

АНДРЕЙ ГАРНАЕВ, ЛАДА РУДИКОВА



Microsoft Office
Excel 2010
разработка
приложений

БОЛЕЕ 300 ПРИМЕРОВ
РАЗРАБОТКИ ПРИЛОЖЕНИЙ
НА VBA

КОНСТРУИРОВАНИЕ
ПОЛЬЗОВАТЕЛЬСКОГО
ИНТЕРФЕЙСА И ФОРМ

АВТОМАТИЗАЦИЯ
ПОВСЕДНЕВНОЙ РАБОТЫ

ИСПОЛЬЗОВАНИЕ ФОРМУЛ
И ФУНКЦИЙ РАБОЧЕГО ЛИСТА

ОБРАБОТКА И АНАЛИЗ
ДАННЫХ

РЕШЕНИЕ РАСЧЕТНЫХ ЗАДАЧ

РАБОТА С БАЗАМИ ДАННЫХ

ИНТЕГРАЦИЯ ПРИЛОЖЕНИЙ



PRO

ПРОФЕССИОНАЛЬНОЕ
ПРОГРАММИРОВАНИЕ

УДК 681.3.06
ББК 32.973.26-018.2
Г20

Гарнаев, А. Ю.

Г20 Microsoft Office Excel 2010: разработка приложений / А. Ю. Гарнаев, Л. В. Рудикова. — СПб.: БХВ-Петербург, 2011. — 528 с.: ил. + (CD-ROM) — (Профессиональное программирование)

ISBN 978-5-9775-0042-5

Продемонстрированы широкие возможности Microsoft Office Excel 2010 по созданию приложений средствами VBA, работе с макросами, технологии ООП, конструированию пользовательского интерфейса и форм. Рассмотрены вопросы автоматизации операций с рабочим листом и диаграммами, в том числе при обработке и анализе данных и принятии решений. Изложены методы интеграции офисных приложений, работы с Интернетом и базами данных, применения XML. Книга содержит более 300 примеров тщательно разработанных приложений: от создания пользовательских функций до построения информационных систем по сбору и обработке данных, программный код которых может быть непосредственно использован читателем при разработке собственных проектов.

Прилагаемый компакт-диск содержит файлы рассмотренных в книге примеров.

Для программистов, преподавателей и студентов

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Евгений Рыбаков</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Анна Кузьмина</i>
Компьютерная верстка	<i>Натальи Караваевой</i>
Корректор	<i>Виктория Пиотровская</i>
Оформление обложки	<i>Елены Беляевой</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 27.06.11.

Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 42,57.

Тираж 1500 экз. Заказ №

"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию № 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12.

Оглавление

Введение	1
О чем эта книга?	1
Для кого предназначена эта книга?	2
Типографские соглашения	2
От издательства	3
Благодарности.....	3
Глава 1. Быстрое начало — первые программы на VBA	5
Что такое VBA?	5
Объекты и не только	6
Создание функции пользователя в VBA.....	7
Где пишется код функции пользователя?	8
Структура кода функции пользователя	10
Ваша первая функция пользователя	10
Вычисление стоимости партии продаваемых книг при помощи пользовательской функции	12
Использование ссылок на диапазон в качестве параметров пользовательских функций	13
Об элементах автоматизации Microsoft Office Excel	14
Зачем нужны макросы?	14
Запись макроса и размещение его на панели быстрого доступа	16
Структура кода процедуры	21
Процедура обработки события.....	22
Автоматизация работы рабочего листа при помощи элементов управления.....	22
Использование элемента управления <i>Кнопка</i> на рабочем листе	23
Построение шаблона таблицы.....	26
Управление диаграммой	29
Наши итоги	30
Глава 2. Как организуются программы на языке VBA.....	31
Язык Visual Basic for Applications: как он устроен?.....	31
Быстрый взгляд на процедуры и функции	31
Переменные, константы и типы данных	34
Ссылки на объекты.....	38
Область действия переменных и процедур.....	38

Что нужно знать о массивах?	41
Как используются массивы?.....	41
Поэлементная инициализация массива	43
Инициализация массива при помощи функции <i>Array()</i>	44
Массив и диапазон.....	44
Использование динамических массивов	45
Как проверить, содержит ли переменная типа <i>Variant</i> массив значений?.....	45
Повторная инициализация массива и высвобождение памяти, выделенной под массив	46
Структурированные типы данных: что это такое?.....	47
Строки.....	47
Перечисляемый тип.....	47
Тип данных, определенный пользователем	48
Дополнительные элементы языка VBA: как они помогают при написании программ?	50
Комментарии.....	50
Перенос строки кода	50
Расположение нескольких операторов в одной строке	51
Операции VBA.....	51
Математические операции	51
Операции отношения.....	52
Логические операции	52
Директива <i>Option Compare</i>	52
Приоритеты операций	53
Встроенные функции VBA	54
Встроенные диалоговые окна.....	54
Окно ввода.....	54
Как обработать нажатие кнопки <i>Cancel</i> ?	56
Окно сообщения.....	56
Определение нажатой кнопки в окне ввода.....	59
Управляющие конструкции: формируем логику программы	59
Оператор присваивания	60
Ветвления	61
Циклы	64
Выход из циклов и процедур.....	65
Примеры использования операторов цикла.....	66
Оператор <i>For...Next</i>	66
Оператор <i>For Each</i>	67
Оператор <i>While</i>	68
Оператор <i>Do</i>	69
Альтернативный выход из цикла	70
Создание бесконечного цикла оператором <i>Do</i>	70
Оператор безусловного перехода <i>GoTo</i>	70
Процедуры: знакомимся с деталями.....	71
Создание пользовательских функций.....	72
Список параметров процедуры	74

Организация программы на языке VBA.....	75
Вызов процедуры и передача значений параметров.....	76
Процедура с необязательными параметрами.....	76
Специфицирование значений по умолчанию необязательным параметром.....	77
Использование неопределенного количества параметров.....	78
Использование массива в качестве параметра процедуры.....	78
Передача параметров по ссылке и значению.....	79
Рекурсивные процедуры.....	80
Фракталы.....	81
Создаем классы, объекты и семейства.....	83
Объявление класса.....	83
Создание экземпляра класса.....	84
Инициализация значений полей.....	85
Ключевое слово <i>Me</i>	85
Ключевое слово <i>Nothing</i> и удаление объекта из памяти.....	86
Методы.....	86
Свойства как средство ограничения доступа к полям класса.....	87
Свойства "только для чтения" и "только для записи".....	89
События.....	90
Объект <i>Collection</i>	92
Наши итоги.....	92
Глава 3. Обрабатываем данные при помощи формул и функций рабочего листа.....	93
Немного об адресации ячейки.....	93
Методы объекта <i>Range</i>	95
Активизация и выбор диапазона.....	96
Автоматический подбор размеров диапазона так, чтобы в нем помещались введенные данные.....	96
Заполнение диапазона по одному значению.....	96
Обрамление диапазона границей.....	97
Очистка ячейки.....	97
Копирование, вырезание и удаление данных из диапазона.....	97
Специальная вставка.....	98
Вставка диапазона с транспонированием.....	99
Снятие выделения после специальной вставки.....	99
Добавление ячейки, строки или столбца.....	99
Что дает автозаполнение?.....	99
Заполнение диапазона прогрессией.....	100
Автозаполнение ячеек диапазона элементами последовательности.....	101
Табуляция функции.....	103
Используем автозамену.....	105
Ищем значения.....	106
Поиск значения в диапазоне.....	106
Повторный поиск и поиск всех значений.....	107
Замена значений.....	108

