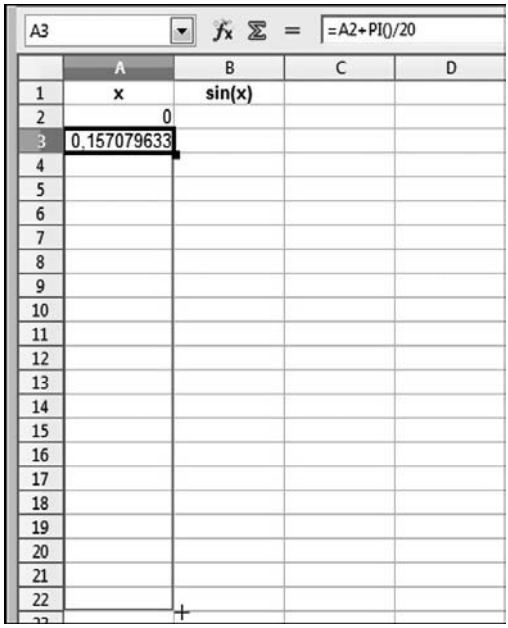


Рис. 16.7
Автоматическое заполнение диапазона ячеек

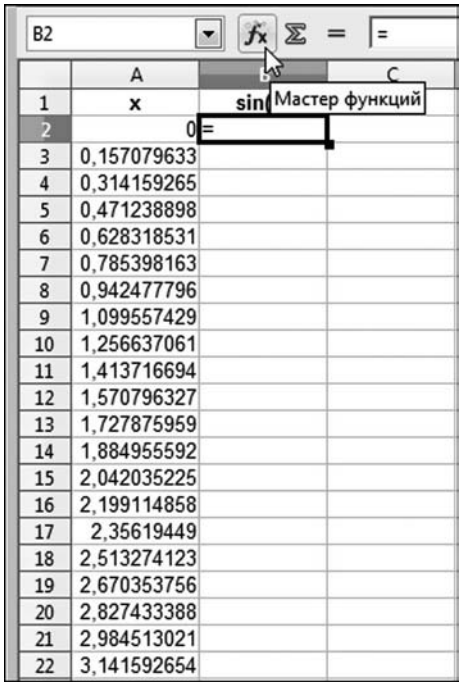


Рис. 16.8
Ввод формулы для вычисления значения функции в узловой точке

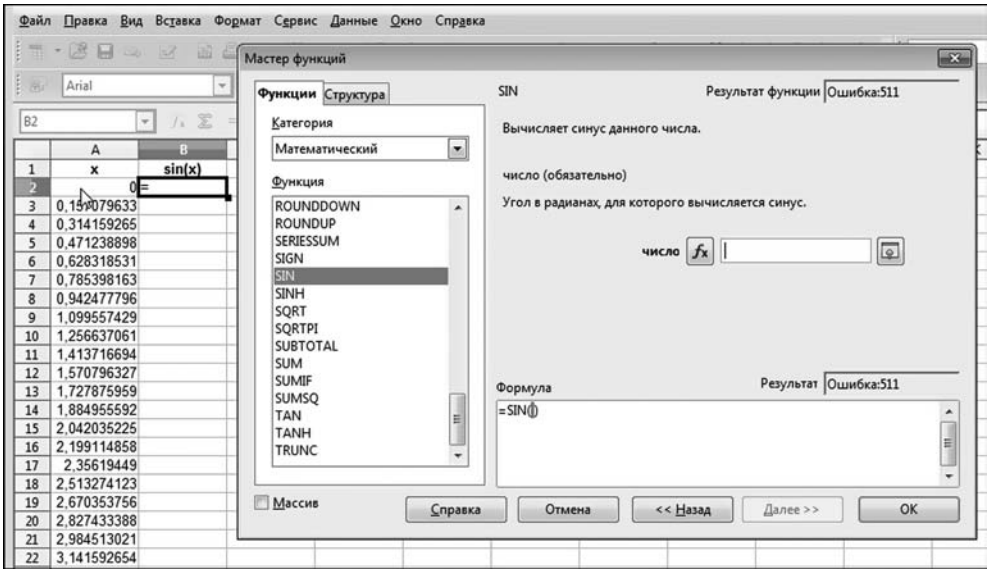


Рис. 16.9
Вставка в формулу функции для вычисления синуса и ее аргумента

значения функции $f(x) = \sin x$. В ячейку A2 внесено начальное нулевое значение, а при заполнении ячейки A3 щелкаем на пиктограмме с изображением функции (пиктограмма находится слева от строки формул — первая из трех). В результате откроется диалоговое окно **Мастер функций**, представленное на рисунке 16.5.

В окне мы находим функцию PI() для вычисления числа π. При этом в области **Формула** можно вводить полностью формулу. Мы ввели =A2+PI()/20, как показано на рисунке 16.6.

После того как в ячейку A3 введено значение (формула), методом автоматического заполнения заполняем диапазон ячеек в столбце A до 22-й строки включительно. Процесс заполнения такой же, как в Excel — захватываем маркером заполнения область для копирования формул, как показано на рисунке 16.7.

Результат заполнения диапазона ячеек A2:A22 проиллюстрирован на рисунке 16.8.

На следующем этапе заполняем столбец B значениями функции в узловых точках, для чего в ячейку B2 вводим соответствующую формулу — для этого опять используем окно **Мастер функций** (рис. 16.9).

В данном случае нас интересует функция SIN() для вычисления синуса. Стоит отметить, что, если у функции есть аргументы, то в окне **Мастер функций** можно указать и аргументы. Более того, если аргумент — адреса ячеек, то выбрать ячейки можно прямо в рабочем документе (рис. 16.10).

В результате ячейка B2 получает формулу =SIN(A2). После этого снова воспользуемся методом автоматического заполнения (рис. 16.11).

