В. В. Ступин

ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ: РАЗРАБОТКА ПРИЛОЖЕНИЙ В MS EXCEL СРЕДСТВАМИ VBA

Учебное пособие

Министерство науки и высшего образования Российской Федерации
Байкальский государственный университет

В. В. Ступин

ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ: РАЗРАБОТКА ПРИЛОЖЕНИЙ В MS EXCEL СРЕДСТВАМИ VBA

Учебное пособие

Иркутск
Издательство БГУ
2019

УДК 004.434
ББК 32.972.13
С88

Печатается по решению редакционно-издательского совета
Байкальского государственного университета

Рецензенты канд. физ.-мат. наук, доц. В. В. Братищенко
канд. техн. наук, доц. Т. И. Ведерникова

Ступин В. В.
С88 Информационные системы и технологии: разработка приложений в MS Excel средствами VBA [Электронный ресурс] : учеб. пособие / В. В. Ступин. — Иркутск : Изд-во БГУ, 2019. — 109 с. — Режим доступа: http://lib-catalog.bgu.ru.

Учебное пособие необходимо для изучения одного из разделов курса «Информационные системы и технологии» с целью освоения основных принципов и навыков создания на языке VBA приложений, автоматизирующих обработку табличных данных.

Для студентов, обучающихся по направлениям, связанным с информационными технологиями.

УДК 004.434
ББК 32.972.13

ОГЛАВЛЕНИЕ

Введение 6
1. Редактор VBA 8
1.1. Отображение вкладки «Разработчик» 8
1.2. Запуск редактора 9
1.3. Элементы интерфейса редактора 9
1.4. Окна Visual Basic Editor 11
1.4.1. Окно Project Explorer 11
1.4.2. Окно Properties Window 12
1.4.3. Окно Object Browser 12
1.4.4. Окно Code 13
1.4.5. Окна UserForm и Toolbox 14
1.4.6. Окно Locals 15
1.4.7. Окно Watches 16
1.4.8. Окно Immediate 17
1.4.9. Окна предоставления справочной информации 17
1.4.10. Окно Options 17
2. Объектная структура языка VBA 19
2.1. Основные определения 19
2.2. Иерархия объектов 20
2.3. Оперирование свойствами объектов 21
2.4. Использование методов объектов 22
2.5. События 23
2.6. Основные объекты Excel 23
2.6.1. Объект Application 24
2.6.2. Объект Workbook коллекции Workbooks 28
2.6.3. Объект Worksheet коллекции Worksheets 31
2.6.4. Объект Range 32
3. Основы программирования на VBA 37
3.1. Оформление кода 37
3.1.1. Комментарии 37
3.1.2. Отступы в коде 37
3.1.3. Переносы строк 37
3.1.4. Запись нескольких операторов в одну строку 38
3.2. Переменные и типы данных 38
3.3. Объявление переменных и констант 39
3.4. Область действия переменных и констант 41
3.4.1. Локальные переменные 41
3.4.2. Переменные уровня модуля 41
3.4.3. Переменные уровня проекта 41
3.4.4. Переменные Static 41
3.4.5. Область действия константы 42

3.5. Выражения и операции ... 42
3.5.1. Арифметические операции ... 42
3.5.2. Операции сравнения ... 43
3.5.3. Логические операции ... 43
3.5.4. Строковая операция ... 44
3.5.5. Приоритеты выполнения операций ... 44
3.5.6. Математические функции ... 44
3.5.7. Функции проверки типов ... 45
3.5.8. Функции обработки строк ... 45
3.5.9. Функции преобразования типов ... 46
3.5.10. Функции даты и времени ... 47
3.5.11. Функции выбора ... 48
3.6. Операторы и управляющие конструкции ... 49
3.6.1. Операторы присваивания ... 49
3.6.2. Оператор With ... 50
3.6.3. Оператор безусловного перехода ... 51
3.6.4. Операторы условного перехода ... 51
3.6.5. Линейная форма записи оператора If ... 51
3.6.6. Блочная форма записи оператора If ... 51
3.6.7. Оператор выбора Select Case ... 53
3.6.8. Операторы циклов ... 54
3.6.9. Оператор For…Next ... 54
3.6.10. Оператор For Each...Next ... 55
3.6.11. Оператор Do … Loop ... 55
3.6.12. Массивы ... 56
3.6.13. Статические массивы ... 57
3.6.14. Динамические массивы ... 58
3.7. Процедуры и функции ... 59
3.7.1. Описание и вызов процедур и функций ... 59
3.7.2. Вызов процедур и функций ... 61
3.7.3. Область действия процедур и функций ... 62
3.7.4. Создание пользовательских функций ... 63

4. Инструменты организации интерфейса приложений ... 68
4.1. Встроенные диалоговые окна ... 68
4.1.1. Окно сообщения ... 68
4.1.2. Окно ввода данных ... 70
4.2. Пользовательские формы ... 71
4.2.1. Вставка новой формы в проект ... 71
4.2.2. Добавление элементов управления в форму ... 73
4.2.3. Использование элементов управления формы ... 74

5. Разработка приложений ... 83
5.1. Приложение «Регистрация и оплата» ... 83
5.1.1. Задание ... 83
5.1.2. Краткое описание решения задачи ... 84

5.1.3. Программная реализация 87
5.2. Приложение «Оплата проживания в гостинице» 91
5.2.1. Задание 91
5.2.2. Краткое описание решения задачи 91
5.2.3. Программная реализация 92
5.3. Приложение «Формирование запросов» 94
5.3.1. Задание 94
5.3.2. Краткое описание решения задачи 94
5.3.3. Программная реализация 95
5.4. Приложение «Гостиница» 102
5.4.1. Задание 102
5.4.2. Краткое описание решения задачи 103
5.4.3. Программная реализация 103
5.5. Задания для самостоятельной работы 104
5.5.1. Общее задание 104
5.5.2. Варианты заданий 104
Список рекомендуемой литературы 108

ВВЕДЕНИЕ

Microsoft Excel предоставляет пользователю широкие возможности для обработки данных. Это операции хранения, доступа, вычисления, анализа, визуализации, реализуемые через стандартный интерфейс. Но, несмотря на богатый набор возможностей стандартного интерфейса, не все ситуации, связанные с автоматизацией обрабатывающих данные операций, предусмотрены разработчиками **Excel**. Для решения подобных задач в приложениях Microsoft Office интегрирован язык программирования **Visual Basic for Applications** (**VBA**), позволяющий расширять возможности этих приложений.

Отличительной особенностью **VBA** является использование обычных переменных и констант с имеющимися объектами приложений **Microsoft Office**. В **Microsoft Excel** это могут быть рабочие книги, рабочие листы, диапазоны ячеек, диаграммы и т. д.

VBA — это сочетание одного из самых простых языков программирования и всех вычислительных возможностей табличного процессора **Excel**. С помощью этого языка можно легко и быстро создавать разнообразные приложения, не являясь специалистом в области программирования. **VBA** содержит графическую среду, позволяющую наглядно конструировать экранные формы и управляющие элементы.

Данное пособие предназначено для оказания помощи в освоении основных принципов и навыков создания на языке VBA приложений, автоматизирующих обработку табличных данных.

Пособие условно можно разделить на две части. В первой в сжатой форме описаны интерфейс редактора **Visual Basic Editor,** объектная модель **MS Excel**, синтаксис языка программирования **VBA,** средства и приемы создания пользовательского интерфейса. Во второй рассматривается несколько примеров решения учебных задач, объединенных одной прикладной областью. Такой подход обусловлен тем, что изучение и анализ кода работающих программ являются эффективным способом освоения основ программирования на **VBA**.

При этом не преследовалась цель создать приложения для использования в реальных условиях. В ряде случаев для выполнения аналогичных действий в разных процедурах демонстрировались разные способы в ущерб эффективности достижения результатов.

В примерах применялся структурный подход программирования. Его основным принципом является модульное программирование, предполагающее разделение основной задачи на отдельные подзадачи и создание для них отдельных автономных программ — модулей. Специально созданная программа объединяет все модули в целое и управляет их работой. Программа на **VBA** всегда состоит из модулей и подпрограмм. При разработке модульных программ применяются два метода проектирования — нисходящее и восходящее. При нисходящем проектировании разработка программного комплекса идет сверху вниз. На первом этапе разработки кодируется, тестируется и отлаживается головной модуль, который отвечает за логику работы программы. Остальные модули заменяются заглушками, имитирующими работу этих модулей.

Применение заглушек позволяет на самом раннем этапе проектирования проверить работоспособность программы и локализовать источник ошибок. На последующих этапах проектирования все заглушки постепенно заменяются рабочими модулями.

В методических целях в примерах иногда допускались нарушения правил и последовательности этапов разработки приложений для демонстрации использования основных элементов **VBA** при создании программных комплексов.

В тексте пособия названия элементов интерфейса, команды меню выделены полужирным начертанием, программный код, свойства, методы и события объектов — гарнитурой **Courier New**.

1. РЕДАКТОР VBA

Редактор **Visual Basic Editor** (**VBE**) представляет собой среду разработки новых и редактирования существующих программ (макросов) и процедур языка **VBA**. Редактор **VBE** включает полный набор средств отладки, обеспечивающих обнаружение ошибок синтаксиса, ошибок выполнения и логических ошибок в программах.

Редактор **Visual Basic Editor** представляет собой отдельное приложение, запускающееся только в программах **MS Office**. Модули и формы **VBA**, т. е. место, где хранится код на языке **VBA**, сохраняются в файлах **MS Office**.

1.1. ОТОБРАЖЕНИЕ ВКЛАДКИ «РАЗРАБОТЧИК»

В **MS Excel** используется ленточный интерфейс. Одной из вкладок на ленте является вкладка **Разработчик**, где можно вызвать редактор **Visual Basic** и другие инструменты разработчика. Поскольку в **MS Excel** вкладка **Разработчик** не показана по умолчанию, необходимо вывести ее на экран.

Для отображения вкладки **Разработчик** выполните следующие действия.

Перейдите на вкладку **Файл** и нажмите кнопку **Параметры**. В открывшемся окне **Параметры Excel** выберите слева категорию **Настройка ленты**, а справа в группе **Настройка ленты** в списке **Основные вкладки** установите флажок **Разработчик** (рис. 1).

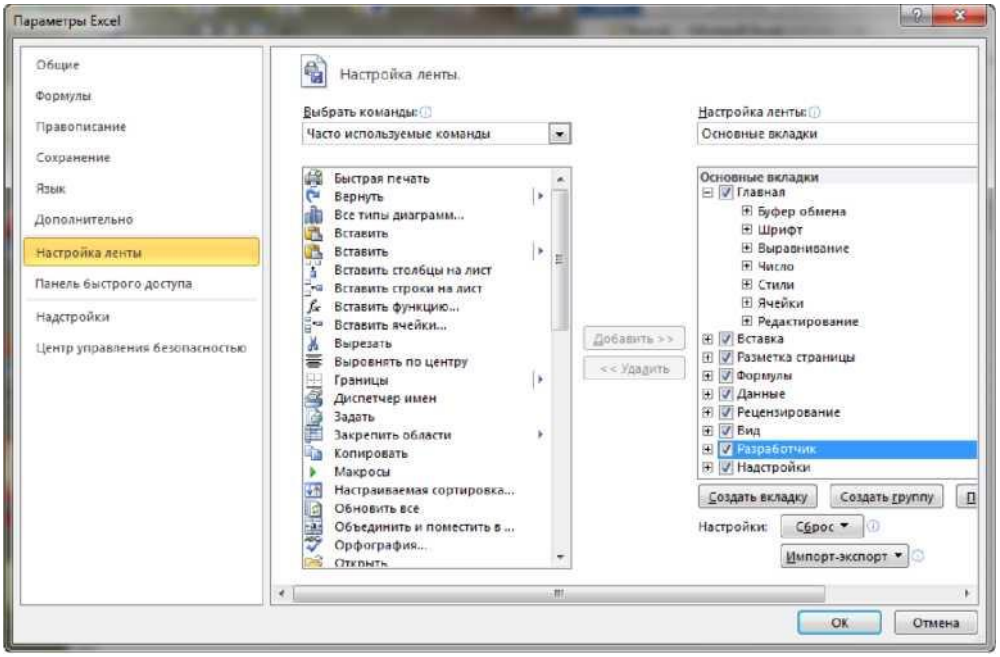

Рис. 1. Размещение на ленте вкладки **Разработчик**

1.2. ЗАПУСК РЕДАКТОРА

Во время работы в **Excel** перейти в окно редактора **Visual Basic Editor** (**VBE**) можно одним из следующих способов:

– на вкладке **Разработчик** в группе **Код** нажать кнопку **Visual Basic**;

– с помощью клавиш-акселераторов **ALT + F11**;

– перейти к модулю кода для рабочего листа, щелкнув правой кнопкой мыши на ярлыке этого листа и выбрав команду **Исходный текст**.

При первом запуске редактора **VBE** откроется окно (рис. 2).

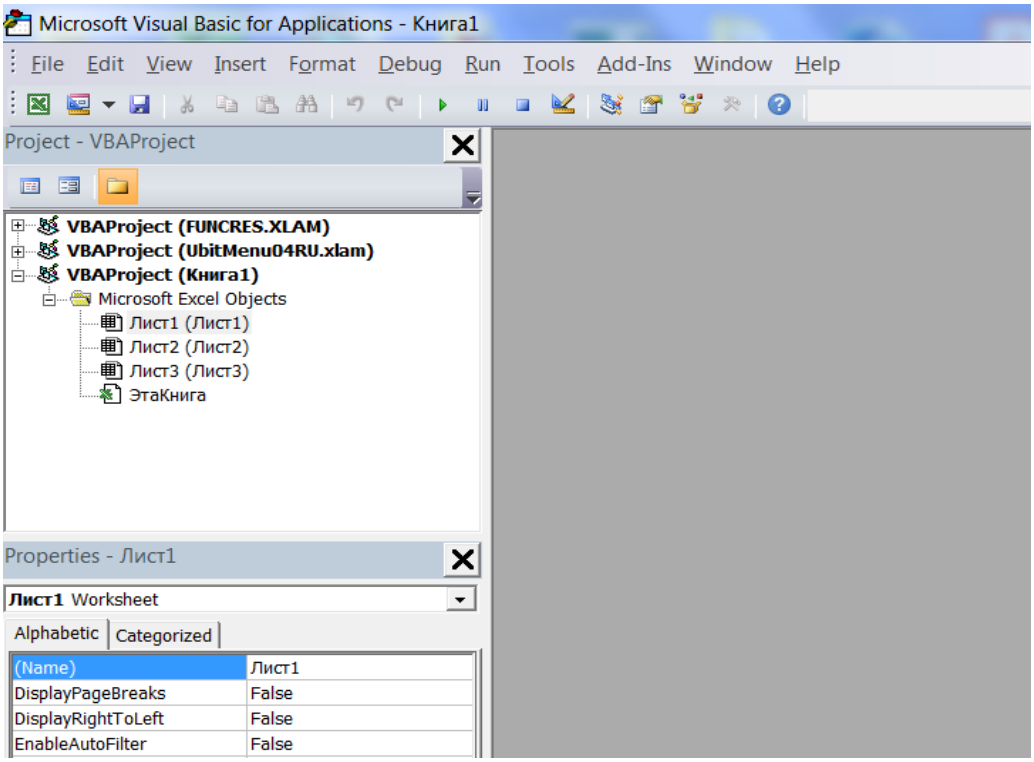

Рис. 2. Фрагмент окна VBE

1.3. ЭЛЕМЕНТЫ ИНТЕРФЕЙСА РЕДАКТОРА

Окно редактора **VBA** состоит из следующих основных компонентов: строка заголовка, строка меню, панели инструментов, окно проекта (**Project**), окно свойств (**Properties**). Посредством меню **View** можно вызвать на экран другие окна, например, окно кода (**Code**) (рис. 3).

В строке заголовка отображается название **Microsoft Visual Basic for Applications** и имя рабочей книги.

Строка меню представляет собой линейку раскрывающихся меню.

Меню **File** (**Файл**) содержит команды, необходимые для сохранения изменений в проекте **VBA**, организации операций экспорта и импорта и вывода на печать исходного кода макросов.

Меню **Edit** (**Правка**) содержит команды, предназначенные для работы с исходным кодом макроса в окне **Code**, а также объектами в формах.

Меню **View** (**Вид**) содержит команды, позволяющие выводить или убирать с экрана различные окна редактора **VBA**.

Команды меню **Insert** (**Вставка**) позволяют добавлять в проект различные объекты: процедуры, модули, формы, содержимое файлов.

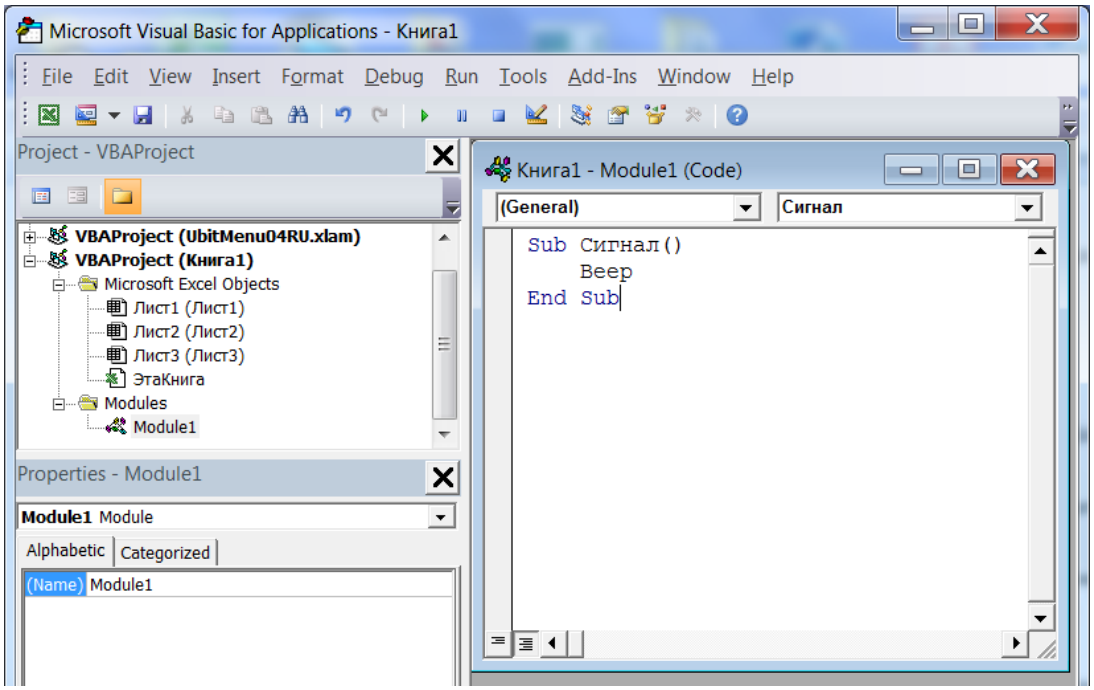

Рис. 3. Интерфейс VBA

Меню **Format** (**Формат**) содержит команды, используемые при создании пользовательских форм. Они позволяют выравнивать объекты в форме по отношению друг к другу, группировать и разгруппировывать элементы управления, настраивать их размеры и внешний вид.

Меню **Debug** (**Отладка**) содержит команды, предназначенные для тестирования и отладки программ пользователя. Команды этого меню позволяют открывать окна отладки, запускать макрос с заданной точки, отслеживать выполнение макроса по шагам и останавливать выполняемую программу в любой момент ее выполнения.

Меню **Run** (**Запуск**) содержит команды, предназначенные для запуска макроса на выполнение, прерывания или возобновления его работы, а также для возврата прерванного макроса в начальное его состояние.

Меню **Tools** (**Сервис**) содержит, в частности, команды, позволяющие выбрать макрос для выполнения или получить доступ к внешним библиотекам макросов. С помощью других команд этого меню можно получить доступ к диалоговому окну **Option** (**Параметры**) редактора **VBA** и окну свойств проекта **VBA**.

Меню **Add-Ins** содержит команду **Add-In Manager**, при выборе которой на экране отображается диалоговое окно **Add-In Manager**. В этом окне можно загружать или выгружать, регистрировать и определять поведение программ-дополнений (надстроек).

Меню **Windows** (**Окно**) позволяет выбирать активное окно, разбивать текущее, размещать окна вертикально, горизонтально и в виде каскада.

Команды меню **Help** (**Помощь**) аналогичны командам меню **Help** в **Word**, **Excel** и других приложениях **Windows**, справочная информация представлена на английском языке.

Панель инструментов включает наиболее часто используемые команды меню в виде кнопок со значками.

По умолчанию под строкой меню располагается панель инструментов **Standard** (**Стандартная**). Кроме панели **Standard**, редактор **VBE** предлагает еще три панели: **Debug** (**Отладка**), **Edit** (**Правка**), **UserForm** (**Форма пользователя**).

Debug (**Отладка**). Кнопки этой панели позволяют запустить программу на выполнение, проследить за ходом ее работы, а также обнаружить различные ошибки в отлаживаемых программах.

Edit (**Правка**). Кнопки этой панели инструментов позволяют редактировать текст в окне **Code** (окне программного кода). Они дублируют команды меню **Edit**.

UserForm (**Форма пользователя**). Эта панель используется при проектировании форм. Многие ее кнопки дублируют команды меню **Format**.

1.4. ОКНА VISUAL BASIC EDITOR

1.4.1. Окно Project Explorer

В окне **Project Explorer** (**Проводник проекта**) отображается древовидная структура всех открытых в данный момент в **Excel** рабочих книг (включая надстройки и скрытые рабочие книги).

Окно **Project Explorer** (**Проводник проекта**) показывает иерархическую структуру открытых в данный момент проектов и предоставляет пользователю средства для быстрого доступа к окнам программного кода, пользовательских форм и других объектов.

Если окно проекта отсутствует, его можно вызвать с помощью команды **View (Вид) → Project Explorer**, или воспользоваться специальной комбинацией клавиш **Ctrl+R**.

В проекте автоматически создается модуль для каждого рабочего листа и для всей рабочей книги, для каждой пользовательской формы, макросов и классов.

Развернутое дерево каждого проекта имеет минимум один узел под названием **Microsoft Excel Objects**. В нем содержатся элементы каждого рабочего листа и лист диаграмм рабочей книги (рабочий лист считается объектом), а также объект **ThisWorkbook** (**ЭтаКнига**), представляющий объект **ActiveWorkbook**. Если в проекте используются модули **VBA**, то в дереве отображается также узел **Modules**, в котором перечислены модули. Проект может включать узел **Forms**, содержащий объекты **UserForm** (пользовательские формы, известные как пользовательские диалоговые окна). Если в проекте находятся модули классов, то в дереве отображается узел **ClassModules**.

Панель инструментов окна проводника содержит три кнопки (рис. 4).

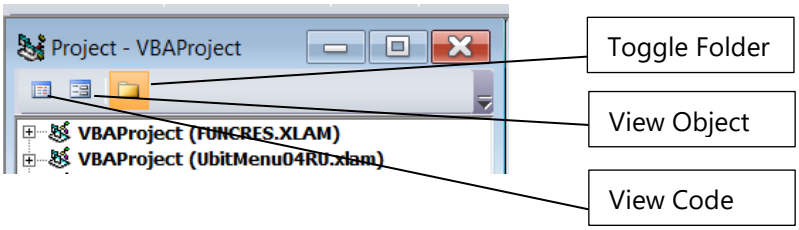

Рис. 4. Панель инструментов окна проводника

Кнопки имеют следующее назначение:

– кнопка **View Code** (**Показать программный код**), позволяет вывести на экран окно программного кода для выделенного объекта;

– кнопка **View Object** (**Показать объект**), выводит на экран выделенный объект;

– кнопка **Toggle Folder** (**Переключение отображения папок**), управляет режимом отображения в самом окне проектов папок среднего уровня иерархии структуры файлов и модулей проектов.

1.4.2. Окно Properties Window

Окно свойств (**Properties Window**) используется для просмотра и изменения свойств любого активного в данный момент объекта (проекта, модуля, формы, элемента управления).

Если окно свойств отсутствует, его можно вызвать с помощью команды **View (Вид) → Properties Window** или нажатием клавиши **F4**.

Свойства можно отображать как в алфавитном порядке, так и по категориям, посредством выбора соответствующих вкладок **Alphabetic** или **Categorized**.

Для изменения свойства выделенного объекта необходимо в окне свойств в левой колонке выбрать свойство, а в правой колонке изменить его значение.

1.4.3. Окно Object Browser

Окно **Object Browser** (**Обозреватель объектов**) предоставляет пользователю средства быстрого доступа к объектам программе **VBA** (рис. 5).

В этом окне можно получить доступ не только ко всем элементам, которые входят в проект, но и к их свойствам, методам, событиям. Окно просмотра объектов по умолчанию не отображается. Оно открывается командой **View (Вид) → Object Browser** или нажатием клавиши **F2**.

Окно **Object Browser** разделено на ряд панелей. В верхней части окна находится панель инструментов, содержащая список **Project/Library** (**Проект/Библиотека**) и кнопки управления обозревателем объектов.

Под панелью инструментов расположена еще одна панель с полем ввода **Search Text** (**Отыскиваемое значение**) для задания критерия поиска и двумя кнопками для запуска поиска и просмотра его результатов.

Ниже находятся две большие панели со списками **Classes** (**Классы**) и **Members** (**Компоненты**), предназначенные для отображения перечня объектов и компонентов выбранного объекта.

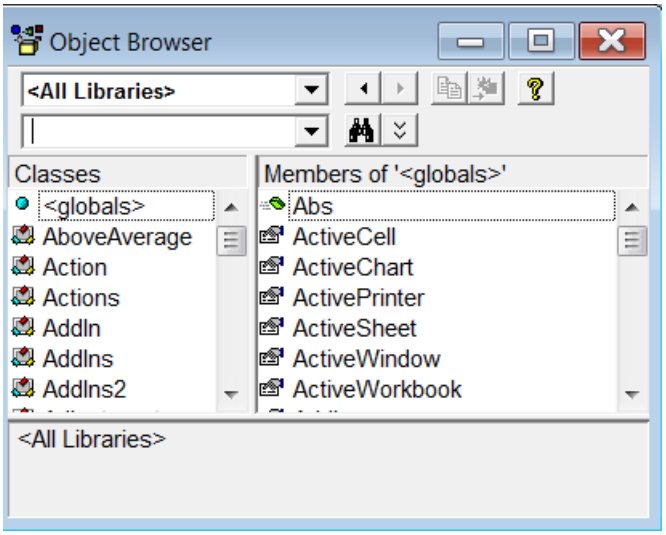

Рис. 5. Окно Object Browser

В нижней части окна расположена панель, на которой отображаются краткие сведения об объекте, выбранном на панели классов или компонентов.

1.4.4. Окно Code

Окно редактирования кода (**Code**) выполняет функции текстового редактора для ввода и изменения процедур и функций проекта (см. рис. 3).

Каждому объекту в проекте соответствует свое окно кода.

Открыть окно редактирования кода можно:

– двойным щелчком по выбранному элементу управления на форме;
– командой меню **View** (**Вид**) → **Code**;
– нажатием клавиши **F7**;
– командой **View Code** контекстного меню;
– щелчком по кнопке **View Code** панели инструментов окна проводника.

Выбирая одну из двух кнопок (**Procedure View** и **Full Module View**), расположенных в левом нижнем углу окна редактирования кода, можно вывести в окне Code код отдельной процедуры или код всего модуля.

Списки **Object** и **Procedure**, расположенные в верхней части окна **Code**, позволяют выбрать для просмотра или редактирования конкретную процедуру. При этом если текущим является объект семейства **Modules**, то в списке **Procedure** будут перечислены процедуры, размещенные пользователем в выбранном модуле, если же обрабатывается объект, обладающий событиями, то список **Procedure** будет включать процедуры-обработчики этих событий. Если данная процедура еще не определена, то будет выведена ее пустая заготовка.

Редактор кода позволяет автоматизировать написание программных операторов, свойств и параметров. При вводе предлагаются варианты продолжения вводимой конструкции. Это может быть список вложенных объектов, свойств и

методов (рис. 6). Вставка выбранного элемента списка осуществляется двойным щелчком или нажатием клавиши **Tab**.

Описанный режим устанавливается при включенном флажке **Auto List Members** вкладки **Editor** диалогового окна **Options**, отображаемого на экране выбором команды **Tools → Options**.

