



**Федеральное государственное образовательное бюджетное учреждение  
высшего образования  
«ФИНАНСОВЫЙ УНИВЕРСИТЕТ  
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»  
(Финансовый университет)**

**Департамент анализа данных и машинного  
обучения**

**С. В. Маркова**

**УЧЕБНОЕ ПОСОБИЕ**

**по дисциплине**

**«ПРОГРАММИРОВАНИЕ НА VBA»**

**Москва 2020**

**Федеральное государственное образовательное бюджетное учреждение  
высшего образования  
«ФИНАНСОВЫЙ УНИВЕРСИТЕТ  
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»  
(Финансовый университет)**

**Департамент анализа данных  
и машинного обучения**

**С.В. Маркова**

**УЧЕБНОЕ ПОСОБИЕ**

**по дисциплине**

**«ПРОГРАММИРОВАНИЕ НА VBA»**

**для студентов, обучающихся по направлению**

**09.03.03 «Прикладная информатика»**

*Одобрено на заседании Совета департамента  
анализа данных, принятия решений и финансовых технологий  
(протокол №12 от 2 июня 2020 г.)*

**Москва 2020**

УДК 004  
ББК 32.973.2  
М26

Рецензент:

**Гайдамака А. И.**, к.в.н., доцент Департамента анализа данных, принятия решений и финансовых технологий Финансового университета при Правительстве РФ

**С.В. Маркова**

Программирование на VBA: Учебное пособие для проведения семинарских занятий по одноименной дисциплине по выбору студентов, обучающихся по направлению по направлению «Прикладная информатика», профиль «ИТ-сервисы и технология обработки данных в экономике и финансах», профиль «Высокопроизводительные вычисления в цифровой экономике» (программа подготовки бакалавра). – М: ФГОБУ ВО «Финансовый университет при Правительстве Российской Федерации», департамент «Анализа данных, принятия решений и финансовых технологий». 2020.- 92 с.

Учебное пособие содержит практические работы для изучения языка программирования Visual Basic for Application (VBA). Содержит множество практических примеров, которые помогут в разработке собственных алгоритмов и написании собственных программ на VBA.

*Учебное издание*

**Маркова Светлана Владимировна**

**УЧЕБНОЕ ПОСОБИЕ ПО ДИСЦИПЛИНЕ  
"ПРОГРАММИРОВАНИЕ НА VBA"**

Учебное пособие

Компьютерный набор, верстка Маркова С.В.  
Формат 60х90/16. Гарнитура *Times New Roman*  
Усл. п.л. 5,7. Изд. № - 2020 Заказ №  
*Электронное издание*

© С.В. Маркова, 2020

© ФГОБУ ВО «Финансовый университет при Правительстве Российской Федерации», 2020

## ОГЛАВЛЕНИЕ

<b>ВВЕДЕНИЕ</b>	<b>6</b>
<b>Практическая работа №1</b>	<b>7</b>
<b>РЕДАКТОР VBA. ЗНАКОМСТВО С ИНТЕРФЕЙСОМ</b>	<b>7</b>
1. Структура редактора VBA	7
2. Окно проекта	8
3. Интеллектуальные возможности редактора кода	9
4. Окно свойств	11
5. Окно просмотра объектов Object Browser	11
6. Работа со справочной системой	13
Контрольные вопросы	14
<b>Практическая работа № 2</b>	<b>16</b>
<b>СТРУКТУРА ПРОГРАММ НА VBA. ПРОЦЕДУРЫ И ФУНКЦИИ</b>	<b>16</b>
1. Структура программного кода	16
2. Алфавит языка VBA	19
3. Имя переменной, постоянной, функции, процедуры	20
4. Типы данных	21
5. Описание простых переменных	22
6. Описание констант	22
7. Описание массивов	22
8. Выражения	23
9. Виды операций	23
10. Ввод и запуск программы на выполнение	24
Задания для самостоятельной работы	25
Контрольные вопросы	26
<b>Практическая работа №3</b>	<b>27</b>
<b>ОСНОВНЫЕ ОПЕРАТОРЫ ЯЗЫКА VBA</b>	<b>27</b>
1. Правила записи операторов	27
2. Оператор присваивания	27
3. Операторы ввода-вывода	27
4. Условный оператор IF	30
5. Оператор выбора Select Case	32
6. Операторы цикла	33
Задания для самостоятельного выполнения	35
Контрольные вопросы	36
<b>Практическая работа №4</b>	<b>37</b>
<b>ОБЪЕКТНАЯ МОДЕЛЬ EXCEL</b>	<b>37</b>
1. Основные сведения об объектах, методах и свойствах EXCEL	37
Задания для самостоятельного выполнения	43
Контрольные вопросы	44
<b>Практическая работа № 5</b>	<b>45</b>
<b>МАКРОСЫ</b>	<b>45</b>
1. Виды макросов	45
2. Способы создания макросов	45
3. Места хранения макросов	46
4. Создание и сохранение макросов	46
5. Запуск макроса	47

6 Редактирование макроса	49
Задания для самостоятельного выполнения	49
Контрольные вопросы	50
<b>Практическая работа №6</b>	<b>51</b>
<b>РАЗРАБОТКА ПОЛЬЗОВАТЕЛЬСКИХ ФОРМ</b>	<b>51</b>
1. Пользовательские формы, свойства и методы	51
2. Элементы управления: общие сведения	55
3. Пример разработки пользовательской формы	56
Задания для самостоятельного выполнения	59
Контрольные вопросы	60
<b>Практическая работа №7</b>	<b>61</b>
<b>ФОРМЫ ДЛЯ РАБОТЫ С БАЗОЙ ДАННЫХ</b>	<b>61</b>
1. Проектирование формы. Типы элементов управления	61
2. Визуализация данных	67
Задания для самостоятельного выполнения	69
Контрольные вопросы	74
<b>Практическая работа №8</b>	<b>75</b>
<b>АВТОМАТИЗАЦИЯ С VBA В MS ACCESS</b>	<b>75</b>
1. В чем отличия работы с базами данных в MS Excel и MS Access	75
2. Основные сведения об объектах ACCESS, DAO и ADO	75
3. Объекты DAO	77
Задания для самостоятельной работы	82
Контрольные вопросы	83
<b>Практическая работа №9</b>	<b>84</b>
<b>ПРОГРАММИРОВАНИЕ В MS WORD</b>	<b>84</b>
1. Основные сведения об объектах Word, их свойствах и методах	84
2. Запись макросов с помощью макрорекордера и способы выполнения макросов в приложении Microsoft Word	87
3. Создание формы пользователя для ввода переменной части текста	89
Задание для самостоятельного выполнения	92
Контрольные вопросы	92

## ВВЕДЕНИЕ

VBA (Visual Basic for Applications) — это упрощенная версия языка Visual Basic, который предназначен для работы с приложениями Microsoft Office и другими приложениями не только от Microsoft, но и других фирм.

VBA встроен в приложения Office, и этот код хранится внутри документов Word, книгах Excel, презентациях PowerPoint и т. п. Эти программы можно запускать из документов на выполнение, так как среда выполнения кода VBA встроена внутрь этих приложений.

В настоящее время VBA внедрён не только во все приложения Microsoft Office — Word, Excel, Access, PowerPoint, Outlook, FrontPage, InfoPath, но и в другие, приложения Microsoft, такие как Visio и Project. Кроме того, VBA используют более 100 приложений третьих фирм, например, в CorelDRAW, AutoCAD и т. п.

Достоинства VBA:

- VBA — универсальный язык. Освоив его, можно создавать полноценные приложения на Visual Basic (поскольку эти языки — близкие родственники), а также использовать все возможности языка VBScript ("урезанный" VBA). Владея этими инструментами, можно создавать скрипты администрирования Windows, Web-страниц (VBScript в Internet Explorer), Web-приложений ASP и много другое.
- В Office встроены мощные средства, которые облегчают работу пользователя: подсказки по объектам и по синтаксису, макрорекордер и т. п., поэтому разработка программ не слишком сложна.
- Приложения на VBA занимают мало памяти, хорошо работают, например, на сервере терминалов.
- Программы на VBA удобно отлаживать в режиме интерпретатора.

Данное учебное пособие содержит комплекс практических работ, предназначенных для получения навыков программирования на VBA в офисных приложениях.

Практическая работа №1  
**РЕДАКТОР VBA. ЗНАКОМСТВО С ИНТЕРФЕЙСОМ**

Цель работы. Изучить основные инструменты и средства редактора VBA в MS Excel, работу со справочной системой

**1. Теоретические сведения. Выполнение работы**

**1.1 Структура редактора VBA**

1. Запустите программу MS Excel.
2. Выполните команду Файл – Сохранить как...
3. В окне Сохранение документа укажите место хранения файла и его имя – Пр1.
4. Выполните команду Файл – Параметр.
5. В категории «Настроить ленту» установите флажок «Показывать вкладку "Разработчик"» на ленте, а затем нажмите кнопку ОК.
6. На вкладке Разработчик в группе Код (Рисунок 1) нажмите кнопку Visual Basic.