Как отобразить примечания?	108
Проверяем данные	110
Что нужно знать о форматах данных?	110
Форматирование числа на VBA	110
Пользовательский формат	112
Форматирование чисел	117
Форматирование процентов	118
Денежный формат	118
Форматирование даты и времени	118
Условное форматирование	119
Форматирование рабочих листов	120
Автоматическое переоформление таблицы при изменении в ней значений	120
Управление стилем границы диапазона и объекта <i>Border</i>	121
Функции <i>RGB()</i> и <i>QBColor()</i>	122
Объект <i>Characters</i> (как форматировать часть содержимого ячейки)	123
Объект <i>Font</i> (задание шрифта)	124
Объект <i>Interior</i> (заливка диапазона)	125
Отмена заливки диапазона	126
Установка числового формата	126
Задание угла, под которым выводится текст в диапазоне	126
Работаем с формулами	127
Ссылки на ячейки в формулах	127
Ссылка на другие листы рабочей книги или на другие рабочие книги	128
Задание групп строк и столбцов	129
Связь объекта <i>Range</i> и свойства <i>Cells</i> объекта <i>Worksheet</i>	129
Свойства объекта <i>Range</i>	129
Ввод или считывание значения из диапазона	130
Ввод в диапазон массива значений	130
Поиск по шаблону подобных значений в диапазоне	131
Ввод или считывание формулы в ячейку в формате A1	131
Ввод или считывание формулы в ячейку в формате R1C1	132
Ввод или считывание формулы локальной версии в ячейку в формате A1	132
Ввод или считывание формулы локальной версии в ячейку в формате R1C1	132
Ввод формулы массива в диапазон	132
Ввод формулы массива локальной версии в диапазон	132
Ввод формулы массива в диапазон с относительными ссылками на ячейки	132
Как узнать, скрыта ли формула на защищенном листе?	133
Как узнать, имеется ли в ячейке формула?	133
Определение адреса ячейки	133
Может ли ячейка быть отредактирована на рабочем листе?	134
Определения числа областей, из которых состоит данный диапазон	134
Операторы	135
Операции с текстом и датами	135
Операции сравнения и адресные операции	136
Автоматическое вычисление	137
Используем функции	137
Логические функции	138

Встроенные функции VBA	139
Ошибки в формулах и отслеживание зависимостей	139
Примеры использования различных функций в Microsoft Office Excel	141
Подготовка различных ведомостей	142
Ведомость о продаже квартир	142
Ведомость, связанная с переоценкой основных средств производства	143
Отчетная ведомость по работе сети компьютерных клубов	144
Ведомость по расчету заработной платы	145
Использование встроенных функций для решения различных задач	149
Принадлежность точек плоскости	149
Пример решения системы линейных уравнений	151
Пример создания итоговой конструкции по заданному образцу	152
Пример разделения информации, находящейся в одной ячейке	153
Пример создания ведомости для учета проката фильмов	154
Использование функций в программах на языке VBA	155
Получение случайного числа из целочисленного интервала	155
Вывод строки посимвольно в окно <i>Immediate</i>	156
Строка, состоящая из указанного числа пробелов	156
Определение числа секунд, прошедших с полуночи	156
Наши итоги	157
Глава 4. Как создаются пользовательские формы	159
Используем элементы управления на рабочем листе	159
О панели инструментов <i>Элементы управления</i>	160
Как расположить элемент управления на рабочем листе и написать код?	161
Ваш первый проект с элементом управления	163
Общие свойства элементов управления	165
Общие методы элементов управления	166
Общие события элементов управления	167
Кнопка (CommandButton)	168
Кнопочное меню	168
Навигация по книге при помощи гиперссылок	169
Кнопочный сценарий	170
Кнопочный сценарий для ввода формул с кнопками, украшенными рисунками, и пользовательским указателем мыши	171
Интерактивная кнопка и определение среднего объема продаж	174
Обмен значений между двумя выбранными ячейками	175
Переключатель (OptionButton)	175
Переключатели и объемы продаж	175
Флажок (CheckBox) и Выключатель (ToggleButton)	176
Флажок и управление отображением элементов диаграммы	177
Выключатель и отображение примечаний	178
Полоса прокрутки (ScrollBar) и Счетчик (SpinButton)	179
Ввод значений в ячейку и управление цветом	180
Ввод в ячейку с помощью полосы прокрутки и счетчика нецелочисленных значений	181

Список (ListBox)	183
Сценарии со списком	184
Защита ячеек рабочего листа	185
Управление печатью элементов управления	186
Создаем пользовательские формы с помощью VBA	187
Добавление формы в проект	187
Семейство форм	187
Свойства формы	188
Методы формы	190
События формы	190
Отображение и скрытие формы	191
Первый проект с формой	191
Как запустить проект на исполнение?	193
Ключевое слово <i>Me</i>	193
Форма с обновляемым фоновым рисунком	193
Удаление рисунка	195
Форма с мозаичным фоном и установкой свойств на этапе инициализации	195
Закрытие формы при нажатии клавиши <Esc>	196
Подтверждение закрытия окна	197
Задание местоположения формы	197
Модальная форма	198
Использование нескольких форм	198
"Пасхальное яйцо"	199
Элементы управления	200
Размещение элемента управления на форме	201
Label (Надпись)	201
TextBox (Поле)	202
Сложение двух чисел	202
Кнопка с "горячей" клавишей	203
Клавиши <Enter> и <Esc>	204
Суммирование с блокировкой результата для пользователя	204
Как сделать, чтобы при нажатии кнопки она не получала фокус?	204
Перемещение фокуса между полями при нажатии клавиши <Enter>	205
Всплывающая подсказка	205
Поле ввода пароля	206
Многострочное поле ввода	206
Обмен значениями между формами	207
Таймер как пример класса, генерирующего события	208
CheckBox (Флажок) и ToggleButton (Выключатель)	208
Управление видимостью элементов управления	208
Управление доступностью для пользователя элементов управления	209
Frame (Рамка)	209
OptionButton (Переключатель)	209
Переключатель и выбор результирующей операции	209
ScrollBar (Полоса прокрутки) и SpinButton (Счетчик)	211
Синхронизированная работа поля ввода и счетчика	211