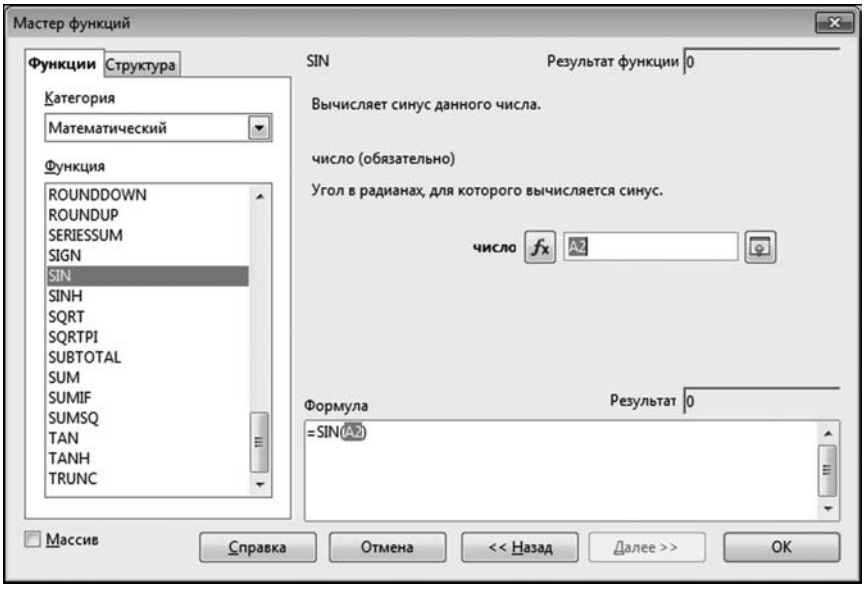
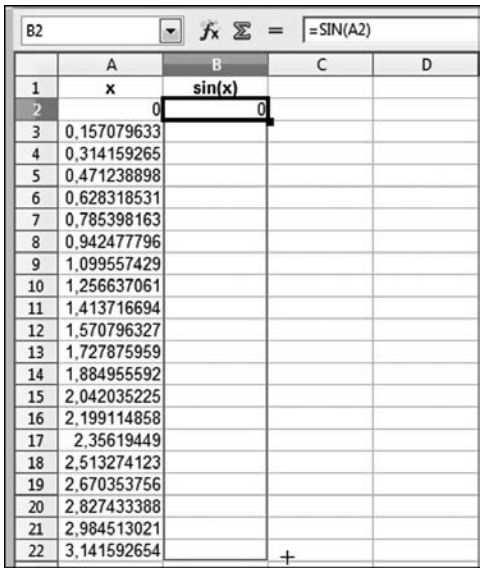
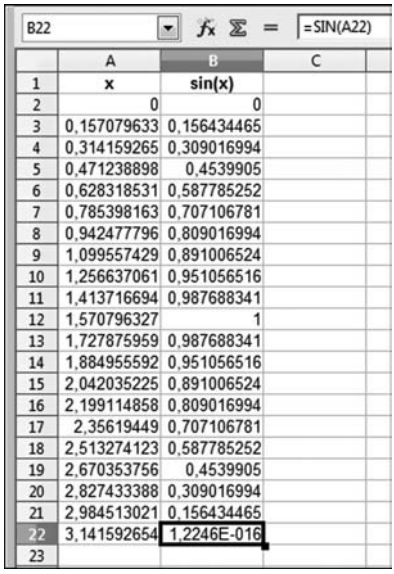


Рис. 16.10
Окно **Мастер функций** с формулой для вычисления значения функции в узловой точке



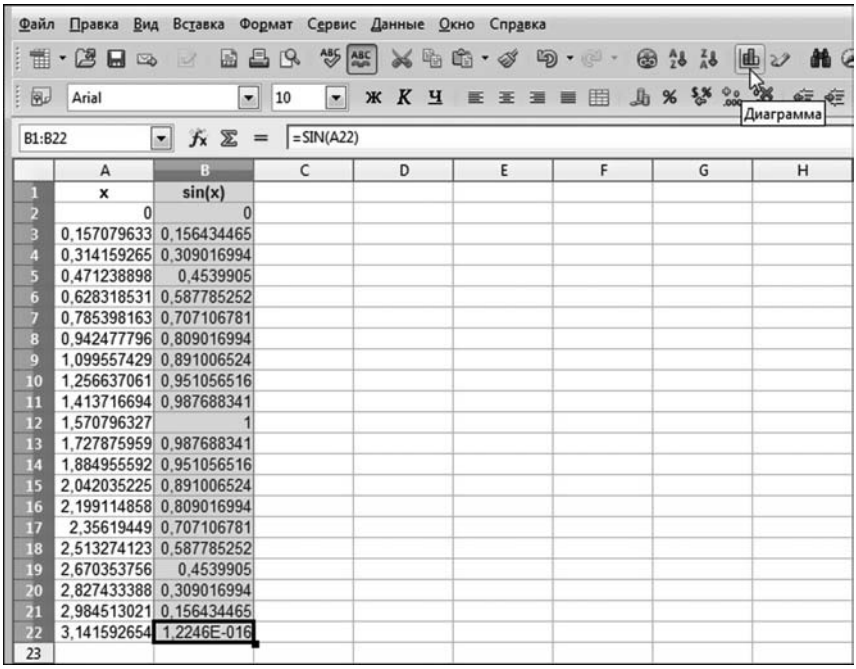
	A	B	C	D
1	x	sin(x)		
2	0	0		
3	0,157079633			
4	0,314159265			
5	0,471238898			
6	0,628318531			
7	0,785398163			
8	0,942477796			
9	1,099557429			
10	1,256637061			
11	1,413716694			
12	1,570796327			
13	1,727875959			
14	1,884955592			
15	2,042035225			
16	2,199114858			
17	2,35619449			
18	2,513274123			
19	2,670353756			
20	2,827433388			
21	2,984513021			
22	3,141592654			

Рис. 16.11
Автоматическое заполнение ячеек для вычисления значения функции в узловых точках