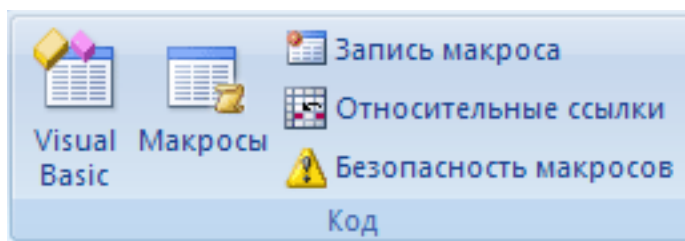


Рисунок 1 Группа Код

На экране появится окно редактора Visual Basic (Рисунок 2). Основные компоненты интерфейса VBA:

- меню;
- окно проекта;
- окно редактирования кода;
- окно свойств;
- окно форм панели инструментов;
- Панели инструментов.

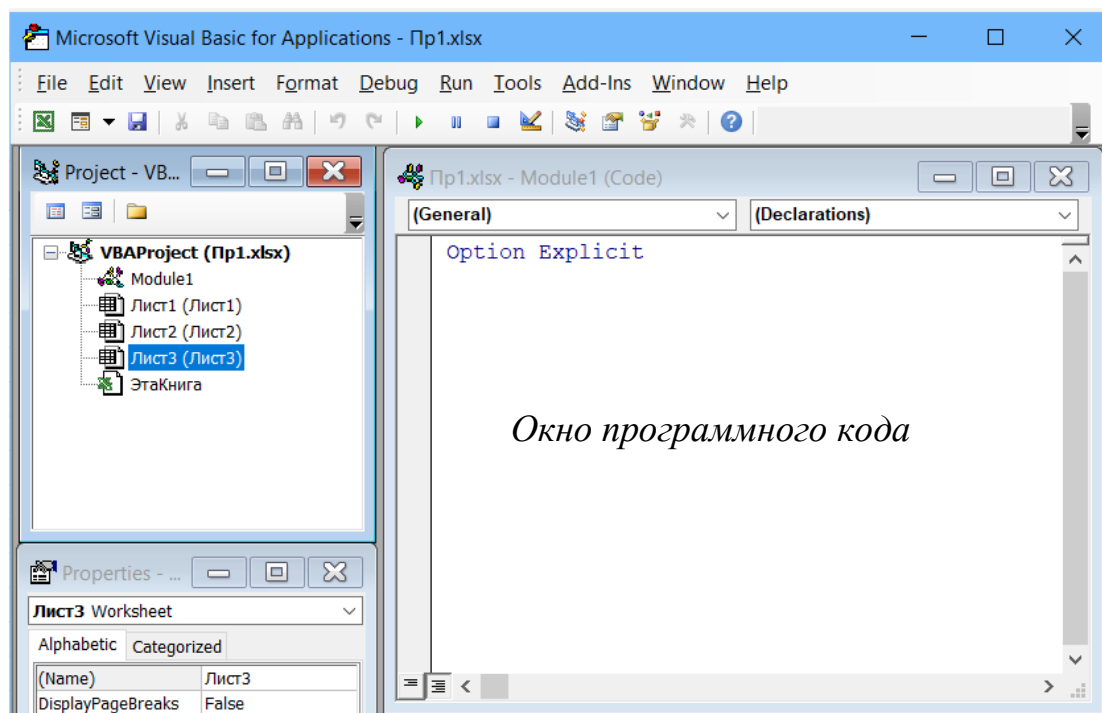



Рисунок 2. Окно редактора Visual Basic

## 1.2 Окно проекта

Окно проекта можно активизировать командой View/Project Explorer или нажатием кнопки . Окно проекта содержит иерархическую структуру файлов форм и модулей текущего проекта (Рисунок 2).

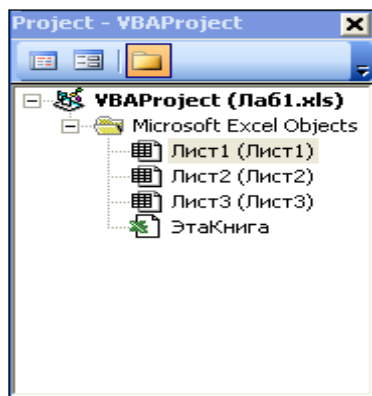


Рисунок 3. Иерархическая структура файла форм и модулей текущего проекта

Модули по назначению делятся на стандартные и модули объектов. Для каждого рабочего листа и для всей рабочей книги модуль в проекте создается автоматически. Модули, связанные с рабочими листами, рабочей книгой, формами, и модули класса относятся к модулям объектов. Стандартным



называется модуль, который содержит макросы. Такие модули добавляются в проект командой Insert/Module. Формы создаются командой Insert/UserForm.

Проекты всех открытых рабочих книг можно увидеть в окне проекта. Это способствует ускорению процесса создания новых приложений, так как позволяет легко копировать формы и коды из одного проекта в другой. Двойной щелчок мышью на значке файла в окне проекта открывает окна редактора для соответствующего модуля.

- Щелкните дважды на объекте Лист1. Справа появится окно для редактирования кода (Рисунок 4).

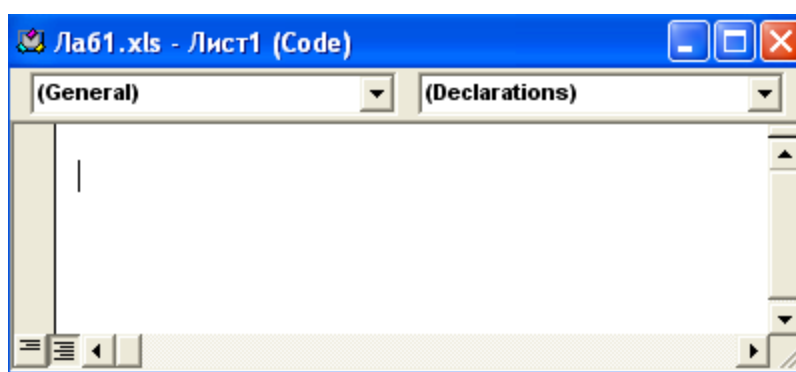


Рисунок 4. Окно для редактирования кода

### 1.3 Интеллектуальные возможности редактора кода

Значительно облегчается написание программ за счет способности редактора кода автоматически завершать написание операторов, свойств и параметров. В процессе набора команды, редактор сам предлагает пользователю список компонентов, логически завершающих вводимую пользователем команду. Сделав двойной щелчок на выбранном элементе из этого списка, или нажав клавишу Tab, выбранное имя вставляется в код программы.

Если курсор расположить на ключевом слове языка VBA, имени процедуры, свойства или метода и нажать клавишу F1, то на экране появится окно со справочной информацией об этой функции.

С помощью команды Tools/Options и используя вкладку Editor диалогового окна Options можно настроить редактор VBA:

- Auto Syntax Check - автоматическая проверка синтаксиса введенной строки программы.
- Require Variable Declaration - устанавливает обязательное явное описание переменных в модулях.
- Auto List Members - выводит список данных, логически завершающих инструкцию.
- Auto Quick Info - сразу после ввода имени процедуры обеспечивает вывод на экран сведений о процедурах: функциях, подпрограммах, свойствах, методах – и их параметрах.
- Auto Data Tips - в режиме прерывания отображает значение переменной, на которой установлен курсор.
- Auto Indent - задание положения табуляции для первой и последующих строк.
- Drag-and-Drop Text Editing. Дает возможность перетаскивать текст с помощью мыши.
- Default to Full Module View - для вновь открываемых модулей устанавливает режим просмотра всех процедур.
- Procedure Separator - отображает или скрывает полосу, разделяющую каждую процедуру в окне модуля.

Вкладка General(*Основные*) содержит опции, которые управляют непосредственно параметрами среды редактора Visual Basic:

- Form Grid Setting(*Установка сетки формы*) - управляет отображением и размером сетки в режиме конструктора экранных форм;
- Edit and Continue - редактирование и завершение;
- Error Trapping - обнаружение ошибок;
- Compile - компилирование.

Вкладка Docking(*Постановка в док*). На этой вкладке расположены флажки всех окон редактора Visual Basic. Если флажок снят, то

соответствующее окно будет плавающим в окне редактора, если флажок установлен, то окно будет "поставлено в док" вдоль одной из сторон окна редактора.

Щелкните на кнопке Отмена для закрытия диалогового окна Options.