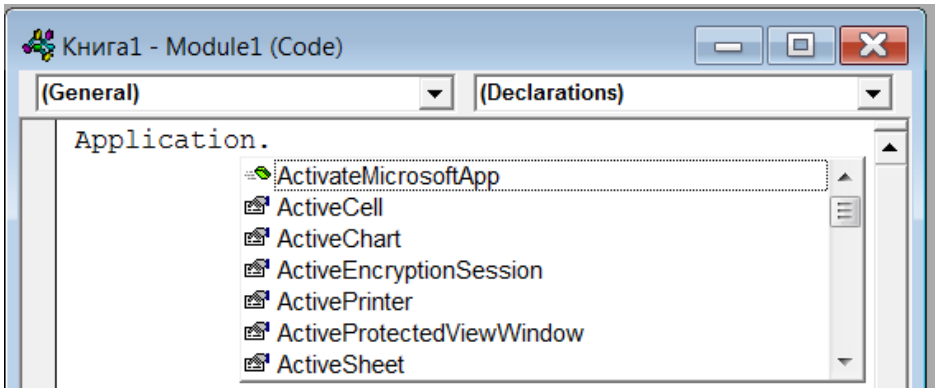

Рис. 6. Список предложений продолжения оператора

Список для завершения вводимого ключевого слова выводится на экран нажатием клавиш **Ctrl+J**.

Редактор кода автоматически отображает на экране описание процедур, функций, свойств и методов после набора их имени (рис. 7).

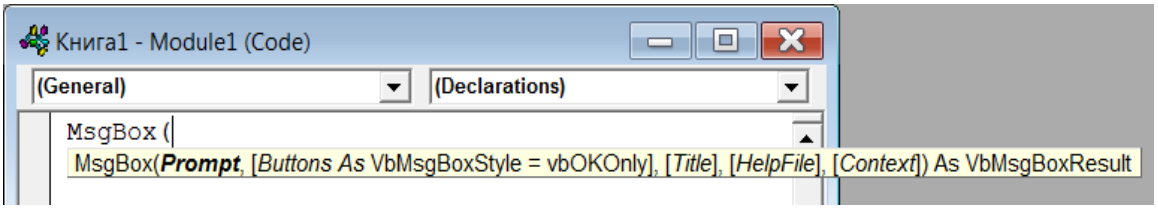

Рис. 7. Вывод описания при вводе оператора

В редакторе кода выполняется автоматическая проверка синтаксиса набранной строки. Строка с синтаксической ошибкой будет выделяться красным цветом. Кроме этого, редактор кода позволяет получить справочную информацию о ключевом слове, процедуре, функции, свойстве или методе. Для этого достаточно установить курсор на нужное ключевое слово и нажать клавишу **F1**.

1.4.5. Окна UserForm и Toolbox

В **VBA** можно создавать в программах интерфейсы пользователя за счет включения в проект объекта **UserForm**.

Пользовательская форма **UserForm** представляет собой пустое диалоговое окно (рис. 8), на котором размещаются необходимые элементы управления. Окно элементов управления **Toolbox** по умолчанию открывается вместе с окном **UserForm**. Если оно отсутствует, отобразить его можно с помощью команды меню **View → Toolbox**.

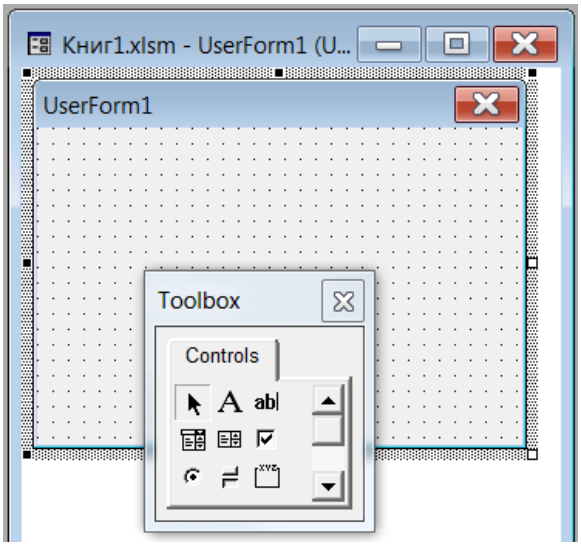

Рис. 8. Окна UserForm и Toolbox

1.4.6. Окно Locals

Окно **Locals Window** (**Локальное окно**) используется при отладке программ и предназначено для просмотра списка локальных переменных приложения и контроля над их значениями (рис. 9). Вызывается это окно командой **View → Locals Window**.

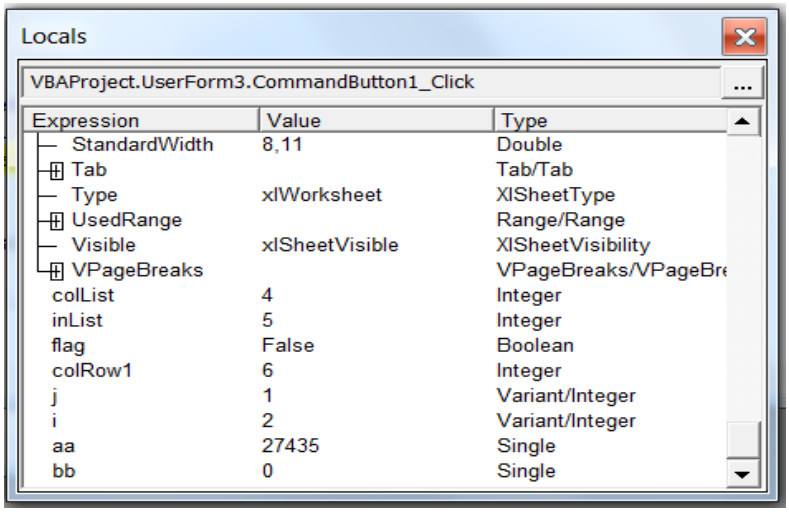

Рис. 9. Окно Locals Window

В окне **Locals Window** можно просмотреть локальные переменные, объявленные в текущей процедуре, их тип и значения. Эта информация автоматически появляется в окне при его вызове. Данное окно используется для отладки и проверки работы приложений. С помощью него можно проконтролировать все локальные переменные в точках останова программы.

При работе с окном **Locals Window** необходимо иметь в виду, что глобальные переменные в этом окне для просмотра недоступны.

Если окно открыто постоянно, то данные между точками останова программы при работе приложения автоматически обновляются.

В окне **Locals Window** можно не только просматривать переменные и их значения в данный момент работы программы, но и иметь возможность изменять значения переменных для проверки реакции программы на эти значения. Для этого в столбце **Value** (**Значение**) необходимо щелкнуть на изменяемом значении, при этом значение переведется в режим редактирования и его можно будет изменить. Клавишей **Enter** или перемещением указателя мыши на другое поле устанавливается новое значение.

1.4.7. Окно Watches

Для более полного контроля работы приложения используется окно **Watches** (**Окно наблюдения**). Это окно вызывается командой меню **View** → **Watch Window** и предназначено для определения значений выражений (рис. 10).

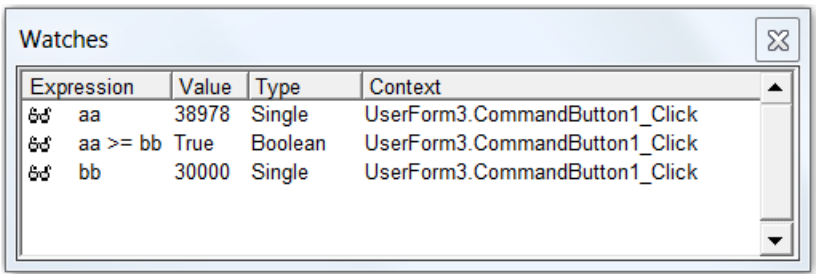

Рис. 10. Окно Watches

В окне **Watches** можно выполнять действия, аналогичные выполняемым в окне **Locals**.

В отличие от **Locals Window** в окно **Watches** нужно добавлять переменную или выражение для контроля. Для этого следует выделить переменную или выражение, которые нужно добавить в окно **Watches**, нажать правую кнопку мыши и выбрать пункт **Add Watch** или выделить выражение и нажать клавиши **Shift+F9**. В появившемся окне **Quick Watch** (рис. 11) нажать клавишу **Add**.

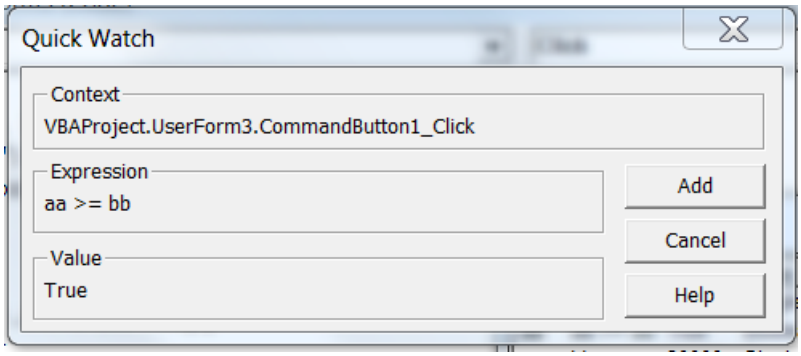

Рис. 11. Окно Quick Watch

Так же как и **Locals Window**, окна **Watches** и **Quick Watch** используется при отладке приложения и проверке его работы.

1.4.8. Окно Immediate

Окно **Immediate** (**Окно непосредственного выполнения**) предназначено для ручного ввода и выполнения команд **VBA**. Это окно вызывается командой меню **View → Immediate Window**.

Для выполнения команды, оператора, просмотра значения переменной или свойства объекта необходимо набрать команду, оператор, имя переменной или название свойства объекта в качестве параметра оператора `Debug.Print` и нажать клавишу **Enter** (рис. 12).

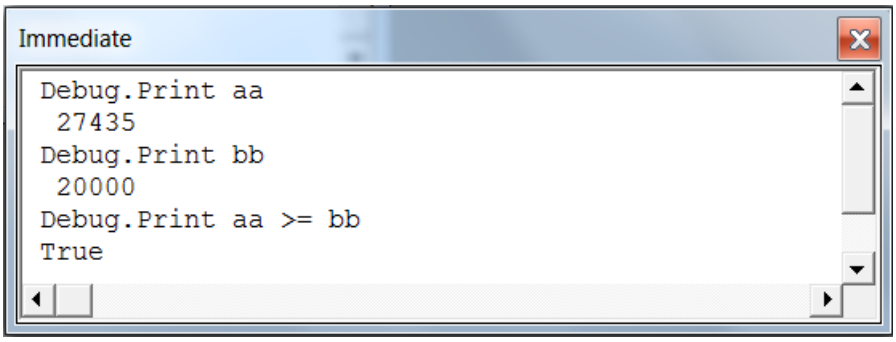

Рис. 12. Окно Immediate

1.4.9. Окна предоставления справочной информации

Для того чтобы открыть главное окно справочной системы **VBA**, необходимо выбрать команду меню **Help → Microsoft Visual Basic for Applications Help** или щелкнуть на кнопке с вопросительным знаком на панели инструментов **Standard**. На экране откроется окно справки с исходной страницей справочной системы редактора **VBA**.

Для вызова контекстно-зависимой справки в редакторе **VBA** предназначена клавиша **F1**. При ее нажатии на экране откроется тот раздел справочной системы, который относится к активному элементу в окне редактора **VBA**. В окне программного кода клавиша **F1** предоставляет сведения о синтаксисе оператора. При работе с пользовательской формой клавиша **F1** позволяет получить справку о выделенном элементе управления формы, или о его свойстве, выбранном в окне свойств. Аналогичным образом можно получить справку об объектах в окне проектов, окне обозревателя объектов и других окнах.

1.4.10. Окно Options

Окно **Options** (**Параметры**) вызывается командой **Tools → Options**.

Окно содержит четыре вкладки: **Editor** (**Редактор**), **Editor Format** (**Формат редактора**), **General** (**Общие**) и **Docking** (**Прикрепление**) для задания параметров редактора (рис. 13).

На вкладке **Editor** устанавливаются параметры для работы в редакторе. Параметры вкладки **Editor Format** определяют параметры кода: цвет, шрифт, размер, полосу индикатора границы. Вкладка **General** содержит общие параметры и рекомендуется настройки оставить по умолчанию. Параметры вкладки **Docking** определяют поведение окон редактора **VBE**.

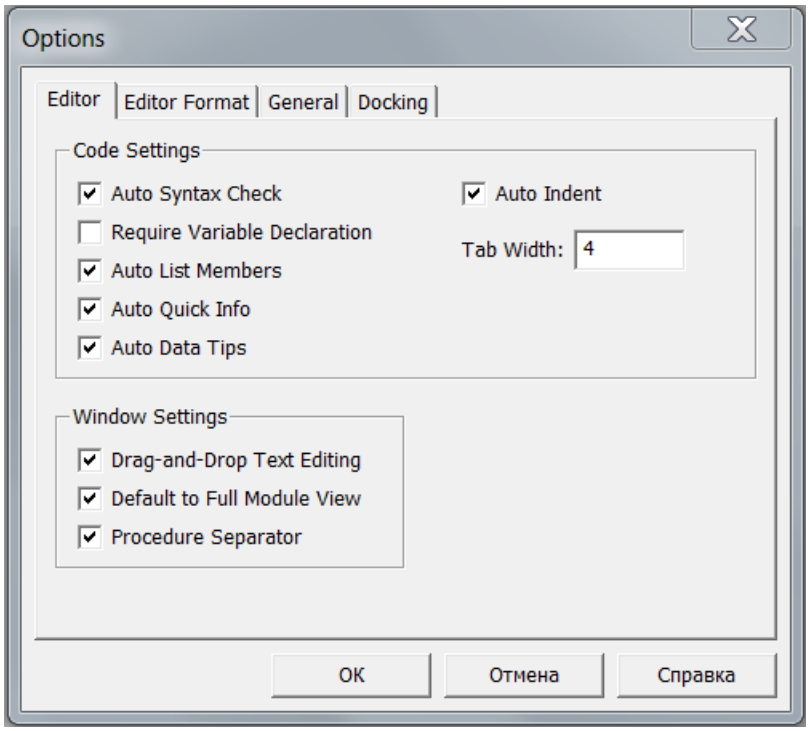

Рис. 13. Окно Options

2. ОБЪЕКТНАЯ СТРУКТУРА ЯЗЫКА VBA

Visual Basic for Applications — упрощенная реализация языка программирования **Visual Basic**, встроенная в линейку продуктов **Microsoft Office**. Не являясь языком объектно-ориентированного программирования в строгом смысле этого слова, поддерживает использование элементов объектно-ориентированного подхода и связанные с ним понятия.

2.1. ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

Язык **VBA** является языком визуального и событийного программирования. Он реализует следующие возможности: создание нестандартного окна диалога в виде формы с набором элементов управления, создание формы на рабочем листе, написание процедур, обрабатывающих события, которые возникают при тех или иных действиях системы и конечного пользователя.

Язык **VBA** не обладает всеми возможностями **VB**, но позволяет работать с набором объектов — в частности, в нем определены все объекты **MS Excel**.

Каждое приложение **MS Office** имеет свою объектную модель, описывающую состав объектов и связи между ними. Язык **VBA** позволяет управлять этими объектами.

В языке **VBA** используются общие принципы объектно-ориентированного программирования.

Объектно-ориентированное программирование (ООП) — это технология программирования, которая основана на представлении программы в виде совокупности объектов, каждый из которых является реализацией определенного класса, а классы образуют иерархию на принципах наследования.

При объектно-ориентированном программировании программа строится как совокупность взаимодействующих объектов.

Объект — это базовое понятие ООП. Каждый объект характеризуется свойствами, методами и событиями Свойства (properties) — это совокупность характеристик и атрибутов, описывающих объект.

Методы объекта — это действия, которые можно выполнить с объектом.

События объекта — это показатели, которые характеризуют реакцию объекта. Класс — это совокупность объектов, характеризующихся общими методами обработки и свойствами.

Основными принципами ООП являются инкапсуляция, наследование и полиморфизм. Инкапсуляция — это сокрытие информации и комбинирование данных и методов внутри объекта. Наследование — это способность объекта сохранять свойства и методы класса-родителя. Полиморфизм — возможность задания в иерархии объектов различных действий в методе с одним именем.

VBA в **Excel** позволяет управлять более чем ста классами объектов, включая рабочую книгу, рабочий лист, диапазон ячеек рабочего листа и др.

Классы объектов организованы в иерархическую структуру.

Объекты могут выступать контейнерами других объектов. Например, **Excel** — это объект самого высокого уровня под названием `Application`

(Приложение). Он содержит другие объекты, например, Workbook (Рабочая книга). Объект Workbook содержит, в свою очередь, другие объекты, например, Worksheet (Рабочий лист) и Chart (Диаграмма). Объект Worksheet может включать такие объекты, как Range (Диапазон), Cell (Ячейка) и др.

Одинаковые объекты образуют коллекцию. Например, коллекция Worksheets включает все рабочие листы в указанной рабочей книге. Коллекции сами являются объектами.

При ссылке на объект, вложенный в другой объект, положение в иерархической структуре задается с помощью точечной нотации, т. е. в виде последовательности имен объектов, разделенных точкой.

2.2. ИЕРАРХИЯ ОБЪЕКТОВ

Объекты **Excel** находятся на разных уровнях иерархии. Для их эффективного использования в программах **VBA** необходимо представлять себе эту структуру (рис. 14).

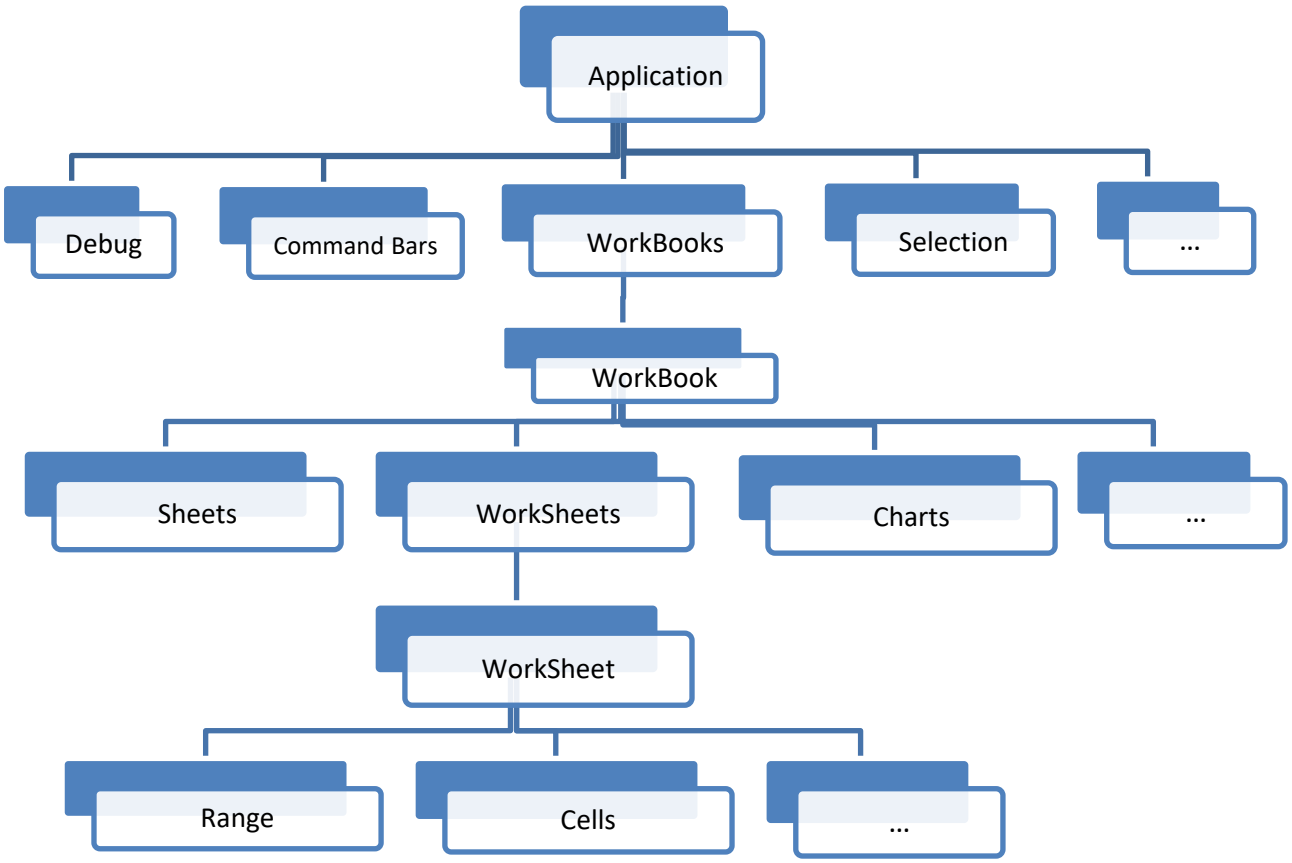

Рис. 14. Фрагмент дерева иерархии объектной модели VBA для Excel

Расположенный на верхнем уровне иерархии объект Application представляет **Microsoft Excel**. Остальные объекты расположены на следующих уровнях иерархии.

Чтобы управлять свойствами и методами объекта, иногда нужно записать ссылки на все объекты, находящиеся в иерархии выше него. Ссылки на объекты

записываются последовательно, начиная с верхнего уровня иерархии, при этом используется точечная нотация: в качестве разделителя между ссылками (именами) ставится точка. Например, оператор `Application.Workbooks ("Книга1").Worksheets(1).Range("A1").Value = 1` определяет ссылку на ячейку `A1` первого листа коллекции листов книги `Книга1` коллекции рабочих книг объекта `Application` и заносит значение в указанную ячейку.

Ссылки на объекты в коллекции могут быть оформлены двумя способами: явным указанием имени объекта в коллекции (`Workbooks("Книга1")`), указанием порядкового индекса в коллекции (`Worksheets(1)`).

Уровни иерархии объектов в сложной ссылке убывают слева направо. Объект, стоящий в последовательности слева от рассматриваемого объекта, является для последнего объектом-контейнером. Так объект `Application` будет контейнером для объекта `Книга1`, объект `Книга1` — контейнером для объекта `Лист` с индексом 1, который, в свою очередь, будет контейнером для объекта `Range("A1")`. В ссылке можно опускать имя объекта верхнего уровня. В этом случае ссылка будет применяться к активному объекту.

Если код выполняется из среды **Excel**, то указание объекта `Application` не является необходимым, и оператор в вышеприведенном примере преобразуется к виду: `Workbooks("Книга1").Worksheets(1). Range("A1").Value = 1`. Если активной рабочей книгой является `Книга1`, ссылку можно записать в виде: `Worksheets("Лист1").Range("A1")`. При активном первом листе в коллекции ссылку можно еще упростить: `Range("A1")`.

Для неявной ссылки на рабочую книгу можно использовать ссылки `ActiveWorkbook` (ссылка на активную рабочую книгу, с которой в данный момент работает пользователь) и `ThisWorkbook` (ссылка на рабочую книгу, в которой хранится исполняемый в настоящий момент код).

Для доступа к активному рабочему листу можно использовать ссылку `ActiveSheet`.

На активный диапазон ячеек можно сослаться, используя свойство `Selection` объекта `Application`. Но это свойство может ссылаться и на другие объекты помимо ячеек, поэтому при использовании такой ссылки не следует полагаться на использование свойства по умолчанию.

Ссылки `ActiveWorkbook`, `ThisWorkbook`, `ActiveSheet` и `Selection` — это ссылки на свойства объектов.

2.3. ОПЕРИРОВАНИЕ СВОЙСТВАМИ ОБЪЕКТОВ

Изменение значений свойств объектов влияет на поведение и внешний вид объекта. С помощью свойств, например, можно задать цвет, значение, шрифт или формат диапазону ячеек.

Для ссылки на свойство также используется точечная нотация: имя свойства записывается через точку после имени объекта. Например, сослаться на значение в ячейке **`A1`** листа **`Лист1`** можно так:

`Worksheets("Лист1").Range("A1").Value`. Изменить свойство объекта можно программно, используя конструкцию:

`Объект.Свойство = Значение`.

Присвоить некой переменной значение свойства объекта позволяет конструкция `Переменная = Объект.Свойство`.

Различные варианты использования свойств объектов приведены в примерах.

Примеры:

1) `Cells(1, 1).Value = 10` — установить значение в ячейке **A1**, равное 10;

2) `Range("A2").Font.FontStyle = "курсив"` — в ячейке **A2** установить начертание курсив;

3) `Range("A2").Font.Color = RGB(255, 0, 0)` — в ячейке **A2** установить красный цвет шрифта;

4) `Range("A2").WrapText = True` — для ячейки **A2** разрешить перенос по словам;

5) `Range("A2").Orientation = 90` — для ячейки **A2** задать ориентацию текста 90°;

6) `Range("A2").Interior.Color = RGB(0, 0, 255)` — для ячейки **A2** установить синий цвет заливки;

7) `stAdd = Range("B2:C3").Address` — сохранить в переменной `stAdd` адрес диапазона **B2:C3**.

2.4. ИСПОЛЬЗОВАНИЕ МЕТОДОВ ОБЪЕКТОВ

При обращении к методу объекта названия объекта и метода так же отделяются точкой. Формы обращения к методу могут различаться:

1) `Объект.Метод Объект.Метод Параметр1, Параметр2, …`

2) `Переменная = Объект.Метод(Параметр1, Параметр2, …)`

3) `Объект.Метод Параметр1:= ЗначениеПараметра1, _ Параметр2:= ЗначениеПараметра2, …`

Нагляднее это иллюстрируют примеры:

1) `Range("A1:D5").Clear` — очистить диапазон ячеек **A1:D5**;

2) `Worksheets(1).Rows(2).Delete` — удалить две строки в первом рабочем листе коллекции рабочих книг;

3) `WorkBooks("MyBook").Close SaveChanges:= True` — закрыть рабочую книгу `MyBook` с сохранением сделанных изменений (`SaveChanges` — параметр, определяющий, сохранить (`True`) или не сохранить изменения (`False`) при сохранении книги;

4) `Set ss = Worksheets.Add(Worksheets("Лист2"), ,1)` — вставить один рабочий лист в активную рабочую книгу и присвоить ссылку на него объектной переменной `ss`; в методе первый параметр — лист, перед которым будет вставлен новый рабочий лист (**Лист2**), второй параметр — лист, по-

сле которого будет осуществлена вставка (опущен), третий параметр — количество вставляемых листов (1).

2.5. СОБЫТИЯ

Событие — это реакция объекта на действие пользователя или программы. Событием может быть щелчок мышью по кнопке, открытие рабочей книги, изменение содержимого ячейки. Все события в **VBA** имеют названия, например, `Click` — щелчок левой кнопкой мыши, `DblClick` — двойной щелчок левой кнопкой мыши, `MouseMove` — перемещение указателя мыши, `Change` — изменение объекта и др.

С такими событиями связаны соответствующие встроенные процедуры изначально без обрабатывающего кода. В процедуру помещается код программы, которая выполняется, когда это событие происходит.

Для включения обрабатывающего кода в процедуру, следует выделить объект в окне **Project**, на панели инструментов в верхней части окна щелкнуть левой кнопкой мыши по кнопке **View Code** (рис. 15). В открывшемся окне **Code** в левом списке под заголовком окна выбрать название объекта, для которого предполагается создать процедуру обработки события. В правом списке выбрать имя обрабатываемого события. В результате генерируется «пустая» процедура-обработчик, состоящая из двух операторов: начального `Sub` и конечного `End Sub`, между которыми необходимо поместить код программы обработки. Программа будет выполняться каждый раз при наступлении этого события.

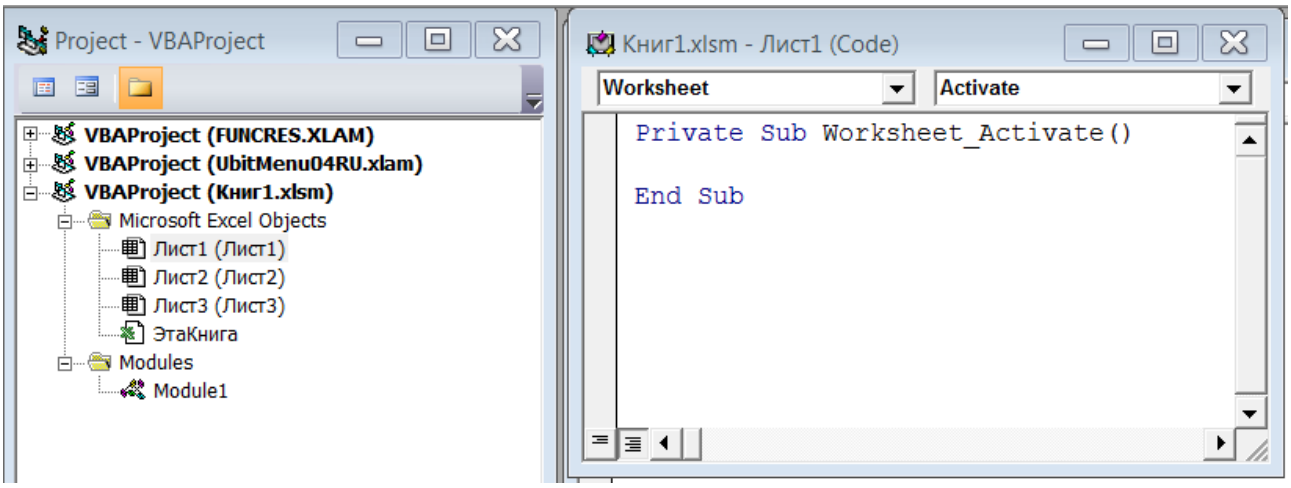

Рис. 15. Интерфейс создания обработчика события

На рис. 15 для объекта `Worksheets("Лист1")` сгенерирована пустая процедура обработки события активизации листа.