<i>ListBox</i> (Список).....	211
Поэлементное заполнение списка.....	212
Заполнение списка из массива и выбор операции.....	212
Заполнение списка из диапазона.....	213
Выбор нескольких элементов из списка.....	214
Согласованная работа двух списков.....	215
Многостолбцовый список.....	216
Заполнение многостолбцового списка из диапазона и нахождение среднего значения выбранных чисел.....	217
Скрытие данных в многостолбцовом списке.....	218
Вывод в многостолбцовом списке выбранного значения при помощи свойств <i>Text</i> и <i>Value</i>	219
Буксировка элементов из одного списка в другой.....	219
<i>ComboBox</i> (Поле со списком).....	220
Поле со списком, ввод данных в алфавитном порядке и объект <i>Collection</i>	221
Добавление и удаление данных в поле со списком.....	221
<i>Image</i> (Рисунок).....	222
Окно <i>О программе</i>	223
Просмотр слайдов.....	223
Модифицированный мастер диаграмм.....	224
Элемент управления <i>RefEdit</i>	226
Определение статистических параметров диапазона.....	226
Решение системы линейных уравнений.....	227
<i>MultiPage</i> (Набор страниц) и <i>TabStrip</i> (Набор вкладок).....	228
Статистика и набор страниц.....	229
Последовательность перехода элементов управления.....	230
Отображение встроенных диалоговых окон.....	230
Открытие документа и метод <i>GetOpenFilename</i>	232
Простейший браузер для графических файлов.....	233
Сохранение документа и метод <i>GetSaveAsFilename</i>	234
Дополнительные элементы управления.....	234
Добавление дополнительного элемента управления.....	235
Удаление дополнительного элемента управления.....	235
Разрабатываем пользовательские приложения.....	235
Заполнение табличного списка данных.....	235
Сервисные возможности для рабочей книги.....	238
Разработка модели склада.....	239
Наши итоги.....	242
Глава 5. Настройка ленты — это так просто!	243
Как настроить панель быстрого доступа?.....	243
Записываем макрос и назначаем его кнопке.....	246
Назначаем кнопкам процедуры VBA.....	250
Очень быстро настраиваем ленту.....	252
Настраиваем ленту с использованием формата Microsoft Office Open XML.....	253
Формат Microsoft Office Open XML в рабочих книгах Microsoft Office Excel 2010.....	253

Настройка ленты прямым редактированием XML-файлов рабочей книги Excel	255
Настройка ленты с использованием XML и VBA	258
Пример создания динамического меню ленты	260
Дополнительные замечания по настройке ленты	263
Создаем панели инструментов из ранних версий MS Excel	265
Конструируем контекстное меню	266
Наши итоги	267
Глава 6. Строим диаграммы	269
Что нужно знать о диаграммах?	269
Создаем шаблон отчета с диаграммой	272
Что представляют собой семейства <i>ChartObjects</i> , <i>Charts</i> и объекты <i>ChartObject</i> , <i>Chart</i> ?	274
Добавление нового элемента в семейства <i>ChartObjects</i> и <i>Charts</i>	275
Свойства объекта <i>Chart</i>	276
Методы объекта <i>Chart</i>	278
События объекта <i>Chart</i>	279
Строим диаграмму с помощью VBA	280
Изменяем диапазон, по которому строится диаграмма	285
Изменяем тип диаграммы	286
Автоматически перестраиваем диаграмму при изменении диапазона данных	288
Последовательно отображаем ряды данных на диаграмме	289
Создаем проект с линией тренда	290
Строим поверхности и управляем ориентацией	293
Устанавливаем защиту на вложенную в рабочий лист диаграмму	296
Защита диаграммы, расположенной на отдельном листе	297
Немного о событиях и диаграммах	298
Привязка события к вложенным в рабочий лист диаграммам	300
Изменение типа диаграммы при помощи контекстного меню	301
Наши итоги	302
Глава 7. Обрабатываем списки в Microsoft Excel	303
Что нужно знать о списке?	303
Сортируем данные	304
Используем VBA для сортировки данных	307
Сортировка данных списка по трем полям	308
Сортировка данных на защищенном листе	310
Сортировка данных в выделенном диапазоне	310
Сортировка всех столбцов списка	311
Фильтруем данные	312
Как найти данные с использованием автофильтра?	312
Как программировать автофильтрацию?	314
Пример приложения, фильтрующего данные	316
Как использовать расширенный фильтр?	317
Немного о методе <i>AdvancedFilter</i>	321
Наши итоги	323

Глава 8. Обрабатываем данные средствами Microsoft Office Excel	325
Подводим промежуточные итоги	325
Простые промежуточные итоги	326
Вложенные промежуточные итоги	328
Метод <i>Subtotal</i>	331
Удаление промежуточных итогов	331
Обобщаем однородные данные с помощью консолидации	332
Консолидация при помощи трехмерных формул на рабочем листе	332
Консолидация при помощи трехмерных формул в коде	333
Консолидация данных по положению и категориям	333
Методы и свойства, используемые при программировании консолидирующей таблицы	336
Пример приложения, консолидирующего данные	337
Структурируем рабочие листы	338
Структура и объект <i>Outline</i>	340
Отображение указанного числа уровней структуры	341
Удаление структуры	341
Отображение значков структуры	341
Автоматическое создание структуры	341
Пример приложения, подводящего промежуточные итоги и управляющего структурой	342
Используем сценарии	343
Расчет внутренней скорости оборота инвестиций	344
Объект <i>Scenario</i>	348
Пример приложения по работе со сценариями	349
Создаем сводные таблицы	350
Пример создания сводной таблицы на рабочем листе Excel	352
Объекты, связанные со сводной таблицей	357
Объект <i>PivotTable</i>	358
Объект <i>PivotCache</i>	359
Объект <i>PivotField</i>	360
Пример построения сводной таблицы средствами VBA	360
Наши итоги	362
Глава 9. Используем поиск решения и подбор параметра	363
Поиск решения: как это работает?	364
Постановка задачи оптимизации в общем случае	364
Настройка <i>Поиск решения</i>	365
Рекомендации по решению задач оптимизации с помощью надстройки <i>Поиск решения</i>	370
Построение математической модели задачи	370
Подготовка рабочего листа MS Excel для решения задачи оптимизации	371
Решение задачи с помощью надстройки <i>Поиск решения</i>	371
Анализ решения задачи оптимизации	372
Решаем задачу линейного программирования	372

Планирование производства материалов	373
Определение состава удобрений	377
Решаем транспортную задачу	381
Пример решения транспортной задачи	382
Что такое дискретное программирование?	386
Решаем задачу нелинейного программирования	389
Какие функции программируют поиск решения?	393
Приложение "Транспортная задача"	395
Решение оптимизационных задач, зависящих от параметра	397
Работаем со средством <i>Подбор параметра</i>	399
Пример определения затрат на проект	399
Нахождение корней уравнения	401
Подбор параметра и решение уравнения с одним неизвестным с использованием VBA	402
Усовершенствование средства <i>Подбор параметра</i>	404
Наши итоги	406
Глава 10. Интеграция Microsoft Office Excel и XML	407
Что необходимо знать о формате XML?	407
Изучаем синтаксис XML	409
Основные компоненты документа XML	409
Структура документа XML	411
Русификация XML	412
Зачем нужны схемы XML?	412
Пространства имен	413
Схема XML, расположенная в документе	414
Внешняя схема XML	415
Экспортируем и импортируем данные XML в рабочую книгу Excel	416
Как выполнить импорт данных XML в Excel?	416
Импорт данных из XML-файла в случае отсутствия схемы XML	417
Создание карты XML и импорт данных из файла XML	418
Как выполнить экспорт данных из Excel в документ XML?	421
Как выполнить импорт и экспорт с помощью VBA?	422
Наши итоги	423
Глава 11. MS Excel и Интернет — рядом!	425
Что нужно знать об Интернете?	425
Работаем с гиперссылками в Microsoft Office Excel	429
Как добавить гиперссылку на документы MS Office?	429
Как задать гиперссылку формулой рабочего листа?	430
Что такое условная гиперссылка?	431
Объект <i>Hyperlink</i> и семейство <i>Hyperlinks</i>	432
Переход по гиперссылке из списка	434
Работаем с веб-страницами	436
Веб-запрос и получение данных с веб-страницы	436