#### 1.4. Окно свойств

Основные параметры активного элемента отображаются в окне свойств (рисунок 5), которое состоит из двух частей: раскрывающегося списка, из которого можно выбрать любой элемент управления текущей формы или саму форму и рабочей части, состоящей из двух вкладок: Alphabetic (По алфавиту) и Categorized (По категориям). Имя элемента управления (свойство Name) идет первым. Значения свойств можно поменять путем ввода с клавиатуры или выбором из раскрывающегося списка.

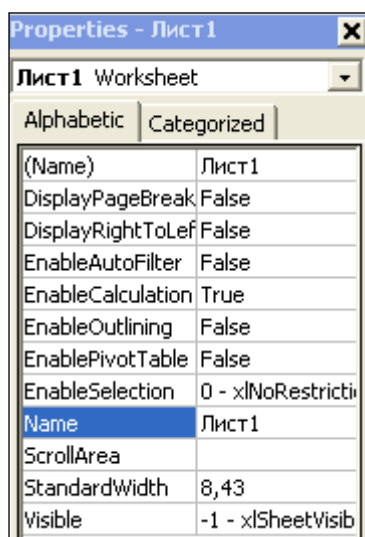



Рисунок 5. Окно свойств

#### 1.5 Окно просмотра объектов Object Browser

Окно просмотра объектов Object Browser позволяет просматривать все объекты проекта. Здесь можно настроить свойства, методы и события, связанные с любым объектом. Доступ к этому окну можно получить одним из способов:

- Щелкнуть на кнопке  Object Browser стандартной панели инструментов редактора Visual Basic.

- Выполнить команду View > Object Browser (Вид - Просмотр объектов).
- Нажать клавишу <F2>.

Окно Object Browser - хорошее обучающее и справочное средство, здесь можно найти все свойства и методы, относящиеся к любому интересующему вас объекту (рис. 6).

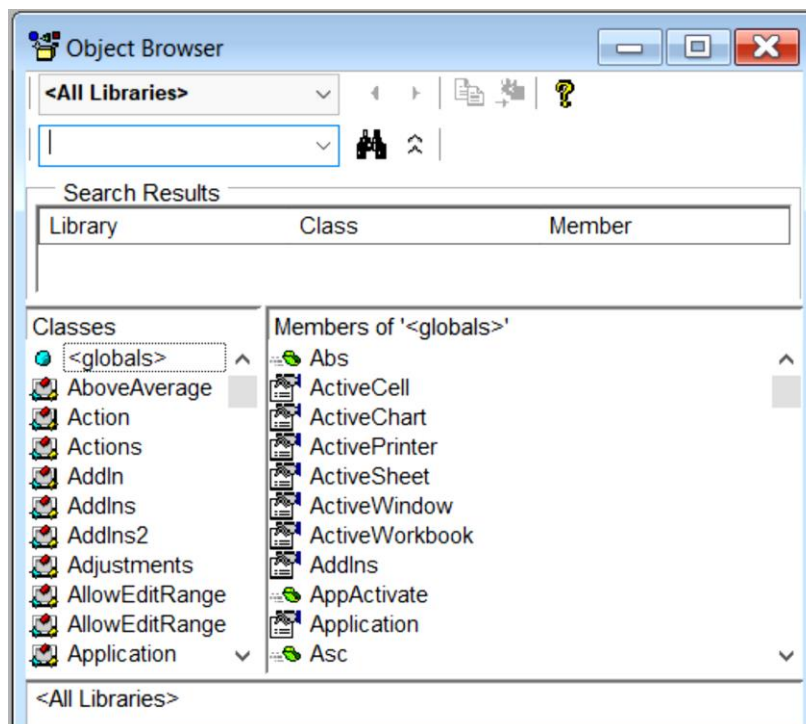


Рисунок 6. Окно просмотра объектов - Object Browser

Данное окно состоит из трех частей:

- <All libraries> (Все библиотеки) в левом верхнем углу окна. В этом раскрывающемся списке можно выбрать различные библиотеки объектов Excel, VBA, Office и VBAProject.
- Classes (Классы). Содержимое данного списка зависит от выбранной библиотеки. Например, после выбора из раскрывающегося списка VBA, все классы этой библиотеки будут выведены в списке Классы.
- Members (Компоненты). Содержимое данного списка зависит от выбора Класа просматриваемой библиотеки, все компоненты выбранного класса выводятся в списке Компоненты.

Из окна Object Browser можно быстро открыть тему справочной системы, описывающую выбранный объект, свойство или метод. Опишем работу в окне Object Browser.

1. Нажмите клавишу <F2> для открытия окна Object Browser
2. В списке Classes (Классы) найдите объект Range и выделите его
3. Прокрутите соседний список Members of 'Range' для просмотра свойств и методов объекта Range
4. Выберите метод Activate
5. Нажмите клавишу <F1>. Откроется окно справочной системы Visual Basic с описанием метода Activate.

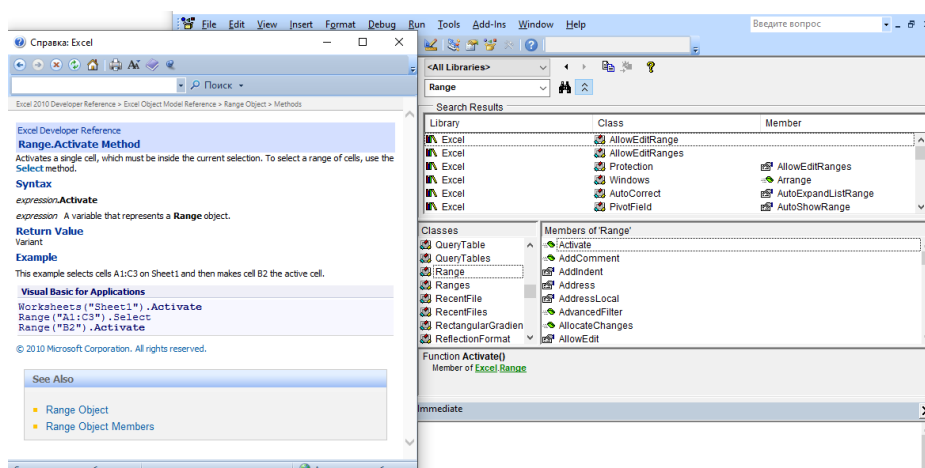


Рисунок 7. Окно справочной системы Visual Basic с описанием метода Activate

6. Закройте окно справочной системы.
7. Закройте окно Object Browser.

## 1.6 Работа со справочной системой

Рассмотрим работу со справочной системой VBA.

1. Выполните команду Help > Справка по Microsoft Visual Basic (рис. 8).

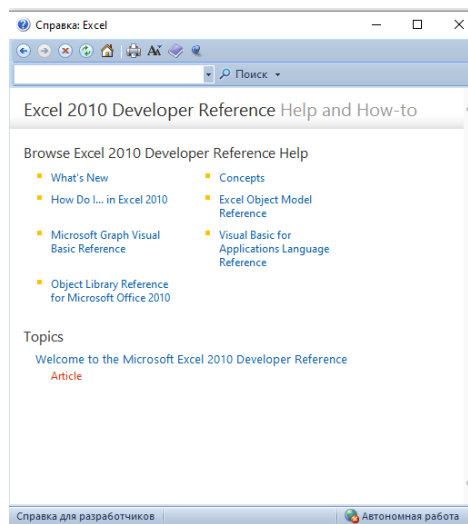


Рисунок 8. Окно Справочной системы

2. Введите вопрос *How do I create a new workbook?* (Как создать новую книгу?) и нажмите клавишу <Enter>.
3. В окне из предложенного списка тем выберите *Creating a New Workbook* (Создание новой книги), которая отобразится в окне справочной системы (рисунок 9).



Рисунок 9. Тема, выбранная в окне помощника, открывает окно справочной системы

Справочная система Visual Basic показывает многочисленные примеры кода. Можно скопировать эти примеры из окна справочной системы и вставить в свои процедуры.

Контекстно-зависимая справка - одно из лучших справочных средств, предлагаемых VBA. При использовании какого-то объекта, свойства, метода или функции можно получить справку о введенном элементе, нажав клавишу <F1>.

### **Контрольные вопросы**

1. Как можно использовать справочную систему, если неизвестно даже имя нужного свойства объекта?
2. Перечислите основные компоненты окна редактора VBA.
3. Какую информацию можно получить, открыв окно Object Browser?
4. Как заменить шрифт Courier, используемый в редакторе Visual Basic?
5. Опишите три способа получения сведений из справочной системы VBA.
6. Окно Object Browser можно использовать для просмотра списков объектов, событий, свойств и \_\_\_\_\_.
7. Какую комбинацию клавиш надо нажать, чтобы в окне кода перейти в начало модуля?
8. \_\_\_\_\_ маленькое окно с информацией о синтаксисе вводимой функции и ее аргументах.
9. Справедливо ли утверждение: в справочной системе VBA можно только просматривать приведенные примеры кода, копировать их в свои процедуры запрещено.
10. Где находятся опции, управляющие такими средствами редактора Visual Basic, как экранная подсказка и автоматическая проверка синтаксиса?