	A	B	C
1	x	sin(x)	
2	0	0	
3	0,157079633	0,156434465	
4	0,314159265	0,309016994	
5	0,471238898	0,4539905	
6	0,628318531	0,587785252	
7	0,785398163	0,707106781	
8	0,942477796	0,809016994	
9	1,099557429	0,891006524	
10	1,256637061	0,951056516	
11	1,413716694	0,987688341	
12	1,570796327	1	
13	1,727875959	0,987688341	
14	1,884955592	0,951056516	
15	2,042035225	0,891006524	
16	2,199114858	0,809016994	
17	2,35619449	0,707106781	
18	2,513274123	0,587785252	
19	2,670353756	0,4539905	
20	2,827433388	0,309016994	
21	2,984513021	0,156434465	
22	3,141592654	1,2246E-016	
23			

Рис. 16.12
В рабочем документе вычислены значения узловых точек и функции в узловых точках



	A	B	C	D	E	F	G	H
1	x	sin(x)						
2	0	0						
3	0,157079633	0,156434465						
4	0,314159265	0,309016994						
5	0,471238898	0,4539905						
6	0,628318531	0,587785252						
7	0,785398163	0,707106781						
8	0,942477796	0,809016994						
9	1,099557429	0,891006524						
10	1,256637061	0,951056516						
11	1,413716694	0,987688341						
12	1,570796327	1						
13	1,727875959	0,987688341						
14	1,884955592	0,951056516						
15	2,042035225	0,891006524						
16	2,199114858	0,809016994						
17	2,35619449	0,707106781						
18	2,513274123	0,587785252						
19	2,670353756	0,4539905						
20	2,827433388	0,309016994						
21	2,984513021	0,156434465						
22	3,141592654	1,2246E-016						
23								

Рис. 16.13
Создание диаграммы на основе выделенного диапазона ячеек

На заметку

Совсем необязательно для вставки функции использовать окно **Мастер функций**. Намного проще ввести имя функции с клавиатуры. Правда, для этого необходимо знать, как называется нужная функция.

Документ с заполненными ячейками A1:B22 показан на рисунке 16.12.

Для создания диаграммы на основе этих данных выделяем, например, диапазон ячеек B1:B22, и щелкаем пиктограмму **Диаграмма** на панели инструментов, как показано на рисунке 16.13.

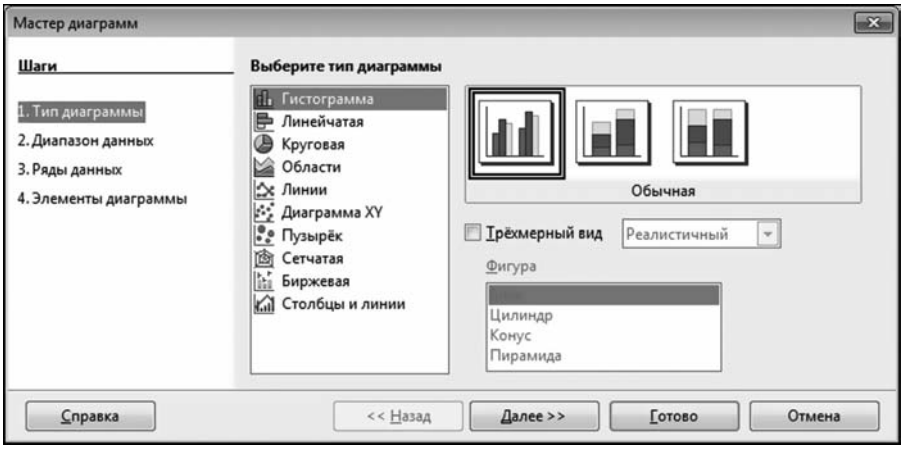
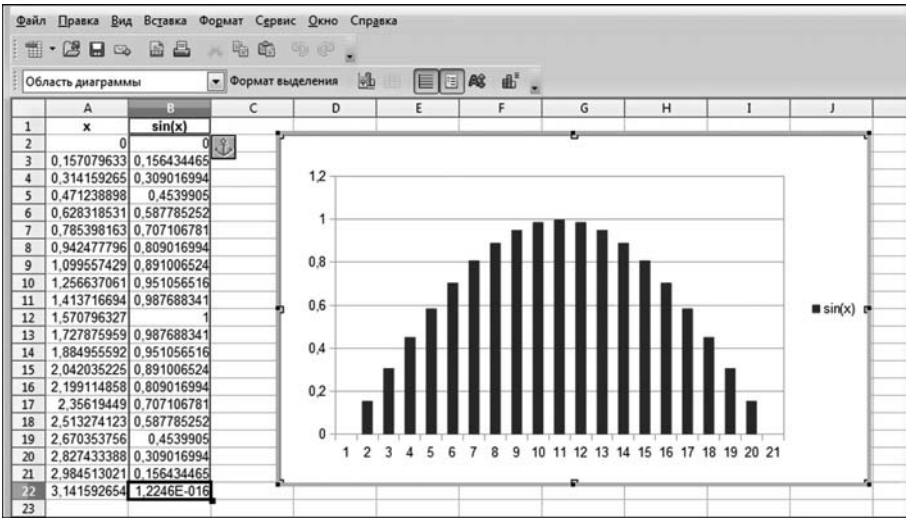


Рис. 16.14
Диалоговое окно **Мастер диаграмм** для настройки вида создаваемой диаграммы



	A	B
1	x	sin(x)
2	0	0
3	0,157079633	0,156434465
4	0,314159265	0,309016994
5	0,471238898	0,4539905
6	0,628318531	0,587785252
7	0,785398163	0,707106781
8	0,942477796	0,809016994
9	1,099557429	0,891006524
10	1,256637061	0,951056516
11	1,413716694	0,987688341
12	1,570796327	1
13	1,727875959	0,987688341
14	1,884955592	0,951056516
15	2,042035225	0,891006524
16	2,199114858	0,809016994
17	2,35619449	0,707106781
18	2,513274123	0,587785252
19	2,670353756	0,4539905
20	2,827433388	0,309016994
21	2,984513021	0,156434465
22	3,141592654	1,2246E-016
23		

Рис. 16.15
Диаграмма в рабочем документе

Автоматически будет добавлена диаграмма, и на ее фоне откроется окно настройки вида диаграммы. Это окно называется **Мастер диаграмм**. Оно представлено на рисунке 16.14.

