ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
Федеральное государственное
образовательное бюджетное учреждение
высшего профессионального образования
«САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ
им. проф. М. А. БОНЧ-БРУЕВИЧА»

Е. В. СТРИГИНА

ПРОГРАММИРОВАНИЕ IT сервисов предприятия

Учебное пособие

СПб ГУТ)))

САНКТ-ПЕТЕРБУРГ
2013

УДК 004 65 (075.8)
ББК 32.973.26–013.2я73
С85

Утверждено
редакционно-издательским советом СПбГУТ
в качестве учебного пособия

Рецензенты:
д.т.н., профессор кафедры ИТЭ СПбГУТ
А. Д. Сотников
к.т.н., доцент кафедры ЭиУС СПбГУТ
А. А. Степаненко

Стригина, Е. В.
С85 Программирование IT сервисов предприятия: учебн. пособие / Е. В. Стригина ; РИЦ СПбГУТ. – СПб., 2013. – 63 с.

Рассмотрены возможности программирования приложений MS Excel 2007 с использованием языка Visual Basic for Applications. Учебное пособие дает возможность познакомиться с основными конструкциями и операторами языка, решить задачи автоматизации вычислительных процессов и обработки табличных данных, а также создания пользовательских интерфейсов с помощью экранных форм.

Предназначено для студентов, обучающихся по направлению (специальности) 080500.62 «Бизнес-информатика».

УДК 004.65(075.8)
ББК 32.973.26–013.2я73

СОДЕРЖАНИЕ

1.

1. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ НА VBA В EXCEL

Excel – это мощное и сложное приложение, предоставляющее в распоряжение пользователей огромный набор средств и инструментов для обработки, анализа и отображения табличных данных. Excel позволяет также решать широкий круг задач, в частности, создание приложений, путем использования программирования. Все, что можно выполнить с помощью клавиатуры и мыши, можно повторить путем написания соответствующей программы. И, как всякая программа, она будет способствовать автоматизации выполнения задач обработки и анализа данных. Применение программирования позволит значительно убыстрить решение задач, уменьшить количество ошибок, создать удобные и функциональные интерфейсы пользовательских приложений, а также обеспечить взаимодействие с другими компонентами Microsoft Office.

Программа представляет собой набор инструкций, определяющий последовательность действий, которые должен выполнить компьютер для решения поставленной задачи. Программы Excel пишутся на языке VBA (Visual Basic for Applications). VBA называют также системой визуального проектирования. Это название особенно очевидно при проектировании интерфейсов с помощью пользовательских форм. На VBA создаются программы для всех пользовательских приложений, которые входят в пакет Microsoft Office.

1.1. Основные принципы объектно-ориентированного программирования (ООП)

Язык VBA относится к классу объектно-ориентированных языков, программирование на которых, в отличие от традиционных языков программирования, базируется на взаимодействии с объектами.

Объектно-ориентированная программа представляет собой совокупность взаимосвязанных разделов (модулей), каждый из которых используется для решения отдельной задачи. Одни модули управляют компонентами программного интерфейса (меню, диалоговыми окнами), другие предназначены для управления рабочими книгами, листами или ячейками. Каждый из таких компонентов является объектом. Программа, написанная с использованием ООП, состоит из объектов, которые могут взаимодействовать между собой.

Объект – это программный компонент, который содержит связанные между собой данные и программный код для их обработки.

Данные, которые являются характеристиками объекта, принято называть его свойствами или атрибутами. Процедуры обработки этих данных называются методами. Выполнение процедур, связанных с определенным методом зачастую происходит при наступлении некоторого события, характерного для конкретного объекта.

Концепция ООП базируется на таких понятиях как инкапсуляция, наследование и полиморфизм.

Инкапсуляция является важнейшим свойством объектов. Если в традиционном программировании была велика вероятность использования процедур обработки данных, не предназначенных для обработки этих данных, то инкапсуляция является средством организации доступа к данным только через соответствующие методы.

Наследование позволяет определять новые объекты, используя свойства прежних, дополняя или изменяя их. Создание новых объектов, не связанных с уже существующими,– менее эффективный путь, чем применение наследования, так как требует затрат времени на отладку и тестирование. Наследование позволяет создавать иерархические структуры данных, связанные отношением подчинения, что облегчает управление большими потоками информации.

Полиморфизм заключается в том, что одно и то же имя метода может соответствовать различным действиям в зависимости от типа объекта. Решение о том, какая операция (метод) должна быть выполнена в конкретной ситуации, принимается во время выполнения программы (тип Variant – ссылка на объект любого типа, поздняя привязка, использование конкретного типа – ранняя привязка).

ООП имеет ряд преимуществ по сравнению с традиционными методами программирования. К таким преимуществам относятся следующие:

-уменьшение количества ошибок из-за изолированности объектов и строгого контроля взаимодействия объектов с другими компонентами программы;

-повторное использование кодов, причем область такого использования не ограничена рамками одной конкретной программы.

1.2. Объектная модель Excel

Доступ программного кода к компонентам Excel обеспечивается с помощью интерфейса, которым является объектная модель Excel.

Объектная модель Excel является основой, на которой строится программирование. Приложение Excel состоит из объектов, обеспечивающих функциональность данной программы. Совокупность объектов Excel, предназначенных для использования программами, называется объектной моделью Excel.

Объектная модель Excel организована в виде иерархической структуры (рис.1). Верхним узлом в этой иерархии является объект Application, представляющий собственно приложение Excel.

В объектной модели Excel часто используются коллекции. Excel использует коллекции всегда, когда существует потенциальная вероятность создания более одного объекта. Имя коллекции образуется как множественное число от имени объектов, в ней содержащихся. Например, рабочая книга Excel может содержать несколько листов, они образуют коллекцию Sheets, которая содержит объекты Sheet для каждого листа рабочей книги. Объект коллекции имеет свойство Count, значение которого соответствует порядковому номеру объекта в этой коллекции. Свойство Count доступно только для чтения.

Предусмотрено два способа обращения к объекту коллекции. Следует либо указать порядковый номер этого объекта в коллекции, либо использовать уникальный ключ данного объекта. Например, в коллекции Sheets таким ключом является название рабочего листа. Так, запись Sheets(1) обозначает обращение к первому листу рабочей книги, тогда как запись Sheets(“Продажи”) ссылается на лист с указанным названием.

Другие примеры использования коллекций приведены далее.

```
Sheets("Заставка").Activate
```

Метод Activate применен к экземпляру (листу) "Заставка" коллекции объектов Sheets.

```
Cells(10,3).Value=25
```

Свойству Value экземпляра коллекции Cells (ячейке), находящейся в 10-строке и 3-ем столбце, присвоено значение 25.

В большинстве коллекций предусмотрены методы, позволяющие создавать новые объекты и удалять старые. Для добавления объектов используется метод Add, а для удаления – Delete.

В графическом виде часть объектной модели представлена на рис.1. В овалах представлены объекты, в прямоугольниках – коллекции.

Как говорилось ранее, объект представляет собой компонент Excel, имеющий имя и способный к изменению. Изменение объекта происходит благодаря его свойствам и методам, т.е. объект – это, по сути, его свойства и методы.

Свойства могут представлять информацию о самом объекте, (например, цветовые или шрифтовые характеристики данных в ячейках таблицы), либо определять особенности его воспроизведения и поведения, (например, видимость объекта).

Некоторые свойства доступны только для чтения, т.е. их значения можно считывать, но не изменять (например, количество ячеек в листе). Другие доступны как для чтения, так и для записи (например, цвет фона ячейки).

Обращение к свойству происходит с помощью точечной нотации:

```
ИмяОбъекта.ИмяСвойства
```

Например, свойству «цвет фона» Interior.Color объекта Range(“A1:A10”) присваивается значение серого цвета в системе rgb следующим кодом:

```
Range(“A1:A10”).Interior.Color=rgb(200,200,200)
```

Другие примеры использования свойств приведены далее.

```
ActiveCell.Value="text"
ActiveCell.Formula="=сумм(A1:A10)"
ActiveCell.FormulaR1C1="text"
```

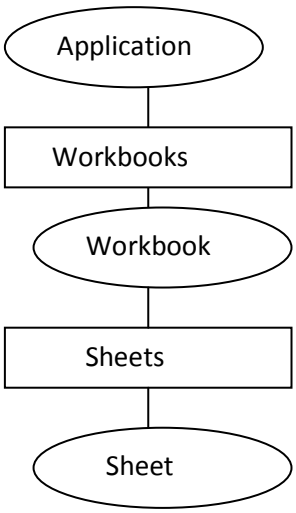


Рис. 1

В этих трех примерах использовались свойства активной ячейки, которые отображают ее содержимое.

UserForm1.TextBox1.Value="Иванов"

Свойство Value текстового поля с именем TextBox1, принадлежащее экранной форме UserForm1, принимает значение Иванов.

Методы представляют собой действия, выполняемые с объектом. Метод может использовать аргументы, уточняющие характер действия, которое необходимо выполнить.

Обращение к методу происходит также с помощью точечной нотации:

ИмяОбъекта.ИмяМетода.

Если метод использует аргументы, то они могут записываться либо в круглых скобках, либо без них, либо используя именованные аргументы:

ИмяОбъекта.ИмяМетода (аргумент1, аргумент2,…)

ИмяОбъекта.ИмяМетода аргумент1, аргумент2,…

Наиболее простым и очевидным способом является использование именованных аргументов:

ИмяОбъекта.ИмяМетода Имя1арг:=Значение1арг,
Имя2арг:=Значение2арг,…

Например, МояКнига.SaveAs Filename:=”Итоговая_таблица.xlsm”

Другие примеры использования методов приведены далее.

Range("A5").Select

Метод Select объекта Range применен для выделения ячейки A5.

Sheets(“Продажи”).Calculate

Метод Calculate применен для пересчета формул листа «Продажи».

1.3. Ссылка на объект

Для получения доступа к объекту можно обратиться к элементу коллекции напрямую, например:
Sheets(“Продажи”).Calculate.

Вторым способом обращения к объекту может быть присвоение ссылки на объект некоторой переменной, а затем уже использование этой переменной для обращения к объекту, его свойствам и методам.

Ссылка – это имя в коде программы, указывающее на данный объект.

Если переменной присваивается ссылка на объект, то ей должно предшествовать ключевое слово Set:

Set МойЛист= Sheets(“Продажи”)

В результате переменная МойЛист теперь ссылается на рабочий лист, имеющий название “Продажи” и может быть использована для обращения к свойствам и методам этого листа, например:

МойЛист. Calculate.

Рассмотрим использование ссылок на объекты на примере объектов Workbook и Worksheet, входящих в соответствующие коллекции.

Каждая открытая в Excel рабочая книга представлена объектом Workbook коллекции Workbooks.

Сначала нужно описать тип ссылки wb:

Dim wb as Workbook,

а затем использовать для создания новой рабочей книги с помощью метода Add:

Set wb=Workbooks.Add

Сохранение книги осуществляется методами Save и SaveAs:

wb.Save

wb.SaveAs FileName:=”mybook”

При необходимости нужно указать полный путь к сохраняемому или к открываемому файлу.

Расширение файла может быть добавлено автоматически.

Для открытия существующей книги используется метод Open:

Set wb=Workbooks.Open(”mybook.xlsx”)

Закрытие файла происходит с помощью метода Close с использованием при необходимости параметров SaveChanges, FileName и RouteWorkbook (имеющего значение True при рассылке файла):

wb.Close SaveChanges:=True

или не используя ссылки:

Workbooks(”mybook.xlsx”). Close SaveChanges:=True

Для ссылки на активную книгу можно воспользоваться ключевым словом ActiveWorkbook.

Добавление и удаление рабочих листов происходит с помощью методов Add и Delete объекта WorkSheet, соответственно:

Dim ws As Worksheet

Set ws = Worksheets.Add

wb.Worksheets("Лист2").Delete

Изменение имени листа происходит с помощью свойства листа Name:

ws.Name = "Total"

Копирование и перемещение листа происходит с помощью методов Copy и Move, которые имеют аргументы After или Before, указывающие на лист, после или перед которым будет вставлен копируемый или перемещаемый лист.

2. VBA. Основные понятия

Программирование в Excel основано на двух компонентах: *языке VBA* и *объектной модели Excel*, включающей совокупность объектов и их иерархию.

2.1.Процедуры в VBA

Базовой структурной единицей программных кодов VBA является *процедура*. Процедура – это совокупность кодов, выполняющих какую-либо задачу. Коды процедур располагаются в модулях. Простейшие программы VBA состоят из одной процедуры, более сложные программы могут содержать ряд взаимодействующих между собой процедур. Различаются процедуры – подпрограммы и процедуры – функции.

Процедура – подпрограмма содержит следующие части:

```
Sub name_sub()
Код процедуры
End sub
```

Здесь Sub и End sub – ключевые слова, ограничивающие процедуру, а name – имя процедуры, которое должно начинаться с буквы и не содержать разделителей.

Процедура – функция содержит следующие части:

```
Function name_function(apr1, apr2,   aprN)
Код функции
End Function
```

Можно выделить следующие типы процедур языка VBA:

- Макросы, - подпрограммы, обычно создающиеся автоматически;
- Подпрограммы, создающиеся вручную;
- Подпрограммы обработки события – создаются вручную и выполняются, когда происходит определенное событие (например, нажатие клавиши или кнопки мыши).
- Функции - создаются вручную;

Файл Excel, содержащий программный код должен иметь расширение .xlsm, т.е. сохраняться как «Книга Excel с поддержкой макросов».

Для выполнения макроса достаточно вызвать его по имени. Чтобы макрос автоматически выполнялся при открытии рабочей книги, ему нужно дать имя Auto_Open, а при закрытии - Auto_Close.

Функции не производят каких–либо действий с помощью команд Excel и не изменяют рабочее пространство, поэтому создаются только вручную и их функционирование заключается в преобразовании аргументов.

Для создания макроса нужно воспользоваться командой Разработчик – Код – Запись макроса, дать макросу имя, указать сочетание клавиш для его запуска и объект Excel, выбранный для его хранения, затем выполнить действия, которые должны быть записаны в макросе и остановить запись с помощью команды Разработчик – Код – Остановить запись.

Выполнить макрос можно с помощью команды Разработчик – Код – Макросы – Выполнить или воспользоваться выбранным сочетанием клавиш.

Коды процедур хранятся в рабочей книге Excel, но не составляют отдельных файлов, тем не менее предусмотрена возможность экспорта кодов в отдельный файл для последующего импорта в другие рабочие книги (расширение .bas для модулей, .frm – для форм).

2.2. Синтаксис VBA

Синтаксис – это совокупность правил, определяющих порядок использования и организацию элементов языка (например, *в русском языке предложение должно заканчиваться точкой*).

Коды VBA состоят из операторов. Каждый оператор располагается в одной строке. Однако если оператор не помещается в видимую область окна редактирования, то для переноса используется сочетание символов пробела и нижнего подчеркивания, затем нажимается клавиша Enter.

2.2.1. Константы

Константы – это данные, которые не изменяются в ходе выполнения программы.

В VBA различаются константы двух типов – литеральные и именованные.

Например, литеральные константы, представляющие собой число 4 и строку ”четыре”:

МоеЧисло=4

МояСтрока=”четыре”

Имена именованных констант объявляются с помощью ключевого слова Const:

Const МОЕИМЯ=”Piter”

Значением именованной константы является литеральная константа. Имя константы должно быть описательным (информативным), что упрощает чтение кодов программы. Согласно принятому соглашению для имен констант используются *прописные буквы*, тогда как другие имена, например имена переменных, набираются из сочетаний строчных и прописных букв. Важным преимуществом использования именованных констант является то, что при необходимости изменения значения константы, это делается только в единственном месте программы – там, где она была объявлена.