## Практическая работа № 2

### СТРУКТУРА ПРОГРАММ НА VBA. ПРОЦЕДУРЫ И ФУНКЦИИ

Цель работы. Изучить структуру программного кода на VBA, синтаксис VBA

#### 1. Теоретические сведения. Выполнение работы

##### 1.1 Структура программного кода

Процесс разработки программы на языке VBA – проекта, может состоять из нескольких этапов, в зависимости от конечного результата. Если необходимо получить программу, которая будет производить определенные вычисления или действия, расширяющие математические возможности стандартного приложения Microsoft Office, то достаточно создать программный модуль. Для применения этой программы можно поместить в рабочей области приложения кнопку, нажатие которой будет вызывать выполнение программы. Для этого в приложении необходимо включить панель инструментов с помощью команды Вид - Панели инструментов - Элементы управления, а затем создать кнопку с соответствующим программным кодом. Либо выполнять программу с помощью команды Разработчик - Макросы.

Разработка программы, для выполнения которой требуется отдельное окно, с различными элементами управления будет включать два этапа. Первый этап – этап визуального программирования, на котором создается окно (форма) программы, где располагаются необходимые элементы управления. Второй – этап программирования, на котором создаются части программы (процедуры), выполняющиеся в ответ на определенные события. **Событием** является, например, щелчок левой кнопкой мыши на командной кнопке (событие Click), нажатие клавиши на клавиатуре (событие KeyPress) и т.д.

Программы на VBA хранятся в проектах. Проект содержит модули различных типов, а модули включают различные процедуры. Добавление модуля осуществляется по команде Insert - Module (в этом случае программный код составляется пользователем) или при создании макроса (когда программа



создается автоматически). Созданному модулю присваивается стандартное имя Module1, Module2 и т. д. Проект может содержать несколько модулей.

**Модули класса** содержат описание объекта, который является членом класса. Процедуры, написанные в модуле класса, используются только в этом модуле. Модули класса это могут быть модули форм и отчетов, которые связаны с конкретной формой или отчетом. Модули форм и отчетов часто содержат процедуры обработки событий, которые срабатывают в ответ на событие в форме или отчете. Процедуры обработки событий используются для управления поведением форм и отчетов и их реакцией на действия пользователя типа щелчка мыши на кнопке.

Самое начало модуля называется общей областью, в которой располагаются общие описания, например, типа данных, используемого по умолчанию (DefТип), инструкция Option Explicit, требующая явного описания всех используемых в модуле переменных, а также описания общих (глобальных) для всех модулей и для данного модуля переменных. Если выполнить команду Tools - Options, в диалоговом окне Options, во вкладке Editor поставить флажок Require Variable Declaration, то VBA будет вставлять оператор Option Explicit автоматически в начале каждого нового модуля.

Модули содержат описания и процедуры - наборы описаний и инструкций, сгруппированных для выполнения. Существуют типы процедур:

- процедура Sub - набор команд, с помощью которого можно решить определенную задачу. При ее запуске выполняются команды процедуры, а затем управление передается в приложение пакета MS Office или процедуру, которая вызвала данную процедуру. В упрощенном виде структура любой процедуры выглядит:

```
Sub <имя> [(<аргументы>)]  
.....  
End Sub
```

- процедура Function (функция) также представляет собой набор команд, который решает определенную задачу. Такие процедуры

обязательно возвращают значение, тип которого можно описать при создании функции.

```
Function  
...  
End Function
```

Приведем пример процедуры Function на VBA именем SmMn, которая получает три аргумента x, y, z типа Double (число, с плавающей запятой двойной точности).

```
Function SmMn(x As Double,y As Double, z As Double) As Double  
    SmMn = x + y - z  
End Function
```

Вызвать процедуру Function можно из другой процедуры VBA при помощи простого присваивания этой процедуры переменной. В следующем примере показано обращение к процедуре SmMn, которая была создана выше.

```
Sub main()  
    Dim tal as Double  
    tal = SmMn(5, 4, 3)  
End Sub
```

Область действия процедуры, также как и область действия переменных, можно задать при помощи ключевых слов Public и Private.

Оператор – это элементарная единица программного VBA-кода, способная выполняться. Например, оператор присваивания, оператор цикла и др.

На данном этапе мы будем создавать программы в рамках следующей синтаксической конструкции:

```
[Option Explicit]  
[Private | Public] Sub <Имя> ([<Список аргументов>])  
    [Dim <Имя> [As <Тип>]]  
    [Const <Имя> [As <Тип>] = <Выражение>]  
    [Инструкции]  
    [Exit Sub]  
    [Инструкции]  
End Sub
```

В этой записи:

- Option Explicit – инструкция, которая обязывает явно описывать все переменные, встречающиеся в программе;
- Public – ключевое слово, которое указывает на область видимости процедуры - доступна для всех других процедур во всех модулях;
- Private – процедура доступна для других процедур модуля, в котором она описана;
- <Имя> – имя процедуры;
- <Список аргументов> – список переменных, представляющий аргументы, которые передаются в процедуру при ее вызове. Имена переменных разделяются запятой.
- Dim <Имя> [As <Тип>] – блок описания переменных;
- Const <Имя> [As <Тип>] = <Выражение> – блок описания констант;
- Инструкции – набор любых операторов VBA;
- Exit Sub – немедленный выход из процедуры;
- Sub, End Sub – служебные слова VBA.

*Внимание! В приведенных синтаксических конструкциях операторов VBA угловые скобки < > свидетельствуют об обязательном аргументе, а квадратные [ ] – необязательный параметр, т.е. может быть, а может не быть.*

## 1.2 Алфавит языка VBA

Для записи операторов, функций, имен, математических выражений используются:

- все прописные и строчные буквы латинского алфавита;
- арабские цифры;
- специальные знаки ! & ' \$ ? , . { } ( ) [ ] = - + \_ ^ % / ~ < > : ;
- допускается использование букв кириллицы в именах.

## 1.3 Имя переменной, постоянной, функции, процедуры

В VBA пользователь определяет имена переменных, функций, процедур, постоянных и других объектов. Вводимые пользователем имена должны

отражать суть обозначаемого объекта так, чтобы делать программу легко понимаемой. В VBA имеются следующие ограничения на имена:

- длина имени не должна превышать 255 символов;
- имя не может содержать стандартные разделители (точку, запятую, двоеточие, дефисов, пробелов и т.п.) и следующих символов: %, &, !, @, #, \$;
- имя может содержать любую комбинацию не запрещенных символов, но начинаться должно с буквы;
- имена должны быть уникальны внутри области, в которой они определены;
- запрещено использовать имена, совпадающие с ключевыми словами VBA и именами встроенных функций и процедур.

Примеры имен переменных (допустимых и недопустимых) приведены в табл. 1.

Таблица 1

Примеры имен переменных

Допустимые имена	Недопустимые имена	Ошибка!
A	Имя более 255 символов	Не допустимо более 255 символов
Go4Ln	1p	Начинается с цифры
SUMMA	P 1	Содержит пробел
P1	W!	Содержит !
S_1	Sub	Ключевое слово VBA

В VBA прописные и строчные буквы не различаются, но введенные прописные буквы сохраняются.

#### 1.4 Типы данных

Все объекты, которыми оперирует язык программирования VBA, относятся к определенному типу. Тип данных определяет:

- область возможных значений переменной;
- структуру организации данных;
- операции, определенные над данными этого типа.

Типы данных делятся на простые (скалярные) и сложные (структурированные). У простых типов данных возможные значения данных едины и неделимы. Сложные же типы имеют структуру, в которую входят

различные простые типы данных. Основные типы данных в VBA приведены в таблице 2.

Таблица 2

#### Типы данных VBA

Тип данных	Содержимое переменной	Диапазон значений
Boolean	Логический	True (1) или False (0)
Byte	Целое число, размером в 1 байт	0 - 255
Integer	Целое число	-32768 - 32767
Long	Целое длинное число	От -2147483648 до 2147483647
Single	Число, с плавающей запятой одинарной точности	От -3,402823E38 до -1,401298E-45 для отрицательных значений и от 1,401298E-45 до 3,402823E38 для положительных значений
Double	Число, с плавающей запятой двойной точности	От -1,79769313486231E308 до 1,79769313486232E308
Currency	Число с 19 разрядами в целой части и 4 в дробной	От -922337203685477,5808 до 922337203685477,5807
Object	Объектный тип	Указывает на любой объект
String	Строка	От 0 до 65400 символов
Variant	Значение любого типа	
Пользовательский Type	Набор переменных, определяемых пользователем вместе как единое целое	

Переменные в программе можно описывать или не описывать. Если не описывать, то переменной будет присвоен тип Variant. Явно описывать переменную можно как в начале блока, так и в любом месте, где возникла необходимость использовать новую переменную. Лучше все переменные описывать явно и, как правило, в начале блока.