Создаем скрипты	439
Как создать скрипты для веба на стороне клиента?	440
Как создать скрипты для веба на стороне сервера?	441
Как передать данные от клиента к серверу?	445
Наши итоги	447
Глава 12. Об интеграции приложений	449
Что такое технология ActiveX?	449
Связываем и внедряем объекты	450
Связь данных	450
Внедрение данных из других приложений	453
Немного примеров	454
Управляем объектами с помощью технологии Automation	458
Программные идентификаторы приложений-серверов Automation	459
Функции доступа к объектам Automation	459
Позднее и раннее связывание	460
Организуем совместную работу Microsoft Excel и Microsoft Word	461
Создание нового документа Microsoft Word функцией <i>CreateObject()</i>	461
Открытие документа Microsoft Word функцией <i>GetObject()</i>	462
Отправка отчета из MS Excel в MS Word	462
Используем Access в качестве сервера автоматизации	464
Отправляем сообщения по электронной почте	465
Создаем презентацию в MS PowerPoint	467
Наши итоги	468
ПРИЛОЖЕНИЯ	469
Приложение 1. Краткая справка по Visual Basic for Applications	471
Основные понятия объектной модели	471
Объектная модель Visual Basic для приложений	472
Объектная модель Microsoft Office 2010	473
Объектная модель Microsoft Office Excel 2010	474
Полная и неявная ссылка на объект	478
Объект <i>Application</i> и его некоторые свойства	478
Ссылка на активную рабочую книгу, лист, ячейку, диаграмму и принтер	478
Инсталлированные надстройки	479
Диапазон ячеек	482
Столбцы и строки рабочего листа	483
Установка заголовка окна MS Excel	483
Семейство встроенных диалоговых окон	484
Отображение строки формул, полосы прокрутки и строки состояния	485
Полноэкранное отображение рабочего листа	485
Установка высоты и ширины окна приложения	485
Семейство всех имен активной рабочей книги	485
Ссылка на выбранный объект	486

Методы объекта <i>Application</i>	486
События объекта <i>Application</i>	487
Наши итоги	489
Приложение 2. Интегрированная среда разработки Microsoft Visual Basic	490
Где набирается код VBA?	490
Структура редактора VBA.....	491
Окно <i>Project - VBAProject</i>	491
Копирование модулей и форм из одного проекта в другой	492
Окно редактирования кода	492
Интеллектуальные возможности редактора кода.....	493
Окно <i>UserForm</i> (Редактирование форм)	495
Окно <i>Properties</i> (Свойства).....	497
Окно <i>Object Browser</i> (Просмотр объектов).....	498
Наши итоги	499
Приложение 3. Отладка приложений.....	500
Ошибки компиляции.....	500
Ошибки выполнения	501
Логические ошибки.....	503
Инструкция <i>Option Explicit</i>	503
Пошаговое выполнение программ.....	504
Точка прерывания	505
Вывод значений свойств и переменных.....	506
Окно <i>Watches</i>	506
Окно <i>Locals</i>	506
Окно <i>Immediate</i>	507
Программный способ вывода значений в окно <i>Immediate</i>	507
Наши итоги	507
Приложение 4. Описание компакт-диска.....	508
Рекомендуемая литература	509
Предметный указатель.....	511

Глава 1

Быстрое начало — первые программы на VBA

Что такое VBA?

Visual Basic for Applications (VBA, Visual Basic для приложений) — это объектно-ориентированный язык программирования, который специально разработан для приложений Microsoft Office. Отличительной особенностью VBA является использование наряду с обычными переменными и константами также и имеющихся объектов приложений Microsoft Office, например, в Microsoft Office Excel 2010 это могут быть рабочие книги, рабочие листы, диапазоны ячеек, диаграммы и т. д. С помощью VBA можно разрабатывать приложения, которые включают различные компоненты нескольких приложений Microsoft Office и способствуют тем самым интеграции и совместному использованию данных.

VBA — достаточно легкий язык программирования. Он прост в освоении и позволяет быстро получать ощутимые результаты — конструировать профессиональные приложения, решающие практически все задачи, встречающиеся в среде Windows, а также удобен при создании клиент-серверных приложений. При этом проектирование многих приложений с использованием VBA проще и быстрее, чем с применением других языков программирования.

VBA использует технологию визуального программирования, т. е. конструирование рабочей поверхности приложения и элементов его управления непосредственно на экране, а также запись всей программы или ее частей при помощи макрорекордера.

VBA позволяет создавать универсальные, автоматизированные приложения на платформе MS Excel. Кроме того, используя технологию ActiveX, VBA позволяет как внедрять в разрабатываемое приложение объекты других приложений, так и, не выходя из созданного приложения, управлять другими Windows-приложениями. VBA позволяет создавать клиент-серверные приложения, связывая ваш компьютер со всем остальным миром.

Для того чтобы писать программы на языке VBA, необходимо четко представлять себе технологию объектно-ориентированного программирования (ООП). ООП можно описать как совокупность принципов, технологии и инструментальных средств для создания программных систем, основу которых составляет архитектура взаимодействия объектов. Объектная модель Microsoft Office и, в частности, объектная модель Microsoft Office Excel 2010 содержит множество различных объектов, которые образуют достаточно сложную иерархию. В свою очередь, каждый

объект обладает набором функциональных возможностей, а также способами воздействия на его состояние.

Итак, прежде чем приступить к созданию первых программ на языке VBA, познакомимся вкратце с такими важными понятиями ООП, как объект, свойство, метод, событие, класс и семейство объектов.

ПРИМЕЧАНИЕ

Примеры, относящиеся к данной главе, вы можете найти в папке Glava_1 на прилагаемом к книге компакт-диске.

Объекты и не только

Объект (object) является основным понятием (основной парадигмой) ООП и представляет собой изменяемый элемент, содержащий данные вместе с кодом, предназначенным для их обработки. Любой объект в ООП обладает определенными *свойствами* (properties), которые описывают данный объект или его состояние, и *методами* (methods), отвечающими за действия, которые можно выполнить над объектом. На любой объект можно воздействовать одним из двух способов, изменяя его состояние: во-первых, можно изменить одно из свойств данного объекта и, во-вторых, можно выполнить некоторые действия, применив один из методов рассматриваемого объекта.

Объекты, которые имеют одинаковые свойства и методы, объединяются в ООП в абстрактное понятие — *класс* (class). Важной особенностью классов является возможность их организации в виде некоторой древовидной *иерархической* структуры. Верхний уровень данной иерархии занимает класс (родительский класс или предок), который охватывает наиболее общие свойства и методы, характерные для всех его подчиненных объектов. На низшем же уровне располагаются лишь те объекты (потомки), дальнейшая конкретизация которых невозможна.