В этом окне выбирается тип диаграммы, диапазон данных, на основе которых строится диаграмма, названия, подписи и т. д. Мы выбираем то, что предложено по умолчанию. Результат представлен на рисунке 16.15.

На заметку

Откровенно говоря, в данном случае лучше бы подошла диаграмма типа **Диаграмма XY**. Такого типа диаграммы используют для отображения графиков функций.

ОБЗОР НАСТРОЕК И РЕЖИМОВ

— Вы меня понимаете, Проня Прокоповна?
— Очень! Мене тоже после пансиона все другим кажется.

Из к/ф «За двумя зайцами»

В этом разделе мы очень кратко пройдемся по содержимому панели меню приложения Open Office Calc, поскольку именно в меню «спрятаны» все основные «полезности» приложения. На рисунке 16.16 представлен в раскрытом виде пункт меню **Файл**.

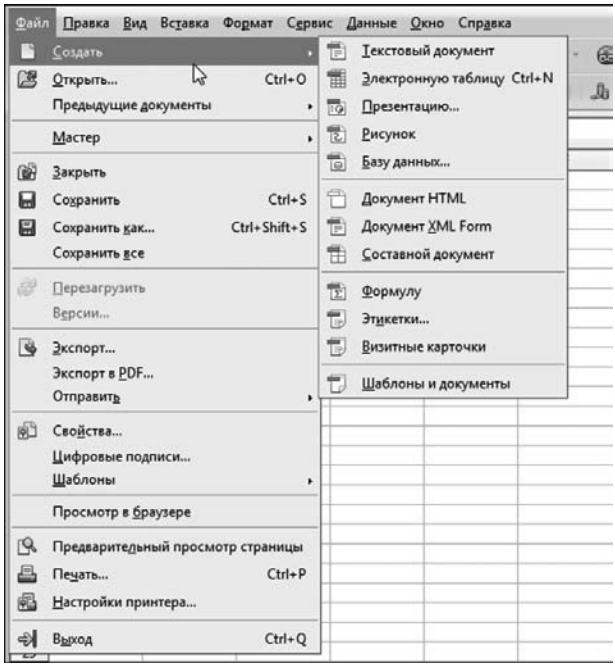


Рис. 16.16
Содержимое меню **Файл**

Команды этого меню предназначены в основном для работы с файлами и документами приложения. Редакторские операции с рабочими документами выполняются с помощью команд меню **Правка** (рис. 16.17).

Настройки внешнего вида приложения (вроде размещения/удаления панелей инструментов) выполняются посредством команд меню **Вид** (рис. 16.18).

В меню **Вставка** собраны в основном команды, которые предназначены для вставки всевозможных элементов и объектов (рис. 16.19).

В этом же меню, кстати, представлены команды для вставки функций, всевозможных объектов (вроде графических формул) и диаграмм. Что касается утилит форматирования, то соответствующие команды собраны в основном в меню **Формат** (рис. 16.20).

Достаточно полезные и «тяжеловесные» команды есть в меню **Сервис** (рис. 16.21).

Для работы с данными предназначены команды в пункте меню **Данные** (рис. 16.22).

Команды меню **Окно** обычно нужны в том случае, если приходится иметь дело сразу с несколькими рабочими окнами (рис. 16.23).

В меню **Справка** собраны утилиты, которые позволяют получить справочную или дополнительную информацию по приложению (рис. 16.24).

Каково резюме всего этого беглого обзора? А резюме таково: то, что в Excel представлено на ленте, в Open Office Calc собрано в командах панели меню. То есть, если нужно выполнить определенное действие, команду ищем в списках пунктов меню. Для наиболее популярных команд представлены специальные пиктограммы на панелях инструментов. Эти самые панели можно добавлять или убирать по своему усмотрению (подменю **Вид** ▷ **Панели инструментов**, рис. 16.18). Кроме того, есть группа настроек, которые выполняются в окне **Настройка** (рис. 16.25).

Здесь, в частности, можно настроить и панель меню. Окно настроек открывается командой **Сервис** ▷ **Настройка**. Если воспользоваться командой **Сервис** ▷ **Параметры**, увидим диалоговое окно **Параметры**, в котором задается ряд важных режимов и характеристик приложения (рис. 16.26).

Фактически, эти два диалоговых окна «имитируют» окно настроек **Параметры Excel**, хотя полной аналогии, конечно, нет.