#### 1.5 Описание простых переменных

Описание простых переменных имеет следующий синтаксис:

*Dim имя\_переменной As имя\_типа*

Одним оператором Dim можно описать произвольное число переменных, но конструкция As должна быть указана для каждой из них, иначе переменным без As будет присвоен тип Variant.

Например, Dim A As Byte, B As Integer, C, Parus As String - здесь переменная A - это переменная байтового типа, переменная B - целого типа, переменная C - типа вариант (не задан тип, по умолчанию), переменная Parus - строкового типа.

### 1.6 Описание констант

Данные, не изменяющиеся внутри программы, можно считать константами. Их можно описать следующим образом:

Const *имя\_константы* As *имя\_типа* = *постоянное\_выражение*

Например, Const Pi As Double = 3.141593

### 1.7 Описание массивов

Массив — это структурированный тип данных, который представляет собой последовательность ячеек памяти, имеющих общее имя и хранящих данные одного типа. Каждый элемент массива определяется индексом (номером). Количество элементов в массиве называется размерностью массива. Данная структура используется для хранения векторов, матриц.

Массив описывается следующей конструкцией:

Dim *имя\_массива*(*список\_размерностей*) As *имя\_типа*

В списке размерностей массива каждое измерение отделяется запятой и определяется заданием нижней и верхней границ изменения индексов.

Например, Dim B(1 TO 8) As Integer, D(1 To 5, 1 To 10) As Double

Здесь B - одномерный массив, состоящий из 8 элементов целого типа, Y - двумерный массив, у которого 5 строк и 10 столбцов с элементами числового типа двойной точности.

### 1.8 Выражения

Выражения состоят из знаков операций и операндов. Операндами являются константы, переменные, указатели функций, выражения, взятые в скобки. Выражения устанавливают порядок выполнения действий над элементами данных.

### 1.9 Виды операций

VBA разделяет операции на три главные категории: арифметические, логические и операции сравнения. Если выражение содержит знаки операций из двух или более категорий, то VBA выполняет операции в следующем порядке:

1. арифметические;
2. операции сравнения;
3. логические операции.

Чтобы изменить последовательность выполнения операций, в выражении используют скобки. Внутри каждой категории операций тоже имеются правила порядка выполнения операций (таблица 3)

Таблица 3

Порядок выполнения операций в VBA

Знаки операций	Операции
<b>Арифметические</b>	
^	Возведение в степень
-	Отрицание
* или /	Умножение или деление
\	Целочисленное деление
Mod	Вычисление остатка от деления
+ или -	Сложение или вычитание
<b>Сравнение</b>	
=	Равенство
< >	Неравенство (не равно)
<	Меньше
>	Больше
< =	Меньше или равно
> =	Больше или равно
<b>Логические</b>	
Not	Логическое НЕТ
And	Логическое И
Or	Логическое ИЛИ

Арифметические выражения записываются с помощью операндов числовых типов и арифметических операций, а результатом является числовое значение. В арифметическом выражении можно использовать стандартные математические функции, которые приведены в таблице.

## Математические функции VBA

Функция	Значение
Abs(число)	Модуль числа
Atn(число)	Арктангенс от числа
Cos(число)	Косинус от числа
Exp(число)	Экспонента
Fix(число)	Выделение целой части числа
Int(число)	Целая часть числа. Функция Int возвращает ближайшее меньшее целое, Fix отбрасывает дробную часть числа
Log(число)	Натуральный логарифма числа
Rnd	Случайное число
Sgn(число)	=1, если число $>0$ ; =0, если число $=0$ ; = -1, если число $<0$ .
Sin(число)	Возвращает синус числа
Sqr(число)	Извлекает квадратный корень из числа
Tan(число)	Возвращает тангенс числа

## 1.10 Ввод и запуск программы на выполнение

Будем осуществлять написание собственного кода на языке VBA в среде редактора Visual Basic приложения MS Excel.

- Для написания кода выполните команду Insert (Вставить)/Procedure(Процедуру).
- В появившемся окне Add Procedure (Рисунок 7) в поле Name необходимо ввести имя процедуры или функции и задать область ее действия.

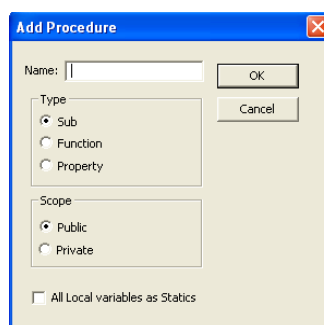



Рисунок 7. Окно Добавление процедуры



Введите в поле Name *пример*, затем нажмите ОК. После этого будут автоматически сформированы операторы начала и конца процедуры и можно переходить непосредственно к набору операторов процедуры.

Для набора следующей процедуры в том же модуле необходимо повторить команду Insert(*Вставить*)/Procedure(*Процедура*). Если нужно создать новый модуль повторяется команда Insert(*Вставить*)/Module(*Модуль*).

Проверка правописания осуществляется на этапе компиляции командой Debug(*Отладка*)/Compile VBAproject(*компилировать*).

Для запуска программы требуется выполнить команду Run(*Запуск*)/Run Sub/UserForm (*Запуск подпрограммы/UserForm*) или нажать клавишу <F5> или соответствующую кнопку панели инструментов .

## 2. Задания для самостоятельной работы

1. Допустимо ли данное имя, если нет, то объясните почему?

- a) First\_Program
- б) const
- в) firma@ru
- г) Наименьшее

2. Определите значение переменных по фрагменту программы:

А)  
Dim x As String, y As String, z As String  
x = "Завтрак"  
y = "Ужин"  
z = " на "  
x = x & z & "утро,"  
y = y & z & "вечер"  
z = x & y

Б)  
Dim f As Boolean, h As Boolean, k As Boolean  
f = 6\*6 = 25  
h = (f+25) >= 35  
k = f Or h

3. Написать функцию, которая вычисляет сумму квадратов двух чисел

## 3. Контрольные вопросы

1. Что определяет тип данных?
2. Какие категории операций существуют в VBA?
3. Почему в программе необходимо описывать используемые переменные?

## Практическая работа №3 ОСНОВНЫЕ ОПЕРАТОРЫ ЯЗЫКА VBA

Цель работы. Изучить синтаксис операторов VBA

### 1. Теоретические сведения. Выполнение работы

#### 1.1 Правила записи операторов

При записи операторов необходимо придерживаться следующих правил:

- Каждый новый оператор записывается с новой строки;
- Чтобы записать несколько операторов на одной строке, их разделяют между собой двоеточием (:);
- Если оператор не помещается в одной строке, то необходимо поставить в конце строки пробел и знак подчеркивания ( \_ ), а затем продолжить на следующей строке.

#### 1.2 Оператор присваивания

Оператор присваивания используется, если какой-то переменной нужно присвоить новое значение. Он имеет следующий синтаксис:

имя\_переменной = выражение

Сначала вычисляется выражение в правой части, а затем результат присваивается переменной, стоящей в левой части.

Например, выражение  $Y = \frac{\sqrt{a+b} + b^2}{(a+b+c)^3} \cdot \operatorname{tg} a$  на VBA запишется как

$y = (\operatorname{Sqr}(a+b) + b^2) / (a+b+c)^3 * \operatorname{Tan}(a)$ .

#### 1.3 Операторы ввода-вывода

##### Оператор и функция MsgBox

Вывод информации в VBA осуществляется с помощью оператора вывода. Процедура MsgBox выводит на экран диалоговое окно, содержащее сообщение, устанавливает режим ожидания нажатия кнопки пользователем.

Синтаксис:

MsgBox (Prompt, [, button][, title])

Аргументы:

Prompt – строковое выражение, отображаемое как сообщение в диалоговом окне.

button – числовое выражение, представляющее сумму значений, которые указывают число и тип отображаемых кнопок, тип используемого значка, основную кнопку. Значение этого аргумента по умолчанию равняется 0.