Имена констант, а также любые другие именованные элементы языка, подчиняются следующим правилам:

- начинаются с буквы;
- состоят не более чем из 255 символов;
- не совпадают с ключевыми словами VBA;
- не содержат пробелов и точек, а также символов !, @, #, &, %, $.

Существует ряд констант, которые имеют предопределенные (зарезервированные) имена, например, xlDouble, обозначающая двойную рамку по краям выделенной области или vbNo, соответствующая нажатию клавиши No в диалоговом окне. С их именами и значениями можно ознакомиться в справочной системе VBA.

2.2.2. Переменные

Переменные – это данные, которые изменяются в ходе выполнения программы. Прежде чем использовать переменную в программе, ее необходимо *объявить* и *инициализировать*.

Объявление переменной включает в себя ключевые слова Dim (от Dimension – размерность) и As, а также имя переменной и ее тип. Именно тип переменной определяет допустимые преобразования, а в случае попытки некорректного использования обеспечит соответствующие сообщения транслятора об ошибках.

Объявление переменной в кодах программы выглядит следующим образом:

Dim ИмяПеременной As тип

В одной строке можно объявить несколько переменных:

```
Dim ИмяПеременной1 As тип1, ИмяПеременной2 As тип2
```

2.2.3. Комментарии

Комментарии – это текст, который игнорируется транслятором, и не влияет на ход выполнения программы. В комментариях обычно записываются пояснения к программе, облегчающие ее понимание.

Первый способ создания комментария заключается в использовании апострофа перед текстом комментария. Комментарий может располагаться вначале строки и внутри нее.

```
‘ Это комментарий
Dim MyWb As Workbook  ‘ Это тоже комментарий
```

Второй способ создания комментариев - использование ключевого слова REM (от remark – замечание). Комментарий также может располагаться вначале строки и внутри нее.

```
REM  Это комментарий
Dim MyWb As Workbook   REM  Это тоже комментарий
```

2.2.4. Типы данных

Числа

Числа разделяются на целые (без дробной части) и с плавающей запятой (с дробной частью).

Для каждого из этих типов определены три категории, отличающиеся диапазоном допустимых значений, а для типов с плавающей запятой еще и точностью.

Обобщенная информация об этих типах представлена в таблице 1. При выборе типа нужно отдавать предпочтение тому типу, который требует меньше памяти, а также диапазон и точность которого будут достаточными для сохранения и обработки значений, хранящихся в описываемой переменной. Для представления денежных значений предназначен тип Currency.

Таблица 1

Тип	Категория	Диапазон	Точность
Byte	Целочисленный	От 0 до 255	Нет
Integer	Целочисленный	-32 768 до 32 767	Нет

(2 байта)			
Long (4 байта)	Целочисленный	-2 147 483 648 до 2 147 483 647	Нет
Single	С плавающей запятой	$-3{,}4*10^{38}$ до $3{,}4*10^{38}$	6 знаков
Double	С плавающей запятой	$-1{,}79*10^{308}$ до $1{,}79*10^{308}$	14 знаков
Currency	С плавающей запятой	$-9{,}22*10^{11}$ до $9{,}22*10^{11}$	4 знака

При объявлении числовых переменных им присваивается нулевое значение.

Строки

Переменные типа String используются для хранения и обработки строковых данных. Строки *переменной длины* могут включать любое количество символов. При объявлении переменных этого типа используется ключевое слово String:

```
Dim Имя As String
```

Другой тип строковых данных – строки *фиксированной длины*. Максимальная длина таких строк определяется в момент их объявления и может колебаться от 1 до 64000 символов. Для объявления строки фиксированной длины после ключевого слова String добавляется символ * и число, соответствующее максимальному количеству символов:

```
Dim Имя As String*12
```

Если переменной при этом присваивается значение, превышающее длину, указанную в объявлении, лишние символы отбрасываются.

В VBA встроены средства, позволяющие преобразовывать строковые данные в числовой эквивалент.

Даты

В VBA термином «дата» обозначаются как собственно значения даты, так и значения времени. Тип данных Date предназначен для обработки информации о дате и времени. Для указания значения даты его нужно заключить в символы # и представить в любом известном формате:

```
Dim день1 As Date, Dim день2 As Date, Dim день3 As Date
день1=#Январь 19, 2013#
```

```
день2=#19.1.2013#
день3=#19/1/2013#
```

Переменные - объекты

Переменные типа Object могут содержать ссылку на любой объект. Однако это приводит к тому, что изначально неизвестно, ссылка на объект какого типа будет использована в программе. Это станет известно только после инициализации переменной. Можно объявлять переменные с указанием конкретного типа объекта:

```
Dim  ws  As Worksheet
```

С помощью этого объявления создается переменная ws, которая будет ссылаться на объекты типа Worksheet. Само объявление не дает ссылки на конкретный объект и первоначально переменная имеет значение Nothing. Для инициализации этой переменной необходимо присвоить ей ссылку на конкретный объект. В нашем случае новый рабочий лист создается с помощью метода Add объекта Worksheet. Таким образом, инициализация переменной будет иметь вид:

```
Set  ws= Worksheets. Add
```

Ключевое слово Set должно использоваться каждый раз, когда переменной присваивается ссылка на объект.

Как правило, используются ссылки на объекты, составляющие коллекции. Если же объект не входит в коллекцию, то инициализация выглядит так:

```
Set  переменнаяОбъект=New  названиеОбъекта
```

Boolean

Переменные, объявленные с использованием типа Boolean, могут принимать одно из двух значений True (истина) и False (ложь). Такие переменные называются логическими, в момент объявления им присваивается значение False. Например:

```
Dim  Flag  As  Boolean
Flag=True
```

Variant

Переменные, объявленные с помощью типа Variant, могут принимать значения практически всех типов за исключением типа фиксированной строки и определенного пользователем. Недостатком этого типа является то, что для обработки данных требуется больше системных ресурсов, а для хранения данных – больше памяти. Данный тип рекомендуется использовать в тех случаях, когда на разных этапах выполнения программы данные могут интерпретироваться как числовые и как текстовые или для сохране-

ния данных из ячеек листа, когда заранее неизвестно, значения каких именно типов могут содержаться в этих ячейках.

2.2.5. Область видимости переменных

Переменные, объявленные в программе VBA, остаются доступными для использования только в пределах определенной части данной программы – области видимости переменных.

В VBA различают три уровня видимости.

- Уровень процедуры
- Уровень модуля
- Уровень проекта

Переменная на уровне процедуры объявляется внутри процедуры с помощью выражения Dim. Чтобы создать переменную с областью видимости на уровне модуля, нужно объявить ее внутри этого модуля за пределами всех процедур. Такой уровень видимости необходим, когда доступ к данной переменной необходим сразу нескольким процедурам этого модуля. Тот же эффект может быть достигнут в том случае, если вместо слова Dim используется ключевое слово Private:

```
Private Total As Integer
```

Область видимости на уровне проекта обеспечивается объявлением переменной с помощью ключевого слова Public:

```
Public Total As Integer
```

3. Основы VBA

3.1. Операции

Преобразование данных происходит с помощью выражений – совокупности операндов, связанных *операциями*.

3.1.1. Операция присваивания

Для присвоения значений свойствам или переменным используется операция присваивания, например, X=2 или X=X+2. При этом вначале вычисляется выражение в правой части и результат присваивается переменной, находящейся в левой части. При попытке присвоения числовой переменной текстового значения или значения, находящегося за пределами диапазона значений данного типа, переменной типа Boolean значений, отличных от True или False или объектной переменной необъектного значе-

ния возникает ошибка. Если строковой переменной присваивается числовое значение, то оно преобразуется в текстовый эквивалент. Если значение с плавающей точкой присваивается переменой целого типа, то оно округляется до ближайшего целого.

3.1.2. Числовые операции

Числовые операции выполняют действия над числами. Результатом выполнения выражений с числовыми операциями является число. К числовым операциям ним относятся +,-,*,/,^ - сложение, вычитание, умножение, деление и возведение в степень. К числовым операциям относится также *целочисленное деление* \, возвращающее целую часть результата. Так, результат 12\5 дает результат 2, а также *остаток от деления* или модуль (mod). Так, 12 mod 2 дает результат 0.

3.1.3. Логические операции

В VBA поддерживается шесть логических операций, приведенных в таблице 2. Результатом выполнения выражений с логическими операциями является значение True или False.

Таблица 2

Операция	Пример	Значение True	Значение False
And (И)	A And B	A и B имеют значение True	A или B, или оба имеют значение False
Or (Или)	A Or B	A или B, или оба имеют значение True	A и B имеют значение False
XOr (Исключающее или)	A XOr B	A и B имеют разные значения	A и B имеют одинаковые значения
Eqv (Эквивалентность)	A Eqv B	A и B имеют одинаковые значения	A и B имеют разные значения
Imp (импликация)	A Imp B	A имеет значение False, B имеет значения True или False, A и B име-	A имеет значение True, а B имеет значение False

		ют значение True	
Not He)	Not A	A имеет значение False	A имеет значение True

3.1.4. Строковые операции

В VBA существует один строковый оператор – оператор конкатенации &. Так, выражение “Micro”&”soft” дает результат “Microsoft”. Результатом выполнения выражений со строковыми операциями является строка.

3.1.5. Операции сравнения

Операции сравнения служит для сравнения выражений. Результатом выполнения выражений сравнения с операциями сравнения является значение True или False.

Операндами выражений сравнения являются числа либо строки. Сравнение строк основано на значениях кода ASCII для символов, из которых они состоят. К операциям сравнения относятся следующие:

= равно, > больше, < меньше, >= больше или равно, <= меньше или равно, <> не равно.

Для сравнения строк используется также операция Like: *строка* Like *шаблон*.

В качестве шаблона можно использовать символы * для обозначения любой последовательности символов, ? –любого отдельного символа, # - любой цифры.

3.1.6. Приоритет операций

Правила приоритета операций VBA определяют порядок выполнения операций в процессе вычисления выражений. В VBA установлен следующий приоритет операций:

^ возведение в степень

* умножение и /деление

\ целочисленное деление

mod - остаток от деления

+ сложение и – вычитание

& конкатенация строк

Для изменения порядка вычислений нужно использовать круглые скобки.

3.2. Управляющие конструкции

VBA содержит операторы, которые могут управлять ходом выполнения программы. Эти операторы известны как управляющие конструкции. Их можно разделить на две группы:

-условные операторы (If…Then и Select Case)

-операторы цикла (Do…Loop, For…Next и For Each Next).

3.2.1. Оператор If…Then

Этот оператор используется в программе для выполнения блока инструкций, если заданное логическое выражение (условие) имеет значение True (т.е. выполняется). Оператор будет иметь вид If условие Then оператор, если блок инструкций состоит только из одного оператора. В противном случае оператор имеет вид:

```
If условие Then
Блок инструкций
End If
```

Если предусмотрено выполнение блока инструкций при значении условия False, то эти инструкции записываются после ключевого слова Else и оператор выглядит так:

```
If условие Then
Блок инструкций1
Else
Блок инструкций2
End If
```

Для проверки нескольких условий добавляется ключевое слово ElseIf. Выполняется тот блок инструкций, для которого условие окажется истинным, если ни одно условие не возвращает значения True, то выполняется блок инструкций, следующий за ключевым словом Else:

```
If условие1 Then
Блок инструкций1
ElseIf условие2
Блок инструкций2
…
```

```
ElseIf  условиеN Then
Блок инструкцийN
Else
Блок инструкций-Else
End  If
```

Пример 1.

Пусть в ячейки A2, B2 и C2 внесены фамилия, пол и возраст человека. В зависимости от его возраста и пола определяется статус: школьник – от7 до 17 лет, студент – от 18 до 22 лет, работающий – от 23 до 55 или 60 лет, соответственно, для женщин и мужчин и пенсионер – старше 55 или 60 лет также в зависимости от пола. В клетку D2 нужно поместить определяемый статус.

```
Public Sub Data_If()
Dim status As String
Dim vozrast As Byte
Dim sex As String * 1

Sheets("Лист1").Activate
Range("a2").Value = InputBox("Введите  имя")
sex = InputBox("Введите пол")
Range("b2").Value = sex
vozrast = InputBox("Введите возраст")
Range("c2").Value = vozrast

If vozrast > 7 And vozrast <= 17 Then
   stutus = "школьник"
   ElseIf vozrast > 17 And vozrast <= 22 Then
    stutus = "студент"
   ElseIf vozrast > 22 And ((vozrast < 55 And sex = "ж") Or (vozrast < 60 And
sex = "м")) Then
      stutus = "работающий"
   ElseIf (vozrast > 55 And sex = "ж") Or (vozrast > 60 And sex = "м") Then
    stutus = "пенсионер"
End If
Range(“d2").Value = stutus
```

```
End Sub
```

3.2.2. Оператор Select Case

Оператор Select Case дает возможность выбрать один из нескольких вариантов в зависимости от значения проверяемого выражения. Оператор Select Case имеет следующий синтаксис:

```
Select Case выражение
Case выражение1
Блок1
Case выражение2
Блок2
…
Case выражениеN
БлокN
Case Else
Блок Else
End Select
```

Конструкция работает следующим образом. Вначале вычисляется выражение, которое может быть числом, строкой или булевым значением. Значение этого выражения сравнивается с выражением выражение1, выражение2 и так далее. При нахождении совпадения этих выражений выполняется соответствующий блок операторов. Если совпадение не найдено, выполняется Блок Else. В качестве выражения может быть определен также диапазон значений с помощью ключевого слова To (например, 7 To 17), а также выражение сравнения с ключевым словом Is (например, Is<100). Также допускается использовать несколько выражений, записанных через запятую (например, Case 25, Is>100, 1 To 5). В этом случае совпадение ищется с одним из этих выражений.

Пример 2.

В упрощенных условиях предыдущего примера используем конструкцию Select Case. Для простоты будем считать статусом людей старше 55 лет – «пенсионер», людей, которым исполнилось 100 лет – «долгожитель», а детей до 7 лет – «дошкольник». В клетку D2 нужно поместить определяемый статус.

```
Public Sub Data_case()
```

```
Dim status As String
Dim vozrast As Byte

Sheets("Лист1").Activate
Range("a2").Value = InputBox("Введите фамилию")
vozrast = InputBox("Введите возраст")
Range("c2").Value = vozrast
Select Case vozrast
Case Is<7
status = "дошкольник"
Case 7 To 17
status = "школьник"
Case 18 To 22
status = "студент"
Case 23 To 55
status = "работающий"
Case Is > 55
status = "пенсионер"
Case 100
status = "долгожитель"
Case Else
status = "ошибка ввода"

End Select
Range(“d2").Value = stutus

End Sub
```

3.2.3. Оператор цикла For…Next

Оператор For…Next выполняет блок инструкций заданное число раз. Синтаксис этого оператора следующий:

```
For параметр_цикла=начальное_значение To конечное_значение
Блок инструкций (тело цикла)
Next параметр_цикла
```

Параметр цикла – это счетчик циклов, который обычно имеет тип целого числа, начальное_значение и конечное_значение – выражения, определяющие начальное и конечное значения параметра цикла. Значение параметра цикла при прохождении цикла увеличивается на единицу и так продолжается до тех пор, пока параметр цикла не достигнет конечного значения. Чтобы применить шаг изменения параметра цикла, отличный от единицы, применяется ключевое слово Step и его значение, и оператор выглядит следующим образом:

```
For параметр_цикла=начальное_значение To конечное_значение Step шаг
Блок инструкций (тело цикла)
Next параметр_цикла
```

Если шаг отрицательный, то начальное значение должно быть больше конечного значения.

Если выражение, определяющее шаг, возвращает значение с плавающей точкой, то параметр цикла должен быть объявлен как переменная одного из типов, соответствующих значениям с плавающей точкой.

Например, значения параметра цикла должны меняться от 5 до 1 с шагом -0.5. В этом случае фрагмент программы будет выглядеть следующим образом:

```
Dim Count As Single
For  Count=5  To  1 Step -0.5
Блок инструкций (тело цикла)
Next Count
```

Для досрочного выхода из цикла используется оператор Exit For.