Введем еще одно понятие, связанное с объектами, которое встретится вам при написании программ на VBA. Довольно часто можно столкнуться с набором однотипных объектов, которые, в свою очередь, образует объект *семейство*. Следует отметить, что если любой объект является реализацией конкретного класса, то семейство представляет собой коллекцию уже имеющихся похожих объектов.

Естественно, что, знакомясь с общей методологией ООП, нельзя обойти вниманием и такие его основные принципы, как наследование, инкапсуляция и полиморфизм.

Наследование (inheritance) связано непосредственно с иерархией классов и определяет возможность обладания определенными свойствами и методами. Так, если некоторый общий (родительский) класс обладает определенным набором свойств и методов, то класс, который связан с ним и находится на уровне ниже (потомок), обязан включать эти же свойства и методы, а также дополнительные, которые будут характеризовать уникальность данного потомка.

Скрытие некоторых деталей реализации объектов по отношению к внешним объектам или пользователям называется *инкапсуляцией* (encapsulation). Как правило, в действительности не столь важно, каким образом реализован, например, тот или иной метод класса, к которому обращается пользователь. Идея инкапсуляции взята от деления модулей в некоторых языках программирования на две части: интерфейс и реализацию, и отражает наше представление об окружающем мире. Так, в ин-

терфейсной части дается вся информация, которая необходима для взаимодействия с другими объектами. Вся остальная информация, которая не относится к процессу взаимодействия объектов, скрывается в части, относящейся к реализации объекта.

Полиморфизм (polymorphism) предполагает, что действия, которые выполняются одноименными методами, но для различных классов объектов, могут существенно отличаться.

Если же мы обратимся к VBA, то убедимся, что объектная модель Microsoft Office содержит много различных объектов и образует сложную иерархию. Например, все визуальные элементы, такие как рабочий лист (Worksheet), диапазон (Range), диаграмма (Chart), форма (UserForm), являются в Microsoft Office Excel 2010 объектами. Более того, практически, все элементы, которые можно увидеть в этом приложении, включая само окно рабочей книги, являются объектами. Исключение составляют, например, кнопки **Свернуть** (Minimize) и две взаимозаменяемые кнопки **Свернуть в окно** (Restore) и **Развернуть** (Maximize), находящиеся в заголовке окна приложения Microsoft Excel.

Многие однотипные объекты объединяются в VBA в семейства (объект Collection). Например, объект Workbooks содержит все открытые объекты Workbook (рабочая книга). Каждый элемент семейства нумеруется и может быть идентифицирован либо по номеру, либо по имени. Например, Worksheets(1) обозначает первый рабочий лист активной книги, а Worksheets("Лист1") — рабочий лист с именем **Лист1**.

Если в программе на VBA нам необходимо указать ссылку на объект, то для конкретного объекта следует уточнить весь его иерархический путь. Например, чтобы присвоить значение 10 первой ячейке второго листа третьей рабочей книги Excel, следует записать такой оператор VBA:

```
Application.Workbooks(3).Worksheets(2).Range("A1")=1
```

Здесь используется не только простой объект "Диапазон" (Range), объекты, которые являются коллекциями (Collection) — Workbooks (семейство "Рабочие книги") и Worksheets (семейство "Рабочие листы").

В дальнейшем, при детальном изучении VBA, мы убедимся, что он не является объектно-ориентированным языком программирования в строгом понимании этого слова. Однако объектный подход играет в VBA основную роль. В последующих главах книги даются более корректные понятия для основных объектов VBA и особенности их использования.

Создание функции пользователя в VBA

Если вы уже использовали Microsoft Excel для каких-либо расчетов, то по достоинству могли оценить тот набор встроенных функций, которые предоставляет библиотека данного приложения. С другой стороны, возможно, какие-то расчеты вы хотели бы ускорить, однако нужной функции найти не удалось. Сейчас мы непосредственно займемся созданием собственных функций и убедимся, что ничего сложного в этом нет.

Итак, на нескольких примерах, которые приводятся далее, мы рассмотрим, как строятся функции пользователя. Начнем с простейшего примера — функции, вычисляющей значение по одной формуле. Затем обсудим более сложный пример, а именно расчет стоимости партии книг, когда значение функции зависит от условия.

Где пишется код функции пользователя?

Для того чтобы написать пользовательскую функцию, необходимо перейти в редактор VBA. Для начала убедитесь в том, чтобы в Microsoft Office Excel 2010 на ленте отображалась вкладка **Разработчик** (рис. 1.1).

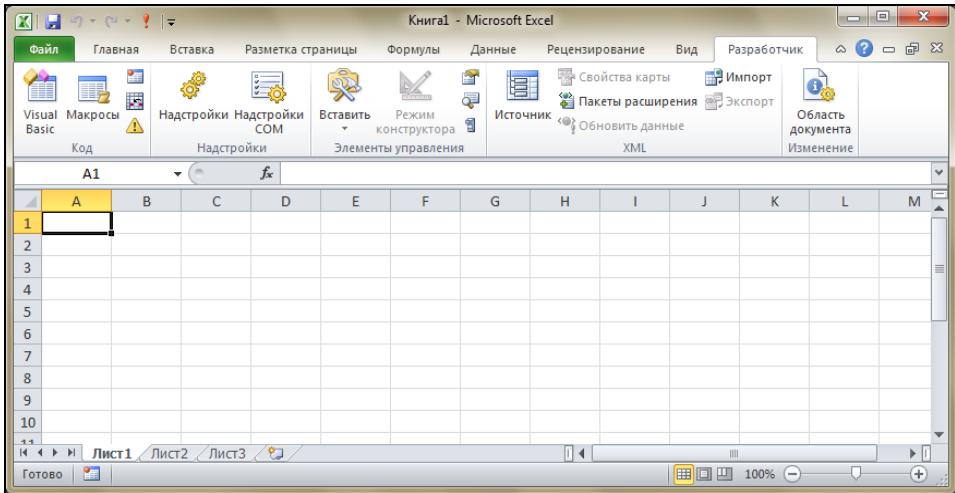


Рис. 1.1. Вкладка **Разработчик** на ленте

Если же она отсутствует, выполните следующие действия:

1. Перейдите на вкладку **Файл** ленты и нажмите кнопку **Параметры**.
2. В открывшемся окне **Параметры Excel** выберите слева категорию **Настройка ленты**, а справа в группе **Настройка ленты** выберите из раскрывающегося списка **Основные вкладки**.
3. Установите флажок **Разработчик** (рис. 1.2) и нажмите кнопку **ОК**.

Итак, перейдите на вкладке **Разработчик** к группе **Код** и нажмите кнопку **Visual Basic**: у вас открылась интегрированная среда разработки приложений IDE редактора Visual Basic (рис. 1.3).

ПРИМЕЧАНИЕ

Для быстрого запуска редактора VBA достаточно всего лишь нажать комбинацию клавиш **<Alt>+<F11>**.