*Получить информацию о том, какие значения при нажатии какой кнопки возвращаются из окна сообщения, можно при помощи справки по функции MsgBox().*

Таблица 1

Значения аргумента, определяющие отображаемые кнопки

Константа	Значение	Отображаемые кнопки
VbOKOnly	0	ОК
VbOKCancel	1	ОК, Отмена
VbAbortRetryIgnore	2	Стоп, Повтор, Пропустить
VbYesNoCancel	3	Да, Нет, Отмена
VbYesNo	4	Да, Нет
VbRetryCancel	5	Повтор, Отмена

Таблица 2

Значения аргумента, определяющие отображаемые значки:

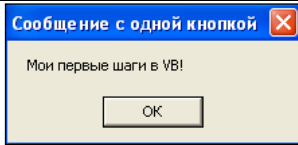
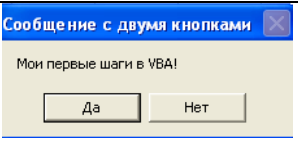
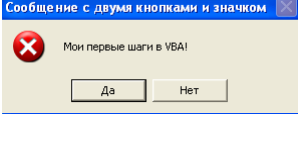
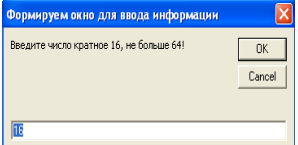
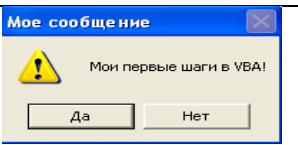
Константа	Значение	Значок сообщения
VbCritical	16	
VbQuestion	32	
VbExclamation	48	
VbInformation	64	

title – строковое выражение, отображаемое в строке заголовка диалогового окна. Если этот аргумент опущен, в строку помещается имя приложения.

Этот оператор выводит на экран диалоговое окно, содержащее сообщение, устанавливает режим ожидания нажатия пользователем кнопки, а затем возвращает в программу.

## Пример

В результате выполнения приведенных ниже строк программы на экране будут появляться окна. В таблице отражены окна, соответствующие команде вывода.

MsgBox “Мои первые шаги в VB!”, , “Сообщение с одной кнопкой”	
MsgBox “Мои первые шаги в VBA!”, vbYesNo, “Сообщение с двумя кнопками”	
MsgBox “Мои первые шаги в VBA!”, vbYesNo + vbCritical, “Сообщение с двумя кнопками и значком”	
x = InputBox (“Введите число, кратное 16, не больше 64!”, ”Формируем окно для ввода информации ”, 16)	
MsgBox “Мои первые шаги в VBA!”, vbYesNo + 48, “Мое сообщение”	

## Функция InputBox

Функция InputBox выводит на экран диалоговое окно, содержащее сообщение и поле ввода, устанавливает режим ожидания ввода текста пользователем или нажатия кнопки. Затем возвращает введенному значению тип String (текстовый).

Синтаксис:

InputBox (Prompt, [, title] [, default])

Аргументы:

Prompt – текст в диалоговом окне;

**title** – заголовок диалогового окна. Если этот аргумент не указан, диалоговое окно называется по имени приложения;

**default** – значение по умолчанию в поле ввода. Если этот аргумент опущен, поле ввода изображается пустым.

Пример. Введите команду  $A = \text{InputBox}(\text{"Введите A"}, 2)$

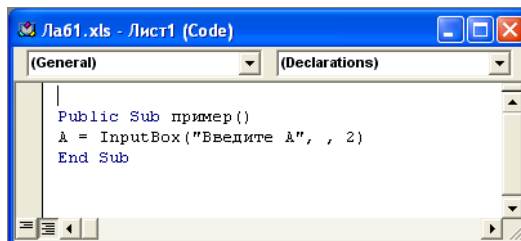


Рисунок 1. Процедура Пример

Запустите на выполнение. На экране появится диалоговое окно, где в поле ввода по умолчанию будет число 2 (Рисунок 2).

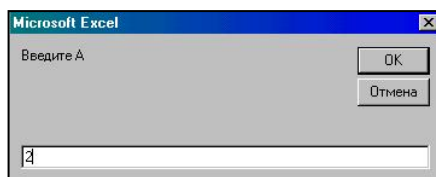


Рисунок 2. Стандартное окно ввода

В программе можно использовать данные, расположенные в ячейках рабочего листа MS Excel. Например,  $A = \text{Cells}(1, 3)$ . После выполнения этого оператора переменной  $A$  присвоится значение, которое хранится в ячейке, находящейся в первой строке и в третьем столбце  $C$ , т.е. в ячейке  $C1$  электронной таблицы.

#### 1.4 Условный оператор IF

Для реализации разветвляющегося вычислительного процесса в VBA используется оператор **If...Then...Else**, который представляет собой простейшую форму проверки условий.

Синтаксис

**if условие then оператор\_1 else оператор\_2**

Оператор\_1 выполняется, если условие истинно, в противном случае выполняется оператор\_2, при этом оператор if...then...else записывается в одну строку.

Условие – это выражение логического типа. результат выражения всегда имеет булевский тип. выражение может быть простым и сложным. При записи простых условий могут использоваться все возможные операции отношения, указанные в табл. 3.2.

Для записи условий могут быть использованы знаки сравнений, представленные в табл. 3.

Таблица 3

Знаки сравнения

Операция	Название	Пример выражения
=	Равно	A = B
<>	Не равно	A <> B
>	Больше	A > B
<	Меньше	A < B
>=	Больше или равно	A >= B
<=	Меньше или равно	A <= B

Сложные условия образуются из простых путем применения логических операций и круглых скобок. Например, A > 10 And A < 20 или (B > 4 Or B < 2) And A > 5

В таблице указаны логические операции, предназначенные для составления сложных условий.

Таблица 4

Логические операции

Операция	Название	Пример выражения
Not	Логическое отрицание	Not A
And	Логическое И	A And B
Or	Логическое ИЛИ	A Or B

В условном операторе допустимо использование блока операторов вместо любого из операторов. В этом случае условный оператор имеет вид:

**if условие then**

```

        блок_операторов_1
    else
        блок_операторов_2
    end if

```

В условном операторе может проверяться несколько условий. В этом случае условный оператор имеет вид:

```

if условие_1 then
    блок_операторов_1
elseif условие_2 then
    блок_операторов_2
else
    ....
end if

```

### 1.5 Оператор выбора Select Case

Оператор Select Case удобно использовать, когда в зависимости от значения некоторого выражения, имеющего конечное множество допустимых значений, необходимо выполнить разные действия. Он также относится к условным операторам, но имеет другой вид:

```

select case проверяемое_выражение
    case значения_1
        операторы_1
    case значения_2
        операторы_2
    ...
    case значения_n
        операторы_n
    [case else
        иначе_операторы]
end select

```

Проверяемое\_выражение может иметь любой скалярный тип, кроме вещественного. значения состоят из произвольного количества значений или диапазонов, отделенных друг от друга запятыми. Тип значений должен совпадать с типом проверяемого\_выражения. Сначала вычисляется проверяемое\_выражение. Если его значение совпадает с одним из значений значения\_i, то выполняются операторы\_i и управление передается оператору, стоящему после end select. Если его значение не совпадает ни с одним из



значений значения\_i, то выполняются иначе\_операторы и управление передается оператору, стоящему после end select.

Рассмотрим пример начисления комиссионных на основе оператора выбора Select Case. В этом примере размер комиссионных зависит от объема проданной продукции по правилу, приведенному в таблице.

Таблица 5

Объем продаж, тыс. руб.	Комиссионные, %
От 0 до 9999	8
От 10000 до 39999	10
40000 и более	14

```
Option Explicit
Public Sub PR()
Dim opr As Double
Dim prem As Double
opr = Val(InputBox("Введите объем продаж"))
Select Case opr
Case 0 To 9999
    prem = 0.08 * opr
Case 10000 To 39999
    prem = 0.1 * opr
Case Is >= 40000
    prem = 0.14 * opr
End Select
MsgBox (" Комиссионные=" & prem)
End Sub
```

## 1.6 Операторы цикла

Для реализации циклического вычислительного процесса, т. е. многократного выполнения одного или нескольких операторов, служит оператор цикла **For...Next**, который имеет следующий синтаксис:

```
for счетчик = нач_значение to кон_значение step шаг
    блок_операторов
[exit for]
    блок_операторов
next счетчик
```

Цикл For...Next перебирает значения переменной счетчик, которая является параметром цикла, от начального до конечного значения с указанным шагом

изменения. При этом обеспечивается выполнение блока операторов тела цикла при каждом новом значении счетчика. Если Step шаг в конструкции отсутствует, то по умолчанию считается, что шаг равен 1. По оператору Exit For можно выйти из оператора цикла до того, как счетчик достигнет последнего значения.

Для перебора объектов из группы подобных объектов, например, ячеек из диапазона или элементов массива, удобно использовать оператор цикла For...Each...Next.

```
for each элемент in группа  
    блок_операторов  
[exit for]  
    блок_операторов  
next элемент
```

В VBA для организации циклов с неизвестным заранее числом повторений используются и другие операторы цикла:

- циклы с предусловием – Do While ... Loop, Do Until ... Loop;
- циклы с постусловием – Do ... Loop While, Do ... Loop Until.