2.6. ОСНОВНЫЕ ОБЪЕКТЫ EXCEL

Ниже рассматриваются только основные, наиболее часто используемые объекты **Excel**. Более полную информацию можно получить в справочной системе. Удобным средством является браузер объектов, который позволяет найти по имени нужный объект, его свойства и методы.

Работа, которая выполняется в **VBA**, в основном связана с управлением ячейками и диапазонами на рабочих листах, рабочими листами и книгами. Поэтому далее описываются именно эти объекты.

2.6.1. Объект Application

Объект `Application` — это объект самого верхнего уровня в иерархии объектов **Excel**, представляющий само приложение **Excel**. Его свойства и методы позволяют установить общие параметры приложения **Excel**. Кроме того, объект `Application` через свойство `WorksheetFunction` предоставляет возможность использовать в коде встроенные функции рабочего листа. Объект `WorksheetFunction` являтся контейнером всех функций рабочего листа. Например, выражение `Application.WorksheetFunction.CountA (WorkSheets("Лист1").Range("C:C"))` определяет количество непустых ячеек в столбце `C` рабочего листа `Лист1`.

Функции рабочего листа можно включать в код непосредственно через объект `Application`, опуская свойство `WorksheetFunction`. То есть предыдущее выражение можно записать:

```
Application.CountA(WorkSheets("Лист1").Range("C:C"))
```

Рассмотрим некоторые свойства объекта в табл. 1.

Таблица 1

Свойства объекта `Application`

Свойство	Описание
`ActiveWorkbook`	Возвращают активную рабочую книгу
`ActiveSheet`	Возвращают активный лист
`ActiveCell`	Возвращают активную ячейку
`Activechart`	Возвращают активную диаграмму
`Calculation`	Определяет режим вычислений. Допустимые значения: `xlCalculationManual` (вручную); `xlCalculationAutomatic` (автоматически); `xlCalculationSemiAutomatic` (автоматически, кроме таблиц)
`Caption`	Возвращает или устанавливает текст из заголовка главного окна **Excel**. Задание значения свойства `Empty` возвращает заголовок, используемый по умолчанию
`DisplayScrollBars`	Регулирует отображение полос прокрутки. Принимает значения `True` или `False`
`DisplayStatusBar`	Регулирует отображение строки состояния. Принимает значения `True` или `False`
`Height`	Возвращает или устанавливает высоту окна приложения
`Left`	Возвращает или устанавливает расстояние от левой границы окна приложения до левого края экрана
`Right`	Возвращает или устанавливает расстояние от правой границы окна приложения до правого края экрана
`Selection`	Возвращает выбранный в текущий момент объект в активном листе для объекта `Application`. Возвращает значение `Nothing`, если объект не выбран

Свойство	Описание
StatusBar	Возвращает или устанавливает текст, выводимый в строке состояния
ThisWorkbook	Возвращает рабочую книгу, содержащую выполняющийся в данный момент макрос. Если макрос выполняется в неактивной книге, то, в отличие от ActiveWorkbook, может возвращать неактивную рабочую книгу
Top	Возвращает или устанавливает расстояние от верхней границы окна приложения до верхнего края экрана
Width	Возвращает или устанавливает ширину окна приложения
WindowState	Устанавливает размер окна. Допустимые значения: xlMaximized (максимальный); xlMinimized (минимальный); xlNormal (нормальный)

В табл. 2 описаны основные методы объекта Application.

Таблица 2

Методы объекта Application

Метод	Описание
Calculate	Вызывает принудительное вычисление в объектах (открытых рабочих книгах, рабочих листах, диапазоне), для которых используется метод
Help	Отображает справку. Параметры: HelpFile — имя HLP-файла (по умолчанию отображается файл справки **Microsoft Excel**); HelpContextID — номер раздела справки (по умолчанию отображается оглавление справки)
OnKey	Выполняет заданную процедуру при нажатии определенной клавиши или комбинации клавиш. Параметры: Key — обязательный строковый аргумент определяет комбинацию клавиш или клавишу, которая назначена процедуре; Procedure — необязательный строковый аргумент, который определяет имя процедуры, запускаемой при нажатии клавиши или комбинации клавиш
OnTime	Назначает выполнение процедуры на заданное время. Параметры: EarliestTime — время запуска процедуры; Procedure — имя выполняемой процедуры; LatestTime — последний момент запуска процедуры, если на момент, заданный параметром EarliestTime, Excel не может запустить указанную процедуру из-за того, что выполняет другую операцию. Если до указанного времени запуск процедуры не произошел, то она не исполняется; Schedule — позволят отложить выполнение процедуры на сутки, если его значение равно True
Run	Запускает макрос или подпрограмму. Параметры: Macro — имя макроса; Arg1, Arg2 … — аргументы макроса
Volatile	Вызывает перевычисление функции пользователя при изменениях в ячейках рабочего листа
Wait	Временно приостанавливает работу приложения. Параметр: Time — время предполагаемого возобновление работы
Quit	Закрывает **Excel**

В программах на VBA можно обрабатывать следующие события объекта `Application` (табл. 3).

Таблица 3

События объекта `Application`

Процедура	Событие
`NewWorkbook`	Создание новой рабочей книги
`SheetActivate`	Активация любого листа книги
`SheetBeforeDoubleClick`	Двойной щелчок по листу
`SheetBeforeRightClick`	Перед нажатием правой кнопки мыши на листе
`SheetCalculate`	После пересчета листа или после изменения данных, которые отображаются на диаграмме
`SheetChange`	Изменение содержимого ячеек на любом листе
`SheetFollowHyperlink`	Переход по гиперссылке на листе
`SheetSelectionChange`	Изменение выделения на листе
`WindowActivate`	Активация окна книги
`WindowDeactivate`	Потеря фокуса окном книги
`WindowResize`	Изменение размера окна книги
`WorkbookActivate`	Активизация любой рабочей книги
`WorkbookBeforeClose`	Перед закрытием рабочей книги
`WorkbookBeforePrint`	Перед печатью рабочей книги
`WorkbookBeforeSave`	Перед сохранением рабочей книги
`WorkbookDeactivate`	Потеря фокуса открытой рабочей книгой
`WorkbookNewSheet`	Добавление нового рабочего листа
`WorkbookOpen`	Открытие рабочей книги

Если при создании процедур-обработчиков событий для объектов второго и нижерасположенных уровней в структуре объектной модели принципиальных трудностей не возникает (см. раздел События), то для доступа к процедурам обработки событий объекта `Application` необходимо проделать предварительные действия.

Чтобы получить возможность обрабатывать события объекта `Application`, следует создать модуль класса в окне **VBE**, выполнив команду **Insert → Class Module** (см. раздел «ЭЛЕМЕНТЫ ИНТЕРФЕЙСА »), и объявить в нем объектную переменную, используя ключевое слово `WithEvents`:

```
Public WithEvents ПеременнаяПриложения As Application
```

Данная инструкция определяет, что описанная в ней переменная ссылается на экземпляр класса, который может порождать события.

После этого в списке объектов (**Object**) окна кода модуля класса появится объект `ПеременнаяПриложения`, а в списке процедур (**Procedure**) — соответствующие ему события, некоторые из которых приведены в вышерасположенной таблице. Требуемая «заготовка» для процедуры обработки выбранного события (ее заголовок со стандартным именем и списком параметров и инструкция, завершающая процедуру) будет создан автоматически при выборе имени `ПеременнаяПриложения`, описанного в модуле класса с ключевым словом `WithEvents`, в списке объектов и нужного события из списка процедур.

Общий синтаксис «заготовки» для процедуры обработки события можно представить в следующем виде:

```
Private Sub ПеременнаяПриложения_Событие (СписокПараметров)

End Sub
```

Параметры, передаваемые процедурам обработки событий, позволяют уточнить условия возникновения и обработки событий (например, могут задать ссылку на рабочую книгу или рабочий лист, с которой связано событие).

После создания модуля класса для обработчиков событий приложения можно в любом модуле «подключиться» к этим процедурам. Для этого нужно описать переменную `Dim Переменная As New ИмяМодуляКласса` и записать код, который установит ссылку на приложение, для которого должны обрабатываться события:

```
Sub Процедура()
Set Переменная.ПеременнаяПриложения = Application
End Sub
```

Выполнение процедур обработки событий можно отключить, если присвоить свойству `EnableEvents` объекта `Application` значение `False`.

На рис. 16 приведен фрагмент окна **VBE**. Фрагмент иллюстрирует пример создания обработчика события создания нового рабочего листа в книге `Книг1.xlsm`.

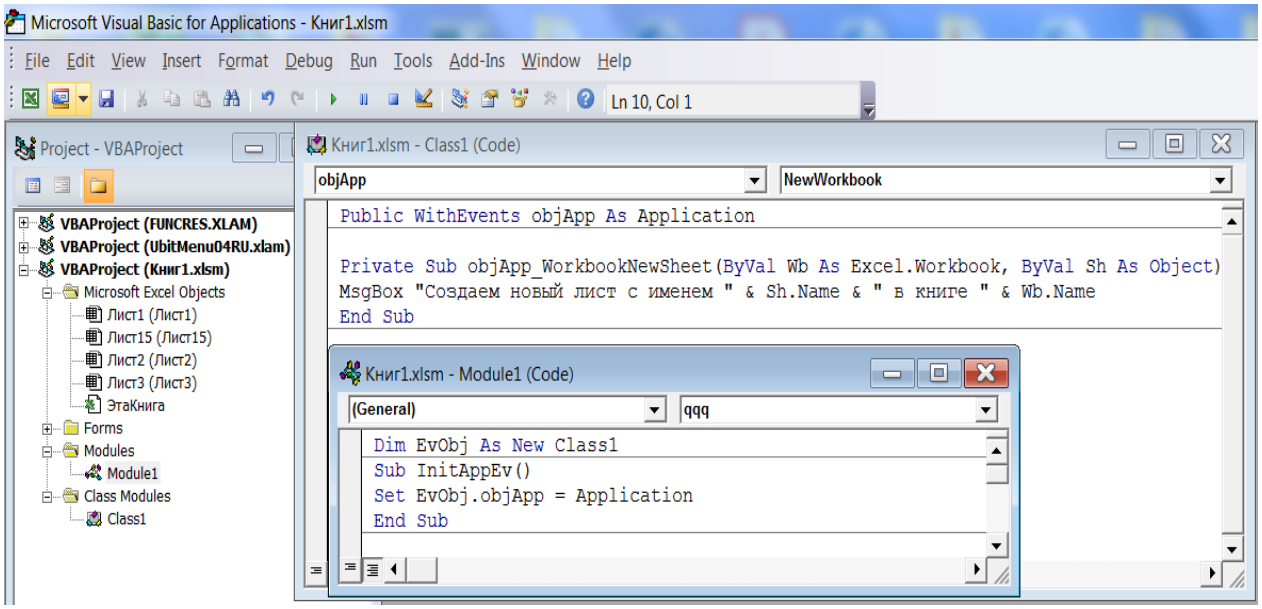

Рис. 16. Пример создания обработчика события объекта Application

Предварительно были созданы модуль `Module1` (**Insert → Module**) и модуль класса `Class1` (**Insert → Class Module**) с именами, установленными по умолчанию.

В модуле класса Class1 описана объектная переменная objApp как экземпляр класса Application, с возможностью порождать события:

```
Public WithEvents objApp As Application
```

В списке объектов выбрано имя этой переменной (objApp), а в списке процедур — событие WorkbookNewSheet. В сгенерированную «заготовку» для процедуры обработки выбранного события записан оператор вывода окна сообщения:

```
Private Sub objApp_WorkbookNewSheet(ByVal Wb As _
Excel.Workbook, ByVal Sh As Object)
MsgBox "Создан новый лист с именем " & Sh.Name & _
" в книге " & Wb.Name
End Sub
```

Параметрами процедуры являются объектные переменные Wb и Sh. В них записываются для передачи в процедуру, соответственно, ссылка на рабочую книгу, в которой создается новый рабочий лист, и ссылка на этот лист.

В модуле Module1 записывается код процедуры, который связывает объект созданного класса Class1 с работающим приложением:

```
Dim EvObj As New Class1
Sub InitAppEv()
Set EvObj.objApp = Application
End Sub
```

После выполнения этого кода, процедуры обработки событий объекта Application будут реагировать на наступление соответствующих событий.

Таким образом, при создании нового рабочего листа в рабочей книге будет выведено окно с сообщением (рис. 17).

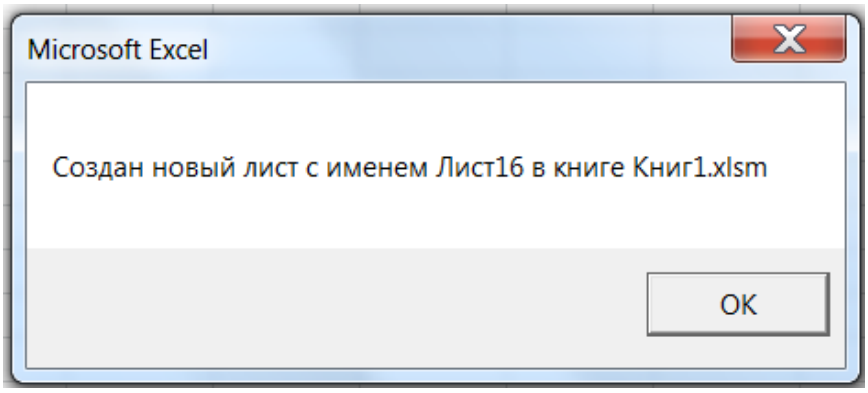

Рис. 17. Результат обработки события WorkbookNewSheet

2.6.2. Объект Workbook коллекции Workbooks

В иерархии объектной модели **Excel** объект Workbook занимает второй уровень и представляет файл рабочей книги. Ссылка на объект WorkBook осуществляется через коллекцию Workbooks. Свойства и методы этого объек-

та позволяют работать с файлами рабочих книг. Некоторые свойства объекта приведены в табл. 4.

Таблица 4

Свойства объекта `WorkBook`

Свойство	Описание
`ActiveSheet`	Возвращает активный лист книги
`DisplayStatusBar`	Регулирует отображение строки состояния. Принимает значения `True` или `False`
`FullName`	Возвращает полное имя книги, включая путь
`Height`	Возвращает или устанавливает высоту окна приложения
`Left`	Возвращает или устанавливает расстояние от левой границы окна приложения до левого края экрана
`Name`	Возвращает или устанавливает имя книги
`Path`	Возвращает полное имя папки, в которой находится книга
`Right`	Возвращает или устанавливает расстояние от правой границы окна приложения до правого края экрана
`Sheet`	Возвращает семейство всех листов книги
`Saved`	Возвращает `True`, если не производились изменения в документе со времени его последнего сохранения, `False` — в противном случае
`StatusBar`	Возвращает или устанавливает текст, выводимый в строке состояния
`Top`	Возвращает или устанавливает расстояние от верхней границы окна приложения до верхнего края экрана
`Width`	Возвращает или устанавливает ширину окна приложения
`WindowState`	Устанавливает размер окна. Допустимые значения: `xlMaximized` (максимальный); `xlMinimized` (минимальный); `xlNormal` (нормальный)
`Worksheets`	Возвращает семейство всех рабочих листов книги
`WriteReserved`	Возвращает `True`, если книга закрыта для записи, `False` — в противном случае

Основные методы объекта `WorkBook` приведены в табл. 5.

Таблица 5

Методы объекта `WorkBook`

Метод	Описание
`Activate`	Активизирует рабочую книгу с активным первым рабочим листом
`Add`	Создает новый объект в семействе `Workbooks`. Параметр: `Template` — задает файл шаблона, на основе которого создается новая рабочая книга
`Close`	Закрывает рабочую книгу. Параметры: `SaveChanges` — принимает значения `True` или `False`, при значении `True` изменения сохраняются в книге, если имя файла, связанное с книгой, еще не задано, используется имя `filename`, если параметр `Filename` опущен, пользователю предлагается указать имя файла; `Filename` — необязательный, задает имя файла для сохранения
`Open`	Открывает существующую рабочую книгу. Параметры: `FileName` — обязательный, задает имя открываемого файла; `Readonly` — необязательный, принимает значения `True` или `False`, при значении `True` файл

Метод	Описание
	открывается в режиме, доступном только для чтения; Password — необязательный, строка пароля для защищенной книги; Notify — необязательный, принимает значения True или False, при значении True происходит извещение о статусе файла «только для чтения»; AddToMRU — необязательный, принимает значения True или False, при значении True файл добавляется в список недавно использованных файлов
Protect	Устанавливает защиту рабочей книги. Параметры: Password — необязательный, задает пароль для защиты книги; Structure — необязательный, принимает значения True или False, при значении True устанавливает защиту структуры книги (взаимное расположение листов); Windows — необязательный, принимает значения True или False, при значении True устанавливает защиту окна книги
Save	Сохраняет рабочую книгу
SaveAs	Сохраняет рабочую книгу в другой файл. Параметры: Filename — строка имени файла, в который будет сохранена рабочая книга; FileFormat — необязательный, задает формат файла; Password — необязательный, пароль для защиты доступа к книге; WriteResPassword — необязательный, пароль для защиты книги от записи; ReadOnlyRecommended — необязательный, принимает значения True или False, при значении True при открытии файла будет отображаться рекомендация открыть файл только для чтения
Unprotect	Снимает защиту рабочей книги. Параметр: Password — необязательный, строка, используемая в качестве пароля для защиты листа

Чаще всего в приложениях обрабатываются следующие события объекта WorkBook (табл. 6):

Таблица 6

События объекта WorkBook

Процедура	Событие
BeforeClose	Закрытие рабочей книги (перед выводом запроса на сохранение изменений)
BeforePrint	Печать рабочей книги или ее части (перед печатью)
BeforeSave	Сохранение рабочей книги (перед сохранением)
Deactivate	Потеря фокуса открытой рабочей книги
NewWorkbook	Создание новой рабочей книги
NewSheet	Добавление нового рабочего листа
Open	Открытие рабочей книги
SheetActivate	Активизация любого рабочего листа книги
SheetBeforeDoubleClick	Двойной щелчок по любому рабочему листу книги (до выполнения стандартной операции, которая производится по умолчанию при двойном щелчке)
SheetBeforeRightClick	Щелчок правой кнопкой по любому рабочему листу книги (до выполнения стандартной операции, которая производится по умолчанию при нажатии правой кнопки мыши)
SheetCalculate	Пересчет значений на любом рабочем листе книги или после вывода измененных данных на диаграмме

Процедура	Событие
SheetChange	Изменение пользователем или ссылкой содержимого ячейки на любом рабочем листе
SheetDeactivate	Потеря фокуса активным рабочим листом
SheetSelectionChange	Изменение выделенного диапазона ячеек

2.6.3. Объект Worksheet коллекции Worksheets

В объектной модели **Excel** объект Worksheet занимает третий уровень иерархии и представляет рабочий лист. Ссылка на объект Workbook осуществляется через коллекцию Workbooks.

В табл. 7 и 8 приведены несколько наиболее часто используемых свойств и методов объекта Worksheet.

Таблица 7

Свойства объекта Worksheet

Свойство	Описание
ActiveCell	Возвращает активную ячейку рабочего листа
Cells	Возвращает объект Range, который представляет все ячейки на указанном листе, при определении строки и столбца, возвращает ссылку на соответствующую ячейку
Columns	Возвращает объект Range, который представляет все столбцы на указанном листе, при определении столбца, возвращает ссылку на соответствующий столбец
Index	Возвращает номер индекса объекта в коллекции рабочих листов
Name	Возвращает или устанавливает имя рабочего листа
Range	Возвращает ссылку на указанный диапазон ячеек
Rows	Возвращает объект Range, который представляет все строки на указанном листе, при определении строки, возвращает ссылку на соответствующую строку
Visible	Устанавливает видимость (значение True) или сокрытие (значение False) рабочего листа
UsedRange	Возвращает используемый диапазон рабочего листа

Таблица 8

Методы объекта Worksheet

Метод	Описание
Activate	Активизирует рабочий лист
Add	Создает новый рабочий лист. Параметры: Before — лист, перед которым будет размещен новый лист; After — лист после которого будет размещен новый лист; Count — число добавляемых листов; Type — тип добавляемого листа
Copy	Копирует рабочий лист в другое место рабочей книги. Параметры: Before — лист, перед которым будет размещен копируемый лист, нельзя указать Before, если задан After; After — лист, после которого будет размещен копируемый лист
Delete	Удаляет рабочий лист

Метод	Описание
Move	Перемещает рабочий лист в другое место рабочей книги. Параметры: Before — лист, перед которым будет размещен перемещаемый лист, нельзя указать Before, если задан After; After — лист, после которого будет размещен перемещаемый лист
Select	Выбирает указанные рабочие листы

В табл. 9 перечислены наиболее часто используемые события объекта Worksheet.

Таблица 9

События объекта Worksheet

Процедура	Событие
Activate	Активизация листа
BeforeDelete	Удаление листа (событие обрабатывается перед реакцией **Excel** по умолчанию)
BeforeDoubleClick	Двойной щелчок мышью на листе (событие обрабатывается перед реакцией **Excel** по умолчанию)
BeforeRightClick	Щелчок правой кнопкой мыши на листе (событие обрабатывается перед реакцией **Excel** по умолчанию)
Calculate	Пересчет значений на рабочем листе
Change	Изменение данных на листе (не происходит, если ячейки изменяются в результате пересчета, а также в результате удаления ячеек)
Deactivate	Лист теряет фокус ввода
FollowHyperlink	Нажатие любой гиперссылки на листе
SelectionChange	Смене выделения на рабочем листе

2.6.4. Объект Range

Вся работа в **Excel VBA** производится с диапазонами ячеек. Объект Range содержится в объекте Worksheet и состоит из одной ячейки или диапазона ячеек на отдельном рабочем листе.

Основные свойства объекта Range описаны в табл. 10.

Таблица 10

Свойства объекта Range

Свойство	Описание
Address	Возвращает адрес диапазона на рабочем листе (параметры определяют, возвращается ли внешняя ссылка, абсолютные или относительные координаты, стиль ссылок)
Borders	Возвращает коллекцию Borders, представляющую границы стиля или диапазона ячеек (в том числе диапазон, определенный как часть условного формата).
Cells	Возвращает ссылку на ячейку (ячейки) диапазона
Column	Возвращает номер первого столбца в диапазоне
CurrentRegion	Возвращает ссылку на текущий диапазон, ограниченный пустыми строками и столбцами
Count	Возвращает количество ячеек диапазона (только для чтения)

Свойство	Описание
Dependents	Возвращает диапазон ячеек, зависящие от исходного диапазона ячеек, т. е. все ячейки, в формулах которых есть ссылки на какие-либо ячейки из исходного диапазона (только чтение)
Formula	Возвращает формулу в формате A1
FormulaLocal	Возвращает неанглоязычные (русифицированные) формулы в формате A1
FormulaR1C1	Возвращает формулу в формате R1C1
FormulaR1C1Local	Возвращает неанглоязычные (русифицированные) формулы в формате R1C1
Name	Возвращает имя диапазона
Offset	Возвращает ссылку на ячейку с заданным смещением
Range	Возвращает ссылку на диапазон в объекте Range
Row	Возвращает номер первой строки в диапазоне
Text	Возвращает значение в виде текстовой строки (только чтение)
Value	Возвращает значение, которое содержит диапазон (если в диапазоне несколько ячеек, то значением свойства является массив, содержащий значения всех ячеек диапазона)

Основные методы объекта Range приведены в табл. 11.

Таблица 11

Методы объекта Range

Метод	Описание
AutoFit	Задает автоматические настройки ширины столбца и высоты строки диапазона
Calculate	Перевычисляет все формулы диапазона
ClearContents	Очищает все значения и формулы диапазона, но оставляет форматы
ClearComments	Очищает комментарии
ClearFormats	Удаляет формулы
ClearNotes	Удаляет примечания
Copy	Копирует значения из диапазона ячеек в другой диапазон или в буфер
Cut	Вырезает значения из диапазона ячеек и помещает их в другой диапазон или в буфер
Insert	Вставляет ячейку или диапазон ячеек
PasteSpecial	Осуществляет специальную вставку из буфера обмена (реализация команды специальной вставки).
Select	Выбирает диапазон
Show	Отображает диапазон

В VBA используются несколько способов задания ссылок на объекты Range, основанных на следующих свойствах:

– свойство Range объекта Worksheet;
– свойство Cells объекта Worksheet;
– свойства Rows и Columns объекта Worksheet;
– свойство ActiveCell объекта Application;
– свойство Selection объекта Application;
– свойство Offset объекта Range;

– свойство `CurrentRegion` объекта `Range`;

– свойство `UsedRange` объекта `Worksheet`.

Синтаксис обобщенной формы обращения к свойству `Range` можно записать следующим образом:

`объект.Range(диапазон1[, диапазон2])`

`объект` — ссылка на объект, например, на рабочий лист или на интервал ячеек (по умолчанию используется активный лист);

`диапазон1`, `диапазон2` — интервалы ячеек произвольного размера.

Примеры:

1) `Worksheets("Лист1").Range("A1").Value = 100` — ввод значения 100 в ячейку `A1` на листе `Лист1` активной рабочей книги[1];

2) `Range("A1:B10") = 10` — ввод значения 10 в диапазон из двадцати ячеек на активном листе;

3) `x = Range("Ячейки ").Rows.Count` — присвоение переменной x значения количества строк в диапазоне с именем "Ячейки";

4) `Range("1:3")` — ссылка на диапазон строк `1:3`;

5) `Range("A:C")` — ссылка на диапазон колонок `A:C`;

6) `Range("A1:C3, B5:D8")` — ссылка на объединение двух несмежных интервалов `A1:C3` и `B5:D8`;

7) `Range("A1:C10 B10:D20")` — ссылка на пересечение двух интервалов `A1:C10` и `B10:D20` (интервал `B10:C10`);

8) `Range("A5","C10")` — ссылка на интервал `A5:D10`.

Другим способом сослаться на диапазон является применение свойства `Cells`. Свойство возвращает единственную ячейку рабочего листа, которая находится на пересечении строки и столбца, задаваемых целыми числами. Синтаксис записи: `объект.Cells(номер строки, номер столбца)`. При этом используется адресация `R1C1`.

Примеры:

1) `ActiveSheet.Cells` — ссылка на все ячейки активного рабочего листа;

2) Range("B2:C5").Cells(2,2) = 10 — ввод числа 10 в ячейку C3;

3) Range(Cells(3,2),Cells(5,4)) — ссылка на интервал ячеек B3:D5.

Свойство `Columns` возвращает объект `Range`, представляющий колонку или коллекцию колонок в объекте, к которому это свойство было применено.

Синтаксис: `объект.Columns(индекс)`,

`объект` — ссылка на объект. Указание необязательно, по умолчанию используется активный рабочий лист;

`индекс` — номер или идентификатор колонки в объекте.

Примеры:

1) `Columns(2)` — ссылка на колонку B активного рабочего листа;

---

[1] Для объекта `Range` свойством по умолчанию является `Value`. Следовательно, выражение `.Value` в этом и приведенном ниже коде можно опустить.

2) `Columns("B")` — ссылка на колонку B активного рабочего листа;

3) `Range("C1:D5").Columns(2)` — ссылка на колонку `D` заданного интервала `C1:D5`;

4) `Range(Columns(1), Columns(5))` — ссылка на интервал, содержащий первые пять столбцов рабочего листа.

Если не указан индекс колонки, то возвращаются ссылки на все колонки объекта `Range`.

Свойство `Rows` возвращает объект `Range`, представляющий строку или коллекцию строк в объекте, к которому это свойство было применено.

Синтаксис: `объект.Rows(индекс)`,

`объект` — ссылка на объект. Указание необязательно, по умолчанию используется активный рабочий лист;

`индекс` — номер строки в объекте.

Примеры:

1) `Rows(2)` — ссылка на строку `2` активного рабочего листа;

2) `Range("C2:D5").Rows(2)` — ссылка на строку `3` заданного интервала `C1:D5`;

3) `x = Selection.Rows(Selection.Rows.Count).Row` — определение номера последней строки в выделенном интервале ячеек.

Если не указан номер строки, то возвращаются все строки объекта в виде объекта `Range`.

Свойство `Offset` позволяет ссылаться на ячейки или интервалы при помощи числа строк и колонок, которые задают смещение относительно исходной ячейки. Например, `Range("A5").Offset(-2,1)` возвращает ячейку `B3`.

Синтаксис `объект.Offset([количество_строк][, количество_столбцов])`

`объект` — ссылка на объект `Range`. Ссылка обязательна и определяет объект, относительно которого задается смещение;

`количество_строк` — смещение строки относительно исходной ячейки;

`количество_столбцов` — смещение столбца относительно исходной ячейки.