Среда разработки имеет стандартный интерфейс, характерный для Windows-приложений: строка заголовка, линейка меню, панель инструментов (в данном случае **Standard**), а также два окна: **Project - VBAProject** и **Properties**.

Отметим, что в окне **Project - VBAProject** перечисляются все модули и формы, входящие в создаваемый проект. Модуль представляет собой окно **Module**, в котором основную часть занимает рабочая область — лист (не путать с рабочим листом), в котором набирается код. Для открытия модуля в окне **Project - VBAProject** достаточно выполнить двойной щелчок на соответствующем значке. Для активного модуля его значок выделяется в окне **Project - VBAProject** серым цветом.

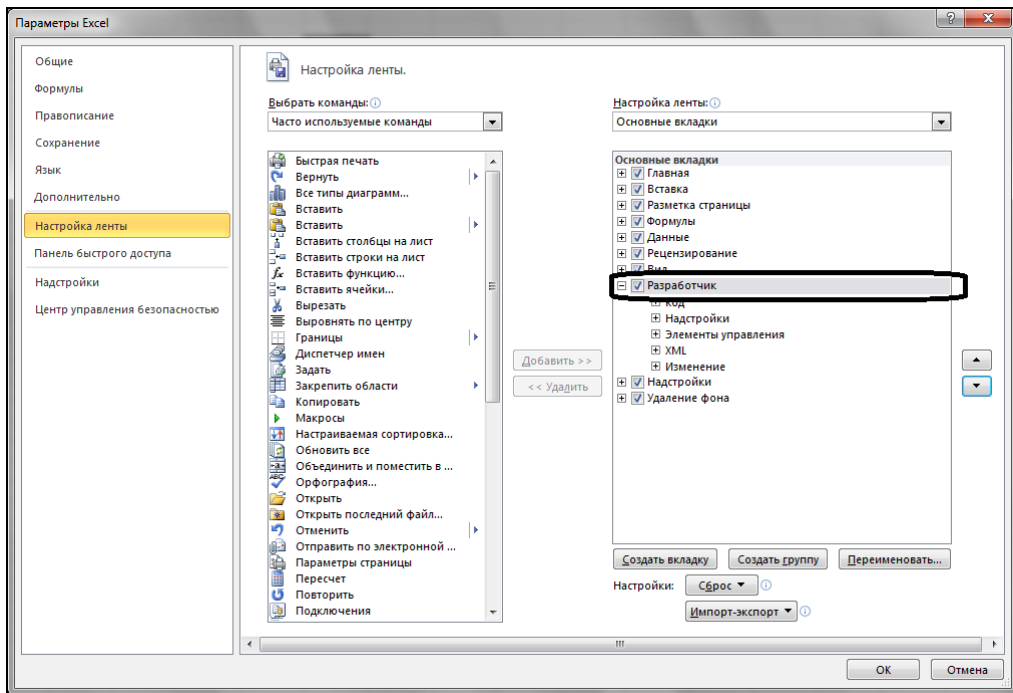


Рис. 1.2. Окно Параметры Excel

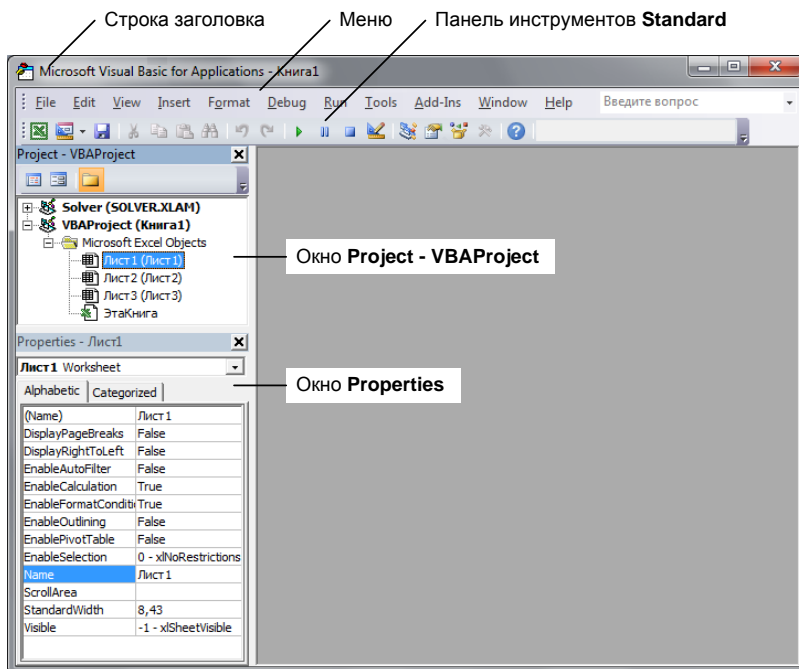


Рис. 1.3. Редактор Visual Basic

В VBA у каждого рабочего листа есть собственный модуль, и у рабочей книги также имеется свой. Более того, если в проекте создаются пользовательские формы, то каждая из них имеет по модулю. Вы можете добавлять в разрабатываемый проект модули классов для описания создаваемых пользовательских классов. Однако для создания пользовательской функции нам потребуется стандартный модуль, добавление которого в проект осуществляется командой **Insert | Module**.

Структура кода функции пользователя

В общем случае, функция пользователя имеет следующий вид:

```
Function ИмяФункции(СписокПараметров)
```

```
    Инструкции
```

```
End Function
```

- *СписокПараметров* — это список параметров, от которых зависит функция. Разделителем в списке параметров является запятая.
- *Инструкции* — это последовательность инструкций, выполняемых при нахождении значения функции. В совокупности они образуют так называемое *тело функции*. Важной особенностью функции пользователя является то, что носителем возвращаемого ею значения является *ИмяФункции*. Поэтому в теле функции должен присутствовать, по крайней мере, один оператор, который присваивает имени функции значение какого-либо выражения.

ПРИМЕЧАНИЕ

Допустим досрочный выход из функции по инструкции `Exit Function`. В теле функции может располагаться несколько инструкций `Exit Function`.

Ваша первая функция пользователя

Теперь можно непосредственно перейти к написанию функции пользователя. Давайте для начала напишем код для вычисления простой функции, например,

$$F(x) = x^3 + x^2.$$

Для реализации данной задачи вам необходимо выполнить следующие действия.

1. В окне редактора VBA добавьте лист стандартного модуля (если он вами еще не создан), выполнив команду **Insert | Module**.
2. В окне созданного модуля (рис. 1.4) наберите код из листинга 1.1 (см. также файл *1-Функции пользователя.xlsm* на компакт-диске).

Листинг 1.1. Пользовательская функция

```
Function F(x As Double) As Double
    F = x ^ 3 + x ^ 2
End Function
```

Следует отметить, что в VBA имеется универсальный тип данных `Variant`, который подразумевается по умолчанию, если явно не был объявлен тип переменной или функции. Поэтому ту же самую функцию можно было бы закодировать следующим образом (листинг 1.2).