3.2.4. Операторы Do…Loop

Операторы Do...Loop используются для циклического выполнения блока операторов. Повторение блока операторов тела цикла происходит в зависимости от истинности или ложности значения условия, которое является неотъемлемой частью этого типа операторов. Синтаксис операторов Do...Loop приведен в таблице 3.

Таблица 3

Синтаксис	Действие	Комментарии
Do While условие Блок операторов Loop	Повторяет выполнение блока операторов, пока условие *остается* истинным	Если условие изначально ложно, цикл не выполняется ни разу
Do Until условие Блок операторов Loop	Повторяет выполнение блока операторов, пока условие *не станет* истинным	Если условие изначально истинно, цикл не выполняется ни разу
Do Блок операторов Loop While условие	Выполняет блок операторов один раз и затем повторяет, пока условие *остается* истинным	Блок операторов обязательно выполняются один раз (до проверки условия)
Do Блок операторов Loop Until условие	Выполняет блок операторов один раз и затем повторяет, пока условие *не станет* истинным	Блок операторов обязательно выполняются один раз (до проверки условия)

Первые два вида цикла называются циклами с предусловием, а последние два – с постусловием.

Например, цикл должен продолжаться, пока пользователь не введет в диалоговое окно InputBox число 10.

Используем цикл While с предусловием:

```
Dim Number As Byte
Do While Number<>10
Number=InputBox(“Введите число 10”)
Loop
```

Используем цикл Until с предусловием:

```
Dim Number As Byte
Do Until Number=10
Number=InputBox(“Введите число 10”)
```

```
Loop
```

Используем цикл While с постусловием:

```
Dim Number As Byte
Do
Number=InputBox(“Введите число 10”)
Loop While Number<>10
```

Используем цикл Until с постусловием:

```
Dim Number As Byte
Do
Number=InputBox(“Введите число 10”)
Loop Until  Number=10
```

Досрочное прекращение цикла Do…Loop возможно при использовании оператора Exit Do.

4. Среда программирования VBA

4.1. Редактор VBA

Редактор VBA является средством написания и редактирования кодов создаваемых программ, их отладки, выполнения, защиты от несанкционированного доступа и пользования справочной системой.

Редактор VBA является частью инсталляционного пакета Excel и вызывается командой Сервис – Макрос – Редактор Visual Basic или клавишами Alt+F11.

Окно редактора представлено на рис. 2.

Совокупность программных кодов, сохраненных в одной рабочей книге, образует *проект* VBA.

В верхнем левом углу редактора VBA находится окно *проектов* (Project Explorer), отображающее иерархическую структуру открытых в данный момент проектов VBA.

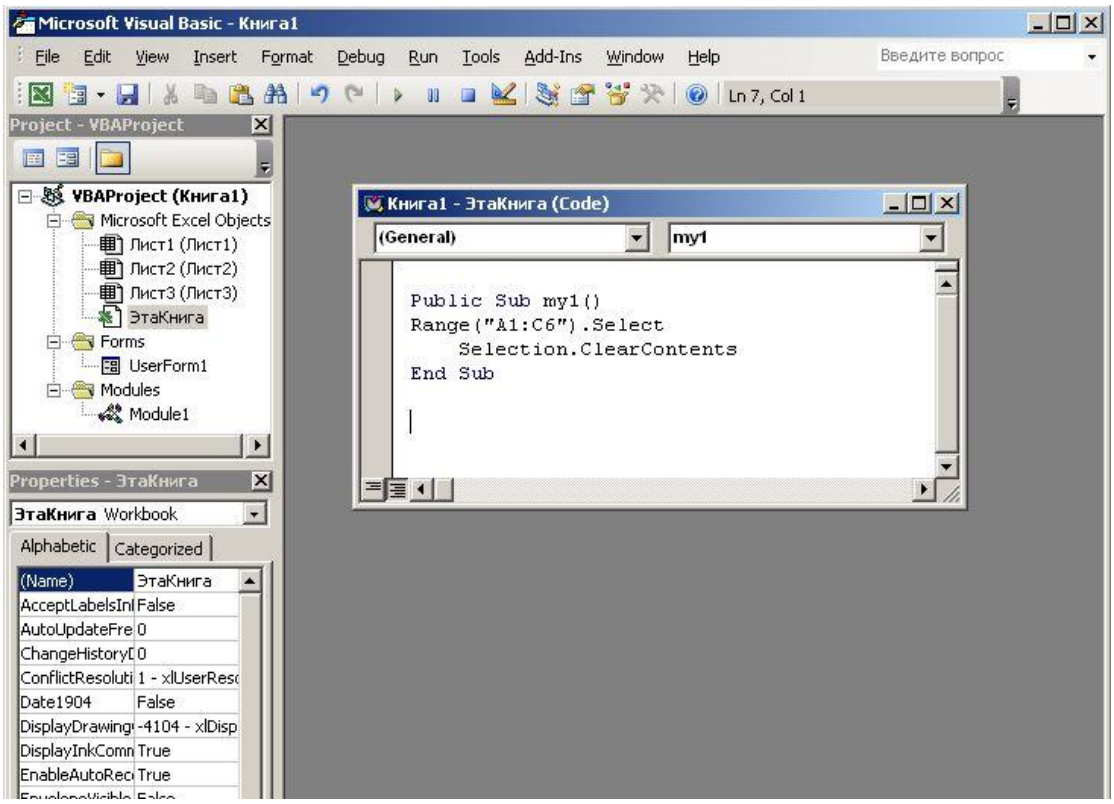


Рис. 2

Каждый проект содержит следующие компоненты:

- Набор объектов MS Excel, включающий по одному объекту для каждого рабочего листа, плюс один специальный объект, имеющий название "Эта книга". Эти компоненты всегда присутствуют в проекте.
- Один или несколько модулей.
- Одна или несколько экранных форм.
- Один или несколько модулей классов.

Последние три компонента присутствуют в тех случаях, когда пользователь их добавляет специально.

Любой из компонентов может содержать в себе коды. Коды, расположенные в рабочем листе, должны быть непосредственно связаны с этим листом. Если коды относятся ко всему проекту, то они располагаются в объекте "Эта книга". В модуле могут располагаться не только коды макросов, но и другие коды, по усмотрению разработчика. Место расположения кодов не принципиально, поскольку можно обращаться к кодам независимо от того, где они находятся.

Для того чтобы открыть окно редактирования кодов какого-либо компонента, нужно дважды щелкнуть мышью на его названии. Окно редакти-

рования откроется справа от окна проектов в рабочей области редактора VBA.

В левой нижней части находится окно свойств (Properties), в котором отображаются свойства выделенного объекта.

4.2. Отладчик кода

Фрагмент кода, который является причиной неправильного функционирования программы, содержит ошибки.

Различают ошибки *времени выполнения* программы (runtime error) и *синтаксические* и *логические* ошибки (bug), не позволяющие получить правильный результат. Ошибки выполнения программы приводят к остановке выполнения программы и получению сообщения о коде ошибки и краткому описанию, по которому можно установить ее причину. В диалоговом окне, содержащем код и описание ошибки, также находятся кнопки End и Debug, нажав на которые можно либо завершить работу программы, либо остановить выполнение программы с выделением той строки кода, в которой обнаружена ошибка.

Синтаксические ошибки, нарушающие правила VBA, обычно обнаруживаются редактором и выделяются красным цветом. Такие ошибки разработчик может устранить в процессе набора кодов программы.

Логические дефекты программ являются причиной неправильной работы программы и получения непредсказуемых результатов. Отладчик кода редактора VBA предназначен для поиска и устранения таких ошибок.

В большинстве случаев причинами ошибок, приводящих к неверным результатам, является следующее:

- неверна логика программы (ход ее выполнения);
- переменные принимают неправильные значения.

4.2.1. Пошаговое выполнение кода

Обычно программный код выполняется слишком быстро для определения источника сбоя.

Отладчик поддерживает пошаговое выполнение кода. Режим пошагового выполнения кода вызывается командой Debug – Step Into (Отладка – Пошаговое выполнение) или нажатием клавиши F8.

При каждом нажатии клавиши F8 будет выполняться одна строка кода. Она выделяется желтым цветом. Определив проблемный участок, остановить выполнение кода можно командой Run – Reset (Выполнить – Сброс).

В режиме пошагового выполнения программы отслеживать значения переменных можно при наведении на эти переменные курсора мыши. Текущее значение переменной отобразится в небольшом окне рядом с курсором мыши.

Пошаговый режим позволяет изменять порядок выполнения строк кода. Для того чтобы выполнить нужный фрагмент кода, нужно перетащить желтую стрелку слева от строки в требуемое место программы и продолжить пошаговое выполнение программы.

4.2.2. Точки прерывания

В программном коде зачастую содержится значительное число строк. Чтобы перейти к проблемному участку, не обязательно выполнять весь код. Достаточно создать точку прерывания в нужном месте программы и ее выполнение будет остановлено перед проблемным участком.

Для создания точки прерывания нужно щелкнуть мышью на полосе слева от строки кода, перед выполнением которой нужно сделать прерывание или выполнить команду Debug – Toggle Breakpoint (Отладка – Установить точку прерывания). Строка кода будет выделена красно-коричневым цветом, а слева от нее появится маркер такого же цвета в виде точки. После выбора команды Run – Run Sub/UserForm (Выполнить – Выполнить подпрограмму/Пользовательскую форму) (F5) выполнение программы остановится на границе точки прерывания, а соответствующая строка будет выделена желтым цветом. Далее можно продолжить выполнение программы в пошаговом режиме (F8).

Завершив отладку кода, следует удалить все точки прерывания. Для удаления нужно щелкнуть мышью на маркере прерывания, а чтобы удалить все точки прерывания – выбрать команду меню Debug – Clear all Breakpoints (Отладка – Удалить все точки прерывания).

4.2.3. Вычисление значений выражений и переменных с помощью окна Immediate (Быстрое выполнение)

Для открытия окна Immediate нужно воспользоваться командой View – Window Immediate (Вид – Окно быстрого выполнения) (Ctrl+G) (рис. 3).

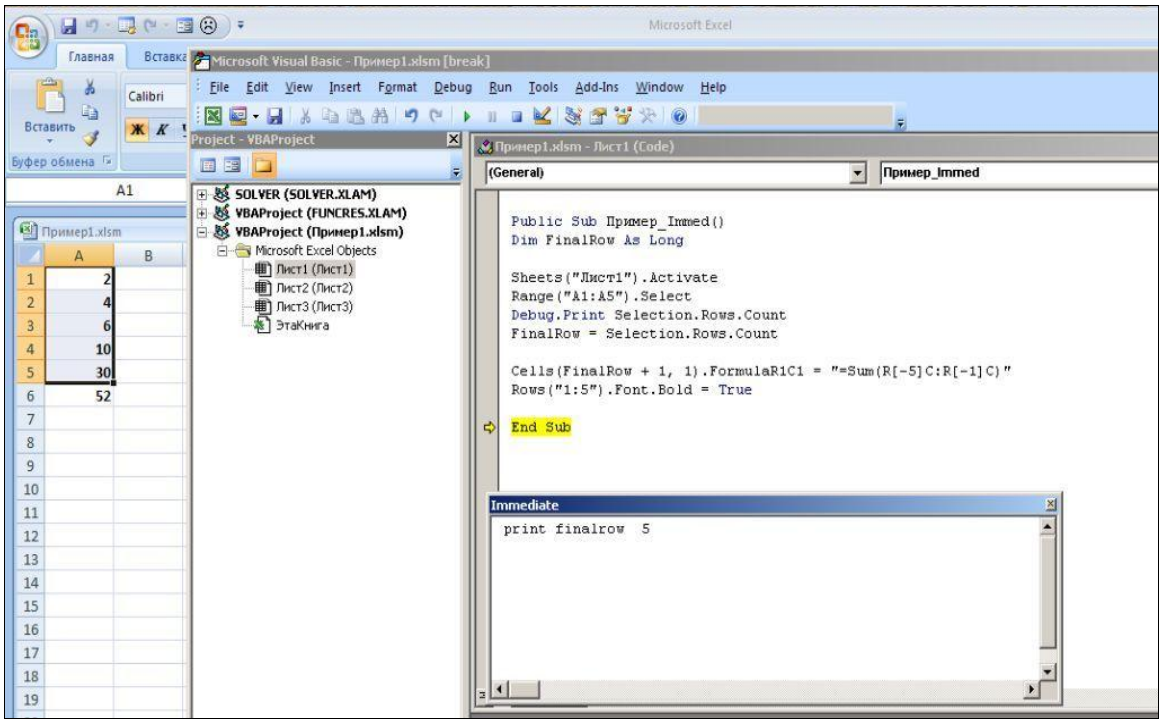


Рис. 3

Окно Immediate обычно располагается под окном с программным кодом. В этом окне помещают конструкцию Print выражение, где в качестве выражения может быть использованы разнообразные коды, помогающие программисту прояснить ситуацию. Результат появляется после нажатия клавиши Enter, когда в режиме отладки данное выражение выполнилось. Рассмотрим пример программного кода:

```
Public Sub Пример_Immed()
Dim FinalRow As Long

Sheets("Лист1").Activate
Range("A1:A5").Select
‘Debug.Print Selection.Rows.Count
FinalRow = Selection.Rows.Count
Cells(FinalRow + 1, 1).FormulaR1C1 = "=Sum(R[-5]C:R[-1]C)"
Rows("1:5").Font.Bold = True
End Sub
```

В данном фрагменте выделяется диапазон ячеек A1:A5 на активном листе Лист1, определяется последняя строка диапазона и в следующую ячейку записывается формула суммирования значений в клетках A1:A5. Затем к строкам с первой по пятую применяется формат Bold. На рис. 3 показано окно Immediate и значение переменной FinalRow, определяющей по-

следнюю строку диапазона. (Для диапазона, начинающегося с ячейки A1, последняя строка диапазона может быть определена свойством Rows.Count, которое в общем случае определяет число строк в диапазоне).

Для получения информации в окне Immediate можно также поместить в текст программы конструкцию Debug.Print выражение как показано на рис. 3.

Увидеть значение переменной или других выражений можно также при наведении мыши на переменную или выражение в коде и удерживании ее в течение нескольких секунд.

4.2.4. Окно Watches

Окно Watches (Просмотры) позволяет отслеживать значение любого выражения во время выполнения кода. Для активизации окна Watches нужно выполнить команду Debug – Add Watch (Отладка – Добавить окно просмотра). Отследим значение переменной FinalRow и значение в клетке A6 из предыдущего примера. Для этого в текстовое поле Expression (выражение) нужно ввести конструкцию Selection.Rows.Count и нажать на OK.