Необязательные аргументы `количество_строк` и `количество_столбцов` — числовые выражения. Если какой-то аргумент не задан, то соответствующее смещение равно нулю.

Примеры:

1) Range("A5").Offset(-2,1) — ссылка на ячейку B3;

2) `Selection.Offset(2,1).Select` —выделение интервала `C3:D5` (предполагается, что выделен интервал `B1:C3`).

Свойство `ActiveCell` указывает на ячейку или объект `Range`, которые обладают фокусом ввода. Свойство `Selection` может указывать не только на одну ячейку или диапазон ячеек, но и на другие объекты, например графические.

Примеры:

1) `ActiveCell.Font.Italic = True` — установка начертания курсив для шрифта для активной ячейки;

2) `Selection.Clear` — очистка выделенных ячеек на активном листе (предполагается, что выбран диапазон ячеек).

Текущий регион (`CurrentRegion`) — диапазон ячеек, ограниченный пустыми строками и колонками или сочетанием пустых строк, колонок и границ рабочего листа, содержащий указанную ячейку.

Использованный диапазон (`UsedRange`) — диапазон ячеек, ограниченный левой верхней и правой нижней заполненными ячейками, наиболее удаленными друг от друга. В этом диапазоне содержатся все заполненные ячейки листа, а также расположенные между ними пустые ячейки. На листе может быть только один такой диапазон.

Эти два свойства очень полезны, когда программа работает с диапазонами, размеры которых изменяются динамично и не известны в каждый момент времени.

Примеры:

1) `ActiveCell.CurrentRegion.Select` — выделение диапазона ячеек текущего региона, содержащего активную ячейку;

2) `x = Cells(2, 4).CurrentRegion.Rows.Count` — подсчет в переменой x числа строк текущего региона, содержащего ячейку `D2`;

3) `ActiveSheet.UsedRange.Select` — выделение диапазона всех заполненных ячеек активного рабочего листа.

3. ОСНОВЫ ПРОГРАММИРОВАНИЯ НА VBA

3.1. ОФОРМЛЕНИЕ КОДА

Программный код **VBA** состоит из следующих конструкций.

Оператор — синтаксическая единица языка программирования, которая определяет некоторое действие. Операторы могут быть простыми и составными, т. е. состоять из обязательного набора ключевых слов.

Процедура — это отдельная единица программного кода **VBA**, которую можно вызывать по имени для выполнения; она может выполняться самостоятельно. Любая процедура содержит один или несколько операторов.

Модуль — это именованная единица, состоящая из одной или нескольких процедур и раздела объявлений, в котором объявляются переменные, константы и пользовательские типы данных, а также устанавливаются параметры компилятора.

Проект — включает в себя все модули, формы и связанные с приложением объекты, относящиеся к конкретному документу, причем проект сохраняется вместе с самим этим документом.

3.1.1. Комментарии

Комментарии — это строки в коде, которые исполняют роль пояснений и описаний и помогают разобраться, какие действия выполняет та или иная часть кода.

Комментарии не участвуют в процессе выполнения программы и не влияют на результат работы макроса. Каждая строка, начинающаяся апострофом (`'`), будет считаться в **VBA** комментарием. Редактор **VBA** выделит такую строку зеленым цветом шрифта.

3.1.2. Отступы в коде

Другой прием, делающий написанный код более читаемым, — правильная расстановка отступов. Для каждого вложенного блока кода делается отступ внутри главной процедуры, который увеличивается в зависимости от уровня вложения. Такие увеличенные отступы помогают понять, где каждый отдельный блок кода начинается и заканчивается.

3.1.3. Переносы строк

Еще один способ сделать код более читаемым и облегчить работу с ним — делать переносы и разбивать одну длинную строку кода на несколько коротких. В **VBA**, чтобы разбить строку, нужно вставить символы « _» (пробел+подчеркивание) непосредственно перед переносом строки. Это сообщает компилятору **VBA**, что текущая строка кода продолжается на следующей строке. При этом надо помнить, что нельзя разбивать переносом строковые константы, допустимо не более семи продолжений одной и той же строки, сама строка не может состоять более чем из 1024 символов.

3.1.4. Запись нескольких операторов в одну строку

Иногда для записи блока операторов удобнее их разместить в одной стоке. Для этого операторы следует записать через разделитель двоеточие «`:`».

3.2. ПЕРЕМЕННЫЕ И ТИПЫ ДАННЫХ

Переменная — это именованный фрагмент памяти, выделяемый или резервируемый для хранения данных, которые могут изменяться во время выполнения программы.

Наряду с переменными в **VBA** используются константы. Как и переменной, константе соответствует фрагмент оперативной памяти. Однако, в отличие от переменной, содержимое фрагмента, соответствующего константе, в программе изменить нельзя.

Существуют определенные правила задания имен переменным, процедурам и константам.

В **VBA** следующие правила назначения имен:

– первым символом имени обязательно должна быть буква;
– нельзя использовать символы «`.`», «`!`», «@», «&», «$», «#» и пробел;
– имя может содержать буквы, цифры и знак подчеркивания;
– имя не может содержать более 255 знаков;
– имя не должно совпадать с ключевыми словами **VBA**;
– в одной процедуре не могут быть объявлены две переменные с одним и тем же именем.

Переменная характеризуется именем и типом данных, которые могут храниться в переменной.

Тип данных определяет:

– формат представления данных в памяти компьютера;
– область возможных значений;
– допустимые операции, применимые к данным.

Краткий перечень используемых типов данных **VBA** приведен в табл. 12.

Таблица 12

Типы данных **VBA**

Типы данных	**Размер (байт)**	**Диапазон значений**
`Byte` (байт)	1	От 0 до 255
`Boolean` (логический)	2	`True` или `False`
`Integer` (целое число)	2	От –32 768 до 32 767
`Long` (длинное целое число)	4	От –2 147 483 648 до 2 147 483 647
`Single` (число с плавающей запятой обычной точности)	4	От –3,402823E38 до –1,401298E-45 для отрицательных значений; от 1,401298E-45 до 3,402823E38 для положительных значений
`Double` (число с плавающей запятой двойной точности)	8	От –1,7976931346232E308 до –4,9406565841247E-324 для отрицательных значений; от 4,9406565841247E-324 до

Типы данных	Размер (байт)	Диапазон значений
		1,7976931346232E308 для положительных значений
`Currency` (денежный)	8	От –992 337 203 685 477,5808 до 992 337 203 685 477,5807
`Date` (дата и время)	8	От 1 января 0100 г. до 31 декабря 9999 г. и время от 0:00:00 до 23:59:59
`Object` (объект)	4	Любая ссылка на объект
`String` (строка переменной длины)	1 на символ	От 0 до 65400 символов
`Variant` (вариант)	16 + 1 байт на каждый символ строковых значений	Может использоваться для хранения различных типов данных: даты/времени, чисел с плавающей точкой, целых чисел, строк, объектов. Данные типа `Variant` могут иметь особое значение `Null`, которое означает, что данные отсутствуют, неизвестны или неприменимы
Определяемый пользователем тип (с помощью ключевого слова `Type`)	Размер выделяемой памяти зависит от определения	Назначение зависит от определения. Используется для описания структур данных. Позволяет хранить в одной переменной такого типа множество различных значений разного типа

3.3. ОБЪЯВЛЕНИЕ ПЕРЕМЕННЫХ И КОНСТАНТ

Прежде чем использовать переменную или константу, ее нужно объявить (описать), т. е. задать ее тип и область использования. Осуществляется это операторами описания — неисполняемыми программными инструкциями, в которых определяются имена процедур, констант или переменных и задаются их характеристики.

При объявлении переменной до присваивания ей значения она инициализируется значением по умолчанию:

– текстовые строки — инициализируются пустыми строками;
– числа — значением 0;
– переменные типа `Boolean` — `False`;
– даты — 30 декабря 1899.

Общий вид оператора объявления переменных имеет следующий синтаксис:

`Dim | Public | Private | Static <ИмяПеременной1> [As <type1>] [, …, <ИмяПеременнойN> [As <typeN>]],`

где `<ИмяПеременной1>, …, <ИмяПеременнойN>` — имена описываемых переменных; `<type1>, …, <typeN>` — соответствующие описываемым переменным типы переменных.

Примеры:

1) Dim i As Integer, j As Integer
2) Dim x As Double

Существуют два типа констант — пользовательские и встроенные. Пользовательские константы требуют объявления. Для этого используется оператор вида:

```
[Public | Private] Const <константа> [As <тип>] =
                        <выражение>,
```

в котором <константа> — имя константы; <тип> — один из поддерживаемых типов данных; <выражение> — литерал, другая константа или любое сочетание, которое включает арифметические или логические операторы, за исключением Is.

Примеры:

1) `Const pi as Double = 3.141592654`
2) `Const Message As string = "Завершение работы"`
3) `Const Day As Date = #1/01/2015#`
4) `Const N As Date = #12:30:00#`

В VBA дата и время определяются как значения, заключенные между символами #. Дата всегда определяется в формате месяц/день/год, даже если система настроена на отображение данных в другом формате.

Встроенные константы, относящиеся к объектам **Excel**, начинаются с префикса xl (например, xlPageBreakManual — разрыв страницы), к объектам **VBA** — vb (например, vbTab — вставка символа табуляции). Встроенные константы не требуют объявления.

Как правило, используемые переменные и константы описываются в начале процедур. Группировка объявлений переменных в начале процедуры позволяет при необходимости быстро отыскать эти операторы в программе.

Объявлять переменные в **Excel** не обязательно. По умолчанию все введенные, но не объявленные переменные в **Excel** будут иметь тип Variant и смогут принять как числовое, так и текстовое значение.

Таким образом, в любой момент написания макроса можно использовать новую переменную (даже если она не была объявлена), и **Excel** будет рассматривать ее как переменную типа Variant. Однако есть несколько причин, почему так поступать не следует:

Устанавливаемый по умолчанию тип Variant использует больше памяти, чем другие типы данных. Казалось бы, несколько лишних байт на каждую переменную — не так уж много, но на практике в создаваемых программах могут быть тысячи переменных (особенно при работе с массивами). Поэтому излишняя память, используемая переменными типа Variant, по сравнению с переменными типа Integer или Single может сложиться в значительную сумму. К тому же операции с переменными типа Variant выполняются гораздо медленнее, чем с переменными других типов, соответственно лишняя тысяча переменных типа Variant может значительно замедлить вычисления.

Объявление переменных позволяет выявить попытку присвоения несоответствующих данных.

По этим причинам при написании макроса **VBA** рекомендуется объявлять все переменные. Для включения режима обязательного объявления переменных

служит оператор `Option Explicit`, который позволяет при компиляции выявить все не объявленные переменные.

3.4. ОБЛАСТЬ ДЕЙСТВИЯ ПЕРЕМЕННЫХ И КОНСТАНТ

После объявления переменную можно неоднократно использовать в той части программы, где она будет доступной. Та часть проекта, в рамках которой некоторая переменная является доступной, называется областью действия данной переменной (областью видимости переменной). Любая переменная может использоваться только в области своего действия.

Область действия каждой переменной зависит от двух взаимосвязанных факторов — места объявления этой переменной и указанных при ее объявлении ключевых слов: `Dim, Public, Private, Static`. При этом для областей действия переменных можно выделить три уровня.

3.4.1. Локальные переменные

Локальная переменная — это переменная, используемая без объявления, или объявленная в процедуре с помощью ключевого слова `Dim`.

Локальные переменные могут использоваться только в процедуре, в которой они объявлены. После выполнения процедуры **VBA** освобождает соответствующую область памяти.

Если переменная объявлена как локальная, другие процедуры в том же модуле могут использовать подобное имя, но каждая переменная считается уникальной в своей процедуре.

3.4.2. Переменные уровня модуля

Переменная уровня модуля — это переменная, объявленная в разделе объявлений модуля с помощью ключевых слов `Private` или `Dim`. Переменная будет доступной только в пределах этого модуля. Ключевые слова `Private` и `Dim` в этом случае работают одинаково.

3.4.3. Переменные уровня проекта

Чтобы сделать переменную доступной во всех процедурах всех модулей проекта **VBA**, необходимо объявить переменную в разделе объявлений модуля (перед объявлением первой процедуры) с помощью ключевого слова `Public`.

Этот оператор должен быть размещен в стандартном модуле **VBA**, а не в коде модуля листа, книги или формы.

При наличии в разделе объявлений модуля оператора `Option Private Module` переменные, объявленные с использованием ключевого слова `Public`, будут доступны всем модулям только данного проекта.

3.4.4. Переменные Static

Кроме типа и области видимости, переменная характеризуется временем жизни — периодом, в течение которого она сохраняет свое значение. Переменные, объявленные как `Public`, существуют в течение времени выполнения при-

ложения. Локальные переменные — только во время выполнения процедуры. По завершении процедуры локальные переменные прекращают свое существование, а использовавшаяся ими память возвращается операционной системе. При повторном вызове процедуры переменная создается вновь и инициализируется.

Для сохранения значения переменной между вызовами процедуры используется оператор `Static`.

Переменные `Static` — объявляются на уровне процедуры и сохраняют свои значения после нормального завершения процедуры. Но если выполнение процедуры прерывается с помощью оператора `End`, статические переменные теряют свои значения.

3.4.5. Область действия константы

Как и переменные, константы имеют область действия. Если требуется, чтобы константа была доступна только в одной процедуре, т. е. была локальной, она объявляется внутри процедуры. Сделать константу доступной для всех процедур в модуле можно, если объявить ее в разделе объявлений модуля с помощью ключевого слова `Private`. Чтобы сделать константу доступной для всех модулей рабочей книги, используется ключевое слово `Public` и объявляется константа в разделе объявлений модуля.

3.5. ВЫРАЖЕНИЯ И ОПЕРАЦИИ

Выражение — часть кода, при выполнении которой вычисляется некоторое значение. Выражение может содержать любую комбинацию чисел, символов, констант, переменных, свойств объектов, встроенных функций и процедур типа `Function`, связанных между собой допустимыми знаками операции. При вычислении могут быть получены значения различных типов. Тип результата зависит от типов операндов и операций, которые над ними выполняются.

Операции, реализуемые в программах на **VBA**, делятся на следующие группы: арифметические операции; операции сравнения; логические операции; операция конкатенации.

Ниже в таблицах приведены операции соответствующих групп. Операнды в описании обозначены символами `x` и `y`.

3.5.1. Арифметические операции

Операции, в том числе и арифметические (табл. 13), удобнее рассматривать в табличной форме.

Таблица 13

Арифметические операции

Знак операции	Операция	Описание
+	x + y	Сложение
–	x – y	Вычитание
–	–x	Перемена знака
*	x * y	Умножение
/	x / y	Деление

Знак операции	Операция	Описание
\	x \ y	Целочисленное деление
Mod	x Mod y	Вычисление остатка от деления
^	x ^ y	Возведение в степень

3.5.2. Операции сравнения

Операторы сравнения используются для сравнения числовых и строковых значений переменных, констант и результатов вычисления выражений. В результате выполнения операции сравнения всегда получается значение типа `Boolean`: `True` (Истина), либо `False` (Ложь). В табл. 14 представлены операции отношения, используемые в **VBA**.

Таблица 14

Операции отношения

Знак операции	Операция	Описание
<	x < y	Меньше
>	x > y	Больше
<=	x <= y	Меньше или равно
>=	x >= y	Больше или равно
<>	x <> y	Не равно
=	x = y	Равно
Is	x Is y	Сравнение двух операндов, содержащих ссылки на объекты
Like	x Like y	Сравнение двух строковых выражений

Знаки операции сравнения можно использовать как для чисел, так и для строковых значений. При сравнении строковых значений компилятор **VBA** последовательно сравнивает отдельные символы слева направо, определяя старшинство в соответствии с алфавитным порядком для букв и в соответствии с двоичным значением кода **ASCII** для любых прочих символов. В языке **VBA** при сравнении строк одна строка будет признана равной другой, если обе строки имеют одну и ту же длину и содержат одинаковые символы, расположенные в одинаковом порядке. Результатом такого сравнения будет значение `True`. Если же хоть одно из перечисленных условий не будет выполнено, результатом сравнения строк будет значение `False`.

3.5.3. Логические операции

Основные логические операции, используемые в **VBA**, приведены в табл. 15.

Таблица 15

Логические операции

Знак операции	Операция	Описание
Not	x Not y	Логическое отрицание
And	x And y	Логическое умножение — конъюнкция (логическое И)
Or	x Or y	Логическое сложение — дизъюнкция (логическое ИЛИ)
Xor	x Xor y	Исключающее ИЛИ, возвращает `True`, когда значения `x` и `y` не совпадают

Знак операции	Операция	Описание
Eqv	x Eqv y	Эквивалентность, возвращает True, когда значения x и y одновременно истинны или одновременно ложны
Imp	x Imp y	Импликация (следование), возвращает False, если x = True, а y = False, и значение True — в остальных случаях

3.5.4. Строковая операция

В языке **VBA** для строковых значений используется единственный оператор, выполняющий операцию объединения двух строк в одну. Для задания операции объединения (операции конкатенации, «склеивания» строк) используется знак «&»: x & y. Результат операции всегда имеет тип данных String.

3.5.5. Приоритеты выполнения операций

Если выражение содержит больше одного оператора, то вычисление значения таких выражений регулируется правилами о приоритете выполнения операций, принятыми в языке **VBA**. Порядок выполнения основных операторов следующий:

1) вызов функции;
2) возведение в степень (^);
3) изменение знака (−);
4) умножение и деление (* и /);
5) деление нацело (\);
6) деление по модулю (Mod);
7) сложение и вычитание (+ и −);
8) операции конкатенации (&);
9) операции сравнения (>, <, >=, <=, <>, =);
10) логическое отрицание (Not);
11) логическое умножение (And);
12) логическое сложение (Or);
13) исключающее ИЛИ (Xor);
14) эквивалентность (Equ);
15) импликация (Imp).

Операторы, имеющие одинаковый приоритет, выполняются в выражении последовательно, слева направо. Изменить стандартный порядок выполнения операций в выражениях можно с помощью скобок.

3.5.6. Математические функции

Основные математические функции описаны в табл. 16.

Таблица 16

Математические функции

Функция	Возвращаемое значение
Abs(<число>)	Модуль (абсолютная величина) числа
Atn(<число>)	Арктангенс
Cos(<число>)	Косинус

Функция	Возвращаемое значение
Exp(число)	Экспонента
Log(<число>)	Натуральный логарифм
Rnd(<число>)	Значение равномерно распределенного на интервале [0,1] случайного числа
Sgn(<число>)	Знак числа
Sin(<число>)	Синус
Sqr(<число>)	Квадратный корень из числа
Tan(<число>)	Тангенс
Fix(<число>)	Целая часть числа. Для отрицательного значения аргумента <число> функция возвращает ближайшее отрицательное целое число, большее либо равное указанному
Int(<число>)	Целая часть числа, для отрицательного значения аргумента <число> функция возвращает ближайшее отрицательное целое число, меньшее либо равное указанному

Аргумент <число> — это переменная, константа, принимающие числовые значения, выражение, результатом которого является число, или обыкновенное число.

3.5.7. Функции проверки типов

Функции проверки типов (табл. 17) проверяют, является ли переменная выражением специфицированного типа. Возвращает значение True, при положительном результате проверки, False — в противоположном случае.

Таблица 17

Функции проверки типов

Функция	Вид проверки
IsArray(<переменная>)	Является ли переменная массивом
IsDate(<переменная>)	Является ли переменная датой
IsEmpty(<переменная>)	Была ли переменная описана инструкцией Dim
IsError(<переменная>)	Является ли переменная кодом ошибки
IsNull(<переменная>)	Является ли переменная пустым значением (Null)
IsNumeric(<переменная>)	Является ли переменная числовым значением
IsObject(<переменная>)	Является ли переменная объектом

3.5.8. Функции обработки строк

Представители функций обработки строковых данных описаны в табл. 18.

Таблица 18

Функции обработки строк

Функция	Назначение
Asc(<строка>)	Возвращает ASCII-код начальной буквы строки
Chr(<код>)	Преобразует ASCII-код в строку
LCase(<строка>)	Преобразует строку к нижнему регистру
UCase(<строка>)	Преобразует строку к верхнему регистру

Функция	Назначение
Left(<строка>, <число>) Аргументы: <число> — число символов; <строка> — исходная строка	Возвращает подстроку, состоящую из заданного числа первых символов исходной строки
Right(<строка>, <число>) Аргументы: <число> — число символов; <строка> — исходная строка	Возвращает строку, состоящую из заданного числа последних символов исходной строки
Mid(<строка>, <позиция> [,<число>]) Аргументы: <строка> — строковое выражение, из которого извлекается подстрока; <позиция> — позиция символа в строке, с которого начинается нужная подстрока; <число> — число возвращаемых символов подстроки	Возвращает подстроку строки, содержащую указанное число символов
Len(<строка>)	Возвращает число символов строки
Space(<число>)	Возвращает строку, состоящую из указанного числа пробелов
String(<число>, <символ>) Аргументы: <число> — число повторений символа; <символ> — повторяемый символ	Возвращает строку, состоящую из указанного числа повторений одного и того же символа

3.5.9. Функции преобразования типов

Некоторые функции преобразования типов данных приведены в табл. 19.

Таблица 19

Функции преобразования типов данных

Функция	Назначение функции	Тип результата
Str(N)	Возвращает строку, эквивалентную численному выражению N. В качестве допустимого десятичного разделителя воспринимает только точку. При наличии другого десятичного разделителя (например, запятой) для преобразования чисел в строки следует использовать функцию CStr	String
Val(S)	Возвращает численное значение, соответствующее числу, представленному строкой S, которая должна содержать только цифры и одну десятичную точку, иначе **VBA** не может преобразовать ее в число. Если **VBA** не может преобразовать строку S в число, то функция Val возвращает 0	Variant
Asc(S)	Возвращает число кода символа, соответствующее первой букве строки S. Например, буква «А» имеет код 65	Integer
Chr(N)	Возвращает строку из одного символа, соответствующего коду символа N, который должен быть числом между 0 и 255. Код символа 66 возвращает букву «В»	String

Кроме функций Val и Str, в **VBA** имеется ряд функций преобразования любого строкового или числового выражения к нужному типу данных.

Функция	Тип возвращаемых данных
CBool(<выражение>)	Boolean
CByte(<выражение>)	Byte
CCur(<выражение>)	Currency
CDate(<выражение>)	Date
CDbl(<выражение>)	Double
CDec(<выражение>)	Decimal
CInt(<выражение>)	Integer
CLng(<выражение>)	Long
CSng(<выражение>)	Single
CVar(<выражение>)	Variant
CStr(<выражение>)	String

3.5.10. Функции даты и времени

В **VBA** имеются следующие функции времени и даты (табл. 20).

Таблица 20

Функции времени и даты

Функция	Назначение
Date	Возвращает значение типа Variant (Date), содержащее текущую системную дату
Time	Возвращает значение типа Variant (Date), содержащее текущее время по системным часам компьютера
Now	Возвращает значение типа Variant (Date), содержащее текущую дату и время по системному календарю и часам компьютера
Hour(<время>), Minute(<время>), Second(<время>) <время> — аргумент типа Variant, числовое выражение, строковое выражение или любая комбинация, которая может представлять время	Возвращают значения типа Variant (Integer), содержащее целое число, которое представляет, соответственно, часы, минуты, секунды в значении времени (<время>)
Day(<дата>), Month(<дата>), Year(<дата>) <дата> — аргумент типа Variant, числовое выражение, строковое выражение или любая комбинация, которая может представлять дату	Возвращает значение типа Variant (Integer), содержащее целое число, которое представляет, соответственно, день, месяц, год в значении даты (<дата>)
DateAdd(<интервал>, <число>, <дата>) <интервал> — аргумент строкового типа, интервал времени, который требуется добавить;	Возвращает значение типа Variant (Date), содержащее результат прибавления к дате (<дата>) указанного интервала времени

Функция	Назначение
<число> — числовой аргумент; количество интервалов, которые требуется добавить; <дата> — аргумент типа Variant (Date), дата, к которой добавляется интервал	
DateDiff(<интервал>, <дата1>, <дата2>) <интервал> — аргумент строкового типа, интервал времени, используемый для вычисления разницы значений <дата1> и <дата2>; <дата1> и <дата2> — аргументы типа Variant (Date), даты, которые требуется использовать в расчете	Возвращает значение типа Variant (Long), указывающее на количество интервалов времени между двумя датами

Допустимые значения аргумента <интервал> функций DateAdd и DateDiff приведены в следующей таблице:

Значение аргумента	Интервал
yyyy	Год
q	Квартал
m	Месяц
y	День года
d	День
w	День недели
ww	Неделя
h	Часы
n	Минуты
s	Секунды

3.5.11. Функции выбора

Для осуществления выбора значений в соответствии с заданным условием применяют функции, описанные в табл. 21.

Таблица 21

Функции выбора по условию

Функция	Назначение
IIf(<условие>, <аргумент1>, <аргумент2>) <условие> — проверяемое выражение); <аргумент1> — значение или выражение, возвращаемое, если <условие> имеет значение True;	Возвращает одну из двух значений в зависимости от оценки условия

Функция	Назначение
<аргумент1> — значение или выражение, возвращаемое, если <условие> имеет значение False	
Choose(<индекс>, <аргумент1>[, <аргумент2>, ... [,<аргументN>]]) <индекс> — числовое выражение или поле, значением которого является число, лежащее между 1 и числом элементов в списке; <аргумент> — выражение типа Variant, содержащее один из элементов списка	Возвращает значение, выбранное из списка аргументов на основе значения аргумента <индекс>
Switch(<выражение1>, <значение1>, <выражение2>, <значение2> ... [, <выражениеN>, <значениеN>]]) <выражение> — выражение типа Variant, подлежащее вычислению; <значение> — возвращаемое значение или выражение, если соответствующее выражение принимает значение True	Возвращается значение, соответствующее первому истинному выражению в списке

Чтобы представить числовое значение как дату, время, денежное значение или в специальном формате, следует использовать функцию Format, которая возвращает значение типа Variant (String), содержащее выражение, отформатированное согласно инструкциям, заданным в описании формата.

Синтаксис: Format(<выражение>, [<формат>]).

В инструкции <выражение> — любое допустимое выражение; <формат> — любое допустимое именованное или определенное пользователем выражение формата. Примером именованного формата является Fixed — формат действительного числа с двумя значащими цифрами после десятичной точки.

При построении пользовательского формата возможно использование следующих символов:

0 — резервирует позицию цифрового разряда. Отображает цифру или нуль;

— резервирует позицию цифрового разряда. Отображает цифру или ничего не отображает;

. — резервирует позицию десятичного разделителя;

% — резервирует процентное отображение числа;

: — разделитель часов, минут и секунд в категории форматов Time;

/ — разделитель дня, месяца и года в категории форматов Date;

E+, E-, e+, e- — разделитель мантиссы и порядка в экспоненциальном формате.

3.6. ОПЕРАТОРЫ И УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ

3.6.1. Операторы присваивания

Оператор присваивает значение выражения переменной, константе или свойству объекта. Оператор присваивания всегда включает знак равенства (=).

Синтаксис:

[Let]<переменная> | <константа> | <свойство объекта> = <значение>

Оператор присваивания предписывает присвоить <значение> переменной, имя которой указано в левой части. В качестве <значение> могут быть использованы: любой числовой или строковый литерал, константа, выражение, свойство объекта. Ключевое слово Let необязательное и обычно опускается.

Выражение значения может быть назначено переменной, константе или свойству, только если его тип данных совместим с переменной. Нарушение этого запрета вызовет ошибку во время компиляции.

Для присвоения переменной ссылки на объект применяется инструкция Set. В следующем примере инструкция Set присваивает переменной Область диапазон A1:B3:

```
Set Область = Range("A1:B3")
```

В общем случае инструкция Set имеет следующий синтаксис:

```
Set <ОбъектнаяПеременная> = [New] <ОбъектноеВыражение> | Nothing
```

Ключевое слово New используется при создании нового класса. Ключевое слово Nothing позволяет освободить все системные ресурсы и ресурсы памяти, выделенные для объекта, на который имелась ссылка (удаляет объект из памяти).

3.6.2. Оператор **With**

Оператор позволяет выполнять несколько операций над одним объектом.

Синтаксис оператора:

```
With <объект> <операторы> End With
```

<объект> — имя объекта или пользовательского типа, <операторы> — один или несколько операторов, которые выполняются в отношении объекта.

Допускаются вложение операторов With в другие структуры With. Тем не менее, поскольку элементы внешних блоков With маскируются во внутренних блоках With, во внутреннем блоке With необходимо указывать полную ссылку на любой элемент объекта во внешнем блоке With.

В общем случае не рекомендуется использовать переходы в блоки With или из них. Если выполнены операторы в блоке With, однако оператор With или End With остался невыполненным, после выхода из процедуры временная переменная со ссылкой на объект останется в памяти.

Рассмотрим пример использования оператора With...End With для изменения шрифта и размера шрифта у выделенной ячейки A1.