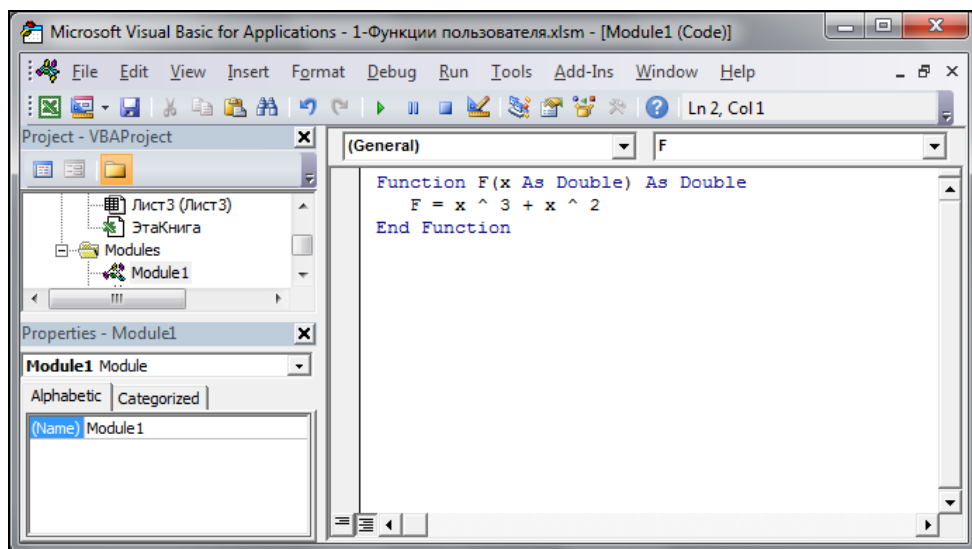


Рис. 1.4. Код пользовательской функции в модуле

Листинг 1.2. Пользовательская функция с использованием типа Variant

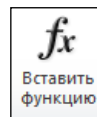
```
Function F(x)
    F = x ^ 3 + x ^ 2
End Function
```

ПРИМЕЧАНИЕ

Еще раз обратите внимание на то, что код пользовательской функции вводится в стандартном модуле, который добавляется в проект командой **Insert | Module**. В проекте много модулей, случайно не перепутайте. Активный модуль в окне **Project - VBAProject** выделяется серым цветом.

Итак, пользовательская функция нами создана. По умолчанию она попадает в раздел **Определенные пользователем** списка **Категория** окна **Мастер функций**. Найдем, например, значение этой функции при $x = 4,7$. Для этого:

1. Перейдите к окну рабочей книги Microsoft Office Excel 2010.
2. Введите число 4,7 в ячейку **A1** рабочего листа (выбрав, например, **Лист1**).
3. Перейдите к ячейке **B1**, в которой найдем значение функции.
4. Перейдите на вкладку **Формулы** ленты и в группе **Библиотека функций** щелкните по кнопке **Вставить функцию**.
5. В первом окне мастера функций укажите из списка категорию **Определенные пользователем** и выберите функцию **F**. Нажмите кнопку **ОК**.
6. Во втором окне мастера функций в поле **X** введите ссылку на ячейку **A1** (или щелкните по соответствующей ячейке рабочего листа мышью) и нажмите кнопку **ОК** (рис. 1.5). Значение функции найдено.



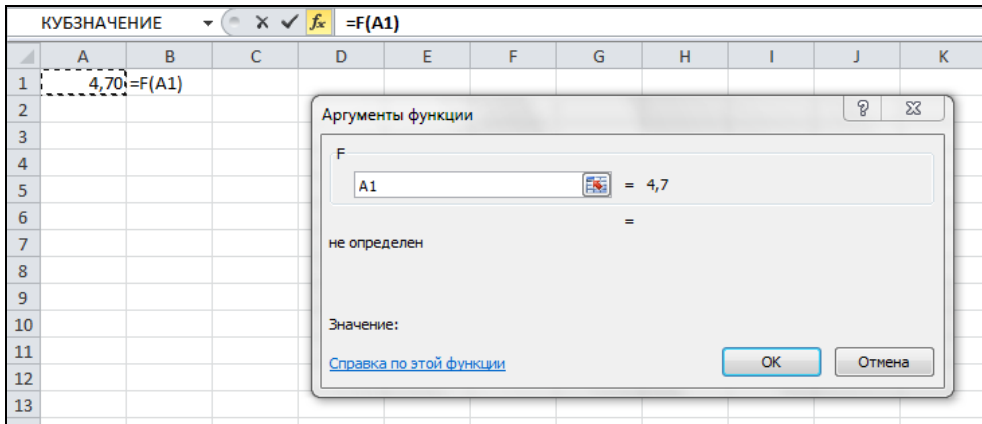


Рис. 1.5. Вычисление значения пользовательской функции при помощи мастера функций

Вычисление стоимости партии продаваемых книг при помощи пользовательской функции

Рассмотрим более сложный пример построения функции пользователя. Представим себе, что вы являетесь менеджером по оптовой продаже книг в издательстве. Для привлечения покупателей в вашем издательстве введена прогрессивная шкала цен. Так, если продается от 100 до 200 экземпляров книги, то скидка от ее отпускной цены составляет 7%, если продается от 201 до 300 экземпляров, то скидка составляет 10%, а если свыше 300 экземпляров, то — 15%. Кроме того, для постоянных клиентов предусмотрена дополнительная скидка в размере 5%. Создадим функцию пользователя с именем *Стоимость* для расчета стоимости партии книг. Параметры этой функции назовем *ЦенаОднойКниги*, *Количество* и *Скидка*. Для параметра *Скидка* предусмотрим только два допустимых значения: 1 — для постоянных клиентов и 0 — для всех остальных. Определим пользовательскую функцию *Стоимость* следующим кодом (листинг 1.3, а также файл *1-Функции пользователя.xlsm* на компакт-диске).

Листинг 1.3. Расчет стоимости партии книг

```
Function Стоимость(ЦенаОднойКниги, Количество, Скидка)
    If Количество < 100 Then
        СтоимостьБезСкидки = ЦенаОднойКниги * Количество
    ElseIf Количество <= 200 Then
        СтоимостьБезСкидки = ЦенаОднойКниги * Количество * 0.93
    ElseIf Количество <= 300 Then
        СтоимостьБезСкидки = ЦенаОднойКниги * Количество * 0.9
    Else
        СтоимостьБезСкидки = ЦенаОднойКниги * Количество * 0.85
    End If

    If Скидка = 0 Then
```

```
    Стоимость = СтоимостьБезСкидки  
Else  
    Стоимость = СтоимостьБезСкидки * 0.95  
End If  
End Function
```

Итак, функция пользователя `Стоимость` создана. Благодаря тому, что в VBA допустимо применение русскоязычных имен, текст написанной программы очевиден для понимания. Кроме того, это обеспечивает и простоту использования диалогового окна мастера функций для данной функции (рис. 1.6).