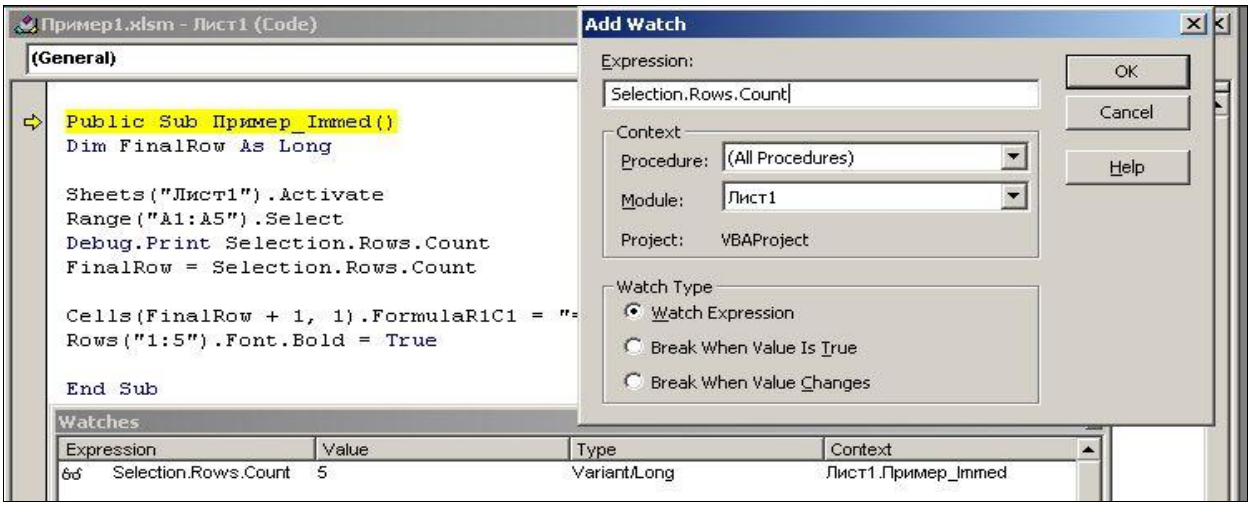


Рис. 4

В режиме пошагового выполнения программы при остановке на соответствующей строке кода в окне Watches будет отображено значение переменной FinalRow (рис. 4) После получения результата нужно отредактировать значение в поле Expression и записать туда выражение Range("A6"). Результат показан на рис. 5.

В диалоговом окне Add Watch в секции Watch Type установка переключателей Break When Value Is True и Break When Value Change обеспечит

переход в режим прерывания при достижении выражения значения True или при изменении состояния (элемента управления) или значения.

Возможность быстрого просмотра (Quick Watch) используется с целью быстрого просмотра значений, принимаемых переменной или выражением. Когда программа находится в пошаговом режиме, нужно поместить курсор на интересующее программиста выражение или переменную и выбрать команду Debug - Quick Watch (shift +F9) (Отладка – Быстрый просмотр). В открывшемся диалоговом окне Quick Watch будет отображено текущее значение переменной или выражения.

Щелчок на кнопке Watch добавит переменную или выражение в окно Watches в качестве обычного наблюдаемого выражения.

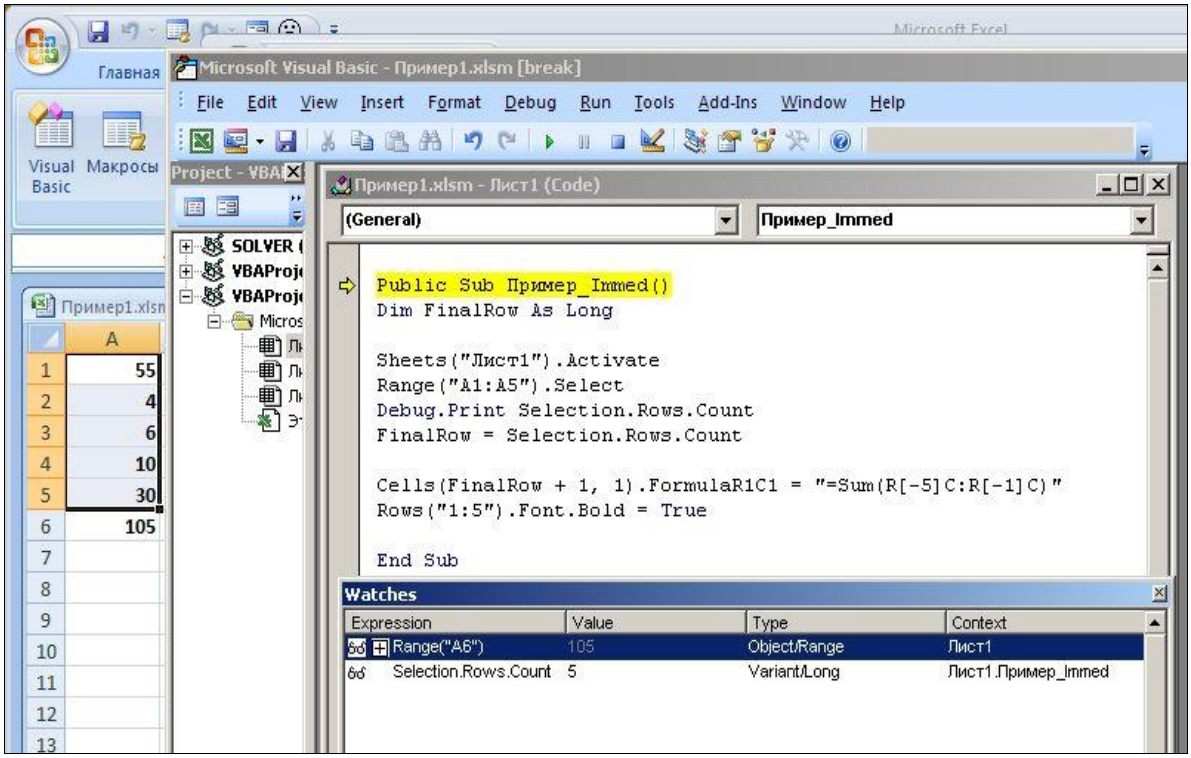


Рис. 5

Редактор VBA позволяет отслеживать в окне Watches состояние объектов, например, объекта Range, как показано на рис. 6. Щелкнув на значке «плюс» можно просмотреть все свойства выбранного объекта. Возле тех объектов, которые составляют коллекции, также находится знак «плюс», щелкнув на котором можно получить более детальную информацию об объекте.

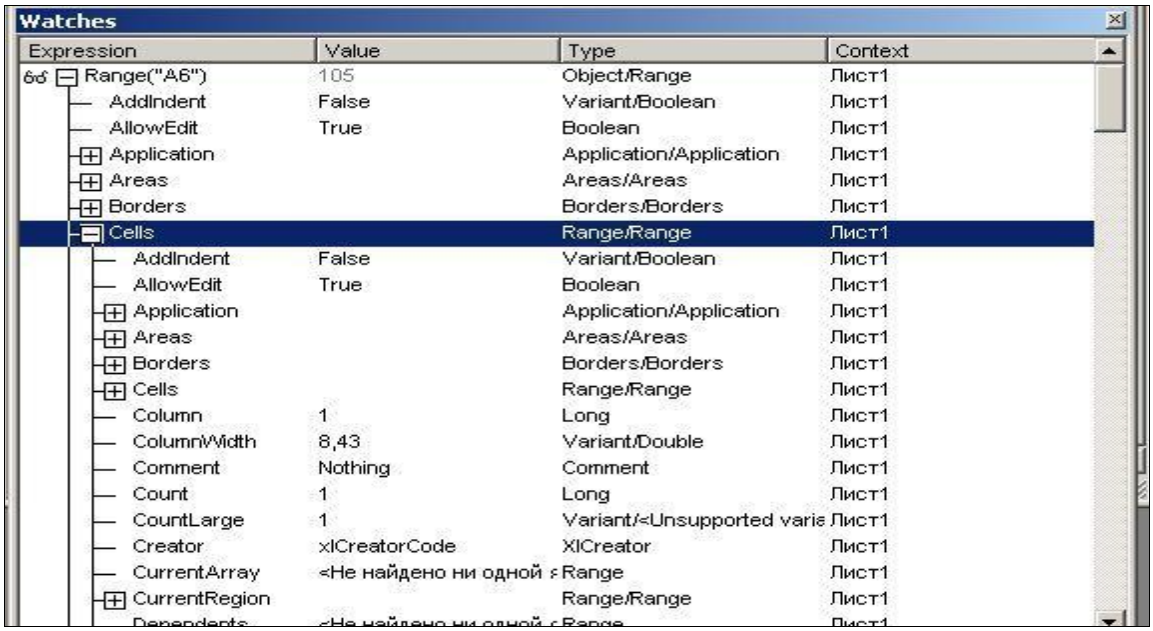

Watches
Expression
Value
Type
Context
Range("A6")
105
Object/Range
Лист1
AddIndent
False
Variant/Boolean
Лист1
AllowEdit
True
Boolean
Лист1
Application
Application/Application
Лист1
Areas
Areas/Areas
Лист1
Borders
Borders/Borders
Лист1
Cells
Range/Range
Лист1
AddIndent
False
Variant/Boolean
Лист1
AllowEdit
True
Boolean
Лист1
Application
Application/Application
Лист1
Areas
Areas/Areas
Лист1
Borders
Borders/Borders
Лист1
Cells
Range/Range
Лист1
Column
1
Long
Лист1
ColumnWidth
8,43
Variant/Double
Лист1
Comment
Nothing
Comment
Лист1
Count
1
Long
Лист1
CountLarge
1
Variant/<Unsupported varia
Лист1
Creator
xlCreatorCode
XlCreator
Лист1
CurrentArray
<Не найдено ни одной з
Range
Лист1
CurrentRegion
Range/Range
Лист1

Рис. 6

5. Работа с текстом и датами

При написании программ часто возникает необходимость в обработке данных, представленных в виде текста. В VBA предусмотрен специальный тип данных String, предназначенный для хранения текстовой информации. Кроме того имеется целый ряд функций позволяющих выполнять вывод и преобразование текстовых данных.

5.1. Функция MsgBox

Функция MsgBox позволяет вывести текстовое сообщение на экране. Сообщение расположено в диалоговом окне, имеющем заданные программистом элементы управления. Функция возвращает значение, которое зависит от реакции пользователя (нажатия соответствующих кнопок).

Функция MsgBox имеет следующий синтаксис:

MsgBox("*сообщение*", *кнопки*, "*заголовок*"), где

сообщение – выводимое в окне сообщение,

кнопки – константа, определяющая какие кнопки и значки отображаются в окне, а также какая кнопка считается выбранной по умолчанию

заголовок – текст, который отображается в заголовке окна.

Для аргумента "*кнопки*" определены именованные константы, которые приведены в таблице 1.

Таблица 1

Константа	Значение	Отображаемые кнопки и значки
vbOKCancel	1	OK и Cancel
vbAbortRetryIgnore	2	Abort, Retry, Ignore
vbYesNoCancel	3	Yes, No, Cancel
vbYesNo	4	Yes, No
vbRetryCancel	5	Retry, Cancel
vbCritical	16	Значок критического сообщения
vbQuestion	32	Значок вопроса
vbExclamination	48	Значок предупреждающего сооб-

		щения
vbInformation	64	Значок информационного сообщения
vbDefaultButton2	256	Фокус по умолчанию на второй кнопке
vbDefaultButton3	512	Фокус по умолчанию на третьей кнопке

Функция возвращает целое значение, позволяющее определить, какая кнопка была нажата пользователем. Значения, которые может вернуть функция, представлены следующими константами, приведенными в таблице 2.

Таблица 2

Константа	Значение
vbOK	1
vbCancel	2
vbAbort	3
vbRetry	4
vbIgnore	5
vbYes	6
vbNo	7

Использование функции MsgBox показано в Примере 1. В диалоговом окне с заголовком "Выход" выводится сообщение "Завершить работу? ", отображаются кнопки Yes и No, а также вопросительный значок как показано на рис. 7.

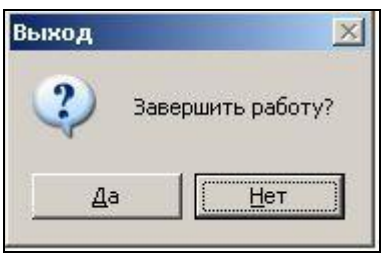


Рис. 7

Пример 1

```
Dim msg As Byte
msg=MsgBox("Завершить работу?", VbYesNo Or vbQuestion Or vbDefaultButton2, "Выход")
If msg=vbYes Then
‘ Код выхода из программы с сохранением данных
Else
‘ Код выхода из процедуры
End If
```

Если возвращаемое значение функции MsgBox не важно, то можно вызывать эту функцию как подпрограмму для вывода сообщений, например, MsgBox "Ваш возраст " & Age & "лет", где Age – переменная, содержащая значение возраста.

5.2. Функция InputBox

Окно для ввода данных активизируется с помощью функции InputBox. Функция возвращает введенные в текстовое поле данные, которые могут быть числом, текстом или формулой. Функция InputBox имеет следующий синтаксис:

InputBox("*сообщение*", "*заголовок*", "*умолчание*", *поз*_x, *поз*_y), где *сообщение* – сообщение в диалоговом окне, *заголовок* – заголовок окна, *умолчание* – сообщение в текстовой строке окна, *поз*_x и *поз*_y – координаты окна относительно верхнего левого угла экрана (в твипах – twip, в одном дюйме 1440 твипов).

В примере 2 показано использование функции InputBox для ввода данных в таблицу, состоящую из трех столбцов и пяти строк, расположенную в диапазоне A1:C5 и имеющую в A1:C1 заголовки столбцов ИМЯ РОСТ ВЕС.

Пример 2

```
Public Sub inp_1()
Dim counter_row As Byte, counter_col As Byte
Dim msgvar As String, my_var As Variant
'формирование таблицы из 5 строк с заголовками столбцов ИМЯ РОСТ ВЕС
For counter_col = 1 To 3
   Select Case counter_col
```

```
    Case 1
    msgvar = "Введите имя"
    Case 2
    msgvar = "Введите рост"
    Case 3
    msgvar = "Введите вес"
    End Select
      For counter_row = 2 To 5
        my_var = InputBox(msgvar)
        Cells(counter_row, counter_col).Value = my_var
      Next counter_row
Next counter_col
End Sub
```

5.3. Преобразование строк

5.3.1. Функции LCase и UCase

Данные функции преобразуют все символы строки к верхнему регистру (UCase) или к нижнему регистру (LCase). Они имеют следующий синтаксис:

LCase(строка) и UCase(строка), где строка – любое строковое выражение.

Например, LCase(“STRing”) вернет значение string, а UCase(“STRing”) вернет значение STRING.

5.3.2. Функция Val

Функция Val преобразует строку в ее числовой эквивалент. Синтаксис функции: Val(строка).

Функция возвращает числовое значение, соответствующее той части строки, которая может быть идентифицирована как число. Числовыми символами считаются цифры, символы + и -, десятичная точка. (Запятая и знак $ к числовым символам не относятся, пробелы и знаки табуляции игнорируются).

Например, Val(“Мойка, 65”) вернет значение 0
Val(“65, Мойка ”) вернет значение 65.

5.3.3. Функция Str

Функция Str преобразует числовое значение в строку. Она возвращает значение типа Variant, содержащее текстовое представление числа. Ее синтаксис: Str(значение),

Например, Str(15.5) вернет строку 15.5, причем строка будет начинаться с пробела, если число – положительное.

5.3.4. Функция Len

Функция Len возвращает длину строки (количество символов в строке). Синтаксис функции Len(строка). Например, Len(Str(15.5)) вернет значение 5, Len("ГУТ") вернет значение 3.

5.4. Извлечение или изменение фрагментов строк

5.4.1. Функции Left и Right

Функции извлекают заданное количество символов с левого или правого конца строки. Синтаксис функций следующий: Left(строка, длина) Right(строка, длина), где *строка* – исходная строка, а *длина* – количество символов, которые необходимо извлечь. Например, Left("ГУТ им. М.А.Бонч-Бруевича", 3) вернет строку ГУТ, а функция Right("ГУТ им. М.А.Бонч-Бруевича", 8) вернет строку Бруевича.

5.4.2. Функция Mid

Функция Mid извлекает символы из строки. Ее синтаксис: Mid(строка, начало, длина), где *строка* – исходная строка, *начало* – позиция символа, с которого необходимо извлекать фрагмент строки, *длина* – количество извлекаемых символов. Если *длина* опущена, то извлекаются все символы от позиции, заданным аргументом *начало*. Например, Mid("М.А.Бонч-Бруевич", 5,4) вернет строку Бонч.

Выражение Mid(строка1, начало, длина)=строка2 позволит заменить часть исходной строки другой строкой. Например, s="Nokia123", Mid(s, 6, 3)= "3 ". При этом исходная строка будет заменена строкой Nokia3 . Замыкающие пробелы можно убрать функцией RTrim.