Ниже приведен синтаксис этих операторов цикла:

*' Цикл с предусловием Do While ... Loop*

```
do while условие  
    блок_операторов  
[exit do]  
    блок_операторов  
loop
```

*' цикл с предусловием do until ... loop*

```
do until условие  
    блок_операторов  
[exit do]  
    блок_операторов  
loop
```

*' цикл с постусловием do ... loop while*

```
do
```

```

        блок_операторов
    [exit do]
        блок_операторов
loop while условие

```

'цикл с постусловием **do ... loop until**

```

do
    блок_операторов
[exit do]
    блок_операторов
loop until условие

```

Оператор **do while...loop** обеспечивает многократное повторение блока операторов до тех пор, пока *условие* соблюдается, а оператор **do until...loop** пока *условие* не соблюдается. Операторы **do...loop while**, **do...loop until** отличаются от перечисленных выше операторов тем, что сначала блок операторов выполняется по крайней мере один раз, а потом проверяется *условие*.

Для избегания заикливания в теле цикла должен быть хотя бы один оператор, который изменяет значения переменных, стоящих в условии. Оператор **exit do** обеспечивает досрочный выход из оператора цикла.

#### Задания для самостоятельного выполнения

1. Создайте новую книгу Excel и сохраните ее как Пр3.xlsm. Откройте редактор Visual Basic в Excel и создайте в этой книге новый стандартный модуль.
2. Введите в этом модуле следующий код:

```

Public Sub IfThenSub()
    Dim nResult As Integer
    nResult = MsgBox("Нажмите кнопку", vbYesNo, "Окно сообщения")
    ThisWorkbook.Worksheets(1).Range("A1").Value = _
        "Вы кликнули по кнопке: " & nResult
    ThisWorkbook.Worksheets(1).Range("A1").Columns.AutoFit
End Sub

```

3. Запустите код на выполнение. Этот код должен вставлять в ячейку A1 первого листа вашей книги текстовое значение вида *"Вы кликнули по кнопке: б"*, в зависимости от того, какая кнопка была нажата в окне сообщения.
4. Измените код этой процедуры таким образом, чтобы вместо чисел в ячейку вписывалось строковое значение нажатой кнопки (например, *"Вы нажали кнопку: Да"*). Используйте при этом синтаксическую конструкцию If...Then...Else.
5. Замените в вашей процедуре строку:  

```
nResult = MsgBox("Нажмите кнопку", vbYesNo, "Окно сообщения")
```

на:  

```
nResult = MsgBox("Нажмите кнопку", vbAbortRetryIgnore, "Окно сообщения")
```
6. Измените вашу процедуру таким образом, чтобы она вставляла в ячейку A1 значения "Прервать", "Повторить" или "Пропустить", в зависимости от того, какая кнопка в окне сообщения была нажата. Используйте при этом синтаксическую конструкцию Select Case.
7. Используя оператор Select Case написать программу. Работа дорожного светофора для водителей запрограммирована следующим образом. Начиная с начала каждого часа, в течение трех минут горит зеленый свет, затем в течение одной минуты – желтый, в течение двух минут – красный, в течение трех минут опять зеленый и т.д. Дано вещественное число x, означающее время в минутах, прошедшее с начала очередного часа. Определить, сигнал какого цвета горит для водителей в этот момент.
8. Написать программу вычисления суммы двух наименьших из трех заданных положительных целых чисел x, y, z.

## Практическая работа №4

### ОБЪЕКТНАЯ МОДЕЛЬ EXCEL

Цель работы. Изучить понятие объектов Excel, их методы и свойства

#### 1. Теоретические сведения. Выполнение работы

##### 1.1 Основные сведения об объектах, методах и свойствах EXCEL

При создании приложения в VBA в основном происходит работа с объектами. Можно использовать объекты, предоставляемые VBA: элементы управления, формы и объекты доступа к данным. Можно также управлять объектами других приложений из приложения VBA. Можно даже создавать свои собственные объекты и определять для них дополнительные свойства и методы.

Объект — это комбинация кода и данных, которую можно рассматривать как одно целое. Объект может быть частью приложения, как элемент управления или форма. Целое приложение также может быть объектом. В таблице 1 приведены примеры типов объектов, которые можно использовать в VBA:

Таблица 1

Примеры типов объектов

Объект	Описание
Кнопка управления	Элементы управления на форме, например кнопки управления и рамки, являются объектами
Форма	Каждая форма в проекте VBA является отдельным объектом
База данных	Базы данных являются объектами и содержат другие объекты, например, поля и индексы
Диаграмма	Диаграмма в Microsoft Excel является объектом

Таким образом, объект – это программный элемент, который имеет свое отображение на экране, содержит некоторые переменные, определяющие его свойства, и некоторые методы для управления объектом. Примеры встроенных объектов в MS Excel:

- Range(“Адрес”) - диапазон ячеек (может включать только одну ячейку);

- Cells(i, j) - ячейка, находящаяся на пересечении i-й строки и j-го столбца рабочего листа MS Excel (i и j – целые числа);
- Rows(№ строки) - строка с заданным номером;
- Columns(№ столбца) - столбец с заданным номером;
- Sheets("Имя") - лист с указанным именем;
- Sheets(№ листа) - лист с указанным номером;
- WorkSheet - рабочий лист.

Объекты VBA поддерживают свойства, методы и события. В VBA данные объекта (установки или атрибуты) называются свойствами, тогда как процедуры, которые оперируют с объектом, называются его методами. Событие — это действие, распознаваемое объектом, например, щелчок кнопкой мыши или нажатие клавиши клавиатуры, и программист может написать код, реагирующий на это событие.

Можно изменять характеристики объекта, меняя его свойства. Установка значений свойств – это один из способов управления объектами. Синтаксис установки значения свойства объекта, следующий:

Объект.Свойство = Выражение

Основным свойством объектов Cells и Range, является Value(*значение*), которое, допускается не указывать.

Например, в диапазон ячеек A5:A10 заносится значение 0:

Range("A5:A10").Value = 0

Синтаксис чтения свойств объекта:

Переменная = Объект.Свойство

Например, переменной Y присваивается значение из ячейки B1 текущего рабочего листа:

Y = Cells(3, 5).Value

или

Y = Range("E3").Value.

Кроме свойств объекты обладают методами.

**Методы** — это такая же часть объектов, как и свойства. Например, показать форму на экране или убрать её можно с помощью методов Show и Hide. В целом, методы — это действия, которые можно выполнить, тогда как свойства — это атрибуты, которые устанавливаются или восстанавливаются.

Синтаксис применения методов к объекту: Объект. Метод

Например:

Sheets(3).Activate – активировать Лист 3;  
Sheets("Диаграмма").Delete – удалить лист "Диаграмма";  
Range("A1:A15").Clear – очистить диапазон ячеек A1:A15;  
Range("A1:B30").Select – выделить диапазон ячеек A1:B30.

В MS Excel имеются объекты, которые содержат другие объекты.

Например, рабочая книга содержит рабочие листы, рабочий лист содержит диапазон ячеек и т.д. Объектом самого высокого уровня является Application (приложение).

Например: Application.Quit - завершение работы с Excel.

Отметим, что точка после имени объекта может использоваться для перехода от одного объекта к другому.

Например, следующее выражение очищает пятую строку рабочего листа Январь в рабочей книге Итог:

Application.Workbooks("Итог").Worksheets("Январь").Rows(5).Delete

Программируя для Microsoft Excel, преследуют такие цели:

- Автоматизация вычислений.
- Автоматизация ввода и обработки информации.
- Работа с базами данных - вывод, ввод, анализ, визуализация информации.
- Анализ финансовой и другой информации.
- Создание систем для организации автоматизированного ввода данных
- Математическое моделирование.

Для вызова MS Excel из другого приложения, необходимо создать объект Excel.Application (при этом при помощи меню Tools-References добавить ссылку на библиотеку Microsoft Excel 16.0 Object Library). Создание этого объекта может выглядеть так:

```
Dim oExcel As New Excel.Application  
oExcel.Workbooks.Add  
oExcel.Visible = True
```

Если работать из уже запущенного Excel, создавать объект Application не потребуется. Он будет доступен всегда. Если обращаться к какому-либо свойству без указания вышестоящего объекта, то редактор Visual Basic в Excel будет считать, что обращение идет к свойству объекта Application. Поэтому эти две строки кода в Excel равнозначны:

```
Application.Workbooks.Add  
и  
Workbooks.Add
```

Чтобы в окне редактора кода для форм появился объект Application, необходимо в разделе Declarations кода формы объявить объект Application с ключевым словом WithEvents: Public WithEvents App As Excel.Application.

В модели объектов Excel имеются более 100 объектов и семейств. Однако, практически использование только небольшого количества объектов достаточно, чтобы выполнить большинство всех возможных действий. Наиболее часто используемыми объектами являются объекты: Application, Workbook (Workbooks), Worksheet (Worksheets) и Range.