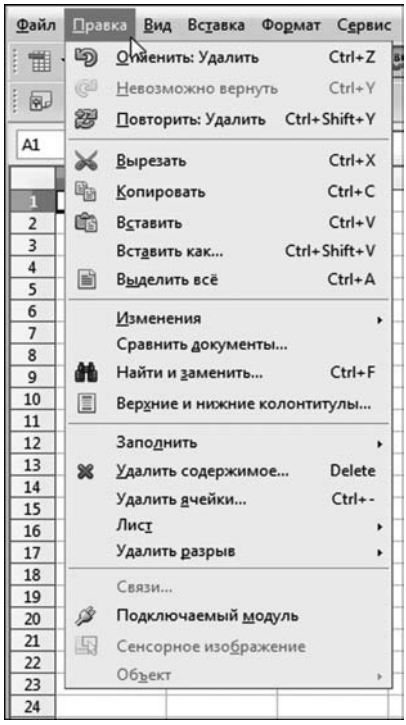


Рис. 16.17
Содержимое меню **Правка**

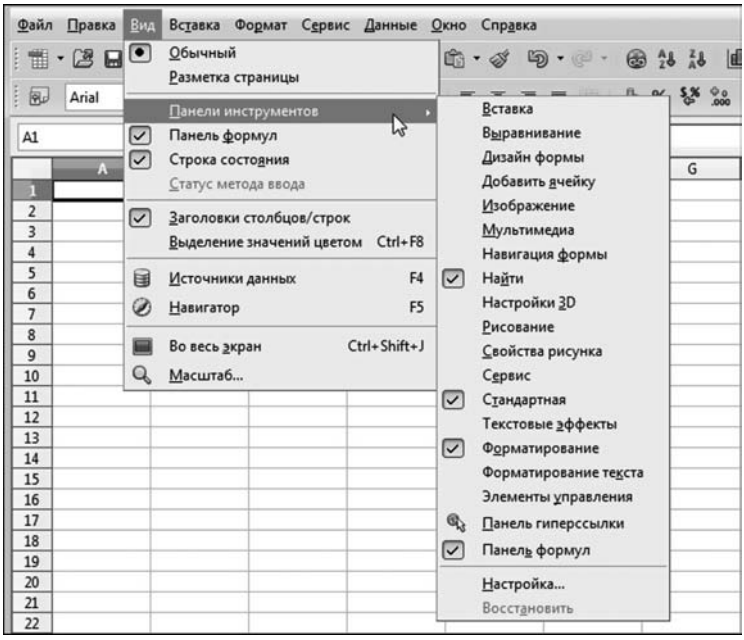


Рис. 16.18
Содержимое меню **Вид**

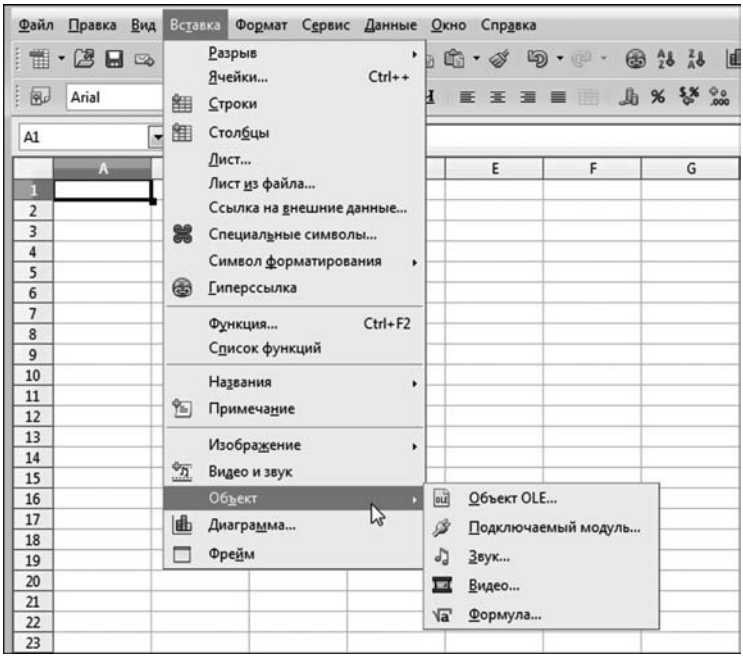


Рис. 16.19
Содержимое меню **Вставка**

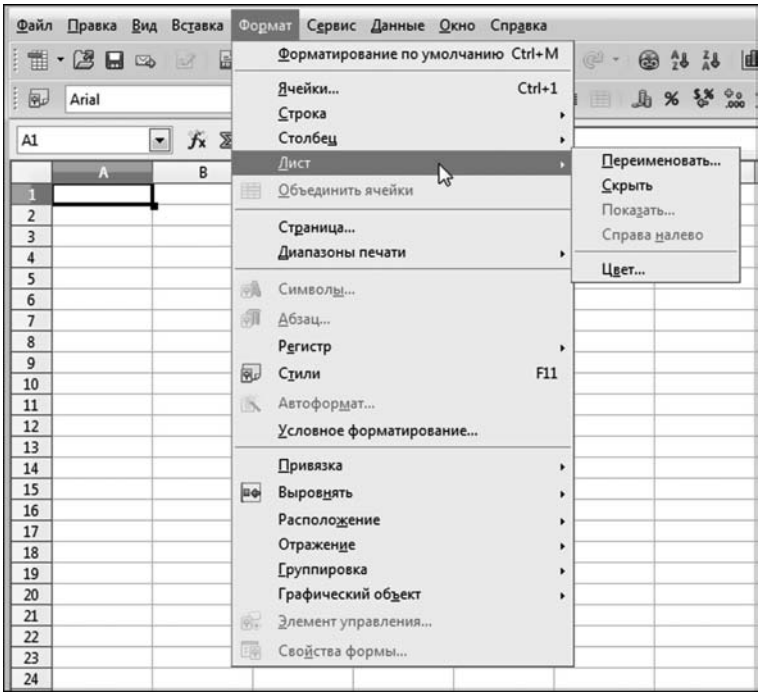


Рис. 16.20
Содержимое меню **Формат**

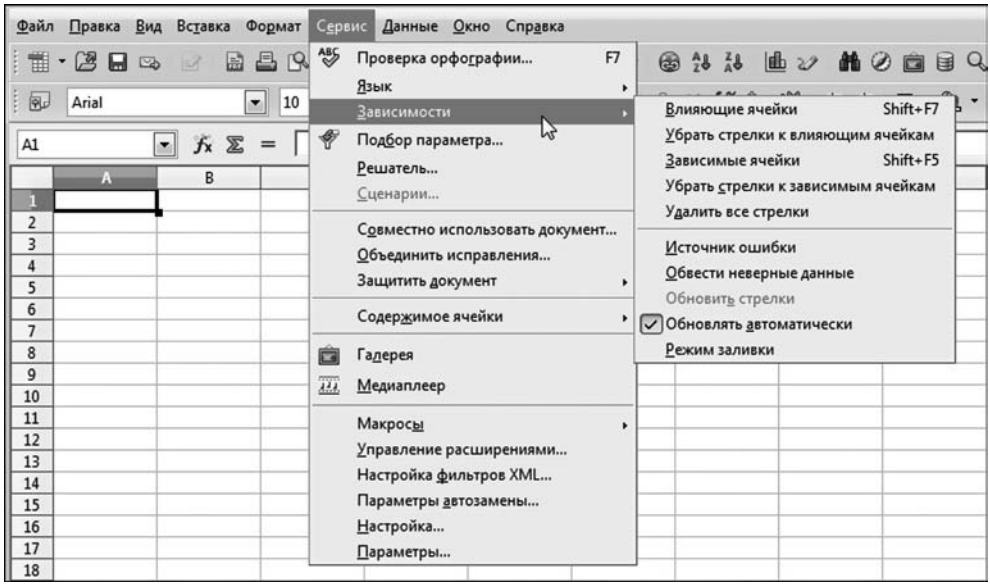


Рис. 16.21
Содержимое меню **Сервис**

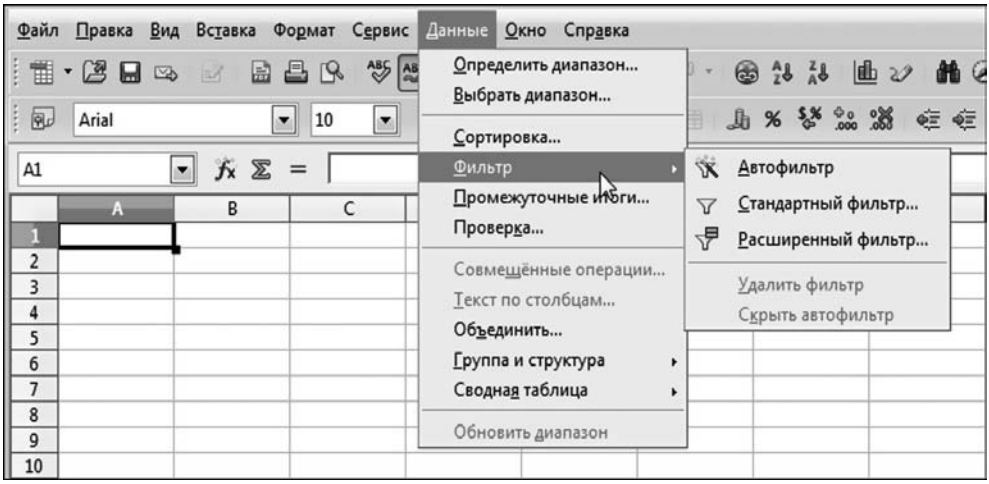


Рис. 16.22
Содержимое меню **Данные**

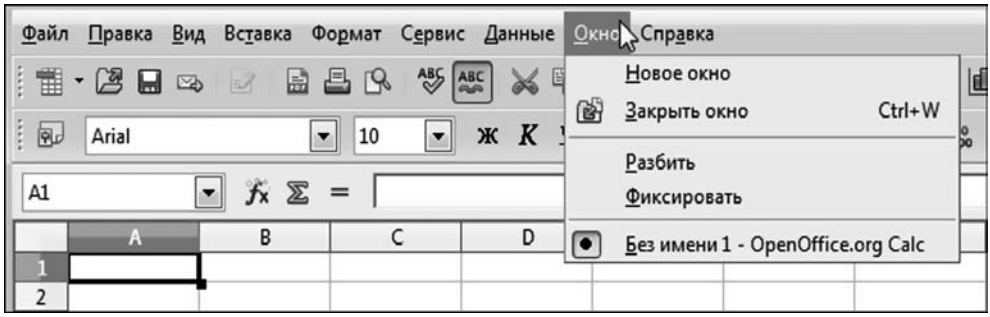


Рис. 16.23
Содержимое меню **Окно**

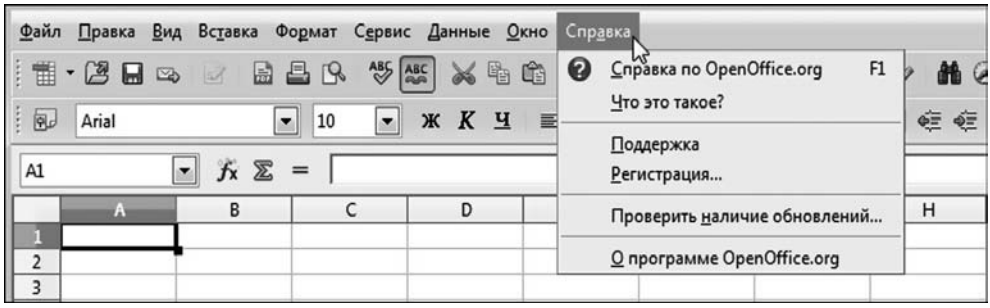


Рис. 16.24
Содержимое меню **Справка**

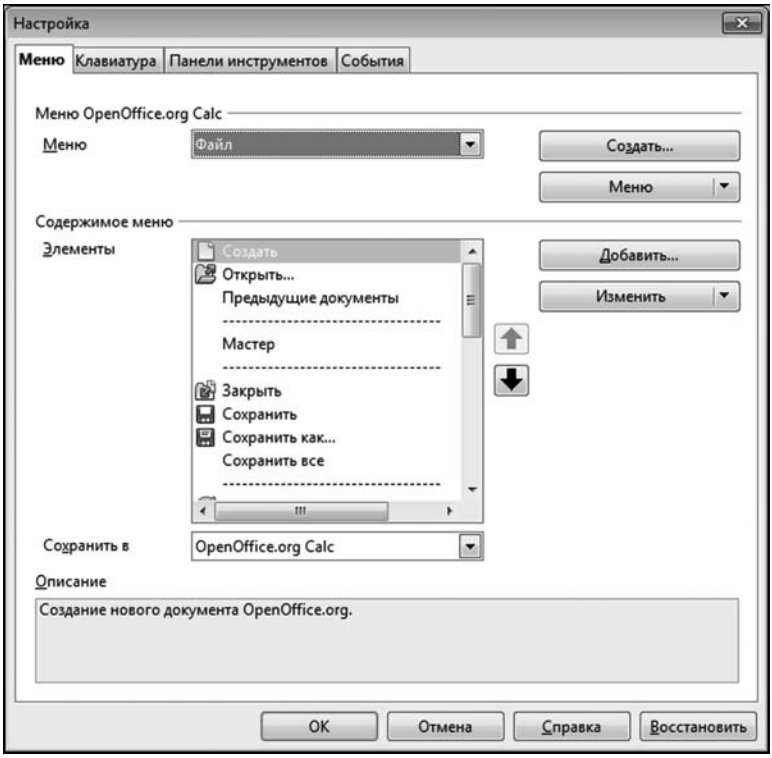


Рис. 16.25
Диалоговое окно **Настройка**