5.5. Работа со значениями ASCII

Во внутренних вычислениях компьютеры используют числовое (двоичное) представление данных. Каждому символу присваивается свой двоичный код. Первым широко распространенным стандартом был символьный набор ASCII (American Standard Code for Information Interchange). В нем для представления символов латиницы, чисел и других знаков используются значения от 0 до 128. Затем он был расширен до 255 символов для представления букв других алфавитов. Наиболее широко совокупность символов, с которыми работает компьютер, представлена в кодировке Unicode, которая поддерживает 65535 комбинаций, включающих национальные алфавиты, а также математические и графические символы. В VBA предусмотрены функции для работы с кодами ASCII и Unicode.

5.5.1. Функция Asc

Функция Asc возвращает код ASCII заданного символа. Ее синтаксис: Asc(символ), например, Asc(“a”) вернет код 97.

В коде ASCII цифры от 0 до 9 имеют коды от 48 до 57, строчные буквы латиницы – от 97 до 122, прописные – от 65 до 90. Непечатаемые символы имеют коды от 0 до 32.

5.5.2. Функция Chr

Функция Chr возвращает символ по его коду. Ее синтаксис: Chr(число), например,Chr(97) вернет символ «a» .

5.6. Сравнение строк

5.6.1. Функция StrComp

Функция StrComp сравнивает две строки. Ее синтаксис: StrComp(стр1, стр2, сравнение), где *cmp1* и *cmp2* – сравниваемые строки, а *сравнение* – способ сравнения (значение 0 означает бинарное сравнение, с учетом регистра, а значение 1 – сравнение без учета регистра). Если *cmp1* меньше *cmp2*, то функция вернет значение -1, если строки одинаковы – то 0, а если *cmp1* больше *cmp2*, то 1. Например, стр1=«Name», стр2=«name» StrComp(стр1, стр2, 0) вернет значение -1, поскольку N(78)<n(110).

5.7. Функции даты и времени

Системные значения даты и времени служат функции
Now – возвращает значения системной даты и времени как значения типа Date,
Date – возвращает значение системной даты,
Time – возвращает значение системного времени.

5.7.1. Функция DateAdd

Функция предназначена для вычитания или прибавления заданного интервала к дате. Ее синтаксис:DateAdd(интервал, число, дата), где *интервал* – строковое выражение, определяющее прибавляемый к дате интервал, возможные значения приведены в таблице 3, *число* – число прибавляемых (или вычитаемых – со знаком минус) интервалов, *дата* – исходная дата.

Таблица 3

Аргумент	Интервал	Аргумент	Интервал
yyyy	Год	ww	Неделя
q	Квартал	h	Час
m	Месяц	n	Минута
y, w, d	День года, месяца или недели	s	Секунда

Например, DateAdd(“d”, -24, #2/23/2011#) вернет дату, предшествующую 23 февраля 2011 года на 24 дня, а функция DateAdd(“q”, 3, Date) вернет дату через три квартала после сегодняшней.

5.7.2. Функция DateDiff

Функция DateDiff используется для вычислений количества указанных интервалов времени между двумя датами. Ее синтаксис:
DateDiff(интервал, дата1, дата2, первый_день_недели, первая_неделя_года), где *интервал* определяется согласно данным таблицы 3, *дата1* и *дата2* – исходные даты, *первый_день_недели* – для России – понедельник, что соответствует константе 2, *первая_неделя_года* – по умол-

чанию, неделя, на которую приходится 1 января. Например, DateDiff(“d”, #1/9/2010#, Date) вернет 410 дней – это количество дней, прошедших с 1 сентября 2010 года до сегодняшней даты.

5.7.3. Функция DatePart

Функция DatePart используется для представления даты в указанных интервалах. Ее синтаксис:DatePart(интервал, дата, первый_день_недели, первая_неделя_года), где *интервал* определяется согласно данным таблицы 3, *дата* – исходная дата, *первый_день_недели* – для России – понедельник, что соответствует константе 2, *первая_неделя_года* – по умолчанию, неделя, на которую приходится 1 января. Например, DatePart(“w”, #2/23/2011#, 2) вернет значение 3, поскольку 23 февраля 2011 года приходится на среду, а счет дней недели ведется с понедельника.

6. Организация вычислений с использованием диапазонов и выделенных областей

6.1. Объект Range (диапазон)

Диапазоном может быть одна ячейка, строка, столбец или прямоугольный блок ячеек любого размера. Любые действия над ячейками диапазона начинаются с создания объекта Range, который является свойством объекта Worksheet (в то же время являясь объектом). Например,

Worksheets(1). Range(“A1”) или Worksheets(1). Range(“A1:B3”) или Worksheets(1). Range(“A1”, “B3”) или ActiveSheet. Range(“A1”)

Правомерными действиями для *выделения* диапазона с помощью метода Select будут следующие:

Range(“A1:B3”).Select или Range(“A1”, “B3”).Select или Range(“A1”, ActiveCell).Select

Можно использовать также именованные диапазоны:

Range(“Данные”).Select

Приведенные конструкции можно использовать для присвоения значений диапазону:

ActiveSheet.Range(“A1”)=”Hello” или используя свойство Value объекта Range:

ActiveSheet.Range(“A1”).Value=”Hello”

Для помещения формулы в диапазон, ее представляют в виде строки:

ActiveSheet.Range(“A1”)=”=Sum(A2:B3)”

Для работы с объектом Range можно создать переменную, которая будет ссылаться на определяемый диапазон:

```
Dim r  As Range
Set r=ActiveSheet.Range(“A1”)
```

В этом случае переменная r ссылается на диапазон A1 и может быть использована для работы с этим диапазоном (в данном случае – ячейкой).

6.1.1. Относительные ссылки на диапазоны

Относительные ссылки полезны тогда, когда необходимо выполнить определенные действия на рабочем листе, но неизвестно, к каким именно ячейкам будут относиться эти действия. Относительные ссылки на диапазоны формируются с помощью свойства Offset(RowOffset, ColumnOffset).

Аргументы RowOffset и ColumnOffset представляют собой числовые значения. При положительных значениях они определяют сдвиг вниз по строкам и вправо по столбцам относительно исходного диапазона. Для сдвига влево по столбцам и вверх по строкам используются отрицательные значения. Правомерным является указание только одного параметра, если другой равен нулю:

Range(“A1”).Offset(2) – вернет диапазон A3, а Range(“A1”).Offset(, 2) – вернет диапазон C1.

Например, нужно отформатировать столбец чисел, определив для каждой второй ячейки полужирное начертание при условии, что пользователь активизирует первую ячейку диапазона. Программа должна работать для любого столбца данных.

```
Public Sub string_bold()
'форматирование через строку в одном столбце,
'в нем АКТИВНА ПЕРВАЯ ячейка
   Dim r1 As Range
   Set r1 = ActiveCell
   Do While r1.Value <> ""
       r1.Font.Bold = True
       Set r1 = r1.Offset(2)
   Loop
End Sub
```

В программе создана ссылка r1 на активную ячейку диапазона. Изменение диапазона происходит путем сдвига его на две строки вниз - Set r1 = r1.Offset(2). В цикле Do While осуществляется форматирование с использованием свойства Font.Bold диапазона r1, которому присваивается значение True. Цикл продолжается пока диапазон не окажется пустой строкой - r1.Value <> "".

6.1.2. Свойство CurrentRegion

Свойство CurrentRegion позволяет определить диапазон, прилегающий к активной ячейке, если размер таблицы заранее не известен.

Например, верхний левый угол таблицы находится в ячейке B2, но неизвестно, сколько строк и столбцов содержит таблица. Можно создать объект Range, который ссылается на таблицу с помощью переменной TableRange.

```
Dim  TableRange as Range
Set  TableRange = ActiveSheet.Range(“B2”).CurrentRegion
```

После выполнения этого кода переменная TableRange ссылается на таблицу и может быть использована для ее дальнейшей обработки.

6.1.3. Строки, столбцы и размер диапазона

Свойства Column и Row объекта Range возвращают число, указывающее на *первый столбец* диапазона (левый) и, соответственно, *первую строку* диапазона (верхнюю).

Например, для диапазона B5:D8 приведенный далее код отобразит в окне Immediate переменную N, равную 2, а M – равную 5

```
Dim N As Long, M As Long
N=Range(“B5:D8”).Column
M=Range(“B5:D8”).Row
Debug.Print N;M
```

Размер диапазона можно определить, пользуясь свойствами Columns.Count и Rows.Count. Эти свойства возвращают количество столбцов и строк диапазона, соответственно.

Например, для диапазона B5:D8 приведенный далее код отобразит в окне Immediate переменную N, равную 3, а M – равную 4

```
Dim N As Long, M As Long
N=Range(“B5:D8”).Columns.Count
M=Range(“B5:D8”).Rows.Count
Debug.Print N;M
```

Обращение к конкретной строке и столбцу диапазона осуществляется с помощью свойств Rows(n) и Columns(n), где n – номер строки или столбца. Обращение ко всем строкам или столбцам диапазона происходит с помощью свойств Rows и Columns, соответственно. Данные свойства возвращают диапазон (объект типа Range). Для работы со столбцами и строками можно использовать любые свойства и методы объектов Range.

Для *доступа* к *полным строкам* и *столбцам*, принадлежащим определенному диапазону, можно применить свойства EntireRow и EntireColumn.

Например, при использовании кода Range(“B5:D8”).EntireRow, будет получена ссылка на строки с пятой по восьмую, а при использовании кода Range(“B5:D8”).EntireColumn будет получена ссылка на столбцы от B до D.

Добавление и *удаление* строк и столбцов осуществляется программным кодом, который использует метод Insert и Delete, соответственно. Для добавления и удаления столбцов в диапазоне B2:C2, нужно использовать следующий код:

Range (“B2:C2”). EntireColumn.Insert для добавления столбцов на место B и C или

Range (“B2:C2”). EntireColumn. Delete для удаления столбцов B и C.

6.1.4. Свойство Selection

Свойство Selection (выделенная область) ссылается на диапазон, состоящий из одной или нескольких ячеек, подобно объекту Range. Это свойство ссылается на диапазон, *выделенный* пользователем. В определенный момент времени может быть выделен только один диапазон. Свойство Selection может ссылаться также на другие выделенные объекты, расположенные на рабочем листе.

Чтобы убедиться, что свойство Selection ссылается на диапазон ячеек, используется функция TypeName. Получая свойство Selection в качестве

аргумента, функция возвращает строку “Range”, если выделена одна или более ячеек. Если ничего не выделено, свойство Selection возвращает строку Nothing. Для проверки выделения используется код:

```
Dim r As Range
If TypeName(Selection)=”Range” Then
    ‘Выделенная область – диапазон
    Set r=Selection
    ‘Обработка диапазона
 Else
‘Выделенная область – не диапазон
End If
```

6.1.5. Присвоение имен диапазонам

Для присвоения имени диапазону нужно использовать свойство Name объекта Range, например,

```
Dim r As Range
Set r=Range(“B10”)
r.Name=”Итоги”
Range(“Итоги”).Value=”=Sum(B3:B9)”
```

Присвоенные таким образом имена существуют только во время выполнения программы. Для создания имен, которые будут сохранены в рабочей книге, и могут использоваться при следующих сеансах работы с Excel, нужно явно добавить имя в коллекцию имен Names данной рабочей книги, используя метод Add, например:

```
Names.Add  Name:="Total", RefersTo:="=Лист1!$B$11"
```

Метод Add имеет именованные аргументы Name, который задает имя диапазону и RefersTo, представляющий собой строку, в которой после знака равенства указан лист и диапазон. Для указания диапазона могут использоваться абсолютные (со знаком $) и относительные адреса. Так, ссылка на ячейку B2 означает, что имя будет присвоено ячейке, расположенной на один столбец правее и одну строку ниже активной ячейки.

6.1.6. Добавление комментария

Для добавления к ячейке комментария нужно использовать метод AddComment “Текст комментария”, например,

```
Range(“B3”).AddComment  “Текст комментария”
```

Код добавит комментарий к ячейке B3.

6.1.7. Изменение размера диапазона

Для изменения размера диапазона используется свойство Resize (количество строк, количество столбцов). Например, чтобы изменить диапазон B3 до B3:D13, нужно использовать код:

```
Range(“B3”).Resize(11,3)
```

6.1.8. Копирование формул в заданном диапазоне

Пусть в диапазон B5:F5 нужно внести формулу суммирования значений в диапазоне B1:F4.

Первый способ заключается в использовании ссылки на результирующий диапазон Set r=Range(“B5:F5”) и стиля ссылок R1C1 для формулы r.FormulaR1C1 = "=sum(r[-4]c:r[-1]c)"

Следует обратить внимание на использование функций, которые записываются в виде строки, например, r.FormulaR1C1 = "=sum(r[-4]c:r[-1]c)"

или Range(“B10”).Formula=”=max(B5:F5)”, которая позволит найти максимальное значение в диапазоне B5:F5.

Второй способ использует метод Copy с параметром Destination Range("B5").Formula = "=sum(B1:B4)"

```
Range("B5").Copy  Destination:=Range("C5:F5")
```

Третий способ использует код, формирующийся при записи макроса:

```
Range("B5").Select
Selection.Copy
Range("C5:F5").Select
ActiveSheet.Paste
```

6.2. Обращение к ячейкам с помощью свойства Cells

Одним из наиболее эффективных средств, позволяющих обращаться к ячейкам рабочего листа, является свойство Cells объектов Range и Worksheet. Свойство Range.Cells ссылается на все ячейки данного диапазона, свойство Worksheet.Cells ссылается на все ячейки рабочего листа.

Например, Range(“A2:B10”).Cells.Font.Size=14

Ссылка на ячейку с помощью указания номера *строки и столбца* осуществляется следующим способом:

Cells(RowIndex, ColumnIndex). Аргументы RowIndex, ColumnIndex – это числовые значения, определяющие ячейку по ее строке и столбцу. Такой вид ссылок позволяет организовать цикл для перебора и обработки всех ячеек диапазона. Например, для диапазона r1 цикл выглядит таким образом:

```
For r=1 to r1.rows.count
For c=1 to columns.count
‘обработка ячейки  r1.cells(r,c)
Next c
Next r
```

Ссылка на ячейку с помощью свойства Cells может осуществляться с помощью обозначения позиции данной ячейки в диапазоне. При этом используется синтаксис:Cells(CellPosition).

CellPosition – числовое значение, определяющее позицию ячейки в диапазоне. Нумерация ячеек начинается с номера 1 (верхняя левая ячейка) и увеличивается для ячеек, расположенных правее и ниже. Например, для диапазона r:

```
For  pos To r.Columns.Count*r.Rows.Count
‘код обработки  r.Cells(pos)
Next pos
```

Для выделения определенных ячеек диапазона на основании оценки их содержимого используется метод SpecialCells.

7. Анализ и обработка данных

7.1. Сортировка

Сортировка осуществляется методом Sort объекта Range. Метод включает несколько именованных аргументов:

Key1 - ссылается на ячейку столбца, по значениям которого должна осуществляться сортировка диапазона,

Key2 – ссылается на ячейку столбца, по значениям которого должна осуществляться последующая сортировка,

Order1 – определяет порядок первичной сортировки и имеет значения xlAscending (сортировка по возрастанию) или xlDescending (сортировка по убыванию),

Order2 - определяет порядок последующей сортировки и имеет те же значения, что и Order1,

Header – указывает, имеет ли таблица строку заголовков с названиями полей и может иметь значения xlYes (да), xlNo (нет) и xlGuess (по умолчанию). В последнем случае Excel самостоятельно определяет, есть ли строка заголовков в таблице данных.