Пример:

```
Set MyObject = Worksheets("Лист1").Range("A1")
With MyObject
.Value = "Любой текст"
    With .Font
```

```
        .Color = RGB(255, 0, 0)
        .Bold = True
        End With
    End With
```

В примере объектной переменной MyObject присваивается ссылка на ячейку A1 листа Лист1. С помощью оператора With в ячейку заносится строка "Любой текст", для содержимого ячейки устанавливается начертание Bold и красный цвет шрифта.

3.6.3. Оператор безусловного перехода

Оператор безусловного перехода задает переход на указанную строку внутри процедуры. Синтаксис: GoTo <строка>

Обязательный параметр <строка> может быть любой меткой строки или номером строки процедуры.

Для использования оператора безусловного перехода строке, куда должно быть передано управление, необходимо присвоить метку. Метка должна начинаться с буквы и заканчиваться двоеточием. Действие оператора GoTo состоит в передаче управления меченой строке.

3.6.4. Операторы условного перехода

Управляющие конструкции условного перехода (ветвления) позволяют проверить некоторое условие и, в зависимости от результатов этой проверки, выполнить ту или иную группу операторов. Для организации ветвлений в **VBA** используются различные формы оператора ветвления If и оператор выбора Select Case.

3.6.5. Линейная форма записи оператора If

Простейшая форма записи оператора условного перехода выглядит следующим образом:

```
If <условие> Then <оператор1>
```

или

```
If <условие> Then <оператор1> [Else <оператор2>]
```

Здесь <условие> — логическое выражение, которое должно быть вычислено для проверки выполнения условия.

Если вычисленное значение равно True, выполняется <оператор1> (или блок операторов), следующий за ключевым словом Then, до конца строки.

Если вычисленное значение равно False, то будет выполнен <оператор2> следующий за ключевым словом Else.

3.6.6. Блочная форма записи оператора If

В операторе If может выполняться не только один оператор, но и блок последовательно следующих операторов. В этом случае проверяемое условие и выполняемые операторы записываются в отдельных строках.

```
If <условие> Then
<операторы>
End If
```

Если логическое выражение имеет значение `True`, выполняется группа операторов, стоящая после ключевого слова `Then`.

Для более сложных ситуаций, когда на основании некоторого условия необходимо выбрать одну из двух различных последовательностей операторов, используется оператор `If...Then...Else`.

```
If <условие> Then
<операторы1>
Else
<операторы2>
End If
```

Если значение логического выражения равно `True`, то выполняются операторы блока `<операторы1>`, а затем, управление передается оператору, следующему за ключевыми словами `End If`.

Если значение условного выражения равно `False`, выполняются операторы блока `<операторы2>`, расположенные после ключевого слова `Else`, вплоть до ключевых слов `End If`. После этого **VBA** продолжит выполнение программы, начиная с первого оператора, следующего за `End If`.

Иногда приходится делать выбор одного действия из целой группы действий на основе проверки нескольких различных условий. Для этого можно использовать цепочку операторов ветвления `If...Then...ElseIf`:

```
If <условие1> Then
<операторы1>
ElseIf <условие2> Then
<операторы2>
ElseIf <условие3> Then
<операторы3>
...
ElseIf <условиеN> Then
<операторыN>
Else
<операторыElse>
End If
```

Можно использовать сколько угодно частей `ElseIf` в блоке `If`, но они не должны стоять после `Else`. Блоки оператора `If` могут быть вложенными, то есть содержать внутри другие.

При выполнении блока `If` последовательно проверяется значение проверки условий. Если значение является `True`, тогда выполняются операторы, стоящие после `Then`. Если значение равняется `False`, каждое условие `ElseIf` оценивается по порядку. При обнаружении условия со значением `True` выполняются операторы, следующие за относящейся к ним части `Then`. Если ни одно из значений, следующих за `ElseIf`, не равняется `True`, выполняются опера-

торы, следующие за Else. После выполнения операторов, следующих за Then либо Else, выполняются операторы после End If.

3.6.7. Оператор выбора Select Case

Оператор Select Case используется в том случае, когда необходимо проверять одно и то же значение, сравнивая его с различными выражениями.

```
Select Case <выражение>
Case <значение 1>
<операторы 1>
[Case <значение 2>
<операторы 2>]
[...]
[Case <значение N>
<операторы N>]
[Case Else
<операторы Else>]
End Select
```

После исходного оператора Select Case за проверяемым значением <выражение> может следовать произвольное количество операторов проверки условия Case, причем в каждом из них проверяется выполнение различных условий. В любом случае выполняется либо только одна из ветвей Case (та, в которой было обнаружено первое истинное условие), либо ни одной (если все условия оказались не выполненными). После каждого оператора проверки условия Case можно записывать любое количество выполняемых операторов.

Вычисление значения <выражение> осуществляется в начале работы оператора Select Case. Это выражение может возвращать значение любого типа, например логическое, числовое или строковое, затем полученный результат последовательно сравнивается со значениями в операторах Case, начиная с первого <значение 1>. Если результат вычисления <выражение> совпадает со значением выражения <значение 1>, будут выполнены <операторы 1>, после чего выполнение программы продолжится, начиная с оператора, следующего за оператором End Select. Если результат вычисления выражения не совпадает ни с одним из значений, начиная с выражения <значение 1> и заканчивая выражением <значение N>, то будут выполнены <операторы Else>, следующие за ключевым словом Case Else. Ключевые слова Case Else всегда находятся в конце конструкции Select Case и не являются обязательными — они и соответствующие им <операторы Else> могут быть опущены.

<значение*> представляет собой одно или несколько выражений, разделенных запятой[2]. При выполнении оператора проверяется, соответствует ли

[2] Символ «*» означает любой номер элемента последовательности.

хотя бы один из элементов этого списка значению проверяемого выражения. Элементы списка значений могут иметь одну из следующих форм:

– <выражение> — проверяется, совпадает ли значение проверяемого выражения со значением этого выражения;

– <выражение1> To <выражение2> — проверяется, находится ли значение проверяемого выражения в указанном диапазоне значений этих выражений;

– Is <логический_оператор> <выражение> — проверяемое выражение сравнивается с указанным значением с помощью заданного логического оператора (например, условие Is >= 10 считается выполненным, если проверяемое значение не меньше 10).

В одном операторе Case можно указать несколько списков значений, которые разделяются запятыми: Case 0 To 10, 20, 30, Is >100.

3.6.8. Операторы циклов

Для организации циклов язык **VBA** предоставляет несколько операторов, называемых операторами организации циклов или операторами циклов. Некоторые из этих операторов построены таким образом, что всегда выполняются заданное количество раз. К таким операторам можно отнести операторы For...Next и For...Each...Next.

Другие операторы циклов повторяются переменное количество раз в зависимости от выполнения некоторого условия. К таким операторам относятся различные варианты оператора Do...Loop.

3.6.9. Оператор For…Next

В языке **VBA** оператор предназначен для организации повторного выполнения фрагмента кода заданного количества раз, известного до начала выполнения цикла.

Синтаксис оператора For...Next имеет следующий вид:

```
For <счетчик> = <начало> To <конец> [Step <шаг>]
<операторы>
[Exit For]
[<операторы>]
Next [<счетчик>]
```

Здесь <счетчик> — счетчик цикла, любая числовая переменная, в которой сохраняется информация о количестве выполненных проходов цикла.

Параметры <начало> и <конец> — числовые выражения, задающие начальное и конечное значения счетчика и определяющие количество повторений цикла.

Числовая переменная <шаг> задает приращение, на которое увеличивается счетчик цикла при каждом проходе. Шаг может быть как положительным, так и отрицательным числом. Если используется отрицательное приращение, то конечное значение счетчика должно быть меньше начального значения либо

равно ему. Если ключевое слово Step отсутствует, то по умолчанию шаг считается равным единице.

После завершения работы цикла For...Next переменная, которая использовалась в качестве счетчика, получает значение, обязательно превосходящее конечное значение в том случае, если шаг положительный, и меньшее конечного значения, если шаг отрицательный. Если начальное и конечное значения совпадают, тело цикла выполняется один раз.

Ключевые слова Exit For используются для досрочного выхода из цикла.

Допускается вложенность циклов For...Next.

3.6.10. Оператор For Each...Next

Оператор используется при обработке элементов массивов или совокупности однородных объектов. В этой разновидности цикла счетчик отсутствует, а тело цикла выполняется для каждого элемента массива или совокупности объектов.

Синтаксис цикла следующий:

```
For Each <элемент> In <совокупность>
[<операторы>]
[Exit For]
[<операторы>]
Next [<элемент>],
```

где <элемент> — это переменная, используемая для ссылки на элементы семейства объектов; <совокупность> — имя массива или совокупности объектов.

Ключевые слова Exit For используются для досрочного выхода из цикла.

Циклы For Each...Next можно вкладывать в другие циклы For Each…Next. При этом каждый цикл должен иметь уникальный аргумент <элемент>.

Пример:

```
Dim Item As Worksheet
For Each Item In ActiveWorkbook.Worksheets
MsgBox Item.Name
Next Item
```

При выполнении этой процедуры функция MsgBox отображает свойство Name каждого рабочего листа из коллекции Worksheets активной рабочей книги. Функция MsgBox будет выполнена столько раз, сколько листов в активной рабочей книге.

3.6.11. Оператор Do … Loop

Циклы типа Do … Loop используются в тех случаях, когда заранее неизвестно, сколько раз должно быть повторено выполнение блока операторов, составляющего тело цикла. Такой цикл продолжает свою работу до тех пор, пока не будет выполнено определенное условие.

Существует четыре вида циклов `Do … Loop`, которые различаются типом проверяемого условия и местом выполнения этой проверки.

Цикл `Do … Loop` с проверкой условия в начале цикла:

```
Do While … Loop
Do Until … Loop
```

Цикл `Do … Loop` с проверкой условия в конце цикла:

```
Do … While Loop
Do … Until Loop
```

Синтаксис четырех конструкций цикла `Do … Loop` приведен ниже.

```
Do While <условие>
<операторы>
[Exit Do]
Loop
```

Условие проверяется до того, как выполняется группа операторов, образующих тело цикла. Цикл продолжает свою работу, пока это условие выполняется (т. е. имеет значение `True`).

```
Do
<операторы>
[Exit Do]
Loop While <условие>
```

Условие проверяется после того, как операторы, составляющие тело цикла, будут выполнены хотя бы один раз. Цикл продолжает свою работу, пока это условие остается истинным.

```
Do Until <условие>
<операторы>
[Exit Do]
Loop
```

Условие проверяется до того, как выполняется группа операторов, образующих тело цикла. Цикл продолжает свою работу, если это условие еще не выполнено (т. е. имеет значение `False`), и прекращает работу, когда оно становится истинным.

```
Do
<операторы>
[Exit Do]
Loop Until <условие>
```

Условие проверяется после того, как операторы, составляющие тело цикла, будут выполнены хотя бы один раз. Цикл продолжает свою работу, если это условие еще не выполнено, а когда оно станет истинным, цикл прекращает работу.

3.6.12. Массивы

Массив — это набор элементов одинакового типа, имеющих общее имя. Массивы позволяют работать с некоторым набором однотипных данных как с единым целым. В языке **VBA** всякий массив имеет собственное имя и состоит из некоторого количества элементов. Каждый из этих элементов занимает в

массиве определенное место. Поэтому, чтобы найти в массиве нужный элемент, достаточно указать имя этого массива и номер (индекс) этого элемента.

Массив характеризуется размером и размерностью. Размер — это количество элементов в массиве, размерность — количество индексов, необходимых для указания местоположения элемента в массиве. Таким образом, массив представляет собой совокупность однотипных индексированных переменных. Количество используемых индексов массива также может быть различным. Массивы с одним индексом называют одномерными, с двумя — двумерными и т. д. В **VBA** допускается использование до 60 индексов.

Язык **VBA** поддерживает два типа массивов — статические и динамические. Статическими называют такие массивы, размер которых был указан при их объявлении. В этом случае размер массива остается фиксированным на протяжении всего выполнения программы.

В отличие от статических массивов, динамические массивы могут изменять свой размер в процессе выполнения кода программы. Объявлять динамический массив целесообразно в следующих случаях:

– размер массива неизвестен до момента выполнения программы;
– в ходе выполнения программы размер массива может меняться;
– при выполнении программы после завершения использования массива необходимо освободить занимаемую им память.

Как и другие переменные, массивы описываются с помощью инструкций `Dim`, `Static`, `Private` или `Public`.

3.6.13. Статические массивы

Синтаксис оператора объявления статического массива следующий:

```
Dim|Static|Private|Public   <имяМассива>(<размер 1>[, <размер 2>, …]) [As <тип>]
```

Указанные в скобках величины `<размер 1>, <размер 2>, …` задают размеры массива — количество индексов и максимально допустимое значение для каждого конкретного индекса. По умолчанию индексирование элементов массива начинается с нуля. Чтобы изменить это стандартное значение, нужно воспользоваться оператором `Option Base 1`. Значение индекса будет начинаться с единицы.

Другим способом объявления массивов является использование в операторе ключевого слова `To`:

```
Dim|Static|Private|Public <имяМассива>(<нижняя граница1> To <верхняя граница 1>[, <нижняя граница 2> To <верхняя граница 2>, …]) [As <типДанных>]
```

Аргументы `<нижняя граница *>` и `<верхняя граница *>` задают минимальное и максимальное значения соответствующего индекса.

Обычно элементы массива содержат значения одного и того же типа. Если необходимо, чтобы в массиве содержались данные разных типов, при объявлении массива нужно указать тип `Object`.

Примеры:

1) `Dim myArray1(9) As Integer`

определяет одномерный массив с именем myArray1 из 10 элементов, являющихся переменными целого типа;

2)
```
Option Base 1
  Dim myArray1(9) As Integer
```
объявляется одномерный массив myArray1, состоящий из 9 элементов;

3)
```
Dim myArray2 (9, 9) As Variant
```
определяется двумерный массив 10x10 из 100 элементов, являющихся переменными универсального типа Variant;

4)
```
Dim myArray1(1 To 10) As Integer
```
объявляется одномерный массив myArray1, состоящий из 10 элементов;

5)
```
Dim myArray2 (1 To 3, 1 To 5) As String
```
определяется двумерный массив 3x5 из 15 элементов, являющихся переменными типа String;

6)
```
Dim obArray(3) As Object
obArray(0) = "Строка"
obArray(1) = 100
obArray(2) = -1.2345
obArray(3) = True
```
элементы массива с именем obArray, определенного как Object, могут содержать значения разных типов.

3.6.14. Динамические массивы

Динамические массивы создаются с помощью оператора Dim, Private, Public, Static, аналогично статическим массивам, но список размерностей при этом опускается. Размер массива устанавливается перед его использованием в программе с помощью оператора ReDim.

Оператор ReDim имеет следующий синтаксис:

```
ReDim [Preserve] <имя1>(<индексы1>) [As <тип1>] [, <имя2>(<индексы2>) [As <тип2>] ]
```

Здесь Preserve — необязательное ключевое слово, позволяет сохранять данные после переопределения размера массива;

<имя*> — имя существующего массива;

<индексы*> — измерения массива;

<тип*> — любой тип VBA или определенный пользователем тип.

Оператор ReDim можно использовать повторно, изменяя размер массива по мере необходимости.

Примеры:

1)
```
Dim Month() As String
  Redim Month(1 To 30)
  Redim Preserve Month(1 To 31)
```
объявляется динамический массив Month, определяется размер массива в 30 элементов, изменяется размер массива до 31 элемента с сохранением содержимого;

2)
```
Dim Table() As Integer
ReDim Table(2, 14)
ReDim Table(4, 19)
```
объявляется динамический массив `Table`, определяется размер двумерного массива 3x15 в 45 элементов, изменяется размер двумерного массива 5x20.

3.7. ПРОЦЕДУРЫ И ФУНКЦИИ

Процедуры и функции представляют собой относительно самостоятельные фрагменты программы, оформленные особым образом и снабженные именем. Упоминание этого имени в тексте программы называется вызовом процедуры (функции). В отличие от процедуры имя функции выступает также в качестве переменной и используется для возвращения значения в точку вызова функции.

В языке **VBA** существует несколько типов процедур:

– процедура типа `Sub` (процедура-подпрограмма) — это часть программы, которая может выполняться независимо; при этом одна процедура типа Sub может вызывать другую;

– процедура типа `Function` (процедура-функция) — этот класс процедур отличается тем, что в результате выполнения функции всегда вычисляется единственное возвращаемое значение, которое присваивается переменной с именем данной функции;

– процедура типа `Property` (процедура свойств) используется для доступа к свойствам объекта определенного пользователем класса;

– процедура обработки события (`Event procedure`) — это процедура специального назначения, которая выполняется при возникновении некоторого события (с. 23).

Процедуры-подпрограммы (в дальнейшем — процедуры) и процедуры-функции (в дальнейшем — функции) не связаны с конкретным объектом. Они выполняются только тогда, когда вызываются другими процедурами.

Процедуры, как и переменные, должны быть объявлены до их вызова.

Код функции заключается между операторами `Function` и `End Function`, процедуры — между операторами `Sub` и `End Sub`.

3.7.1. Описание и вызов процедур и функций

Описание процедуры:
```
[Private|Public] Sub <имя процедуры> ([<аргументы>])
<инструкции>
[Exit Sub]
[<инструкции>]
End Sub
```
`<аргументы>` — список формальных параметров, значения которых передаются в процедуру или возвращаются из процедуры при ее вызове. Разделителем в списке параметров является запятая. Если параметры у процедуры отсутствуют, то после имени процедуры ставится пустая пара скобок.

Список параметров процедуры (<аргументы>) имеет следующий синтаксис:

[Optional] [ByVal|ByRef] [ParamArray] <ИмяПеременной> [As <тип>]

Optional — ключевое слово, указывающее на то, что параметр не является обязательным. При использовании этого элемента все последующие параметры, которые содержит список <аргументы>, также должны быть необязательными, и их надо описать с помощью ключевого слова Optional. Все параметры, описанные как Optional, должны иметь тип Variant. Не допускается использование ключевого слова Optional для любого из параметров, если указано ключевое слово ParamArray. Если аргумент необязателен, то для него задается значение, принимаемое по умолчанию. Оно передается процедуре, если при ее вызове данный аргумент будет опущен. В этом случае синтаксическая конструкция описания списка параметров процедуры примет вид: Optional <ИмяПеременной> As <тип> = <значение>, где <значение> — это значение данного аргумента, которое будет использоваться по умолчанию. Все необязательные аргументы должны указываться после обязательных, т. е. в конце списка аргументов.

ByVal — ключевое слово, указывающее на то, что этот параметр передается по значению.

ByRef — ключевое слово, указывающее на то, что этот параметр передается по ссылке. Описание ByRef используется по умолчанию.

ParamArray — ключевое слово, которое используется только для последнего элемента в списке <аргументы> для указания того, что конечным параметром является описанный как Optional массив значений типа Variant, поэтому тип аргументов не указывается. Ключевое слово ParamArray позволяет задавать произвольное количество значений-аргументов. Оно может быть использовано с ключевыми словами ByVal, ByRef.

<тип> — тип значений параметра, переданного процедуре. Допустимы значения: Byte, Boolean, Integer, Long, Currency. Single, Double, Data, String, Object, Variant. Если отсутствует ключевое слово Optional, может быть также указан определяемый пользователем тип или объектный тип.

Если указано только имя аргумента без указания его типа, то такому аргументу присваивается тип Variant.

Оператор Exit Sub вызывает выход из процедуры до ее завершения.

В конце процедуры обязательно должен присутствовать оператор End Sub, указывающий компилятору VBA на завершение процедуры.

Описание функции:

```
[Private|Public]Function <имя функции>([<аргументы>])
[As <тип>]
<инструкции>
<имя функции> = <значение>
```

```
[Exit Function]
[<инструкции>]
[<имя функции> = <значение>]
End Function
```

В теле функции обязательно присутствует оператор присвоения имени функции возвращаемого функцией результата (`<имя функции> = <значение>`). В заголовке описывается тип возвращаемого функцией результата. Если этот тип не указан, тип возвращаемого результата будет `Variant`.

3.7.2. Вызов процедур и функций

Чтобы использовать написанную процедуру или функцию, ее нужно вызвать. Процедура может вызываться двумя способами:

```
<имя процедуры> [<параметры>]
```

или

```
Call <имя процедуры>(<параметры>)
```

В первом случае список фактических параметров — `<параметры>` — задается без скобок, во втором использование скобок обязательно. Список фактических параметров должен полностью соответствовать списку формальных параметров. Все фактические параметры должны быть перечислены в том порядке, в каком они присутствуют в описании процедуры.

Вызов функции имеет следующий вид:

```
<имя переменной> = <имя функции>(<параметры>)
```

Список фактических параметров — `<параметры>` — при вызове функции обязательно должен заключаться в скобки.

Возможны два разных способа передачи переменных процедуре или функции: по ссылке и по значению. Если переменная передается по ссылке, то это означает, что процедуре или функции будет передан адрес этой переменной в памяти. При этом происходит отождествление формального аргумента процедуры и переданного ей фактического параметра. Поэтому вызываемая процедура может изменить значение фактического параметра: если будет изменен формальный аргумент процедуры, то это скажется на значении переданного ей при вызове фактического параметра. Если же фактический параметр передается по значению, то формальный аргумент вызываемой процедуры или функции получает только значение фактического параметра, но не саму переменную, используемую в качестве этого параметра. Тем самым все изменения значения формального аргумента не скажутся на значении переменной, являющейся фактическим параметром.

Способ передачи параметров процедуре или функции указывается при описании ее аргументов: имени аргумента может предшествовать явный описатель способа передачи. Описатель `ByRef` определяет передачу по ссылке, а `ByVal` — по значению. По умолчанию подразумевается передача по ссылке. Например, `Call Prog1(a, ByVal b, ByRef c)`. В процедуру `Prog1` параметр `a` передается по ссылке (по умолчанию), `b` — по значению, `c` — по ссылке.

Содержимое ячеек рабочих таблиц, передаваемые в функцию через список параметров, не изменятся после вызова функции, т. е. эти величины передаются в функцию только по значению.

3.7.3. Область действия процедур и функций

Как и переменные, процедуры и функции имеют определенную область действия, в пределах которой данную процедуру или функцию можно вызвать из других программ (процедур или функций). Область действия каждой процедуры и функции определяется указанием при ее объявлении соответствующих ключевых слов. При этом можно выделить три типа области действия процедур.

Процедура, объявленная с помощью ключевого слова `Private`, доступна для всех процедур в данном модуле и недоступна процедурам, находящимся в других модулях.

```
Private Sub <имя процедуры>([<аргументы>])
<инструкции>
End Sub
```

Процедура, объявленная с помощью ключевого слова `Public`, доступна для всех других процедур во всех модулях рабочей книги. Эта область действия процедуры принята по умолчанию, поэтому ключевое слово `Public` использовать при объявлении процедуры не обязательно.

```
Public Sub <имя процедуры>([<аргументы>])
<инструкции>
End Sub
```

`Static` — ключевое слово, указывающее на то, что локальные переменные процедуры сохраняются в промежутках времени между вызовами этой процедуры.

```
Static Sub <имя процедуры>([<аргументы>])
<инструкции>
End Sub
```

Область действия функций задается аналогичным образом.

Процедуры обработки событий начинаются с ключевого слова `Private`. Это означает, что ее область видимости ограничивается модулем. По умолчанию все процедуры обработки событий объявляются с ключевым словом `Private`.

Процедуры, объявленные ключевым словом `Public`, доступны на уровне приложения, такие процедуры называются открытыми или глобальными. При использовании в модуле, содержащем оператор `Option Private Module`, процедура будет недоступна за пределами проекта.

Пользовательские функции не разрешены для выполнения каких-либо действий, кроме возвращения значения в формулу на листе или в выражение, используемое в другом макросе или функции VBA. Например, пользовательские функции не могут изменять размер окон, редактировать формулы в ячейках, а также изменять шрифт, цвет и параметры узора для текста в ячейке.

3.7.4. Создание пользовательских функций

Пользовательские функции выполняют различные вычисления, а не действия. Некоторые операторы (например, предназначенные для выбора и форматирования диапазонов) исключаются из пользовательских функций. Пользовательские функции создаются в стандартном модуле.

Пользовательские функции должны начинаться с оператора `Function` и заканчиваться оператором `End Function`. Помимо названия функции, оператор `Function` обычно включает один или несколько аргументов. Однако можно создать функцию без аргументов. В **Excel** доступно несколько встроенных функций (например, `СЛЧИС` и `ТДАТА`), в которых нет аргументов.

После оператора `Function` указывается один или несколько операторов **VBA**, которые выполняют действия с использованием аргументов, переданных функции. Наконец, в процедуру функции следует включить оператор, назначающий значение переменной с тем же именем, что у функции. Это значение возвращается в формулу, которая вызывает функцию. Единственное действие, которое может выполнять процедура функции (кроме вычислений), — это отображение диалогового окна. Чтобы получить данные от пользователя, выполняющего функцию, можно использовать в ней оператор `InputBox`. Допускается выводить из функции сведения для пользователей оператором `MsgBox`. В функции можно использовать настраиваемые диалоговые окна (`UserForms`) (с. 70).

Для использования настраиваемой функции необходимо открыть книгу, содержащую модуль, в котором она была создана. Функция будет доступна при использовании обращения к ней в коде программы или включении функции в формулу в ячейке на рабочем листе. Если книга с модулем, содержащим функцию, не открыта, появится сообщение `#NAME?` — ошибка при попытке использования функции. Если необходимо сослаться на функцию в другой книге, перед именем функции следует указать имя книги, в которой она находится.

Например, если создана функция с именем "`NameFunction`" и параметром "`Argument`" в книге с именем "`Book1.xlsm`" (рис. 18), и требуется вызвать ее из другой книги, необходимо ввести `= Book1.xlsm!NameFunction(Argument)`, а не просто `= NameFunction(Argument)`.

Чтобы вставить пользовательскую функцию в ячейку рабочего листа быстрее (и избежать ошибок), ее можно выбрать в диалоговом окне **Мастер функций** (**Формулы → Вставить функцию** или комбинация горячих клавиш **SHIFT+F3**). Пользовательские функции доступны в категории **Определенные пользователем** (рис. 19).

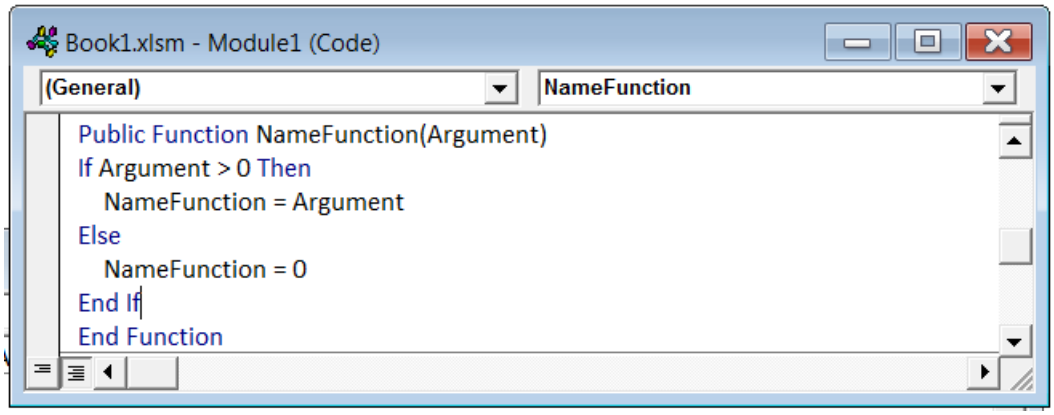

Рис. 18. Код функции `NameFunction` в окне VBE

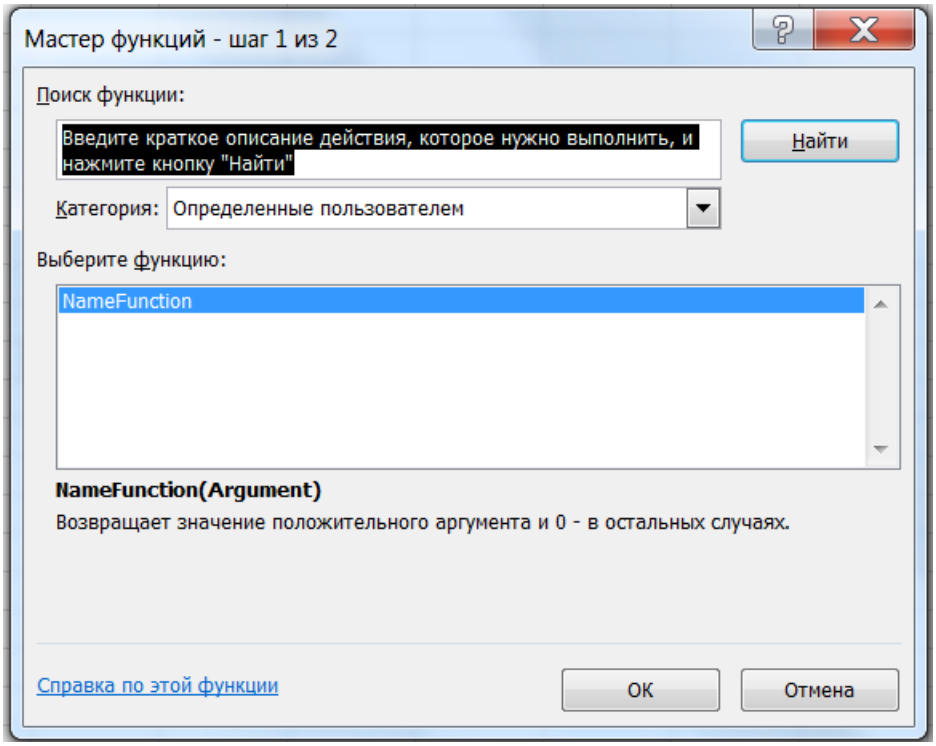

Рис. 19. Выбор пользовательской функции `NameFunction` в окне «Мастер функций»

Встроенные функции Excel содержат пояснения возвращаемого результата и аргументов, которые они принимают. Это можно увидеть на примере любой функции в окне **Мастер функций**. Задокументировать пользовательскую функцию позволяет метод `MacroOptions` объекта `Application`. Для этого необходимо создать процедуру в обыкновенном модуле, как показано на рис. 20, указав требуемое количество переменных (в зависимости от числа аргументов пользовательской функции).