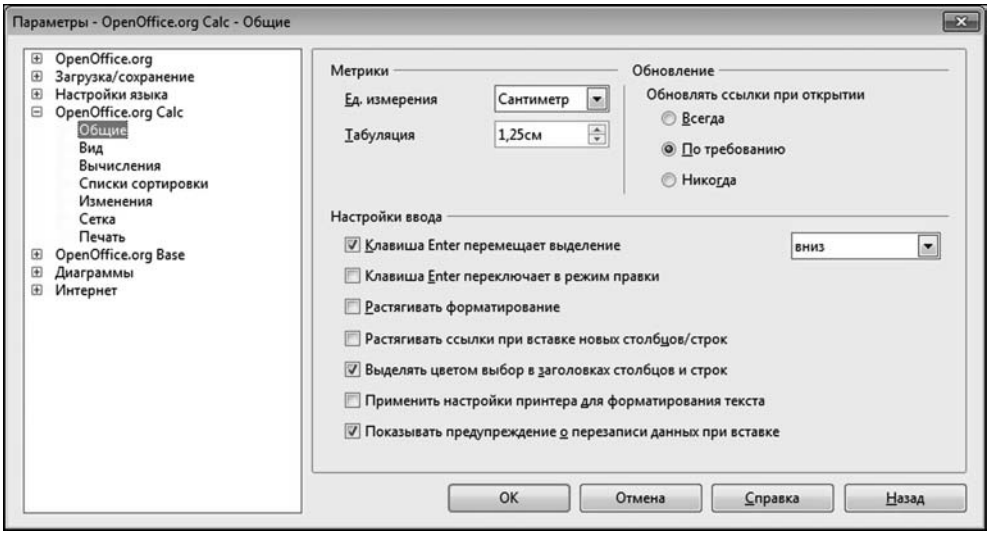


Рис. 16.26
Диалоговое окно **Параметры**

СРАВНЕНИЕ ПОДХОДОВ

Вот это верно. Виктория не самая блестящая.
Однако давайте повременим с самобичеванием,
поскольку изобличить их, кроме нас, некому.

Из к/ф «ТАСС уполномочен заявить»

Может сложиться впечатление, что разница между приложением Open Office Calc и Excel сводится в основном к интерфейсам разных типов. Конечно же, это не так. Приведем лишь несколько примеров. Выше упоминалось, что в Open Office Calc, как и в Excel, можно записывать макросы. Для работы с макросами предназначена группа команд в подменю **Сервис ▷ Макросы** (рис. 16.27).