Сортировка таблицы, приведенной на рис. 8 (слева) по полу, а затем по фамилии происходит с помощью следующего кода.

```
Dim r As Range
Set r = Worksheets("Лист1").Range("A1:B9")
r.Sort Key1:=Worksheets("Лист1").Range("B2"), Order1:=xlAscending, _
Key2:=Worksheets("Лист1").Range("A2"), Order2:=xlAscending, _ Header:=xlYes
```

Результат сортировки представлен на рис. 8 (справа).

	A	B
1	Фамилия	Пол
2	Иванов	м
3	Петрова	ж
4	Сидоров	м
5	Jackson	м
6	Степанов	м
7	Николаева	ж
8	007	м
9	Журавлева	ж
10		

	A	B
1	Фамилия	Пол
2	Журавлева	ж
3	Николаева	ж
4	Петрова	ж
5	007	м
6	Jackson	м
7	Иванов	м
8	Сидоров	м
9	Степанов	м
10		

Рис. 8

7.2. Фильтрация с помощью автофильтра

Автофильтр подключается при использовании метода AutoFilter объекта Range. Объект может ссылаться на весь диапазон или на одну его ячейку. Метод включает несколько именованных аргументов:
Field – числовое значение, указывающее поле, по которому выполняется фильтрация (крайнее левое поле имеет индекс 1),

Criteria1, (Criteria2) – критерии, по которым выполняется фильтрация (используются в случае необходимости, при их отсутствии выбираются все записи),

Operator – константа Excel, определяющая способ интерпретации критерия Criteria1. Константы могут принимать следующие, наиболее часто применяемые значения:

xlAnd – отобранные записи должны удовлетворять обоим критериям,
xlOr - отобранные записи должны удовлетворять одному из критериев,
xlBottom10Items, xlTop10Items – отбирается количество записей, указанных в критерии, соответственно, наименьших или наибольших
xlFilterValues – используется при применении в качества критерия нескольких (больше двух) значений, образующих массив,
xlFilterDynamic – используется при «динамической фильтрации», которая позволяет отбирать значения больше или меньше среднего или отбирать даты, находящиеся в определенном интервале. При этом значения критериев также определяются константами Excel:
xlFilterAboveAverage, xlFilterBelowAverage - для отбора значений, соответственно, больше или меньше среднего,
xlFilterTomorrow, xlFilterNextWeek, xlFilterNextMonth, xlFilterNextQuarter, xlFilterNextYear – для отбора значений будущих периодов,
xlFilterToday, xlFilterThisWeek, xlFilterThisMonth, xlFilterThisQuarter, xlFilterThisYear – для отбора значений текущих периодов,
xlFilterYesterday, xlFilterLastWeek, xlFilterLastMonth, xlFilterLastQuarter, xlFilterLastYear – для отбора значений прошлых периодов,
xlFilterDatesInPeriodQuarter1 до xlFilterDatesInPeriodQuarter4 – для отбора значений заданных кварталов,
xlFilterDatesInPeriodJanuary до xlFilterDatesInPeriodDecember – для отбора значений заданных месяцев.

Применение автофильтра показано на примере таблицы, приведенной на рис. 9. Следующий код позволяет отобрать записи, относящиеся к регионам «Восток» и «Запад».

```
Dim r As Range
Set r = Worksheets("Лист1").Range("A1")
r.AutoFilter Field:=4, Criteria1:="=Восток", Operator:=xlOr, Criteria2:="=Запад"
```

	A	B	C	D
1	Фамилия	Пол	Оклад	Регион
2	Иванов	м	32000	Северо-запад
3	Петрова	ж	18000	Центр
4	Сидоров	м	20000	Северо-запад
5	Jackson	м	100000	Запад
6	Степанов	м	25000	Восток
7	Николаева	ж	28000	Северо-запад
8	007	м	1000	Запад
9	Журавлева	ж	15000	Центр
10				

Рис. 9

Для копирования результата в произвольную табличную область (в случае необходимости дальнейшей обработки) нужно применить метод SpecialCells(xlCellTypeVisible), который позволяет обратиться к видимым ячейкам, а затем скопировать их содержимое в произвольную область.

```
'копирование результата в свободный диапазон
Worksheets("Лист1").AutoFilter.Range.SpecialCells(xlCellTypeVisible).Copy _
Destination:= Worksheets("Лист1").Range("A15")
```

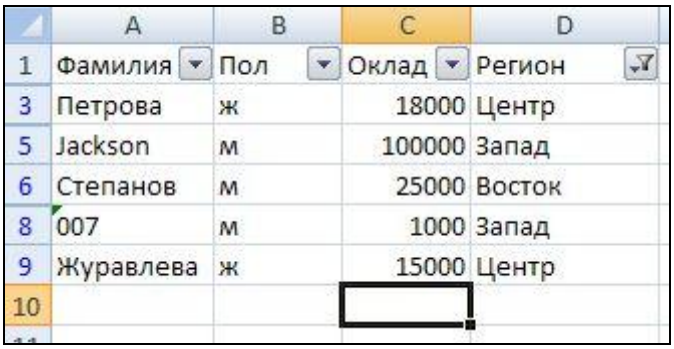
Результат фильтрации и копирования его в диапазон, начинающийся с ячейки A15, показан на рис. 10.

	A	B	C	D	E
1	Фамилия	Пол	Оклад	Регион	
5	Jackson	м	100000	Запад	
6	Степанов	м	25000	Восток	
8	007	м	1000	Запад	
10					
11					
12					
13					
14					
15	Фамилия	Пол	Оклад	Регион	
16	Jackson	м	100000	Запад	
17	Степанов	м	25000	Восток	
18	007	м	1000	Запад	
19					

Рис. 10

На рис. 11 приведен результат фильтрации, по критерию, использующему массив значений регионов, в который вошли регионы «Восток», «Запад», и «Центр». Код приведен далее.

```
Dim r As Range
Set r = Worksheets("Лист1").Range("A1")
r.AutoFilter Field:=4, Criteria1:=Array("Восток", "Запад", "Центр"),_
Operator:=xlFilterValues
```

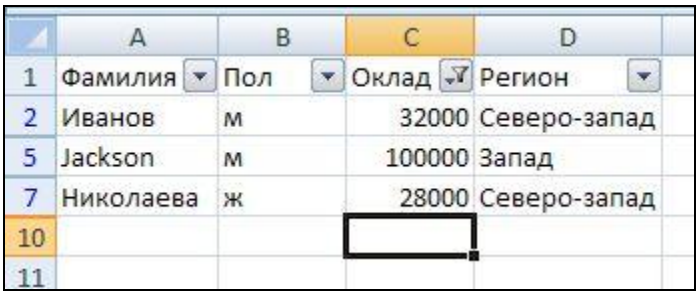


	A	B	C	D
1	Фамилия	Пол	Оклад	Регион
3	Петрова	ж	18000	Центр
5	Jackson	м	100000	Запад
6	Степанов	м	25000	Восток
8	007	м	1000	Запад
9	Журавлева	ж	15000	Центр
10				

Рис. 11

На рис. 12 приведен результат фильтрации, использующей значение аргумента Operator:=xlTop10Items и позволяющего отобрать три записи с наибольшими окладами. Код приведен далее.

```
Dim r As Range
Set r = Worksheets("Лист1").Range("A1")
r.AutoFilter Field:=3, Criteria1:="3", Operator:=xlTop10Items
```

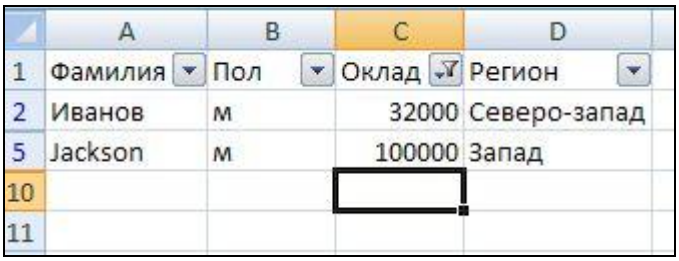


	A	B	C	D
1	Фамилия	Пол	Оклад	Регион
2	Иванов	м	32000	Северо-запад
5	Jackson	м	100000	Запад
7	Николаева	ж	28000	Северо-запад
10				
11				

Рис. 12

На рис. 13 приведен результат фильтрации, использующей значение аргумента Operator:= xlFilterDynamic и позволяющего отобрать записи со значениями окладов, больших среднего значения. Код приведен далее.

```
Dim r As Range
Set r = Worksheets("Лист1").Range("A1")
r.AutoFilter Field:=3, Criteria1:=xlFilterAboveAverage, _
Operator:=xlFilterDynamic
```



	A	B	C	D
1	Фамилия	Пол	Оклад	Регион
2	Иванов	м	32000	Северо-запад
5	Jackson	м	100000	Запад
10				
11				

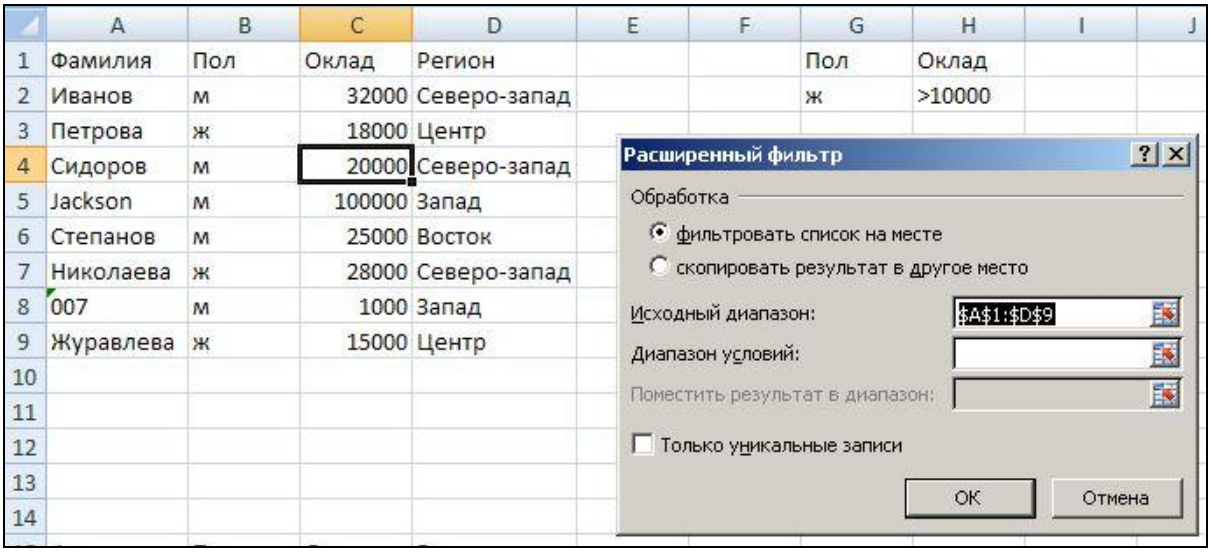
Рис. 13

7.3. Фильтрация с помощью расширенного фильтра

Расширенный фильтр позволяет использовать критерии, дающие возможность извлечь любое подмножество записей таблицы или получить уникальный список значений любого столбца.

Фильтр подключается при использовании метода AdvancedFilter объекта Range. Элементы управления диалогового окна «Расширенный фильтр» (рис. 14) определяют аргументы данного метода:

Action (секция «Обработка») определяет возможность фильтрации «на месте» или копирования результата во вне табличную область. В первом случае аргумент принимает значение xlFilterInPlace, а во-втором - xlFilterCopy, при этом указывается выходной диапазон с помощью аргумента CopyToRange, аргумент CriteriaRange указывает диапазон условий фильтрации, а аргуент Unique – позволяет получить список уникальных значений заданного поля. При этом он должен иметь значение True.



	A	B	C	D	E	F	G	H	I	J
1	Фамилия	Пол	Оклад	Регион			Пол	Оклад		
2	Иванов	м	32000	Северо-запад			ж	>10000		
3	Петрова	ж	18000	Центр						
4	Сидоров	м	20000	Северо-запад						
5	Jackson	м	100000	Запад						
6	Степанов	м	25000	Восток						
7	Николаева	ж	28000	Северо-запад						
8	007	м	1000	Запад						
9	Журавлева	ж	15000	Центр						
10										
11										
12										
13										
14										

Рис. 14

На рис. 15 приведен результат фильтрации, использующей диапазон условий G1:H2, позволяющий отобрать записи, относящиеся к сотрудникам женского пола со значениями окладов, больших 10 000. Код приведен да-

лее. Последняя строка исходного диапазона может быть не известна заранее. Для ее нахождения нужно переместиться в конец рабочего листа (Cells(Rows.Count, 1)), а затем использовать последовательное нажатие клавиш END и ↑ (End(xlUp), и воспользоваться свойством Row.

```
Dim Irange As Range, Orange As Range, Crange As Range
Dim finalRow As Long
'Определение последней строки исходного диапазона
finalRow = Cells(Rows.Count, 1).End(xlUp).Row
'Определение исходного диапазона
Set Irange = Worksheets("Лист1").Range("A1:D" & finalRow)
'Определение выходного диапазона
Set Orange = Worksheets("Лист1").Range("A" & finalRow + 2)
' Определение диапазона критериев
Set Crange = Worksheets("Лист1").Range("G1:H2")
Irange.AdvancedFilter Action:=xlFilterCopy, CriteriaRange:=Crange, _
CopyToRange:=Orange
```

12	Фамилия	Пол	Оклад	Регион
13	Петрова	ж	18000	Центр
14	Николаева	ж	28000	Северо-запад
15	Журавлева	ж	15000	Центр

Рис. 15

На рис. 16 приведен результат отбора уникальных значений в поле «Регион». Код приведен далее.

```
Dim Irange As Range, Orange As Range
Dim finalRow As Long
finalRow = Cells(Rows.Count, 1).End(xlUp).Row
'Определение исходного диапазона
Set Irange = Worksheets("Лист1").Range("A1:D" & finalRow)
'Определение выходного диапазона
Set Orange = Worksheets("Лист1").Range("i1")
Irange.AdvancedFilter Action:=xlFilterCopy, CopyToRange:=Orange, _
Unique:=True
```

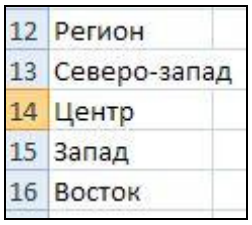


Рис. 16

7.4. Процедуры с передачей параметров

Использование процедуры фильтрации, приведенной в п. 7.3, для разных диапазонов условий возможно при передаче параметра в данную процедуру. В качестве параметра используется необходимый для конкретной задачи диапазон условий. Код процедуры с формальным параметром Crange (диапазон критериев) приведен далее. Формальный параметр записывается в заголовке процедуры (ByVal Crange As Range).

```
Public Sub расш_парам (ByVal Crange As Range)
Dim Irange As Range, Orange As Range
Dim finalRow As Long

finalRow = Cells(Rows.Count, 1).End(xlUp).Row
'Определение исходного диапазона
Set Irange = Worksheets("Лист1").Range("A1:D" & finalRow)
'Определение выходного диапазона
Set Orange = Worksheets("Лист1").Range("A" & finalRow + 2)
Irange.AdvancedFilter Action:=xlFilterCopy, CriteriaRange:=Crange, _
CopyToRange:=Orange
End Sub
```

В процедуре «задание_факт_парам» происходит задание фактического значения диапазона условий. При вызове процедуры «расш_парам», фактический параметр замещает в ней формальный параметр, и процедура выполняется с заданным диапазоном условий. Код этой процедуры приведен далее.

```
Public Sub задание_факт_парам()
'определение диапазона критериев - (фактического параметра)
Set rr = Worksheets("Лист1").Range("G1:I2")
расш_парам (rr)
End Sub
```

7.5. Графическое представление данных

Для добавления диаграммы в коллекцию диаграмм Charts рабочего листа используется метод Add:

```
ActiveWorkbook.Charts.Add
```

Далее указывается диапазон, по которому строится диаграмма, тип диаграммы, а также параметры, требующиеся для удобства ее восприятия.