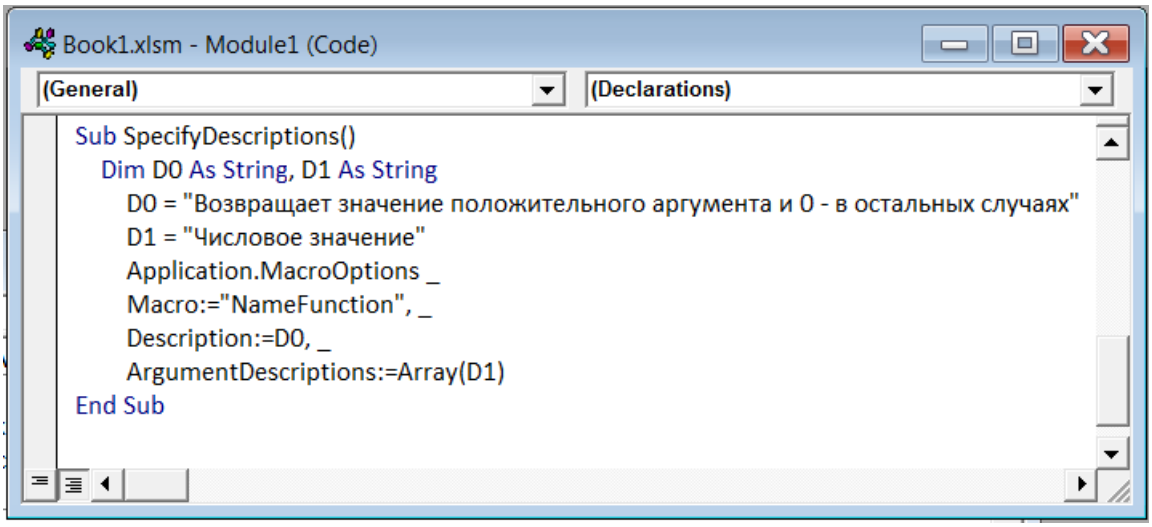

Рис. 20. Процедура создания описания функции пользователя

В качестве параметра `Macro` должна быть передана текстовая строка с названием пользовательской функции, в качестве `Description` — переменная типа `String` с текстом описания возвращаемого значения, в качестве `ArgumentDescriptions` — массив переменных типа `String` с текстами описаний аргументов пользовательской функции.

Для создания описания пользовательской функции достаточно один раз выполнить созданную выше процедуру. После этого при вызове пользовательской функции отображается описание возвращаемого результата и переменной (рис. 21).

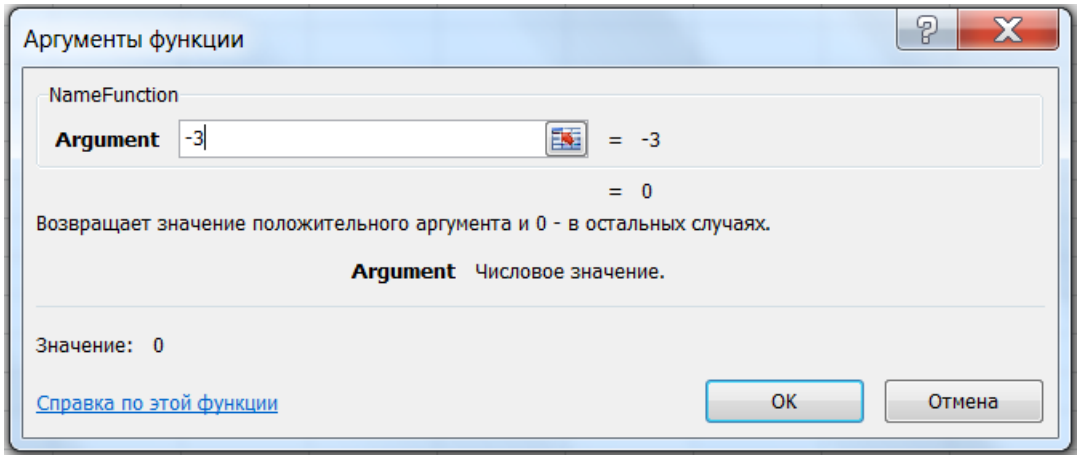

Рис. 21. Окно задания параметров функции

Для организации простого доступа к пользовательским функциям при работе с любой рабочей книгой Excel рекомендуется собрать все пользовательские функции в отдельной книге, а затем сохранить ее в виде надстройки. После этого сделать надстройку доступной при запуске Excel.

Чтобы сохранить открытую рабочую книгу в виде надстройки, в диалоговом окне **Сохранение документа** следует в раскрывающемся списке **Тип файла** выбрать значение **Надстройка Excel**. Книга будет сохранена с заданным

именем и расширением `.xlam` в папке AddIns, которая будет автоматически предложена в диалоговом окне по умолчанию.

Для подключения файла с сохраненными функциями к любой открытой рабочей книге надо выполнить команду **Файл → Параметры → Надстройки**.

В панели **Надстройки** в списке **Управление:** выбрать **Надстройки Excel** и щелкнуть по кнопке **Перейти**.

В диалоговом окне **Надстройки** установить флажок рядом с именем подключаемой книги (рис. 22).

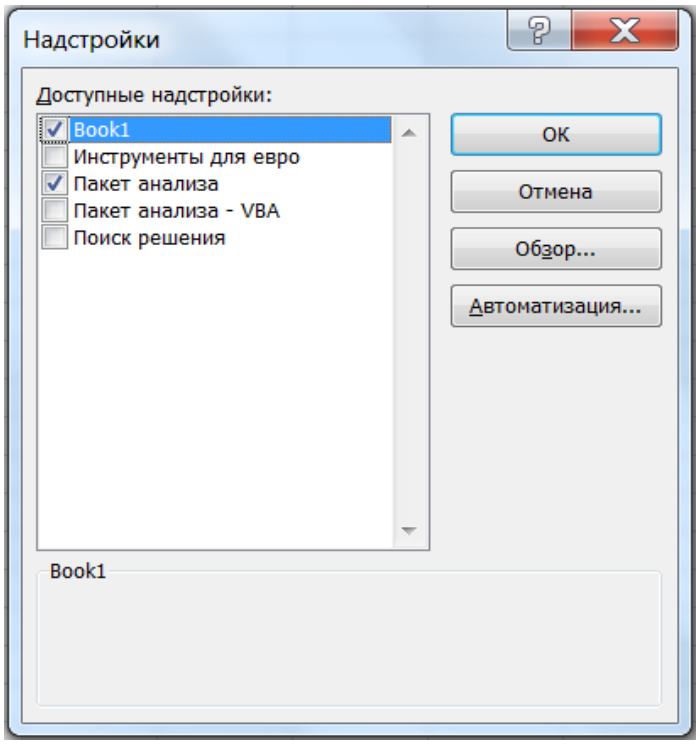

Рис. 22. Подключение книги в окне Надстройки

После выполнения этих действий пользовательские функции будут доступны при каждом запуске Excel. В окне проекта VBE рабочей книги как проект будет отображаться имя книги с процедурами и функциями пользователя (рис. 23).

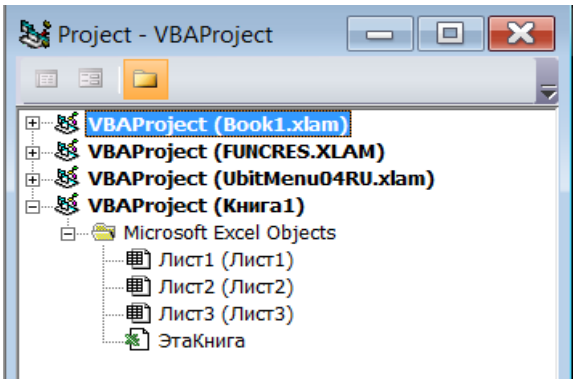

Рис. 23. Фрагмент окна проекта с подключенным файлом Book1.xlam

С проектом «надстройка» можно работать как с проектом «рабочая книга», т. е. добавлять в его модули новые функции, которые будут всегда доступны в категории **Определенные пользователем** диалогового окна **Мастер функций**.

4. ИНСТРУМЕНТЫ ОРГАНИЗАЦИИ ИНТЕРФЕЙСА ПРИЛОЖЕНИЙ

4.1. ВСТРОЕННЫЕ ДИАЛОГОВЫЕ ОКНА

4.1.1. Окно сообщения

Окно сообщения — это диалоговое окно, содержащее сообщение и от одной до четырех кнопок. Для вывода окна сообщения на экран используется функция `MsgBox`. При активизации окна устанавливается режим ожидания нажатия кнопки (одной из кнопок). После нажатия кнопки функция возвращает значение типа Integer, указывающее, какая кнопка была нажата. Значение можно присвоить переменной, либо использовать функцию MsgBox без оператора присваивания для отображения сообщения.

Синтаксис функции создания окна сообщения:

`MsgBox(<Сообщение>[, <Атрибуты>][, <Заголовок>])`

`<Сообщение>` — обязательный аргумент. Строковое выражение, отображающееся в диалоговом окне. Максимальная длина строки аргумента сообщение составляет приблизительно 1024 знака и зависит от их ширины. Если аргумент сообщение содержит несколько строк, их можно разделить с помощью знака возврата каретки (`Chr(13)`), знака перевода строки (`Chr(10)`) или сочетания этих знаков (`Chr(13) & Chr(10)`).

`<Заголовок>` — строковое выражение, отображаемое в заголовке диалогового окна. Если аргумент не указан, в заголовке выводится имя приложения.

`<Атрибуты>` — числовое выражение, являющееся суммой значений, определяющих количество и тип отображаемых кнопок, стиль значков, выбранной по умолчанию кнопки и модальность окна сообщения. Если аргумент пропущен, по умолчанию используется значение 0. Значения констант, определяющих число, тип кнопок и тип используемого значка, приведены в табл. 22 – 25. При определении значения аргумента `<Атрибуты>` следует суммировать не более одного значения из каждой таблицы.

Таблица 22

Константы и их значения, определяющие число и названия кнопок окна сообщений

Константа	Значение	Отображаются кнопки
vbOKOnly	0	**ОК**
VbOKCancel	1	**ОК, Отмена (Cancel)**
VbAbortRetryIgnore	2	**Прервать (Abort) , Повторить (Retry), Пропустить (Ignore)**
VbYesNoCancel	3	**Да (Yes), Нет (No), Отмена (Cancel)**
VbYesNo	4	**Да (Yes), Нет (No)**
VbRetryCancel	5	**Повторить (Retry), Отмена (Cancel)**

Таблица 23

Константы и их значения, определяющие вид значка в окне сообщения

Константа	Значение	Вид сообщения	Значок сообщения
VbCritical	16	Критическое сообщение	
VbQuestion	32	Запрос с предупреждением	
VbExclamation	48	Сообщение с предупреждением	
VbInformation	64	Информационное сообщение	

Таблица 24

Константы и их значения, определяющие номер активной кнопки по умолчанию в окне сообщения

Константа	Значение	Активная кнопка
VbDefaultButton1	0	1
VbDefaultButton2	256	2
VbDefaultButton3	512	3
VbDefaultButton4	768	4

Таблица 25

Константы и их значения, определяющие вид модальности окна сообщения

Константа	Значение	Модальность окна
vbApplicationModal	0	Локальная модальность (на уровне приложения)
vbSystemModal	4096	Глобальная модальность (на уровне системы)

При нажатии любой кнопки в окне сообщения функция `MsgBox` возвращает значение, обработку которого можно реализовать в последующем коде программы как реакцию на нажатие кнопки. Эти значения представлены в табл. 26.

Таблица 26

Константы и их значения, определяющие нажатую кнопку

Константа	Значение	Нажатая кнопка
vbOK	1	**OK**
VbCancel	2	**Отмена (Cancel)**
VbAbort	3	**Прервать (Abort)**
VbRetry	4	**Повторить (Retry)**
vbIgnore	5	**Пропустить (Ignore)**
VbYes	6	**Да (Yes)**
VbNo	7	**Нет (No)**

Пример:

```
a = MsgBox("Сегодня " & Format(Date, "dddd") & vbCr & _
"или " & Format(Date + 1, "dddd") & "?", vbYesNo + _
vbQuestion, "Проверка")
```

Оператор примера позволяет вывести окно сообщения вида «Запрос с предупреждением» с заголовком «Проверка», кнопками **Да**, **Нет** и с сообщением-вопросом о сегодняшнем дне недели. Строка сообщения помимо текстовых символов содержит значение текущей даты (функция `Date`) в формате полного названия дня недели (`dddd`), константу с кодом разрыва строки (`vbCr`), значение завтрашней даты (функция `Date + 1`) в формате полного названия дня недели (рис. 24).

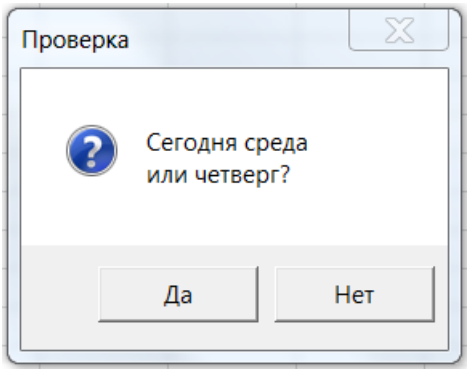

Рис. 24. Пример окна сообщения

При нажатии кнопки **Да** переменной `c` будет присвоено значение 6, **Нет** — 7.

4.1.2. Окно ввода данных

Окно ввода данных — это простое диалоговое окно, которое позволяет пользователю ввести одно значение.

Функция `InputBox` применяется для ввода чисел или текста, выводит на экран диалоговое окно, содержащее сообщение, поле ввода и две кнопки **OK** и **Cancel**. Устанавливает режим ожидания ввода текста пользователем и нажатия кнопки, а затем, при нажатии кнопки **OK**, возвращает значение типа `String`, содержащее текст, введенный в поле ввода. При нажатии кнопки **Cancel** возвращает пустую строку.

Синтаксис функции:

```
InputBox(<Сообщение>[, <Заголовок>][, <Значение>])
```

(`<Сообщение>` — строка-приглашение для ввода, единственный обязательный аргумент, служащий подсказкой пользователю, какую информацию необходимо ввести в поле ввода. Максимальная длина параметра составляет примерно 1024 знаков, в зависимости от используемых символов. Если запрос состоит из нескольких строк, можно разделить строки с помощью символа возврата каретки (`Chr(13)`), символа перевода строки (`Chr(10)`) или сочетания этих символов (`Chr(13) & (Chr(10)`).

`<Заголовок>` — строка, задающая заголовок диалогового окна. Если аргумент не указан, в строку заголовка помещается имя приложения.

`<Значение>` — строка, отображаемая в текстовом поле в качестве значения по умолчанию, если ввод не будет выполнен. Если этот аргумент опустить, то поле ввода отображается пустым.

Пример:

```
b = InputBox("Ваше имя? ", "Вопрос", "Иван")
```

Оператор примера позволяет вывести окно ввода данных с заголовком «Вопрос», кнопками **ОК**, **Cancel** и с приглашением для ввода «Ваше имя?». В функции ввода используется аргумент задания умалчиваемого значения — «Иван» (рис. 25).

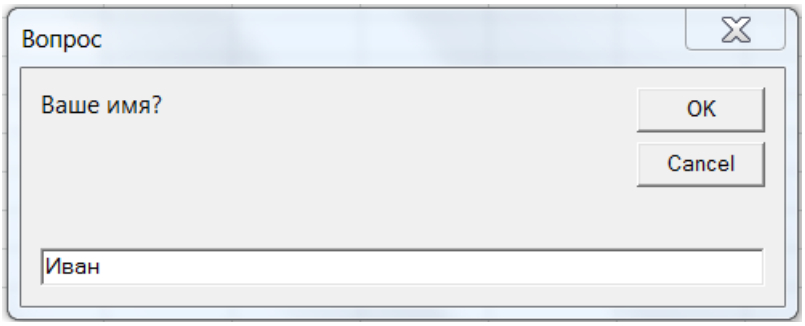

Рис. 25. Пример окна ввода данных

При нажатии кнопки **ОК** переменной `b` будет присвоено значение, находящееся в строке ввода окна, **Cancel** — пустая строка.

4.2. ПОЛЬЗОВАТЕЛЬСКИЕ ФОРМЫ

Пользовательская форма — это диалоговое окно с различными элементами управления.

Для создания формы пользователя обычно выполняют следующие действия:

– вставка новой формы в проект `VBAProject` рабочей книги;
– добавление элементов управления в форму;
– настройка свойств добавленных элементов управления;
– создание в модуле формы процедур обработки событий для элементов управления;
– разработка в обычном модуле процедуры отображения формы;
– реализация способа вызова процедуры отображения формы.

4.2.1. Вставка новой формы в проект

Чтобы добавить новую форму в проект, нужно выполнить команду **Insert (Вставка) → UserForm (Форма)** или щелкнуть кнопке на панели инструментов Стандартная (см. рис. 3).

Для открытия уже созданной формы необходимо выполнить двойной щелчок мышью на ее имени в окне проектов. На экран будет выведено окно с требуемой формой и плавающая панель инструментов **Toolbox**.

Размер формы можно изменять, используя маркеры изменения размера.

В проекте может быть произвольное количество форм, они образуют коллекцию `UserForms`. Как и все коллекции, `UserForms` имеет свойства: `Count` (возвращает число компонентов в семействе), `Item` (возвращает определенный компонент семейства), `Add` (добавляет к семейству новый компонент).

В окне **Properties (Свойства)** представлены свойства формы.

Рассмотрим наиболее важные свойства форм (они применимы и для других элементов управления, за исключением `ShowModal`).

`Name` — определяет имя формы. Имя формы используется для ссылки на нее в программном коде. После создания формы имя `UserFormN` ей присваивается по умолчанию. `N` — это номер формы, присваиваемый по порядку.

`Caption` — задает заголовок формы (по умолчанию совпадает с именем формы).

`Enabled` — устанавливает режим доступности формы: значение `False` закрывает доступ к форме.

`ShowModal` — включает и отключает режим модальности окна формы: значение `False` запрещает передачу фокуса ввода другим объектам из окна формы, пока она не будет закрыта, `True` (по умолчанию) — разрешает.

`BackColor` — возвращает цвет фона формы.

`BorderStyle` — устанавливает тип границ.

`BorderColor` — возвращает цвет границ.

`Picture` — устанавливает рисунок фона формы.

Другие основные свойства относятся к внешнему виду, размерам и местонахождению формы.

Форма имеет методы, позволяющие выполнять с ней различные операции. Ниже описаны наиболее часто используемые методы объекта `UserForm`.

`Show` — отображает объект `UserForm`. Если форма не загружена, то она автоматически загружается. Предварительно форму можно загрузить оператором `Load`, который только загружает объект, но не показывает его.

`Hide` — скрывает объект, но не выгружает его. Выгрузить форму можно оператором `Unload`.

`PrintForm` — печатает изображение формы.

`Move` — изменяет положение и размер формы.

Основные события объекта UserForm приведены в табл. 27.

Таблица 27

События объекта UserForm

Событие	**Описание**
`Activate`	Каждая активизация окна формы
`Initialize`	Первая загрузка формы в память посредством выполнения оператора Load или методом Show
`Click`	Одиночный щелчок мыши
`DblClick`	Двойной щелчок мыши

Событие	Описание
`Error`	Возникновение ошибки в форме
`Resize`	Изменение размеров формы
`Terminate`	Выгрузка формы из памяти

4.2.2. Добавление элементов управления в форму

Элементы управления — это специализированные объекты формы, которые используются для организации интерфейса пользователя. Они создаются в пользовательской форме с помощью панели элементов управления (см. рис. 2).

Панель элементов управления в редакторе VBE вызывается командой меню **View → Toolbox** или щелчком мышью по соответствующей кнопке на панели инструментов **Стандартная**.

В составе панели элементов управления содержатся основные элементы управления форм: метки, текстовые поля, кнопки и другие элементы для быстрого визуального проектирования макета формы.

Чтобы получить доступ к другим элементам управления, необходимо в контекстно-зависимом меню выбрать команду **Additional Controls** (Дополнительные элементы управления). Список элементов управления приведен в табл. 28.

Таблица 28

Элементы управления UserForm

Элемент управления	Назначение
`Select Objects` (Выбор объекта)	Используется для позиционирования маркера (указателя) мыши
`Label` (Метка)	Размещает в форме объекты, предназначенные для создания текстовой информации, надписей и примечаний
`TextBox` (Текстовое поле)	Размещает в форме текстовое поле, предназначенное для ввода и вывода текстовой информации, чисел и дат
`CommandButton` (Командная кнопка)	Размещает в форме кнопки управления для выполнения команд, запуска программ
`ComboBox` (Поле со списком)	Создает в форме объект, содержащий одновременно поле ввода и раскрывающийся список
`ListBox` (Список)	Создает в форме список для выбора одного или нескольких значений из предлагаемого списка
`OptionButton` (Переключатель)	Создает в форме переключатели для выбора режима работы или настроек выполнения программы
`CheckBox` (Флажок)	Размещает в форме флажок, предназначенный для формирования условий, работающих по принципу «да» – «нет»
`ToggleButton` (Выключатель)	Указывает на состояние (да/нет) или режим (вкл./вкл.). При нажатии кнопки меняет свое состояние на противоположное
`Frame` (Рамка)	Создает в форме рамку с заголовком для логической группировки объектов
`ScrollBar` (Полоса прокрутки)	Используется для прокрутки диапазона значений с помощью кнопок со стрелками или путем перетаскивания ползунка полосы прокрутки

Элемент управления	Назначение
SpinButton (Счетчик)	Позволяет увеличивать или уменьшать числовое значение, время или дату на заданное значение
Image (Рисунок)	Размещает в форме графические элементы и анимацию
TabStrip (Набор вкладок)	Создает диалоговое окно с несколькими вкладками
MultiPage (Набор страниц)	Создает диалоговое окно с несколькими страницами

Добавленному элементу управления назначается имя, которое состоит из названия типа элемента управления и порядкового номера элемента.

Удаление выделенного элемента из формы осуществляется нажатием клавиши **Delete**.

Каждый элемент управления характеризуется набором параметров, которые определяют внешний вид и поведение элемента управления. Свойства элемента управления можно изменять следующими способами:

1) в окне **Properties** (Свойства) при разработке пользовательской формы;
2) в процессе выполнения программы посредством операторов VBA.

4.2.3. Использование элементов управления формы

Элемент управления Label (Надпись) используется для отображения надписей, например, заголовков элементов управления, которые поясняют назначение этих элементов.

Основные свойства надписи приведены в табл. 29.

Таблица 29

Свойства элемента управления Label

Свойство	Описание
Caption	Возвращает текст, отображаемый в надписи
Visible	Устанавливает режим отображения: значение True — надпись отображается в форме, False — в противном случае
WordWrap	Устанавливает режим автоматического переноса слов: значение True — разрешается, False — в противном случае
TextAlign	Возвращает параметры выравнивания текста
Font	Возвращает параметры шрифта, его размер, начертание
AutoSize	Устанавливает режим автоматического подбора размера для вводимого текста: значение True — размер изменяется, False — размер фиксированный
ControlTipText	Устанавливает подсказку, при наведении указателя мыши на элемент управления
ForeColor	Возвращает цвет текста
BackColor	Возвращает цвет фона

Элемент управления `TextBox` (Текстовое поле) — используется для ввода и вывода данных. Основные свойства текстового поля приведены в табл. 30.

Таблица 30

Свойства элемента управления `TextBox`

Свойство	Описание
`Text`	Возвращает текст, содержащийся в поле
`Multiline`	Устанавливает многострочный режим ввода текста в поле: значение True — устанавливает, False — отменяет
`ScrollBars`	Устанавливает режим отображения в поле полос прокрутки: значение True — устанавливает, False — отменяет
`SelStart`	Задает позицию фокуса ввода в тексте поля. Крайней левой позиции соответствует 0. Число, равное или больше, чем число символов в текстовом поле, задает позицию за последним символом
`SelLenght`	Задает ширину выделенного текста в поле
`SelText`	Задает текст, замещающий выделенную строку в текстовом поле, если замена не была осуществлена пользователем
`MaxLength`	Устанавливает максимально допустимое количество вводимых в поле символов. Если это свойство равно 0, то нет ограничений на вводимое количество символов
`PasswordChar`	Устанавливает символ, отображаемый при вводе пароля. Если это свойство определено, то вместо вводимых символов в поле будет отображаться установленный символ

Элемент управления `CommandButton` (Командная кнопка) в основном используется для инициирования выполнения действий, вызываемых нажатием кнопки, например запуск программы или остановка ее выполнения, печать результатов и т. п.

Наиболее важные свойства надписи приведены в табл. 31.

Таблица 31

Свойства элемента управления `CommandButton`

Свойство	Описание
`AutoSize`	Устанавливает режим автоматического подбора размера для задаваемого названия кнопки: значение True — размер изменяется, False — размер фиксированный
`BackColor`	Задает цвет фона
`BackStyle`	Устанавливает прозрачность фона. fmBackStyleTransparent — прозрачный фон, fmBackStyleOpaque — непрозрачный
`Caption`	Задает строку текста, отображаемую на элементе управления
`ControlTipText`	Возвращает текст всплывающей подсказки
`ForeColor`	Задает цвет шрифта
`Picture`	Задает ссылку на файл с растровым изображением, используемым в качестве фона элемента управления
`Visible`	Устанавливает режим отображения: значение True — кнопка отображается в форме, False — в противном случае
`WordWrap`	Устанавливает режим автоматического переноса слов в названии: значение True — разрешается, False — в противном случае

Элемент управления OptionButton (Переключатель) позволяет выбрать одну из нескольких взаимоисключающих альтернатив. Переключатели обычно отображаются группами по выбираемым альтернативам. Группировка производится при помощи элемента управления Рамка или свойства GroupName объекта OptionButton. Основными событиями переключателя являются события Click и Change. Основным свойством переключателя является свойство Value, возвращающее или устанавливающее его состояние. Если значение этого свойства равно True, то переключатель установлен, а если False — то сброшен.

Элементы CheckBox (Флажок) и ToggleButton (Выключатель) предоставляют пользователю возможность выбора. Основным свойством этих элементов управления является свойство Value, возвращающее их состояние. Эти элементы управления обычно имеют два состояния: установлен (значение свойства Value равно True), сброшен (значение свойства Value равно False).

Элемент управления ScrollBar (Полоса прокрутки) применяется для установки числового значения, причем этот элемент может устанавливать только целые неотрицательные значения.

Основным событием элемента управления ScrollBar является Change, а основными свойствами — свойства Value, Min, Max и SmallChange, устанавливающие соответственно текущее, минимальное, максимальное значения и величину шага изменения значения при щелчке по одной из стрелок полосы прокрутки.

Элемент управления SpinButton (Счетчик) по своим функциональным возможностям аналогичен полосе прокрутки, но у него нет ползунка. Элемент управления SpinButton (Счетчик) имеет свойства Value, Min, Max, аналогичные свойствам ScrollBar, но свойство Value не может отображать значение этого свойства.

В большинстве случаев требуется, чтобы пользователь мог изменить значение элемента управления SpinButton непосредственно, а не многократно щелкая на элементе управления. Эффективным решением может стать объединение элемента управления SpinButton с элементом управления TextBox, что позволит пользователю вводить значение элемента управления SpinButton непосредственно, используя для этого текстовое поле TextBox. А щелчок на элементе управления SpinButton позволит изменить значение, отображаемое в элементе управления TextBox.