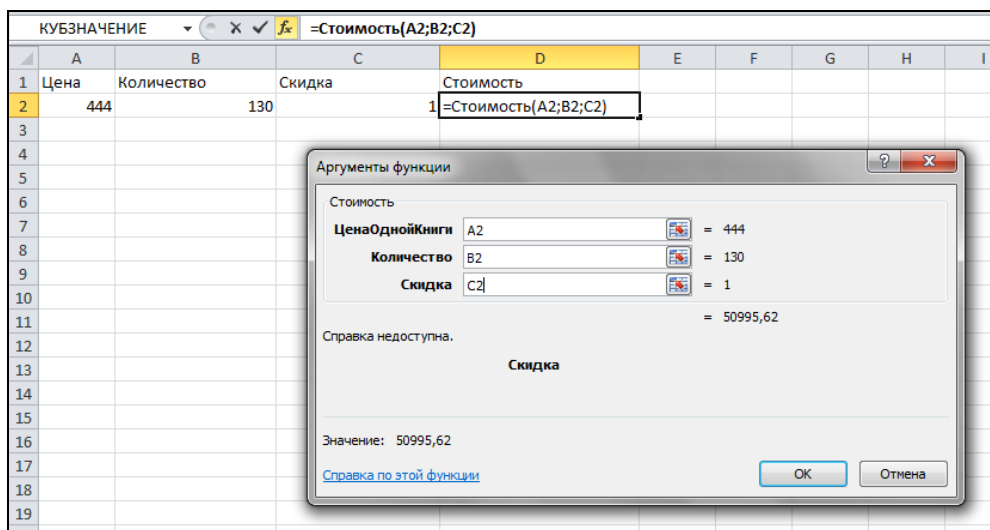


Рис. 1.6. Расчет стоимости партии книг

Названия всех параметров функции `Стоимость` в окне мастера функций выводятся также на русском языке, что позволяет ее применять любому пользователю, даже не владеющему VBA. Следует заметить, что для удобства использования созданной функции `Стоимость` рекомендуется предварительно на рабочем листе задать значения для входных параметров функции, т. е. для `ЦенаОднойКниги`, `Количество` и `Скидка`. Однако это не является обязательным условием: задать необходимые значения вы можете также и сразу в окне мастера функций.

Использование ссылок на диапазон в качестве параметров пользовательских функций

Предыдущий пример о вычислении стоимости партии книг выглядит уже достаточно убедительно. VBA действительно может облегчить жизнь пользователя. Возникает только один вопрос по двум приведенным примерам. В созданных в них пользовательских функциях в качестве значений параметров были только ссылки на ячейки. Можно ли построить пользовательскую функцию, у которой в качестве

значений параметров могут быть ссылки на диапазоны ячеек? Код из листинга 1.4 демонстрирует возможность использования ссылок на диапазон значений и решает задачу суммирования значений для указанного диапазона ячеек (см. также файл *1-Функции пользователя.xlsm* на компакт-диске).

Листинг 1.4. Нахождение суммы

```
Function MySum(ByVal rng As Range) As Double
    Dim c As Range
    Dim s As Double
    s = 0
    For Each c In rng.Cells
        s = s + c.Value
    Next
    MySum = s
End Function
```

Об элементах автоматизации Microsoft Office Excel

Зачем нужны макросы?

А сейчас мы познакомимся с вами с основами автоматизации деятельности, которая не возможна без использования макросов. *Макрос* — это программа, состоящая из списка команд, которые должны быть выполнены приложением. Макрос служит для объединения нескольких различных действий в одну процедуру. Такой список команд состоит, в основном, из *макрооператоров*, тесно связанных с командами приложений из Microsoft Office. Большая часть макрооператоров соответствует командам меню или параметрам, которые задаются в диалоговых окнах.

Можно выделить три основные разновидности макросов:

- ❑ *командные* — наиболее распространенные макросы, которые обычно состоят из операторов, эквивалентных тем или иным командам меню или параметрам диалоговых окон. Основным предназначением таких макросов является выполнение действий, аналогичных командам меню, т. е. изменение окружения и основных объектов приложения. Например, изменение рабочего листа или рабочего пространства Microsoft Excel, сохранение или вывод на печать и т. п. Таким образом, в результате выполнения макроса вносятся изменения либо в обрабатываемый документ, либо в общую среду приложения;
- ❑ *пользовательские функции* — работают аналогично имеющимся встроенным функциям Microsoft Excel. Отличие этих функций от командных макросов состоит в том, что они используют значения передаваемых им аргументов, производят некоторые вычисления и возвращают результат в точку вызова, но не изменяют среду приложения;
- ❑ *макрофункции* — представляют собой сочетание командных макросов и пользовательских функций. Наряду с тем, что они, подобно пользовательским функциям,

могут использовать аргументы и возвращать результат, макрофункции, как и командные макросы, способны еще и изменять среду приложения. Чаще всего макрофункции вызываются из других макросов и активно используются для модульного программирования. Если необходимо в различных макросах выполнить ряд одинаковых действий, то эти действия обычно выделяются в отдельную макрофункцию (подпрограмму).

Как правило, макросы используются для быстрого получения чернового варианта кода. Следует помнить о последовательности действий, связанной с разработкой макросов:

1. *Логическая разработка процедуры.* Прежде всего, вам необходимо точно определить, что следует получить в результате выполнения макроса и какова логическая последовательность действий для получения данного результата.
2. *Подготовка документа.* Произведите предварительные действия, которые не нужно включать в процедуру (например, создание нового рабочего листа или перемещение в конкретную часть рабочего листа и т. д.).
3. *Запись макроса с помощью макрорекордера.* Макрорекордер представляет собой транслятор, создающий программу (макрос) на языке VBA, которая является результатом перевода на языке VBA действий пользователя с момента запуска макрорекордера до окончания записи макроса. Для записи макроса с помощью макрорекордера:
 - перейдите на вкладку **Разработчик** ленты и в группе **Код** щелкните по кнопке **Запись макроса**;
 - в открывшемся диалоговом окне **Запись макроса** установите параметры записываемой процедуры (ее имя, описание, сочетание клавиш для выполнения записанной процедуры, указание, для каких документов доступен макрос) и войдите в режим записи макроса — на экране на вкладке **Разработчик** ленты в группе **Код** кнопка **Запись макроса** изменится на кнопку **Остановить запись**; кроме того, кнопка **Пауза** также станет активной (если вы на время захотите приостановить запись макроса и выполнить другие действия с документом);
 - последовательно выполните все необходимые действия с документом и его содержимым, предусмотренные на первом этапе;
 - остановите запись (кнопка **Остановить запись** в группе **Код** на вкладке **Разработчик**).
4. *Просмотр и редактирование созданной процедуры:*
 - нажмите кнопку **Макросы** в группе **Код** на вкладке **Разработчик**, а затем в открывшемся диалоговом окне **Макрос** выберите в списке имя нужного макроса и нажмите кнопку **Изменить**, далее откроется главное окно редактора Microsoft Visual Basic и окно **Модуль (Module)** с текстом выбранного макроса;
 - внесите в текст макроса необходимые изменения и закройте окно редактора.
5. *Выполнение макроса.* Нажмите кнопку **Макросы** в группе **Код** на вкладке **Разработчик**, затем в открывшемся диалоговом окне **Макрос** в списке выберите имя макроса и нажмите кнопку **Выполнить**.

ПРИМЕЧАНИЕ

Вы можете назначить записанной процедуре кнопку и расположить ее на **Панели быстрого доступа** для упрощения вызова макроса.