Диаграмма, отображающая оклады сотрудников из таблицы, приведенной на рис. 9, приведена на рис. 17.

Код, позволяющий построить диаграмму по диапазонам A1:A9 (фамилии) и C1:C9 (оклады) на отдельном листе приведен далее.

```
Dim ch As Chart
Set ch = ActiveWorkbook.Charts.Add
ch.SetSourceData  Source:=Worksheets("Лист1").Range("a1:a9",  "c1:c9"),
PlotBy:=xlColumns
ch.ChartType = xlColumnClustered
ch.HasTitle = True
ch.ChartTitle.Caption = "Оклады сотрудников"
ch.SetElement msoElementDataLabelCenter
```

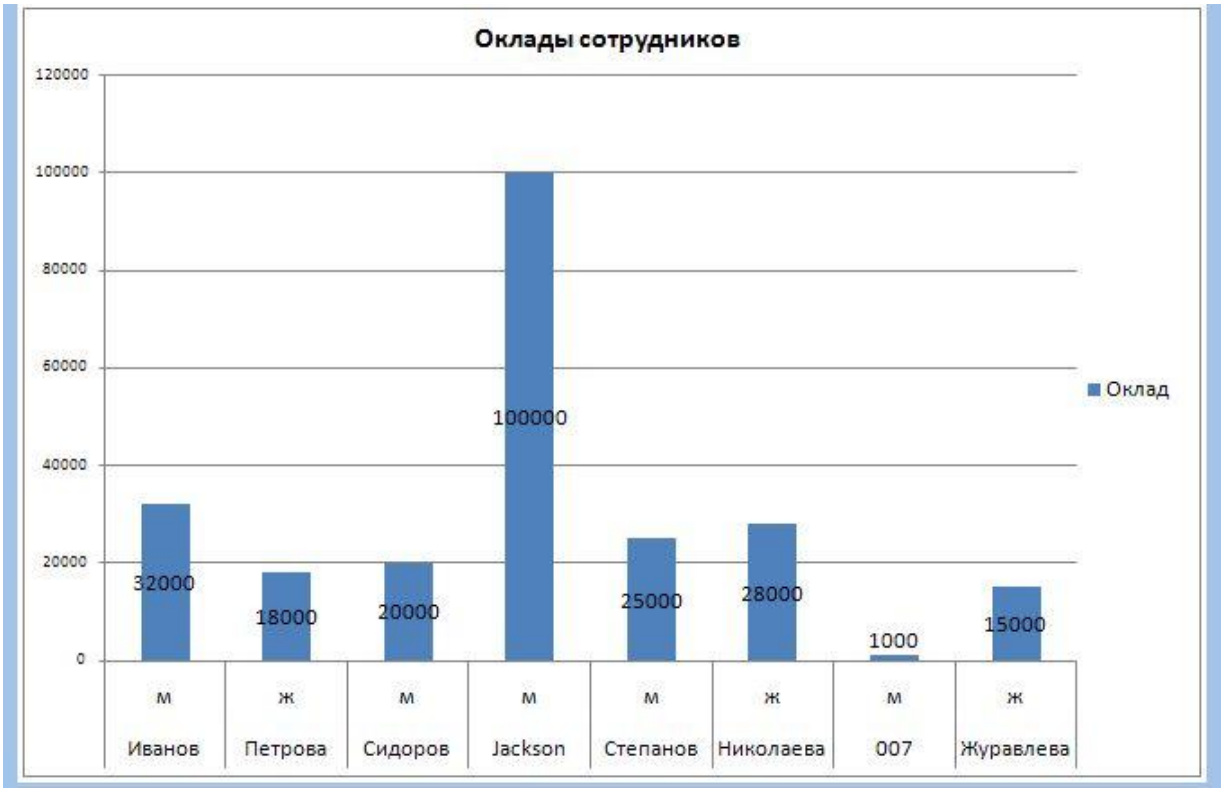


Рис. 17

В случае необходимости построения диаграммы в табличном листе нужно создать новый объект типа ChartObject (диаграмма) и указать область построения диаграммы для нового объекта с помощью кода:

```
Set co = Worksheets("Лист1").ChartObjects.Add(100, 150, 450, 250)
```

Код, позволяющий построить диаграмму по диапазонам A1:A9 (фамилии) и C1:C9 (оклады) в табличном листе приведен далее.

```
Dim co As ChartObject
Dim ch As Chart
Set co = Worksheets("Лист1").ChartObjects.Add(100, 150, 450, 250)
'расстояния слева, сверху, затем ширина и высота
Set ch = co.Chart
ch.SetSourceData Source:=Worksheets("Лист1"). _
Range("a1:a9", "c1:c9"), PlotBy:=xlColumns
ch.ChartType = xlColumnClustered
ch.HasTitle = True
ch.ChartTitle.Caption = "Оклады сотрудников"ливается
ch.SetElement msoElementDataLabelCenter
```

7.6. Форматирование табличных данных

7.6.1. Форматирование шрифтов

Для форматирования шрифта используется объект Font. Его свойства определены стандартными свойствами шрифтов Excel. Например, установки свойств шрифта для диапазона r будут иметь вид:

r.Font.Bold=true (полужирное начертание)

r.Font.Size=12 (размер шрифта в пунктах)

r.Font.Underline=true (подчеркивание)

r.Font.Color=rgb(255,0,0) (красный цвет символов)

Цвет шрифта устанавливается с помощью функции rgb(r,g,b) с соответствующими весовыми значениями составляющих красного, зеленого и синего цветов, которые задаются в диапазоне от 0 до 255.

Кроме этого способа можно использовать значения свойства ColorIndex, меняющегося от 1 до 56. Например, фон ячейки, который пред-

ставлен объектом Interior, можно определить этим свойством так: Cells(1,2). Interior.ColorIndex=4.

Снять формат с ячеек всего листа или диапазона r можно с помощью кода Cells.ClearFormat или r. ClearFormat, соответственно.

В VBA предусмотрен также ряд констант, которые тоже можно использовать для цветового оформления:
vbBlack, vbRed, vbBlue и т.д.

7.6.2. Форматирование чисел

Для установки числового формата используется объект NumberFormat и кодовые значения, применяемые в Excel. Например код Range(“A1”). NumberFormat=”#.000” отобразит число с тремя знаками после запятой (5.670), а код Range(“A2”). NumberFormat=”$#,###.00” отобразит знак $, отделит тройки разрядов и покажет два знака после запятой ($5 670.45).

7.6.3. Выравнивание

Содержащиеся в ячейках значения можно выравнивать в горизонтальном и вертикальном направлении. Для этого используются свойства объекта Range – HorizontalAlignment (горизонтальное выравнивание) со значениями xlLeft (по левому краю), xlCenter (по центру) и xlRight (по правому краю) и VerticalAlignment (вертикальное выравнивание) со значениями xlTop (по верхнему краю) xlCenter (по центру) и xlBottom (по нижнему краю).

7.6.4. Границы ячеек

Объект Range имеет коллекцию Borders, при использовании которой без аргументов, можно воздействовать на все границы диапазона с помощью свойств Color (цвет со значениям rgb), LineStyle (стиль границы, например, xlContinuous – сплошная, xlDouble – двойная, xlDash - пунктирная) и Weight (толщина границы, например, xlThin – тонкоя линия, xlThick,- широкая, xlMedium – средняя толщина).

Если требуется воздействовать только на конкретную границу (в данном случае на верхнюю границу), то используется следующий синтаксис:
r.Borders(xlEdgeTop).LineStyle=xlDash

В этом случае в скобках указывается одна из констант, относящаяся к конкретной границе (xlEdgeBottom – к нижней, xlEdgeTop – к верхней, xlEdgeLeft – к левой, xlEdgeRight – к правой, xlInsideVertical – к внутренним вертикальным, xlInsideHorizontal – к внутренним горизонтальным).

7.6.5. Размер строк и столбцов

Для установки ширины столбца (в символах) или высоту строки (в пунктах) можно использовать свойства объекта Range - ColumnWidth и RowHeight, соответственно.

Можно использовать коллекцию Columns объекта Worksheets, чтобы обратиться к столбцам рабочего листа, например,
Worksheets(“Лист1”). Columns(“A:C”). ColumnWidth =24

Автоматический подбор ширины столбцов осуществляется методом AutoFit, например, Worksheets(“Лист1”). Columns(“A:C”)= AutoFit

Для регулирования ширины столбцов конкретного диапазона можно использовать свойство EntireColumn, например, для диапазона r
r. EntireColumn. AutoFit

Высота строк диапазона r, равная 24pt определится следующим образом: r.RowHeight=24

8. Программирование пользовательского интерфейса. Работа с экранными формами

Экранные формы представляют собой программируемые элементы, позволяющие создать для приложений Excel визуальный пользовательский интерфейс, обеспечивающий взаимодействие с пользователем по заданному алгоритму. Экранные формы (диалоговые окна) содержат программируемые элементы управления, которые обычно встречаются в диалоговых окнах программы Excel.

Экранная форма имеет три компонента:

- Окно со строкой заголовка.
- Элементы управления, составляющие ее визуальный и функциональный интерфейс.
- Коды VBA, управляющие экранной формой и ее элементами управления.

Форма представлена объектом UserForm. Объект UserForm, как и размещенные в окне формы элементы управления, обладают свойствами, методами и событиями, использование которых определяет внешний вид и функциональные характеристики экранной формы.

8.1. Использование редактора VBA для создания экранных форм

Создание экранной формы происходит в редакторе VBA. Для создания экранной формы необходимо выполнить команду меню Insert – UserForm. По умолчанию вновь созданным формам даются стандартные имена UserForm1, UserForm2 и т.д.

Окно редактора с экранной формой представлено на рис. 18.

В рабочей области находится окно создаваемой формы и панель инструментов (Toolbox), отображаемая в случае активизации формы и содержащая элементы управления, которые могут быть размещены в форме.

В верхней части окна проектов находятся три кнопки. Левая – View Code (Просмотр кода) отображает код активного объекта (элемента управления или формы), средняя – View Object (Просмотр объекта) отображает элементы управления активного объекта (доступна только для экранных форм), правая кнопка – Toggle Folders (Переключение папок) переключает режим отображения элементов в окне проектов.

Для того чтобы открыть окно редактирования кодов формы или элемента управления (рис. 19), нужно воспользоваться кнопкой View Code (Просмотр кода) или дважды щелкнуть мышью на форме или элементе управления. Окно редактирования кодов откроется справа от окна проектов в рабочей области редактора VBA.

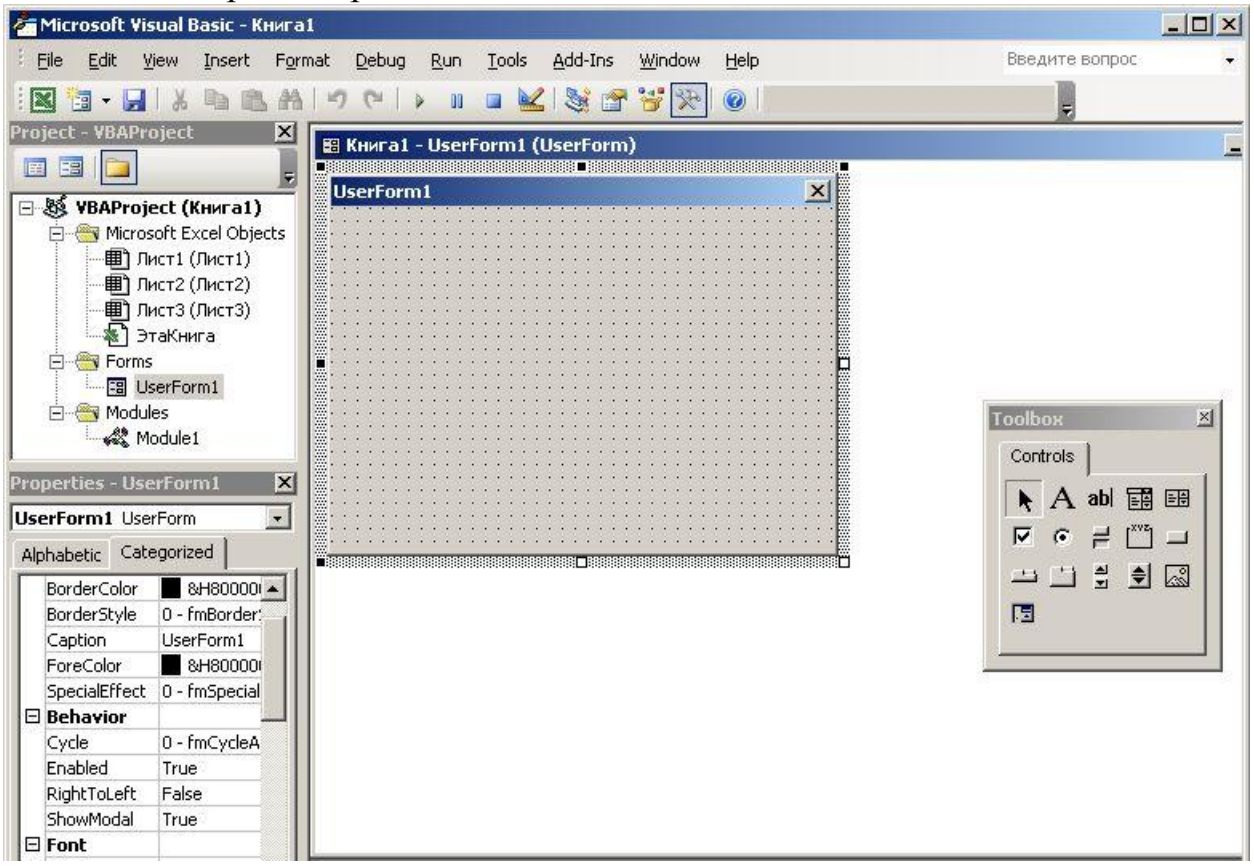


Рис. 18

В левой нижней части находится окно свойств (Properties), в котором отображаются свойства выделенного объекта. При работе с формой такими объектами может быть сама форма или элементы управления, находящиеся в ней.

Свойствами можно управлять как вручную, устанавливая их значения в окне свойств, так и программным способом на этапе выполнения.

8.2. Свойства и методы экранных форм

В окне свойств выделенной формы можно изменить предложенные свойства или установить незаданные изначально значения. В левой части окна находится список свойств, а правая часть предназначена для указания их значений. Свойства могут быть сгруппированы по категориям (вкладка Categorized) или расположены по алфавиту (вкладка Alphabetic).

Рассмотрим основные свойства экранных форм.

Свойство Name

Свойство Name определяет имя объекта UserForm, которое будет использовано для обращения к этому объекту.

Свойство Caption

Свойство Caption задает заголовок, отображаемый в строке заголовка окна формы.

Свойства BackColor, ForeColor и Font

Свойства BackColor, ForeColor и Font дают возможность установить цвет фона, символов (переднего плана) и тип шрифта для разрабатываемой формы.