Элемент управления ListBox (Список) позволяет выбрать из предлагаемого множества вариантов одно или несколько значений, которые в последующем используются в тексте программы.

Существует три типа списков.

– простой список поддерживает выбор только одного элемента. Такой список напоминает группу переключателей, но позволяет более эффективно работать с большим числом элементов;

– список связанного выбора позволяет выбрать один элемент, а также несколько расположенных рядом элементов;

– список, разрешающий несвязный выбор нескольких строк, позволяет выбрать один элемент, расположенные рядом элементы, а также разрозненные элементы.

Наиболее часто используемые свойства элемента управления `ListBox` приведены в табл. 32.

Таблица 32

Свойства элемента управления `ListBox`

Свойство	**Назначение**
`ListIndex`	Возвращает номер текущего элемента списка. Нумерация элементов списка начинается с нуля
`ListCount`	Возвращает количество элементов списка
`TopIndex`	Возвращает элемент списка с наибольшим номером
`ColumnCount`	Устанавливает число столбцов в списке
`TextColumn`	Устанавливает столбец в списке, элемент которого возвращается свойством `Text`
`Enabled`	Устанавливает режим доступа: значение `True` — запрещен выбор значения из списка, `False` — в противном случае
`Text`	Возвращает выбранный в списке элемент
`List`	Возвращает элемент списка, стоящий на пересечении указанных строки и столбца. Параметры: `Row`, `Column`
`RowSource`	Устанавливает диапазон, содержащий элементы списка
`ControlSource`	Устанавливает диапазон (ячейку), куда возвращается выбранный элемент из списка
`MultiSelect`	Устанавливает способ выбора элементов списка. Допустимые значения: 0 (`fmMultiSelectSingle`) — разрешен выбор только одного элемента; 1 (`fmMultiSelectMulti`) — разрешен выбор нескольких элементов; 2 (`fmMultiSelectExtended`) — разрешено использование клавиши Shift при выборе ряда последовательных элементов списка
`Selected`	Используется для определения выделенного текста, когда свойство `MultiSelect` имеет значение `fmMultiSelectMulti` или `fmMultiSelectExtended`: значения `True` — элемент списка выбран, `False` — в противном случае
`ColumnWidths`	Устанавливает ширину столбцов списка
`ColumnHeads`	Устанавливает режим вывода заголовков столбцов списка: значение `True` — заголовки выводятся, `False` — в противном случае
`ListStyle`	Устанавливает режим выбора элементов списка: значение `fmListStylePlain` — выбранный элемент выделяется цветом, `fmListStyleOption` — выбор элемента осуществляется установкой флажка

Свойство	Назначение
BoundColumn	Устанавливает тип, возвращаемый свойством Value: значение 0 — свойство Value возвращает индекс выбранной строки, от 1 до количества столбцов в списке — свойство Value возвращает элемент из выбранной строки, стоящий в столбце, определенном свойством BoundColumn

Наиболее часто используемые методы элемента управления ListBox: Clear — удаляет все элементы из списка, RemoveItem(index) — удаляет из списка элемент с указанным номером (index — номер элемента), AddItem([item [, varIndex]]) — добавляет элемент в список (item — элемент, добавляемый в список, varIndex — номер добавляемого элемента).

При работе с элементом управления ListBox следует обратить внимание на ряд особенностей:

– элементы в список могут быть включены из диапазона ячеек (определяемого свойством RowSource) или добавляться с использованием метода AddItem);

– в списке можно выделить один или несколько элементов (определяется значением свойства MultiSelect);

– если элемент управления ListBox не настроен на выделение нескольких элементов, то значение списка можно связывать с ячейкой листа с помощью свойства ControlSourse;

– список может отображаться без предварительно выбранного элемента, если свойство ListIndex равно –1;

– список может включать несколько столбцов (определяется свойством ColumnCount), заголовки столбцов (устанавливается свойством ColumnHeads);

– элементы списка могут включать флажки, если разрешено выделение нескольких элементов, или в виде переключателей, если поддерживается выделение одного элемента (определяется свойством ListStyle).

Примеры:

1) ListBox.RowSource = ”Лист1!A1:A5”

Пять элементов списка извлекаются из ячеек диапазона **A1:A5**;

2)
```
With ListBox1
 .AddItem "Раз"
 .AddItem "Два"
 .AddItem "Три"
 .AddItem "Четыре"
 .ListIndex = 0
 End With
```

Список из четырех элементов формируется последовательно методом AddItem.

Элемент управления ComboBox (Поле со списком) сочетает в себе функциональные возможности списка и поля ввода. В отличие от списка, в поле со

списком отображается только один элемент списка. У него отсутствует режим выделения нескольких элементов списка. Элемент управления ComboBox позволяет пользователю вводить значение через поле ввода, как это делает элемент управления TextBox.

Свойства элемента управления ComboBox, такие как ListIndex, ListCount, List, RowCource, Text и методы Clear, RemoveItem и AddItem аналогичны соответствующим свойствам и методам элемента управления ListBox.

Кроме того, у него есть ряд уникальных свойств, которые перечислены в табл. 33.

Таблица 33

Свойства элемента управления ComboBox

Свойство	Назначение
DropButtonStyle	Устанавливает вид раскрывающегося списка: значение fmDropButtonStylePlain — кнопка без символов, fmDropButtonStyleArrowDisplays — кнопка со стрелкой, fmDropButtonStyleEllipsis — кнопка с эллипсом, fmDropButtonStyleReduce — кнопка с линией
ListRows	Устанавливает число элементов, отображаемых в раскрывающемся списке
MatchRequired	Устанавливает режим ввода в текстовое поле: значение True — только выбор из раскрывающегося списка, False — выбор из раскрывающегося списка и ввод текста
Value	Возвращает введенное в текстовое поле списка значение

Элемент управления Image (Рисунок) используется для отображения графических файлов в формате BMP, CUR, GIF, ICO, JPG и WMF. Основные свойства элемента управления Image перечислены в табл. 34.

Элемент управления MultiPage (Набор страниц) позволяет создать в форме многостраничное диалоговое окно с набором страниц-вкладок. При этом заголовки страниц будут присутствовать на корешках вкладок, отображаемых в главной форме. Для перехода со страницы на страницу достаточно щелкнуть левой кнопкой мыши на корешке вкладки требуемой страницы.

Набор страниц позволяет группировать элементы интерфейса и размещать каждую группу на отдельной вкладке.

Таблица 34

Свойства элемента управления Image

Свойство	Описание
AutoSize	Устанавливает режим автоматического подбора размера для размещения изображения полностью: значение True — размер изменяется, False — размер фиксированный
Picture	Задает ссылку на отображаемый графический файл (используется с функцией LoadPicture)

Свойство	Описание
PictureSizeMode	Устанавливает масштабирование рисунка: значение fmPictureSizeModeClip — обрезаются не помещающиеся в границах объекта части рисунка, fmPictureSizeModeStretch — рисунок занимает всю поверхность объекта, fmPictureSizeModeZoom — рисунок масштабируется с сохранением относительных размеров так, чтобы он помещался полностью внутри объекта
PictureAlignment	Устанавливает расположение изображения внутри элемента управления
PictureTiling	Устанавливает режим мозаичного заполнения рисунком площади объекта

При создании элемента MultiPage в него автоматически помещаются две страницы с именами Page1 и Page2. Чтобы изменить количество вкладок, следует щелкнуть правой кнопкой мыши на корешке требуемой вкладки. Раскроется контекстное меню, содержащее четыре команды: NewPage (Создать страницу), DeletePage (Удалить страницу), Rename (Переименовать) и Move (Переместить). При выборе команды NewPage в элемент управления будет добавлена новая страница. Все остальные команды предназначены для выполнения соответствующих операций.

Основные свойства элемента управления MultiPage приведены в табл. 35.

Таблица 35

Свойства элемента управления MultiPage

Свойство	Описание
MultiRow	Устанавливает режим отображения корешков страниц: значение True — разрешает вывод корешков в несколько строк, False — разрешает вывод кнопок перехода к невидимым корешкам
SelectedItem	Возвращает ссылку на объект выбранной страницы
Count	Возвращает количество страниц
Value	Возвращает номер активной страницы (нумерация страниц начинается с нуля)

Элемент управления TabStrip (Набор вкладок) позволяет создать в диалоговом окне формы несколько вкладок.

Элемент управления TabStrip обладает теми же свойствами, что и элемент управления MultiPage. Основное различие между этими элементами управления состоит в том, что каждая страница в объекте MultiPage имеет собственную отображаемую поверхность, независимую от остальных страниц этого объекта. Поэтому каждая страница может содержать собственный набор элементов управления и отображаемых в них данных.

В объекте TabStrip отображаемая на всех вкладках поверхность является общей и принадлежит непосредственно форме. Поэтому набор элементов управления на всех вкладках будет одинаков, однако отображаемые в них дан-

ные могут быть различными, так как с каждой вкладкой может быть связан собственный источник данных.

Для TabStrip определены следующие свойства.

TabOrientation — определяет расположение ярлычков вкладок, возможны четыре значения: fmTabOrientationTop — вверху, fmTabOrientationBottom — внизу, fmTabOrientationLeft — слева и fmTabOrientationRight — справа.

Style — позволяет задать стиль ярлычков: fmTabStyleTabs — в виде закладок, fmTabStyleButton — в виде кнопок и fmTabStyleNone — ярлычки отсутствуют.

Для TabStrip основным событием является событие Change, возникающее при переходе между вкладками.

При добавлении элемента TabStrip в форму UserForm, автоматически создаются две вкладки с именами Tab1 и Tab2. Операции по созданию, удалению, переименованию, перемещения вкладок выполняются аналогично операциям со страницами элемента MultiPage.

Элементы управления обладают несколькими общими методами, позволяющими перемещать, располагать их и управлять фокусом. В табл. 36 перечислены эти методы. Сведения о других методах при необходимости можно посмотреть в справочнике по VBA для Office.

Таблица 36

Общие методы элементов управления

Метод	Описание
Move	Перемещает элемент управления
SetFocus	Устанавливает фокус на элементе управления
BringToFront	Располагает элемент управления на переднем плане
SendToBack	Располагает элемент управления на заднем плане
ZOrder	Располагает элемент управления на переднем (значение параметра — fmTop) или заднем (значения параметра — fmBottom) плане по отношению к другим элементам управления

Элементы управления имеют большую коллекцию процедур, способных перехватывать и обрабатывать различные события, происходящие в системе, и действия, произведенные пользователем.

В табл. 37 перечислены общие для элементов управления события. Подробное описание остальных событий можно посмотреть в справочнике по VBA для Office.

Таблица 37

Общие события элементов управления

Событие	Описание
BeforeDragOver	Выполнение буксировки данных
BeforeDropOrPaste	Попытка перетащить или вставить данные в объект
Click	Щелчок мышью на элементе управления

Событие	Описание
DblClick	Двойной щелчок мышью на элементе управления
Enter	Элемент управления получает фокус
Exit	Элемент управления теряет фокус
Error	Элемент управления обнаруживает ошибку и не может возвратить сведения о ней в вызывающую программу
KeyDown, KeyUp	Пользователь нажимает (KeyDown) и отпускает (KeyUp) клавишу
KeyPress	Пользователь нажимает клавишу с кодом ANSI
MouseDown, MouseUp	Пользователь нажимает (MouseDown) и отпускает (MouseUp) любую кнопку мыши
MouseMove	Пользователь перемещает указатель мыши над элементом управления

5. РАЗРАБОТКА ПРИЛОЖЕНИЙ

Эффективный способ освоения основ программирования на VBA — изучение и анализ кода работающих программ. Далее рассматривается несколько примеров решения учебных задач, объединенных одной прикладной областью.

Не преследовалась цель создать приложение для использования в реальных условиях. В ряде случаев для выполнения аналогичных действий в разных процедурах применялись разные способы в ущерб эффективности достижения результатов.

В примерах применялся структурный подход программирования. Его основным принципом является модульное программирование, предполагающее разделение основной задачи на отдельные подзадачи и создание для них отдельных автономных программ — модулей. Специально созданная программа объединяет все модули в целое и управляет их работой. Программа на VBA всегда состоит из модулей и подпрограмм. При разработке модульных программ применяются два метода проектирования — нисходящее и восходящее. При нисходящем проектировании разработка программного комплекса идет сверху вниз. На первом этапе разработки кодируется, тестируется и отлаживается головной модуль, который отвечает за логику работы программы. Остальные модули заменяются заглушками, имитирующими работу этих модулей. Применение заглушек позволяет на самом раннем этапе проектирования проверить работоспособность программы и локализовать источник ошибок. На последующих этапах проектирования все заглушки постепенно заменяются рабочими модулями.

В методических целях в примерах иногда допускались нарушения правил и последовательности этапов разработки приложений для демонстрации использования основных элементов VBA при создании программных комплексов.

5.1. ПРИЛОЖЕНИЕ «РЕГИСТРАЦИЯ И ОПЛАТА»

5.1.1. Задание

Разработать приложение, позволяющее фиксировать сведения о клиенте и вносимую сумму оплаты за проживание в гостинице в базе данных, реализованной в виде двух таблиц: расположенных на листах «**Раз**» и «**Два**». Структура таблицы на листе «**Раз**» приведена на рис. 26.

	A	B	C	D	E	F	G	H
1	Фамилия	Имя	Пол	Тип номера	Оплачено	Паспорт сдан	Срок проживания, суток	Оплатить

Рис. 26. Структура таблицы на листе «Раз»

Типы данных таблицы «**Раз**»: в столбцах **A:D, F** — строковые значения, **E, G, H** — числовые.

Столбец **H** предназначен для расчета стоимости проживания в данном типе номера за указанное количество суток. В столбец **E** будет записываться внесенная на данный момент сумма оплаты.

На листе «**Два**» расположены данные о виде номеров и суточной стоимости проживания (рис. 27):

	A	B
1	**Тип номера**	**Стоимость проживания**
2	одноместный	2000
3	двухместный	2500
4	люкс	3000

Рис. 27. Данные в таблице на листе «**Два**»

Для вычисления значений в столбце **H** можно использовать различные варианты:

1) создание формулы:

```
=G2*ЕСЛИ(D2=Два!$A$2;Два!$B$2;ЕСЛИ(D2=Два!$A$3;Два!$B$3;
                        Два!$B$4))
```

2) создание процедуры-функции и размещение ссылки на нее в соответствующих ячейках:

```
Function Функ1(тип_н, срок)
    With Sheets("Два")
        Функ1 = срок * IIf(тип_н = .Cells(2, 1), .Cells(2, 2), _
        IIf(тип_н = .Cells(3, 1), .Cells(3, 2), .Cells(4, 2)))
    End With
End Function
```

3) создание процедуры-функции (п. 2) и использование ссылки на нее в соответствующей процедуре.

5.1.2. Краткое описание решения задачи

Задача решается посредством предварительно разработанной пользовательской формы (рис. 28) для ввода данных в таблицу (см. рис. 26). Алгоритм обработки элементов управления формы иллюстрируется блок-схемой на рис. 29.

Основные действия при работе с приложением следующие.

– активизация формы и настройка параметров ее элементов управления;

– задание значений элементам управления формы:

1) текстовых полей: фамилия клиента, имя клиента, срок проживания (связано со счетчиком для корректировки значения), сумма к оплате (содержимое формируется автоматически в зависимости от срока проживания и типа номера, корректируется, если оплата вносится не полностью);

2) поле со списком: тип номера;

3) радиокнопки: пол клиента;

4) флажки: сдан или нет паспорт на регистрацию, полностью или нет оплачена услуга (при полной оплате включение флажка закрывает доступ к текстовому полю с суммой);

– запись значений элементов управления формы в базу данных при нажатии командной кнопки **Ок** (командная кнопка **Отмена** предназначена для закрытия формы).

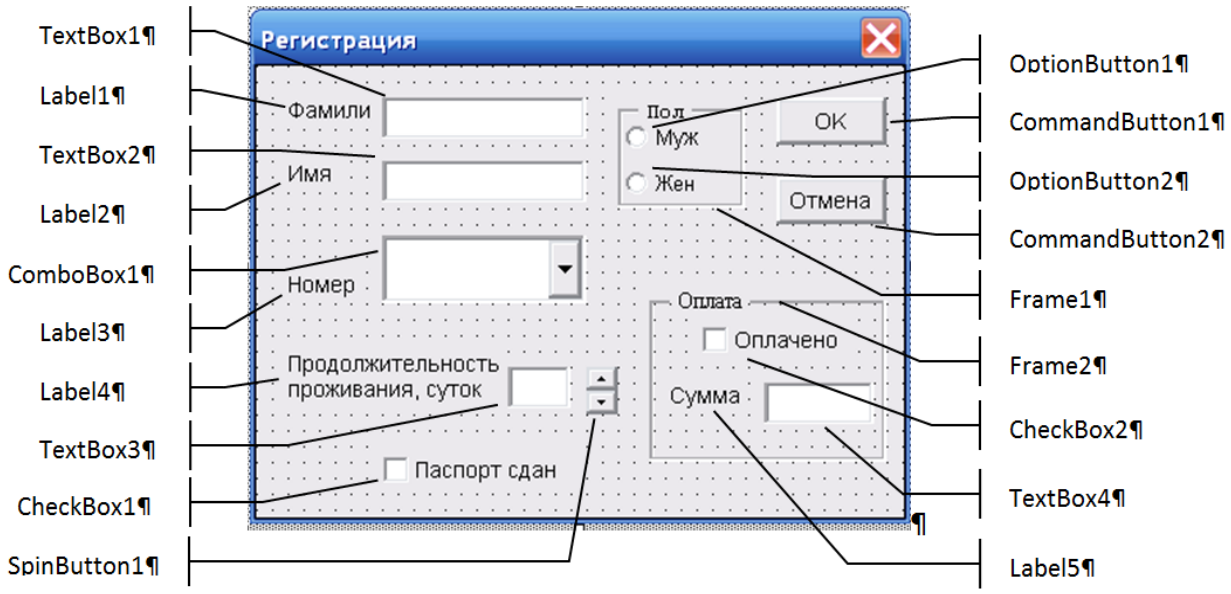

Рис. 28. Форма приложения «Регистрация и оплата»

Рис. 29. Блок-схема программы «Регистрация и оплата»

5.1.3. Программная реализация

Программы, размещенные в модуле

```
    Dim wor As Integer, srok As Integer
    Dim oplach As Single
    Dim fam As String, ima As String, pol As String, _
    pasport As String, nomer As String
    Dim срок As Integer, тип_н As String
'Определение стоимости проживания в номере (параметр "тип_номера")
'за данное количество суток (параметр "срок")
    Function Функ1(тип_н, срок)
    With Sheets("Два")
        Функ1 = срок * IIf(тип_н = .Cells(2, 1), .Cells(2, 4), _
        IIf(тип_н = .Cells(3, 1), .Cells(3, 4), .Cells(4, 4)))
    End With
    End Function

'Управляющая процедура приложения. Загружает форму, подготавливает
'ее к работе, визуализирует форму
    Sub Регистрация()
    Load UserForm1
    Подготовка_формы1
    UserForm1.Show
    End Sub

'Процедура задания начальных значений управляющим элементам формы
    Sub Подготовка_формы1()
    With UserForm1
       .TextBox1.Text = ""
       .TextBox2.Text = ""
       .TextBox3.Text = ""
       .TextBox4.Text = ""
       .CheckBox1.Value = 0
       .CheckBox2.Value = 0
       .SpinButton1.Min = 1
       .SpinButton1.Max = 100
       .OptionButton1.Value = 0
       .OptionButton2.Value = 0
       .ComboBox1.Text = ""
```

```
        .ComboBox1.RowSource = "Два!A2:A4"
    End With
    End Sub

'Процедура заполнения ячеек таблицы значениями элементов
'управления формы

    Sub Заполнить()
'Определение номера строки в таблице для ввода данных

    wor = Application.CountA(Sheets("Раз").Range("C:C")) + 1
'Присвоение переменным значений элементов управления формы
    With UserForm1
    fam = .TextBox1.Text
    ima = .TextBox2.Text
    srok = .TextBox3.Value
    pol = "жен"
    If .OptionButton1.Value = -1 Then pol = "муж"
        If .CheckBox1 = -1 Then
       pasport = "да"
        Else
       pasport = "нет"
        End If
        nomer = .ComboBox1.Text
        oplach = .TextBox4.Value
    End With
'Запись в ячейки таблицы значений переменных

    With Worksheets("Раз")
        .Cells(wor, 1).Value = fam
        .Cells(wor, 2).Value = ima
        .Cells(wor, 3).Value = pol
        .Cells(wor, 4).Value = nomer
        .Cells(wor, 5).Value = oplach
        .Cells(wor, 6).Value = pasport
        .Cells(wor, 7).Value = srok
        .Cells(wor, 8) = Функ1(nomer, srok)
    End With
'Обрамление ячеек вставленной строки границами

    Dim stk As String
```

```
    stk = "A" & wor & ":H" & wor
    With Worksheets("Раз").Range(stk)
        .Borders(xlEdgeBottom).LineStyle = xlContinuous
        .Borders(xlEdgeLeft).LineStyle = xlContinuous
        .Borders(xlEdgeRight).LineStyle = xlContinuous
        .Borders(xlInsideVertical).LineStyle = xlContinuous
    End With
    End Sub

'Вычисление стоимости проживания и запись результата в поле формы
'с меткой "Сумма"
    Sub Оплата()
    With UserForm1
       If .ComboBox1.Text <> "" And .TextBox3.Value <> 0 Then
         .TextBox4.Value = Функ1(.ComboBox1.Text, .TextBox3.Value)
       End If
    End With
    End Sub
```

Программы, размещенные в модуле UserForm1

```
'Процедура обработки события "Щелчок по элементу управления
'"Флажок" с надписью "Оплачено". Управляет режимом доступа к полю
'с надписью "Сумма"
    Private Sub CheckBox2_Click()
    With UserForm1
        If .CheckBox2 = -1 Then
            .TextBox4.Enabled = False
        Else
            .TextBox4.Enabled = True
        End If
    End With
    End Sub

'Процедура обработки события "Изменение состояния элемента
'управления "Поле со списком" с надписью "Номер"
    Private Sub ComboBox1_Change()
    Оплата
    End Sub
```

```
'Процедура обработки события "Щелчок по элементу
'управления "Командная кнопка" с надписью "Ок"

     Private Sub CommandButton1_Click()

'Проверка заполнения текстовых полей формы

     If UserForm1.TextBox1.Text = "" Or UserForm1.TextBox2.Text = _
     "" Or UserForm1.TextBox3.Text= "" Or UserForm1.TextBox4.Text =
     "" Then

         MsgBox "Пустое текстовое поле"

     Else

         Заполнить 'Запись данных из формы в таблицу

         UserForm1.Hide

     End If

     End Sub


'Процедура обработки события "Щелчок по элементу управления
'"Командная кнопка" с надписью "Отмена"

     Private Sub CommandButton2_Click()

     Unload UserForm1

     End Sub


'Процедура обработки события "Изменение значения элемента
'управления "Счетчик". Присвоение значения счетчика текстовому
'полю с надписью "Продолжительность проживания"

     Private Sub SpinButton1_Change()

     UserForm1.TextBox3.Text = CStr(UserForm1.SpinButton1.Value)

     End Sub


'Процедура обработки события "Изменение значения элемента
'управления "Текстовое поле". Присвоение значения текстового
'поля с надписью "Продолжительность проживания" счетчику,
'перерасчет суммы оплаты, присвоение результата полю формы
'с меткой "Сумма"

     Private Sub TextBox3_Change()

     If UserForm1.TextBox3.Text <> "" Then

        UserForm1.SpinButton1.Value = CVar(UserForm1.TextBox3.Text)

        Оплата

     End If

     End Sub
```

5.2. ПРИЛОЖЕНИЕ «ОПЛАТА ПРОЖИВАНИЯ В ГОСТИНИЦЕ»

5.2.1. Задание

Для базы данных (таблицы на листах «**Раз**» и «**Два**» приложения «Регистрация и оплата») разработать приложение, позволяющее учитывать текущую задолженность и вносимую плату за проживание в гостинице.

5.2.2. Краткое описание решения задачи

Поэтапно решение задачи можно представить следующим образом:

1) задание реквизитов для поиска клиента, производящего доплату;
2) поиск клиента в таблице;
3) вывод информации о результатах поиска и, в случае успешного поиска, о клиенте;
4) ввод доплачиваемой суммы;
5) корректировка соответствующего поля в базе на доплачиваемую сумму;
6) вывод сообщения о задолженности или ее отсутствии.

Интерфейс приложения реализован в виде пользовательской формы (рис. 30).

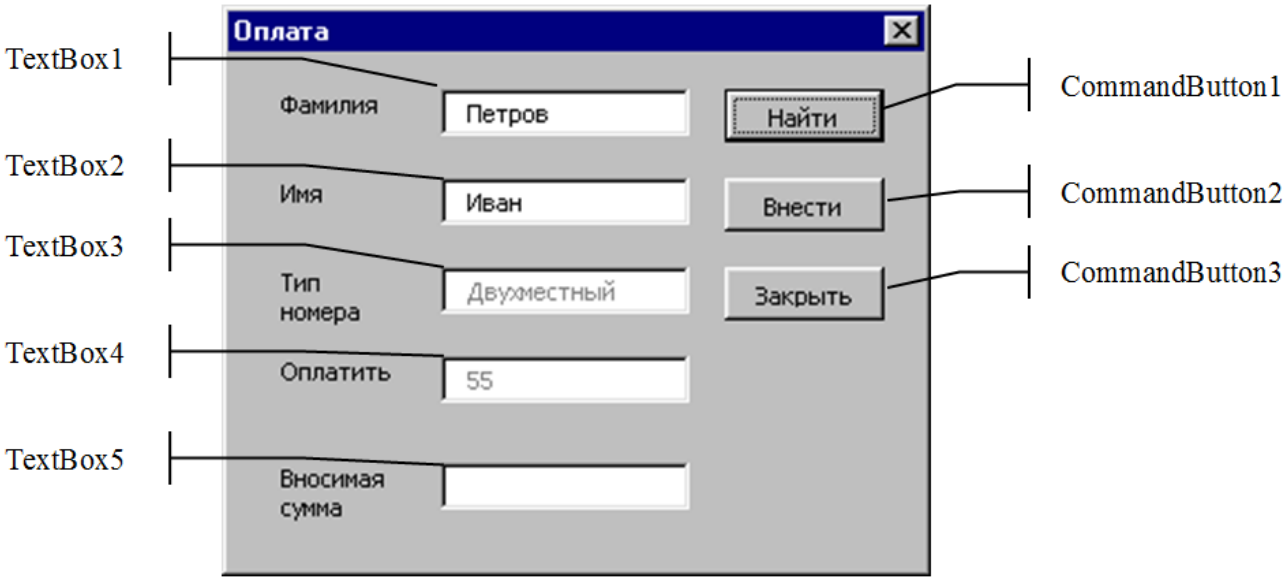

Рис. 30. Форма приложения «Оплата проживания в гостинице»

В качестве реквизитов для поиска клиента выбраны фамилия и имя, которые вводятся в текстовые поля. Функция поиска клиента в базе оформлена в виде отдельной процедуры, и ее активизация связана с событием «нажатие командной кнопки». Результат поиска («найден», «не найден») отображается в информационном окне. В случае успешного поиска информация о клиенте (тип номера и сумма оплаты) автоматически выводится в соответствующие текстовые поля формы пользователя. Поля недоступны пользователю для записи. Для ввода доплачиваемой суммы в форме добавлено текстовое поле. Внесение этой суммы в базу реализуется посредством отдельной процедуры, выполнение которой связано с событием «нажатие командной кнопки». В этой же процедуре реализован вывод результата доплаты («клиент рассчитался», «осталось опла-

тить», «переплачено») в информационном окне. С третьей командной кнопкой связано закрытие формы. Таким образом, форма содержит пять текстовых полей, два из которых недоступны для ввода, и три командных кнопки.