То есть внешне все вроде так же (или почти так же), как в Excel. Тем не менее процесс записи и редактирования макросов в Open Office Calc требует от пользователя более высокого уровня профессиональной подготовки. И это обстоятельство нельзя сбрасывать со счетов.

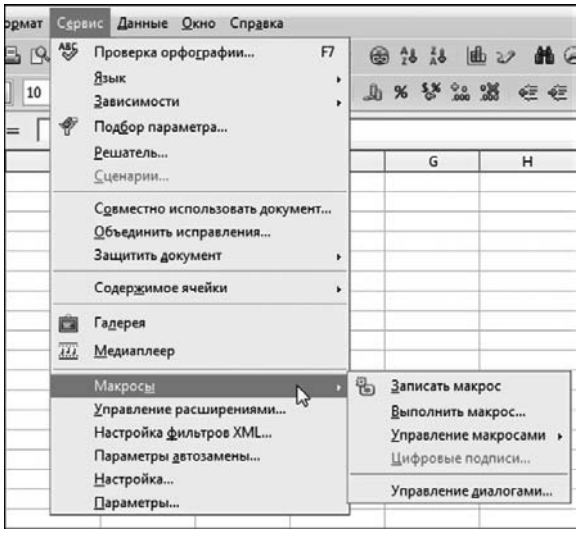


Рис. 16.27
В подменю **Сервис ▷ Макросы** содержатся команды для работы с макросами

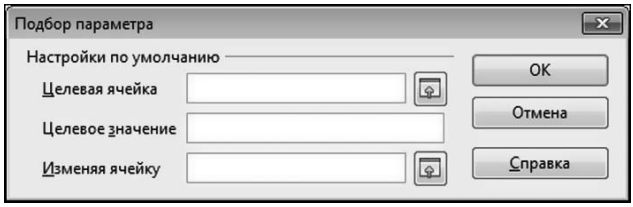


Рис. 16.28
Диалоговое окно утилиты **Подбор параметра**

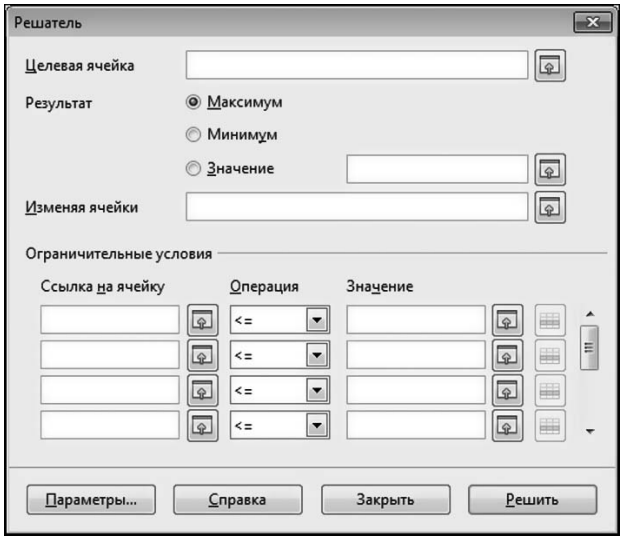


Рис. 16.29
Диалоговое окно утилиты **Решатель**

Вместе с тем, не все так плохо. Так, утилита **Подбор параметра**, которая запускается командой **Сервис ▷ Подбор параметра** и окно которой показано на рисунке 16.28, функционирует практически так же, как одноименная утилита Excel.