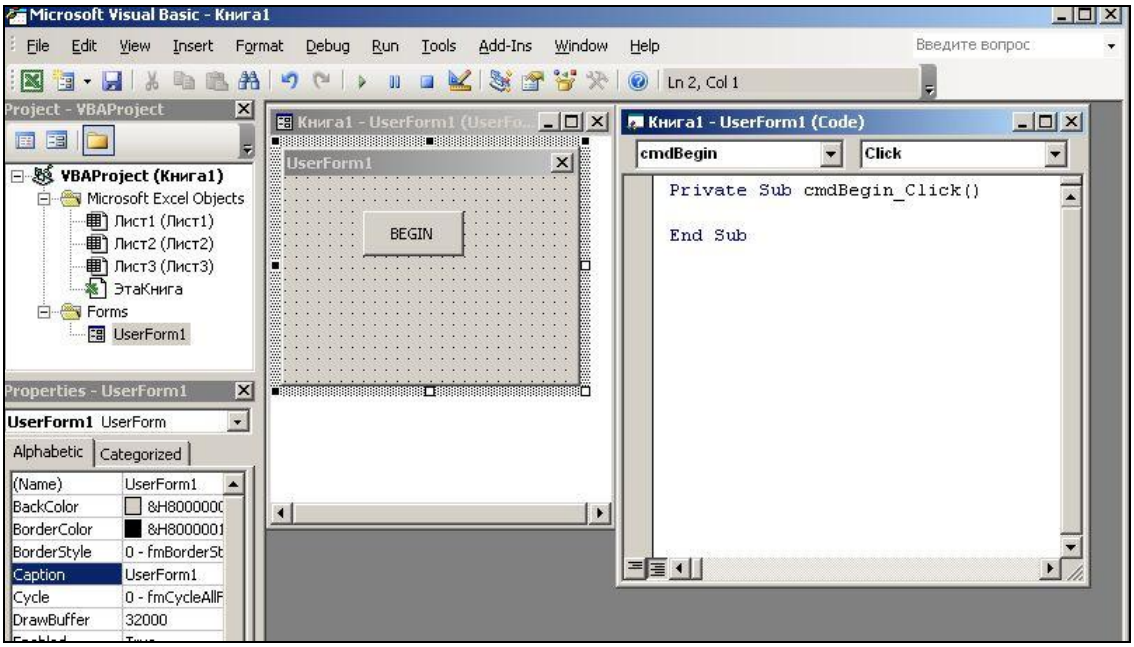


Рис. 19.

Свойства Position

Свойства Width и Height определяют ширину и высоту формы, а свойства Left и Top – положение экранной формы.

Свойство StartUpPosition позволяет установить окно формы по центру экрана (CenterScreen), по центру окна Excel (CenterOwner), произвольно (Manual), по умолчанию в верхней левой части экрана (WindowDefault).

Свойства Behavior

Свойство ShowModal определяет модальность окна. При значении True пользователь должен закрыть форму, прежде чем он будет переходить к выполнению другого кода. Если свойству присвоено значение False, пользователь может оставить эту форму отображенной на экране и перейти к использованию других частей приложения.

Рисунок, который может быть расположен на форме, определяется свойствами категории Picture.

Для получения справки по определенному свойству нужно выделить это свойство в окне Properties и нажать клавишу F1.

Объект UserForm имеет несколько методов:

Hide скрывает форму, если она отображена.

Show отображает форму.

Move (Left, Top, Width, Height) перемещает форму и изменяет ее размеры согласно заданным аргументам.

8.3. Элементы управления экранных форм

Элементы управления позволяют решать задачи отображения и ввода данных. Каждый элемент обладает собственным набором свойств, методов и событий, в то же время существует стандартный перечень, общий для всех элементов.

Для размещения элемента управления на экранной форме достаточно щелкнуть на нем в панели инструментов Toolbox и перетащить в форму. Рассмотрим наиболее часто используемые элементы управления. Они представлены в таблице 9.

Обращение к большинству элементов происходит с помощью конструкции:

Имя формы.Имя элемента.Value или Имя формы.Имя элемента.Text.

Таким образом, обращение к содержимому текстового поля с именем TextBox1, находящемуся в экранной форме с именем UserForm1 будет: UserForm1.TextBox1.Value

При обращении к элементу из самой формы имя формы можно опустить: TextBox1.Value

8.3.1. Текстовое поле

В текстовом поле (TextBox), по умолчанию, отображается одна текстовая строка. Если необходимо отобразить более одной строки, свойству MultiLine нужно присвоить значение True. Переход на другую строку происходит по нажатию клавиш Shift+Enter, при этом свойству WordWrap присваивается значение False. В текстовое поле могут быть добавлены полосы прокрутки с помощью свойства ScrollBars

8.3.2. Флажки

Свойство Value флажков (CheckBox) возвращает значение True, если они установлены и False, если нет.

8.3.3. Комбинированный список

Элемент комбинированный список (ComboBox) используется для выбора элемента, отображаемого в списке.

Для добавления элемента в список используется метод AddItem "Элемент". Например, для списка с именем ComboBox1 с помощью приведенного далее кода можно добавить несколько элементов:

```
ComboBox1.AddItem "Книги"
ComboBox1.AddItem "DVD-диски"
ComboBox1.AddItem "Аудиокниги"
ComboBox1.AddItem "Музыка"
```

Каждый элемент, добавленный в список, получает числовой индекс. Первый элемент имеет индекс 0, второй 1 и т.д. Свойство ListIndex отображает индекс выбранного элемента. Для начального отображения в списке первого элемента, нужно задать свойству ListIndex значение 0:

```
ComboBox1.ListIndex=0
```

Для сокращения записи кода добавления в список, приведенного выше, используется конструкция With…End With:

```
With ComboBox1
.AddItem "Книги"
.AddItem "DVD-диски"
.AddItem "Аудиокниги"
.AddItem "Музыка"
.ListIndex=0
End With
```

Свойство ListCount содержит число элементов, внесенных в список, свойство List (номер элемента) позволяет обратиться к любому элементу списка.

Метод RemoveItem позволяет удалить элемент из списка. Например, использование кода ComboBox1. RemoveItem(2) удалит из списка третий элемент списка.

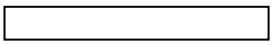
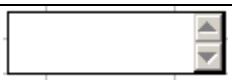
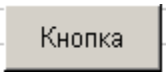
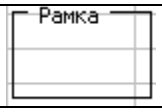
Метод Clear удаляет все элементы списка.

8.3.4. Список

Элемент управления ListBox отображает список. Он может использоваться в двух режимах: для выбора элементов списка и для отображения информации из табличной области Excel.

Добавление и удаление элементов аналогично комбинированному списку.

Таблица 9

Название	Имя	Отображение в форме
Текстовое поле	TextBox	
Флажок	CheckBox	
Переключатель	OptionButton	
Комбинированный список	ComboBox	
Список	ListBox	
Командная кнопка	CommandButton	
Рамка	Frame	
Рисунок	Image	
Надпись	Label	A
Счетчик (полоса значений)	SpinButton	

Для вывода диапазона данных из диапазона A1:D5, первая строка которого содержит названия столбцов, можно применить следующий код:

```
With ListBox1
.ColumnWidths="50;50;50;40"
.ColumnCount=4
.RowSource="A1:D5"
.ColumnHeads=True
End With
```

В данном коде используются свойства ColumnWidths – ширина столбцов в пикселах (8,43 симв=64px), ColumnCount – число столбцов, RowSource – источник строк, ColumnHeads – наличие заголовков столбцов.

8.3.5. Командная кнопка

Элемент управления командная кнопка (CommandButton) используется для выполнения некоторых действий после ее нажатия. Иногда требуется, чтобы в окне формы была определена кнопка, выбранная по умолчанию, и кнопка отмены. Чтобы сделать кнопку выбранной по умолчанию, ее свойству Default нужно присвоить значение True. Командная кнопка становится кнопкой отмены, если ее свойству Cancel присваивается значение True.

8.3.6. Рамка

Элемент управления рамка (Frame) служит для объединения в группу других элементов управления.

8.3.7. Переключатели

Переключатели (OptionButton) допускают выбор только одного переключателя из группы. Обычно переключатели объединяются в группы или с помощью элемента Frame, или приданием свойству GroupName всех переключателей группы одного и того же значения. В экранной форме может быть создано несколько групп переключателей.

8.3.8. Счетчик (полоса значений)

Счетчик (полоса значений) (SpinButton) позволяет вводить числовые значения из ограниченного числового диапазона, который задается с помощью свойств Min и Max, а также начального значения с помощью свойства Value. Нажатие на кнопки позволяет увеличивать или уменьшать считываемые числовые значения. Изменению данных соответствует событие Change, а щелчку на верхней (или правой) кнопке – SpinUp, соответственно щечку на нижней (или левой) кнопке – SpinDown.

8.4. События экранных форм и элементов управления

Экранная форма и ее элементы управления могут реагировать на события определенных типов. Эти события, как правило, являются результатом действий пользователя – щелчков мыши на элементах управления или нажатий клавиш. Перечень таких событий приведен далее.

- Initialize – происходит при первой загрузке формы.

- Click, DblClick – происходит при одинарном или двойном щелчке мыши на объекте.

- Change – происходит при изменении данных элементов управления - CheckBox, ComboBox, OptionButton, TextBox.

- KeyDown – происходит когда элемент управления находится в фокусе и пользователь нажимает клавишу, доступно для TextBox и других элементов управления.

- MouseMove – происходит когда курсор мыши перемещается над объектом, доступно для форм и отдельных элементов управления.

8.4.1. Процедуры обработки события

Отклик на событие размещается в коде процедуры обработки события, которая автоматически выполняется в момент наступления события.

Название процедуры обработки события складывается из двух частей, разделенных дефисом: имени объекта, в котором произошло событие, и названия события. Например, имя процедуры инициализации формы, которая выполняется при наступлении события Initialize, выглядит следующим образом: UserForm_Initialize, а имя процедуры обработки события Click командной кнопки с именем cmdBegin будет такой:

cmdBegin_Click.

Процедуры обработки событий размещаются в окне редактирования кодов, которое активизируется при выборе кнопки View Code или двойном щелчке мыши на элементе управления, в котором происходит события.

В левой верхней части окна редактирования кодов расположен раскрывающийся список объектов, соответствующих элементам управления, а в правой части находится список событий, характерных для выбранного элемента управления. На рис. 19 показано окно редактирования кодов, в котором отображается процедура обработки события Click для командной кнопки с именем cmdBegin.

Ключевое слово Private определяет область видимости процедуры и означает локальный уровень видимости, т.е. данная процедура будет видна только в своем модуле, в данном случае кодах процедур экранной формы

Чтобы процедура была доступна для всех модулей проекта, можно применить ключевое слово Public, однако этот уровень доступности установлен по умолчанию и не требует дополнительных указаний.

8.5. Активизация формы

Активизировать форму в отладочном режиме из редактора VBA можно нажатием клавиши F5 или командой Run – Run Sub/UserForm.

8.6. Пример использования формы для взаимодействия с электронной таблицей

Для ввода данных в таблицу, приведенную на рис. 9, используем форму, показанную на рис. 20.

Рис. 20

Коды, обеспечивающие функционирование данной формы, приведены далее.

Формирование элементов раскрывающихся списков происходит с помощью процедуры обработки события Initialize, которое возникает при первой загрузке формы:

```
Private Sub UserForm_Initialize()
ComboBox1.AddItem "м"
ComboBox1.AddItem "ж"

ComboBox2.AddItem "Северо-запад"
ComboBox2.AddItem "Центр"
ComboBox2.AddItem "Запад"
ComboBox2.AddItem "Восток"
End Sub
```

Процедура очистки элементов формы нужна для ее подготовки к вводу новой записи:

```
Public Sub ClearForm()
TextBox1.Value = ""
ComboBox1.Value = ""
```

```
TextBox2.Value = ""
ComboBox2.Value = ""

End Sub
```

Процедура TextBox2_KeyDown(ByVal KeyCode AsMSForms.ReturnInteger , ByVal Shift As Integer) обеспечивает использование только цифровых клавиш для ввода значений окладов. Согласно таблице кодов ASCII цифровые клавиши имеют коды от 48 (0) до 57 (9). Если используется не цифровая клавиша, то формируется сообщение "Нажмите цифровую клавишу".

```
Private Sub TextBox2_KeyDown(ByVal KeyCode As
MSForms.ReturnInteger, ByVal Shift As Integer)
If (KeyCode > 32 And KeyCode < 48) Or KeyCode > 57 Then
KeyCode = 0
MsgBox "Нажмите цифровую клавишу"
End If
End Sub
```

Для проверки заполнения полей формы используется функция ValidateData(), которая возвращает булеву константу true, если все поля формы заполнены и false - в противном случае

```
Private Static Function ValidateData() As Boolean

If TextBox1.Value = "" Then
MsgBox "Укажите фамилию"
ValidateData = False
Exit Function
End If

If ComboBox1.Value = "" Then
MsgBox "Укажите пол"
ValidateData = False
Exit Function
End If

If TextBox2.Value = "" Then
MsgBox "Укажите оклад"
```

```
ValidateData = False
Exit Function
End If

If ComboBox2.Value = "" Then
MsgBox "Укажите регион"
ValidateData = False
Exit Function
End If

ValidateData = True
End Function
```

Процедура обработки события Click кнопки Ввод позволяет ввести данные из пользовательской формы в таблицу в том случае, если функция проверки заполнения полей возвратит значение true. В процедуре вычисляется номер строки последней записи (finalrow), который используется для нахождения строки нового ввода данных (ws.Cells(finalrow + 1, 1).Value). В конце процедуры происходит очистка формы.

```
Private Sub Ввод_Click()

Dim finalrow As Long
Dim ws As Worksheet
Set ws = Worksheets("Лист1")
finalrow = Cells(Rows.Count, 1).End(xlUp).Row
If ValidateData = True Then

ws.Cells(finalrow + 1, 1).Value = TextBox1.Value
ws.Cells(finalrow + 1, 2).Value = ComboBox1.Value
ws.Cells(finalrow + 1, 3).Value = TextBox2.Value
ws.Cells(finalrow + 1, 4).Value = ComboBox2.Value
Worksheets("Лист1").Columns("a:d").AutoFit
End If
ClearForm
End Sub
```

Процедура закрытия формы приведена далее.

```
Private Sub Закрыть_Click()
ClearForm
Me.Hide
End Sub
```

Активизация формы происходит при нажатии на элемент управления, созданный в листе Excel, которому присоединен макрос с кодом UserForm1.Show

СПИСОК ЛИТЕРАТУРЫ

1. Истомин, Е. П.. Информатика и программирование : Pascal и VBA. учебник для вузов : / [ред. Л. С. Слесарева] ; Рос. гос. гидрометеорологический ун-т : СПб. : Андреевский издательский дом, 2010. – 293 с.

2. Стригина, Е. В. Программирование IT сервисов предприятия. : методические указания к выполнению лабораторных работ. / Е. В. Стригина. – СПб.: Издательство СПбГУТ, 2013. - 40 с.

3. Стригина, Е. В., Средства обработки и анализа данных : методические указания к выполнению лабораторных работ / Е. В. Стригина. – СПб.: Издательство СПбГУТ, 2012. – 51с.

4. Вольфсон, М. Б. , Средства обработки и хранения данных : учебное пособие / М. Б. Вольфсон, Е. В. Стригина. – СПб.: Издательство СПбГУТ, 2012. –54 с.

5. Слепцова, Л. Д. Программирование на VBA в Microsoft Office 2007 / Л. Д. Слепцова. – М. : ООО «Диалектика, Вильямс», 2007. – 688 с.

6. Джелен, Б. VBA и макросы в Microsoft Office Excel 2007 / Б. Джелен, Т. Сирстал ; пер.с англ. –М. : ООО «И.Д. Вильямс», 2008. –212 с.

7. Информатика: учебник / Б. В. Соболь, А. Б. Галин, Ю. В. Панов и др. – Ростов н/Д : Феникс, 2006. - 446 с.

9. Уокенбах, Дж. Microsoft Office Excel 2007 : Библия пользователя / Дж. Уокенбах. – М. : ИД «Вильямс», 2009. – 816 с.

Стригина Елена Владимировна

ПРОГРАММИРОВАНИЕ
IT сервисов предприятия

Учебное пособие

Редактор Л. А. Медведева
План 2013 г., п. 124

Подписано к печати 29.03.2013
Объем 4 усл.-печ. л. Тираж 40 экз. Заказ 169

РИЦ СПбГУТ. 191186 СПб., наб. р. Мойки, 61
Отпечатано в СПбГУТ

Е. В. Стригина

ПРОГРАММИРОВАНИЕ IT сервисов предприятия

Учебное пособие

САНКТ-ПЕТЕРБУРГ
2013