5.2.3. Программная реализация

Программы, размещенные в модуле

```
'в переменной "строка" будет записываться номер найденной строки
     Dim строка As Integer
'Управляющая процедура приложения. Загружает и визуализирует
'форму, подготавливает ее к работе
     Sub Оплата1()
     Подготовка_формы2
     Работа_с_формой2
     End Sub

'Процедура задания начальных значений управляющим элементам формы
     Sub Подготовка_формы2()
     UserForm2.TextBox3.Enabled = False
     UserForm2.TextBox4.Enabled = False
'здесь же могут располагаться операторы задания параметров
'элементов управления формы (см. приложение "Регистрация и
'оплата")
     End Sub

'Процедура загружает и визуализирует форму
     Sub Работа_с_формой2()
     UserForm2.Show
     End Sub

'Процедура поиска клиента в базе по заданным фамилии и имени
'кол_во – число заполненных строк в базе
'найден – индикатор поиска: истина, если клиент найден,
'ложь – в противном случае
     Sub Поиск()
     Dim кол_во As Integer, найден As Boolean
     найден = False
'определение количества заполненных строк в таблице
     Set Область = Sheets("Раз").Cells(1, 1).CurrentRegion
     кол_во = Область.Rows.Count
'организация поиска и вывода информации о результатах поиска
```

```
    строка = 1
    With Sheets("Раз")
        Do While Not найден And строка < кол_во
        строка = строка + 1
        If Trim(.Cells(строка, 1)) & _
           Trim(.Cells(строка, 2)) = _
           Trim(UserForm2.TextBox1.Text) & _
           Trim(UserForm2.TextBox2.Text) Then
           найден = True
           MsgBox "Клиент найден"
           UserForm2.TextBox3.Text = .Cells(строка, 4)
           UserForm2.TextBox4.Text = CStr(.Cells(строка, 8) - _
           .Cells(строка, 5))
        End If
        Loop
        If Not найден Then MsgBox "Клиент не найден"
    End With
    End Sub

    Sub Внести()
'Процедура внесения изменений в базу в связи с оплатой и
'вывод сообщения о задолженности или ее отсутствии
    Dim raz As Single
    With Sheets("Раз")
        .Cells(строка, 5) = Val(UserForm2.TextBox5.Text) + _
        .Cells(строка, 5)
        raz = .Cells(строка, 8) - .Cells(строка, 5)
    End With
    Select Case raz
        Case 0
           MsgBox "Клиент расчитался "
        Case Is > 0
           MsgBox "Осталось оплатить " & Str(raz)
        Case Else
           MsgBox "Переплачено " & Str(-raz)
    End Select
    End Sub
```

Программы, размещенные в модуле UserForm2:

```
'Процедура обработки события "Щелчок по командной кнопке": поиск
'клиента в базе по заданным в форме фамилии и имени

    Private Sub CommandButton1_Click()

    Поиск

    End Sub


'Процедура обработки события "Щелчок по командной кнопке":
'внесение изменений в базу в связи с оплатой

    Private Sub CommandButton2_Click()

    Внести

    End Sub


'Процедура обработки события "Щелчок по командной кнопке":
'закрытие и выгрузка из памяти формы

    Private Sub CommandButton3_Click()

    Unload UserForm2

    End Sub
```

5.3. ПРИЛОЖЕНИЕ «ФОРМИРОВАНИЕ ЗАПРОСОВ»

5.3.1. Задание

Для базы данных (см. приложение «Регистрация и оплата») разработать приложение, позволяющее выполнять запросы, выводящие данные:

1) о конкретном постояльце;
2) списки мужчин или женщин;
3) списки задолжников на сумму, превышающую заданную.

5.3.2. Краткое описание решения задачи

Задача решается посредством предварительно разработанной пользовательской формы (рис. 31) для выбора вида запроса, задания его параметров и выполнения команды реализации запроса.

Основные действия при работе с приложением следующие.

– активизация формы и настройка параметров ее элементов управления. Изначально в форме будут визуализированы только радиокнопки (все выключены) и соответствующие им названия, командные кнопки **Создать** и **Закрыть**. Остальные элементы управления будут скрыты;

– выбор вида запроса. Осуществляется путем включения соответствующей требуемому запросу радиокнопки. После чего визуализируются элементы управления для задания параметров запроса;

– задание значений параметров запроса:

1) для запроса данных о конкретном клиенте — выбор имени и фамилии из раскрывающегося списка;

2) для запроса сведений о клиентах по гендерному признаку — выбор пола из раскрывающегося списка;

3) для запроса о клиентах с задолженностью, превышающей или равной заданной суммы, — ввод значения суммы в текстовое окно с возможностью корректировки вводимого значения с помощью счетчика, связанного с текстовым полем;

– выполнение выбранного запроса при нажатии командной кнопки **Создать** (командная кнопка **Закрыть** предназначена для закрытия формы).

В результате выполнения каждого запроса в рабочий лист с соответствующим названием заносится отфильтрованная по заданным параметрам исходная таблица. Если выбранный запрос выполняется впервые, то рабочий лист для записи результата создается в рабоче книге.

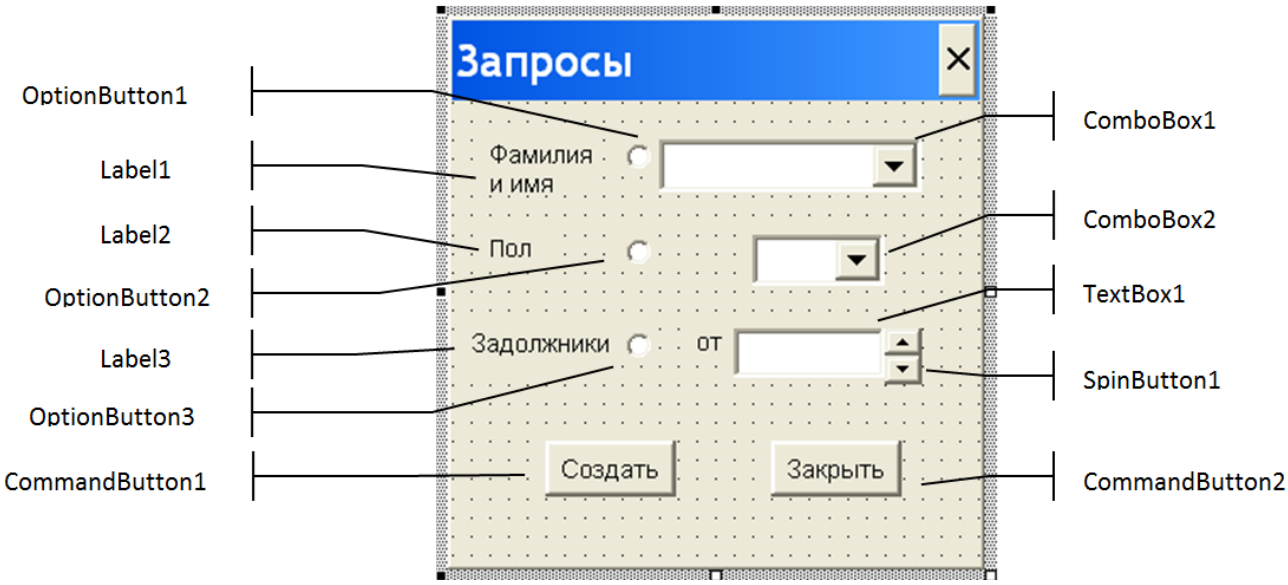

Рис. 31. Форма примера «Формирование запросов»

5.3.3. Программная реализация

Программы, размещенные в модуле:

```
'Управляющая процедура приложения. Загружает и визуализирует
'форму
    Sub Запросы()
    UserForm3.Show 0
    End Sub
```

Программы, размещенные в модуле UserForm3:

```
'Процедура обработки события "Инициализация формы": установка
'начальных значений параметров элементов управления формы
    Private Sub UserForm_Initialize()
        With UserForm3
```

```
            .SpinButton1.Max = 10000000
            .TextBox1.Visible = False
            .SpinButton1.Visible = False
            .Label4.Visible = False
            .ComboBox2.Visible = False
            .ComboBox1.Visible = False
            .OptionButton1.Value = False
            .OptionButton2.Value = False
            .OptionButton3.Value = False
        End With
    End Sub

'Процедура обработки события "Завершение работы формы":
'очистка полей с раскрывающимися списками формы

    Private Sub UserForm_Terminate()
        With UserForm3
            .ComboBox1.Clear
            .ComboBox2.Clear
        End With
    End Sub

'Процедура обработки события "Щелчок по командной кнопке":
'выполнение запроса при нажатии командной кнопки "Создать"

    Private Sub CommandButton1_Click()
    Dim aList1 As Worksheet, colList As Integer, _
    inList As Integer, flag As Boolean
    Dim colRow1 As Integer
'определение числа строк в исходной таблице
    colRow1 = _
    Sheets("Раз").Range("A1").CurrentRegion.Rows.Count
    UserForm3.Hide
      With UserForm3
'проверка: включена радиокнопка с надписью "Фамилия и имя" и
'выбраны имя и фамилия клиента в поле со списком
        If .OptionButton1 And .ComboBox1.Text <> "" Then
            colList = Sheets.Count
            flag = True
            For inList = 1 To colList
'проверка наличия рабочего листа в книге с именем "Клиент"
```

```
                If Sheets(inList).Name = "Клиент" Then flag = _
                False
            Next inList
             If flag Then
'создание листа с именем "Клиент" при отсутствии листа с таким
'именем
                    Set aList1 = Sheets.Add(, Sheets("Два"))
                    aList1.Name = "Клиент"
                Else
                    Set aList1 = Sheets("Клиент")
            End If
'копирование первой строки исходной таблицы и вставка ее
'на лист "Клиент"
            Sheets("Раз").Rows(1).Copy
            aList1.Select
            aList1.Cells(1, 1).Select
            ActiveSheet.Paste
'поиск строки с заданными именем и фамилией в исходной таблице и
'вставка ее на лист "Клиент"
            j = 1
            i = 1
            Do
                   i = i + 1
            Loop While Trim(Sheets("Раз").Cells(i, 1)) & " " & _
            Trim(Sheets("Раз").Cells(i, 2)) <> .ComboBox1.Text
                Sheets("Раз").Rows(i).Copy
                aList1.Select
                aList1.Cells(2, 1).Select
                ActiveSheet.Paste
'выравнивание ширины столбцов по значениям в результирующей
'таблице
                aList1.Range("A1").CurrentRegion.Columns.AutoFit
                aList1.Cells(1, 1).Select
'проверка: включена радиокнопка с надписью "Пол" и
'выбран пол клиента в поле со списком
        ElseIf .OptionButton2 And .ComboBox2.Text <> "" Then
            colList = Sheets.Count
            flag = True
            For inList = 1 To colList
```

```
'проверка наличия рабочего листа в книге с именем "Гендер"
                If Sheets(inList).Name = "Гендер" Then flag = _
                False
            Next inList
             If flag Then
'создание листа с именем " Гендер" при отсутствии листа с таким
'именем
                    Set aList1 = Sheets.Add(, Sheets("Два"))
                    aList1.Name = "Гендер"
                Else
                    Set aList1 = Sheets("Гендер")
            End If
'копирование первой строки исходной таблицы и вставка ее
'на лист "Гендер"
            Sheets("Раз").Rows(1).Copy
            aList1.Select
            aList1.Cells(1, 1).Select
            ActiveSheet.Paste
'поиск строк с заданным значением пола в исходной таблице и
'вставка их на лист "Гендер"
            j = 1
            For i = 2 To colRow1
                If Trim(Sheets("Раз").Cells(i, 3)) = _
                .ComboBox2.Text Then
                    j = j + 1
                    Sheets("Раз").Rows(i).Copy
                    aList1.Select
                    aList1.Cells(j, 1).Select
                    ActiveSheet.Paste
                End If
             Next i
'очистка строк в результирующей таблице, оставшихся после
'выполнения предыдущего запроса
            Do While j < colRow1
                    j = j + 1
                   aList1.Rows(j).Select
                   Selection.Clear
             Loop
```

```
'выравнивание ширины столбцов по значениям в результирующей
'таблице
             aList1.Range("A1").CurrentRegion.Columns.AutoFit
             aList1.Cells(1, 1).Select
'проверка: включена радиокнопка с надписью "Задолжники" и
'введено значение значимого долга в текстовое поле
        ElseIf .OptionButton3 And .TextBox1.Text <> "" Then
            colList = Sheets.Count
            flag = True
            For inList = 1 To colList
'проверка наличия рабочего листа в книге с именем "Должники"
               If Sheets(inList).Name = "Должники" Then _
               flag = False
            Next inList
             If flag Then
'создание листа с именем "Должники" при отсутствии листа с таким
'именем
                    Set aList1 = Sheets.Add(, Sheets("Два"))
                    aList1.Name = "Должники"
                Else
                    Set aList1 = Sheets("Должники")
            End If
'копирование первой строки исходной таблицы и вставка ее
'на лист "Должники"
            Sheets("Раз").Rows(1).Copy
            aList1.Select
            aList1.Cells(1, 1).Select
            ActiveSheet.Paste
'поиск строк в исходной таблице со значением неоплаченной
'стоимости проживания, большей или равной заданной, и вставка
'их на лист "Должники"
            j = 1
            For i = 2 To colRow1
                Dim aa As Single, bb As Single
                aa = Sheets("Раз").Cells(i, 8) - _
                Sheets("Раз").Cells(i, 5)
                bb = CVar(.TextBox1.Text)
                If aa >= bb Then
                    j = j + 1
```

```
                    Sheets("Раз").Rows(i).Copy
                    aList1.Select
                    aList1.Cells(j, 1).Select
                    ActiveSheet.Paste
                End If
            Next i
'очистка строк в результирующей таблице, оставшихся после
'выполнения предыдущего запроса
            Do While j < colRow1
                    j = j + 1
                   aList1.Rows(j).Select
                   Selection.Clear
            Loop
'выравнивание ширины столбцов по значениям в результирующей
'таблице
            aList1.Range("A1").CurrentRegion.Columns.AutoFit
            aList1.Cells(1, 1).Select
        End If
    End With
    UserForm3.Show 0
    End Sub

'Процедура обработки события "Щелчок по командной кнопке":
'закрытие и выгрузка из памяти формы
    Private Sub CommandButton2_Click()
    Unload UserForm3
    End Sub

'Процедура обработки события "Изменение состояния радиокнопки":
'если кнопка включается, формируются элементы поля со списком
'(фамилии и имена клиентов) и поле со списком визуализируется;
'если кнопка выключается, поле со списком очищается и делается
'невидимым
    Private Sub OptionButton1_Change()
    With UserForm3
        If .OptionButton1.Value = True Then
            Dim fiCol As Integer
            fiCol = Worsheets("Раз").Range("A1").CurrentRegion. _
            Columns(1).Cells.Count
```

```
            For ii = 2 To fiCol
                .ComboBox1.AddItem _
                Worksheets("Раз").Cells(ii, 1).Text & _
                " " & Worksheets("Раз").Cells(ii, 2).Text
            Next ii
             .ComboBox1.Visible = True
        Else
             .ComboBox1.Visible = False
             .ComboBox1.Clear
        End If
    End With
    End Sub

'Процедура обработки события "Изменение состояния радиокнопки":
'если кнопка включается, формируются элементы поля со списком
'(пол) и поле со списком визуализируется; если кнопка выключается,
'поле со списком очищается и делается невидимым

    Private Sub OptionButton2_Change()
    With UserForm3
        If .OptionButton2.Value = True Then
            .ComboBox2.AddItem "муж"
            .ComboBox2.AddItem "жен"
            .ComboBox2.Visible = True
        Else
             .ComboBox2.Visible = False
             .ComboBox2.Clear
        End If
    End With
    End Sub

'Процедура обработки события "Изменение состояния радиокнопки":
'если кнопка включается, визуализируются текстовое поле, счетчик и
'метка, если кнопка выключается, эти элементы управления делаются
'невидимыми

    Private Sub OptionButton3_Change()
    With UserForm3
        If .OptionButton3.Value = True Then
             .TextBox1.Visible = True
             .SpinButton1.Visible = True
```

```
            .Label4.Visible = True
        Else
            .TextBox1.Visible = False
            .SpinButton1.Visible = False
            .Label4.Visible = False
        End If
    End With
    End Sub

'Процедура обработки события "Изменение значения элемента
'управления "Счетчик". Присвоение значения счетчика текстовому
'полю с надписью "по"
    Private Sub SpinButton1_Change()
    UserForm3.TextBox1.Text = CStr(UserForm3.SpinButton1.Value)
    End Sub

'Процедура обработки события "Изменение значения элемента
'управления "Текстовое поле". Присвоение значения текстового
'поля с надписью "по" счетчику
    Private Sub TextBox1_Change()
    With UserForm3
        If CVar(.TextBox1.Text) > .SpinButton1.Max Then
            MsgBox "Значение превышает допустимое"
            .TextBox1.Value = .SpinButton1.Max
            .TextBox1.SetFocus
            Exit Sub
        End If
        If Trim(.TextBox1.Text) <> "" Then
            .SpinButton1.Value = CVar(.TextBox1.Text)
        End If
    End With
    End Sub
```

5.4. ПРИЛОЖЕНИЕ «ГОСТИНИЦА»

5.4.1. Задание

Разработать приложение, обеспечивающее выполнение функций приложений «Регистрация и оплата», «Оплата проживания в гостинице», «Формирование запросов».

5.4.2. *Краткое описание решения задачи*

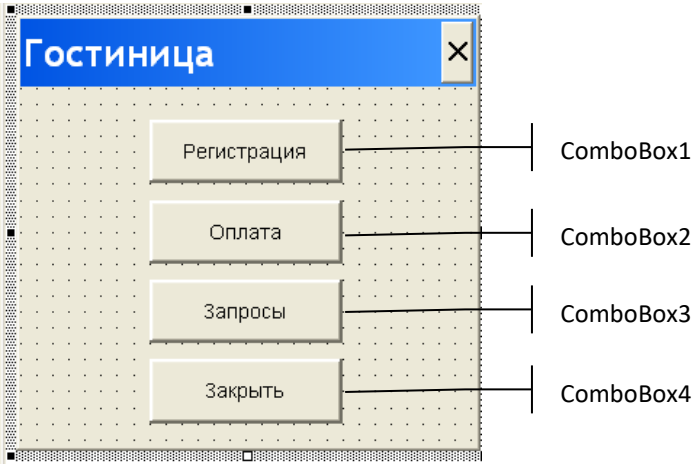

Рис. 32. Окно управляющей формы

Задача решается посредством предварительно разработанной пользовательской формы (рис. 32), позволяющей активизировать перечисленные выше приложения.

5.4.3. *Программная реализация*

Программы, размещенные в модуле:

```
'Управляющая процедура приложения. Загружает и визуализирует
'форму
    Sub Главная()
    UserForm4.Show 0
    End Sub
```

Программы, размещенные в модуле UserForm4:

```
'Процедура обработки события "Щелчок по командной кнопке":
'выполнение управляющей процедуры приложения "Регистрация и
'оплата"
    Private Sub CommandButton1_Click()
    Регистрация
    End Sub

'Процедура обработки события "Щелчок по командной кнопке":
'выполнение управляющей процедуры приложения "Оплата проживания
'в гостинице"
    Private Sub CommandButton2_Click()
    Оплата1
    End Sub
```

```
'Процедура обработки события "Щелчок по командной кнопке":
'выполнение управляющей процедуры приложения "Формирование
'запросов"
    Private Sub CommandButton3_Click()
    Запросы
    End Sub

'Процедура обработки события "Щелчок по командной кнопке":
'закрытие и выгрузка из памяти формы
    Private Sub CommandButton4_Click()
    Unload UserForm4
    End Sub
```

5.5. ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

5.5.1. Общее задание

Средствами Visual Basic разработать приложение для работы с таблицей, структура которой определяется текстом варианта задания. Размер таблицы — не менее 20 записей. Последней строкой таблицы должна быть итоговая строка, в которую включаются формулы подсчета итогов по произвольно выбранным столбцам.

Приложение должно обеспечивать реализацию следующих функций:

1) заполнение таблицы, т. е. добавление новых записей в конец таблицы с пересчетом итоговых строк и проверкой вводимых данных на допустимость значений;

2) корректировка записей таблицы, т. е. изменение значений в доступных для изменения ячейках таблицы;

3) выбор данных из таблицы по заданным условиям (2–3 запроса по одному и по нескольким самостоятельно выбранным условиям);

4) создание отчетов, содержащих выбранную из таблицы информацию в удобочитаемой форме.

Каждую из функций связать с кнопкой, кнопки разместить на главной экранной форме.

5.5.2. Варианты заданий

Вариант № 1

Создать таблицу со следующими реквизитами: Ф.И.О., таб. номер, стаж, категория, ставка, отработано дней, % премии, % соц. защиты, % налога, дополнительная зарплата. В отчете учесть сумму зарплаты, выдаваемую на руки и вычисляемую без учета налога.

Вариант № 2

Создать таблицу со следующими реквизитами: Ф.И.О. студента, факультет, курс, специальность, оценка за 1 экзамен, ..., оценка за 4 экзамен, количество пересдач (<=3) по 1 экзамену, ..., количество пересдач по 4 экзамену, предмет 1, ..., предмет 4. В отчете учесть информацию о начислении стипендии.

Вариант № 3

Создать таблицу со следующими реквизитами: поставщик, код поставщика, дата поставки, наименование гарнитура, стоимость гарнитура, сорт, вид брака, скидка за брак партии, количество гарнитуров. В отчете учесть информацию о стоимости партии гарнитуров с учетом уценки.

Вариант № 4

Создать таблицу со следующими реквизитами: дата, наименование, код товара, наличие на складе, № склада, отпущено, принято, ед. измерения, стоимость ед. товара. В отчете учесть информацию о стоимости принятого, отпущенного, имеющегося в наличии товара.

Вариант № 5

Создать таблицу со следующими реквизитами: фамилия, адрес, дата, стоимость заказа, сумма аванса, задолженность, вид заказа. В отчете учесть итоговую информацию: задолженность + стоимость – аванс.

Вариант № 6

Создать таблицу со следующими реквизитами: фамилия, адрес, номер телефона, аванс, вид льгот, % скидок, абонентная плата, плата за иногородние разговоры, задолженность по оплате, % пени, месяц. В отчете учесть информацию в виде графы "к оплате": (абонентная плата + плата за иногородние разговоры) * (1 – % скидок / 100) + задолженность по оплате * (1 + % пени / 100) – аванс.

Вариант № 7

Создать таблицу со следующими реквизитами: фамилия ребенка, имя ребенка, возраст ребенка, группа детсада, № детсада, % льгот, месяц, год, аванс, оплата за месяц, задолженность, % пени. В отчете учесть графу "к оплате": (оплата за месяц + задолженность * (1 + % пени / 100) * (1 – % льгот / 100) – аванс.

Вариант № 8

Создать таблицу со следующими реквизитами: улица, дом, квартира, кол. проживающих, кол. проживающих старше 18 лет, кол. инвалидов, статус семьи, месячный доход семьи, площадь квартиры, кол. комнат. В отчете учесть информацию о доходе на одного члена семьи.

Вариант № 9

Создать таблицу со следующими реквизитами: корпус, этаж, номер комнаты, вид помещения, метраж, коэф. использования полезной площади, состояние помещения, коэф. аварийности. В отчете учесть состояние помещения: (метраж * коэф. использования) * коэф. аварийности.

Вариант № 10

Создать таблицу со следующими реквизитами: Ф.И.О. ребенка, Ф.И.О. отца, Ф.И.О. матери, дата рождения ребенка, пол, вес при рождении, код патологии ребенка, кол. детей в семье, уровень жизни семьи (низкий, средний, высокий). В отчете учесть информацию о статусе семьи (например, многодетная, среднеобеспеченная).

Вариант № 11

Создать таблицу со следующими реквизитами: Ф.И.О. вкладчика, № сберкнижки, адрес, вид вклада, дата вклада, приход, расход, % вклада. В отчете учесть информацию о сумме вклада.

Вариант № 12

Создать таблицу со следующими реквизитами: корпус, этаж, номер лаборатории, наименование оборудования, код оборудования, стоимость единицы оборудования, дата установки, кол. единицы данного оборудования. В отчете учесть стоимость установленного оборудования.

Вариант № 13

Создать таблицу со следующими реквизитами: поставщик, вид оплаты, наименование товара, код товара, вид товара, количество, стоимость единицы наименования товара. В отчете учесть стоимость товара.

Вариант № 14

Создать таблицу со следующими реквизитами: факультет, специальность, группа, фамилия студента, имя, отчество, средний балл аттестата, пол, стаж работы, форма обучения (дневная, заочная, ускоренная), вид набора (коммерческий, бюджетный), оценка за вступительный экзамен 1, ..., оценка за вступительный экзамен 3. В отчете учесть средний балл за вступительные экзамены (0 для платного обучения).

Вариант № 15

Создать таблицу со следующими реквизитами: факультет, группа, специальность, предмет число студентов в группе, лектор, ассистент, кол. лекционных часов, кол. практических и лабораторных часов, кол. недель в семестре. В отчете учесть кол. часов дисциплины в неделю.

Вариант № 16

Создать таблицу со следующими реквизитами: факультет, кафедра, Ф.И.О. преподавателя, стаж работы, разряд, нагрузка за год, кол. читаемых

дисциплин, должность, ученое звание, степень, пол. В отчете учесть среднюю нагрузку за месяц.

Вариант № 17

Создать таблицу со следующими реквизитами: Ф.И.О. студента, код студента, факультет, группа, курс, семестр, предмет 1, ..., предмет 5, экзам. оценка 1, ..., экзам. оценка 5. В отчете учесть информацию о начислении стипендии.

Вариант № 18

Создать таблицу со следующими реквизитами: тема, подтема, наименование книги, автор, кол. экземпляров, место хранения, цена одного экземпляра, издательство. В отчете учесть стоимость книги данного наименования.

Вариант № 19

Создать таблицу со следующими реквизитами: Ф.И.О. проживающего, номер гостиницы, класс номера, стоимость в сутки, срок проживания, услуги по питанию и быту, непредвиденные расходы. В отчете учесть общую стоимость проживания.

Вариант № 20

Создать таблицу со следующими реквизитами: Ф.И.О. студента, курс, факультет, группа, номер абонемента, наименование книги, автор, тема, дата заявки и выдачи на руки, срок сдачи книги, стоимость экземпляра, штраф за просрочку за день. В отчете учесть сумму штрафа: штраф за день * кол. дней + стоимость книги * 0,2.

Вариант № 21

Создать таблицу со следующими реквизитами: № маршрута, № автобуса, марка машины, Ф.И.О. водителя, дата, смена, кол. ездок, норма горючего на одну ездку. В отчете учесть норму отпуска горючего на смену.

Вариант № 22

Создать таблицу со следующими реквизитами: фамилия, адрес, дата, стоимость заказа, сумма аванса, задолженность, вид заказа. В отчете учесть итоговую информацию: задолженность + стоимость – аванс.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

Берман Н. Д. Разработка приложений в Microsoft Excel 2010 : учеб. пособие / Н. Д. Берман, Н. И. Шадрина ; [науч. ред. Е. Г. Агапова]. Хабаровск : Изд-во Тихоокеан. гос. ун-та, 2015. — 130 с.

Биллиг В. А. Основы офисного программирования и язык VBA / В. А. Биллиг. — М. : НОУ Интуит, 2016. — 191 с.

Гарбер Г. З. Основы программирования на Visual Basic и VBA в Excel 2007 / Г. З. Гарбер. — М. : СОЛОН-ПРЕСС, 2008. — 192 с.

Гарнаев А. Использование MS Excel и VBA в экономике и финансах / А. Гарнаев. — М. : ОЗОН-Пресс, 2012. — 336 с.

Гарнаев А. Ю. Microsoft Office Excel 2010: разработка приложений / А. Ю. Гарнаев, Л. В. Рудикова. — СПб. : БХВ-Петербург, 2011. — 528 с.

Гарнаев А. Ю. VBA / А. Ю. Гарнаев. — СПб. : БХВ-Петербург, 2005. — 825 с.

Джексон М. А. Excel 2016. Профессиональное программирование на VBA / М. А. Джексон, Р. Куслейка. — М. : Диалектика, 2018. — 784 с.

Кашаев С. М. Программирование в Microsoft Excel на примерах / С. М. Кашаев. — СПб. : БХВ-Петербург, 2007. — 320 с.

Комолова Н. Программирование на VBA в Excel 2016. Самоучитель / Н. Комолова, Е. Яковлева. — СПб. : БХВ-Петербург, 2017. — 431 с.

Лядова Л. Н. Microsoft Office: от начинающего пользователя до профессионала. В 2 ч. Ч. 2: Основы офисного программирования : учеб.-метод. пособие / Л. Н. Лядова, В. В. Ланин ; Перм. ун-т. — Пермь, 2007. — 388 с.

Осетрова И. С. Microsoft Visual Basic for Application / И. С. Осетрова, Н. А. Осипов. — СПб. : НИУ ИТМО, 2013. — 120 с.

Слепцова Л. Д. Программирование на VBA в Microsoft Office 2010 / Л. Д. Слепцова. — М. : Вильямс, 2010. — 432 с.

Справочник по VBA для Office [Электронный ресурс]. — Режим доступа: https://docs.microsoft.com/ru-ru/office/vba/api/overview/ (дата обращения: 10.04.2019).

Уокенбах Дж. Excel 2010: профессиональное программирование на VBA : пер. с англ. / Дж. Уокенбах. — М. : Вильямс, 2012. — 944 с.

Уокенбах Дж. Microsoft Excel 2010. Библия пользователя : пер. с англ. / Дж. Уокенбах. — М. : Вильямс, 2011. — 912 с.

Учебное издание

Ступин Виталий Валерьевич

Информационные системы и технологии:
разработка приложений в MS Excel средствами VBA

Учебное пособие

Издается в авторской редакции

ИД № 06318 от 26.11.01.
Подписано в пользование 30.04.19.

Издательство Байкальского государственного университета.
664003, г. Иркутск, ул. Ленина, 11.
http://bgu.ru.