Появился в Open Office Calc и аналог надстройки **Поиск решения**, который в русскоязычном варианте приложения называется **Решатель**. Запускается утилита командой **Сервис ▷ Решатель**. Окно утилиты показано на рисунке 16.29.

Утилита позволяет решать задачи по оптимизации (с ограничениями) и решению уравнений. Однако есть проблема: по умолчанию используются алгоритмы решения линейных задач, что существенно снижает полезность утилиты.

На заметку

Данная проблема устранима. При желании можно «научить» **Решатель** решать и нелинейные задачи — соответствующие дополнения к утилите уже разработаны и находятся в свободном доступе.

С другой стороны, нелинейные задачи — это уже заявка на серьезные математические вычисления, которые обычно выполняют с помощью специальных пакетов. Так что некоторая степень неоднозначности остается. Эта неоднозначность дает перспективу. А наличие перспективы — это однозначно лучше, чем ее отсутствие.

ЛИТЕРАТУРА

— Этнографическая экспедиция.
— Понятно. Нефть ищете?

Из к/ф «Кавказская пленница»

Здесь приводится список литературы, который может быть полезным при работе над примерами, рассмотренными в книге. Для удобства все книги разбиты на две категории. В одной собраны ссылки на книги по методам работы с приложением Excel. В другой представлены книги по числовым методам. Думается, это облегчит читателю процесс «осмысления» материала книги. Списки достаточно субъективные (или оптимизированные — кому как больше нравится). Вместе с тем, эти книги, хочется верить, покажутся интересными читателю.

КНИГИ ПО EXCEL И VBA

1. *Васильев, А. Н.* Научные вычисления в Microsoft Excel. — М. : ООО «И. Д. Вильямс», 2004. — 512 с.
2. *Васильев, А.* Excel 2007 на примерах. — СПб. : БХВ-Петербург, 2007. — 656 с.
3. *Васильев, А.* Excel 2010 на примерах. — СПб. : БХВ-Петербург, 2010. — 422 с.
4. *Кузьменко, В. Г.* VBA. Эффективное использование. — М. : Бином-Пресс, 2012. — 624 с.
5. *Курбатова, Е. А.* Microsoft Office Excel 2010. Самоучитель. — М. : ООО «И. Д. Вильямс», 2010. — 416 с.
6. *Ларсен, Р. У.* Инженерные расчеты в Excel. — М. : ООО «И.Д. Вильямс», 2004. — 544 с.
7. *Леонов, В.* Функции Excel 2010. — М. : Эксмо, 2011. — 560 с.
8. *Сингаевская, Г. И.* Функции в Microsoft Office Excel 2010. — М. : ООО «И. Д. Вильямс», 2011. — 1094 с.
9. *Уокенбах, Д.* Excel 2010. Библия пользователя. — М. : ООО «И. Д. Вильямс», 2011. — 912 с.
10. *Уокенбах, Д.* Excel 2010. Профессиональное программирование на VBA. — М. : ООО «И. Д. Вильямс», 2011. — 944 с.

КНИГИ ПО ЧИСЛОВЫМ МЕТОДАМ И ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКЕ

1. *Березин, И. С.* Методы вычислений. Т. 1 / И. С. Березин, Н. П. Жидков. — М. : ГИФМЛ, 1959. — 464 с.
2. *Березин, И. С.* Методы вычислений. Т. 1 / И. С. Березин, Н. П. Жидков. — М. : ГИФМЛ, 1959. — 620 с.
3. *Демидович, Б. П.* Основы вычислительной математики / Б. П. Демидович, И. А. Марон. — М. : Наука, 1970. — 664 с.
4. *Калиткин, Н. Н.* Численные методы. — М. : Наука, 1972. — 512 с.
5. *Каханер, Д.* Численные методы и математическое обеспечение / Д. Каханер, К. Моулер, С. Неш. — М. : Мир, 1998. — 575 с.
6. *Марчук, Г.* Методы вычислительной математики. — М. : Наука, 1980. — 534 с.
7. *Никифоров, А. Ф.* Основы теории специальных функций / А. Ф. Никифоров, В. Б. Уваров. — М. : Наука, 1974. — 304 с.
8. *Никифоров, А. Ф.* Специальные функции математической физики / А. Ф. Никифоров, В. Б. Уваров. — М. : Наука, 1978. — 320 с.
9. *Самарский, А. А.* Введение в численные методы. — М. : Наука, 1982. — 272 с.

ОГЛАВЛЕНИЕ

ЧАСТЬ ВТОРАЯ

ВЫЧИСЛЕНИЯ И ПРЕДСТАВЛЕНИЕ РЕЗУЛЬТАТОВ

Алексей Николаевич ВАСИЛЬЕВ

ЧИСЛОВЫЕ РАСЧЕТЫ В EXCEL

Учебное пособие

Ответственный редактор *А. Д. Пузовик*
Технический редактор *А. С. Кузьмина*
Корректор *Т. В. Ананченко*
Подготовка иллюстраций *А. П. Маркова*
Верстка *М. И. Хетерели*
Выпускающие *Т. С. Симонова, Н. В. Черезова*

ЛР № 065466 от 21.10.97
Гигиенический сертификат 78.01.07.953.П.007216.04.10
от 21.04.2010 г., выдан ЦГСЭН в СПб

Издательство «ЛАНЬ»
lan@lanbook.ru; www.lanbook.com
192029, Санкт-Петербург, Общественный пер., 5.
Тел./факс: (812) 412-29-35, 412-05-97, 412-92-72.
Бесплатный звонок по России: 8-800-700-40-71

Подписано в печать 16.01.14.
Бумага офсетная. Гарнитура Школьная. Формат 70×100 $^{1}/_{16}$.
Печать офсетная. Усл. п. л. 49,4. Тираж 1500 экз.

Заказ № .

Отпечатано в ОАО «Первая образцовая типография»,
филиал «Чеховский Печатный Двор»
в полном соответствии с качеством предоставленного оригинал-макета.
142300, Московская обл., г. Чехов, ул. Полиграфистов, д. 1.
Тел.: (495) 988-63-76, факс: 8 (496) 726-54-10.