Главным в иерархии объектов Excel является объект Application (Приложение), которое представляет само приложение Excel. Объект Application.Excel имеет свойство - участник (встроенный объект) Workbooks, возвращающее Workbooks – коллекцию всех открытых книг. Каждый элемент коллекции – рабочая книга - является объектом класса Workbook.

Свойствами объекта Application являются:

- ActiveWorkbook – активная рабочая книга
- ActiveWindow – активное окно



Например: MsgBox "Книга1"ActiveWorkbook.Name

Методы объекта Application:

- Quit – завершает работу с Excel;
- Undo – отменяет последнее выполненное действие

Объект Workbook (рабочая книга) – объект этого класса определяет состояние рабочей книги. Например, не является ли она доступной только для чтения, или какой из листов рабочей книги активен в настоящий момент. К этому классу принадлежит объект ActiveWorkbook (активная в настоящий момент рабочая книга).

Объект Workbook, представляющий одну рабочую книгу, имеет свойство Sheets. При обращении к этому свойству возвращается объект Sheets – коллекция листов данной книги. Каждый элемент коллекции лист имеет свой тип и поэтому является объектом класса Worksheet (рабочий лист) или Chart (диаграмма).

Свойства объект Workbook:

- ActiveSheet – активный рабочий лист
- Name – имя рабочей книги

Методы объекта Workbook

- Activate –активизирует рабочую книгу
- Close – закрывает рабочую книгу
- Save – сохраняет рабочую книгу

Объект Worksheet (рабочий лист) этого класса используется при копировании и удалении рабочих листов, их скрытии и показе, проведении вычислений для формул рабочего листа. К этому классу принадлежит объект ActiveWorksheet (активный в настоящий момент рабочий лист).

Свойства объекта Worksheet:

- Name – имя рабочего листа
- Previous – предыдущий рабочий лист
- Visible – режим видимости рабочего листа (скрыт или показан)

### Методы объекта Worksheet:

- **Activate** – активизирует рабочий лист
- **Calculate** – заново вычисляет значения в ячейках рабочего листа
- **Delete** – удаляет рабочий лист
- **Protect** – защищает рабочий лист

Объект **Worksheet**, представляющий один рабочий лист, имеет свойство **Cells**, возвращающее объект **Range**. Он может задавать как все ячейки рабочего листа, так и любую ее часть.

**Range** (диапазон). Объект этого класса позволяет изменять свойства интервала ячеек (например, шрифт), проверять или изменять содержимое ячеек, вырезать или копировать интервал и так далее.

К основным свойствам объекта **Range** относятся:

- **Cells, Columns, Rows** – возвращают коллекции ячеек, столбцов или строк, входящие в объект **Range**;
- **Column, Row** – возвращают соответственно номер первого столбца или первой строки в области объекта **Range** позволяет прочесть или задать формулу;
- **Value** – значение указанной ячейки. Если она пуста, то возвращается значение **Empty**, что можно проверить, вызвав функцию **IsEmpty**;
- **Font** – используемый в интервале шрифт;
- **Formula** – формула интервала;
- **Name** – имя интервала.

Синтаксис установки значения объекта:

Объект.Свойство = значение

Здесь значение может быть константой или формулой, возвращающей постоянное значение, и принадлежать к одному из трех типов:

1. Числовое значение. Например, для установки размера шрифта:  
`ActiveCell.Font.Size = 14`
2. Строка символов. Например: `ActiveCell.Font.Name = "Courier New Cyr"`

### 3. Логическое значение: ActiveCell.Font.Italic = True

К основным методам объекта Range относятся:

- Clear – полностью очищает интервал (в том числе и форматирование)
- ClearContents – очищает содержимое ячеек интервала
- ClearFormats – очищает форматирование ячеек интервала
- Copy – копирует интервал в буфер обмена
- Offset – возвращает интервал с указанным смещением относительно первоначального интервала
- Paste – вставляет содержимое буфера обмена в интервал
- Select – выделяет интервал

Синтаксис вызова метода объекта: Объект.Метод[аргументы].

Например, Range("A1:B2").Select

### 2.Задания для самостоятельного выполнения

1. Создайте новую книгу Excel и сохраните ее с именем Пр4.xlsm. Введите в ячейки A1, A2 и A3 этой книги любые значения.
2. Откройте редактор Visual Basic в Excel и создайте в этой книге новый стандартный модуль.
3. При помощи меню Tools - References добавьте в ваш проект ссылку на библиотеку Microsoft Word 14.0 Object Library.
4. Введите в созданном вами стандартном модуле следующий код:

```
Public Sub FromExcelToWord()  
    MsgBox Range("A1").Text  
    MsgBox Range("A2").Text  
    MsgBox Range("A3").Text  
  
    Dim oWord As Word.Application  
    Dim oDoc As Word.Document  
    Set oWord = CreateObject("Word.Application")  
    oWord.Visible = True  
    Set oDoc = oWord.Documents.Add()  
    oDoc.Activate  
    oWord.Selection.TypeText "Текст из программы"
```

End Sub

Этот код должен выводить в окна сообщений значения ячеек A1, A2 и A3, а затем «Текст из программы» .

5. Проверьте работу кода.

6. Измените код этой процедуры таким образом, чтобы вместо строки "Текст из программы" выводились значения ячеек A1, A2 и A3.

### Контрольные вопросы

1. Какие цели преследуют программируя для MS Excel?
2. Что такое объект? Наиболее часто используемые объекты Excel.
3. Как можно управлять объектами?
4. Перечислите свойства и методы объекта Range.
5. Назовите главный объект в иерархии объектов Excel

## **Практическая работа № 5**

### **МАКРОСЫ**

Цель работы. Научиться работать с макросами: создавать, редактировать, сохранять.

#### **Теоретические сведения. Выполнение работы**

##### **1. Виды макросов**

**Макрос** – это набор инструкций, которые программа выполняет по команде запуска. Инструкции могут соответствовать простым нажатиям клавиш или сложным наборам команд меню. Макросы обычно создаются при необходимости в регулярном выполнении операций, которые повторяются в одном и том же порядке.

Выделяют три основные разновидности макросов:

- командные макросы, которые состоят из операторов, соответствующих тем или иным командам или параграфам диалоговых окон, записанных в определенном порядке. Основным предназначением таких макросов является изменение внешнего вида окна или элемента;
- пользовательские функции. Эти макросы работают аналогичным образом, как встроенные функции Excel. Они не изменяют среды приложения в отличие от командных макросов;
- макрофункции, которые представляют собой сочетание командных макросов и пользовательских функций.

##### **2 Способы создания макросов**

Создать макросы можно следующими способами:

- запустить встроенное средство автоматической записи макросов - макрорекордер и выполнять последовательность действий, которые необходимо включить в макрос. Затем остановить макрорекордер. Обычно так создаются простые макросы;

- ввести коды макроса в окне редактора, используя операторы VBA.

Обычно при создании сложных макросов объединяют два этих способа в один.

### **3 Места хранения макросов**

Для Excel возможно сохранять макросы в текущем документе, в новой рабочей книге, чтобы иметь возможность использовать его из других рабочих книг или в личной книге макросов.

Личная книга макросов (Personal Macro Workbook) - это книга, которая автоматически загружается при запуске Excel. Обычно она невидима и макросы из нее можно вызывать из любой рабочей книги.

Макросы, сохраненные в текущем документе, называются *локальными*. Макросы, сохраненные в личной книге макросов, называются *глобальными*.

Макросы можно запускать, находясь в той книге, где они были сохранены, из другой рабочей книги, если она открыта или из личной книги макросов. При необходимости использовать макросы во многих рабочих книгах, их лучше сохранять

### **4 Создание и сохранение макросов**

Прежде чем создавать макрос, необходимо четко представлять последовательность действий, которые он должен выполнять. Необходимо настроить приложение таким образом, чтобы можно было выполнить команды, подлежащие записи.

Чтобы создать макрос, первым способом, необходимо:

1. Открыть окно Запись макросов: меню Разработчик – вкладка Код - Запись макроса.
2. Ввести имя макроса и краткое его описание в соответствующие поля диалогового окна Запись макроса.

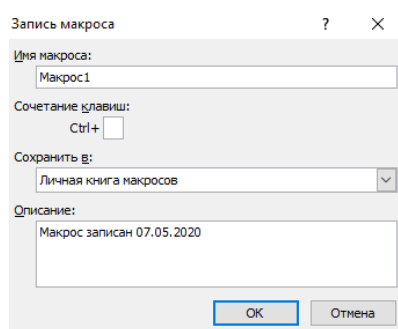



Рисунок 1. Окно **Запись макроса**

3. Укажите в раскрывающемся списке место, где будет сохранен макрос (Личная книга макросов, Новая книга, Эта книга)
4. Можно задать комбинацию клавиш для вызова макроса
5. Щелкните на кнопке Ок, запись макроса начнется
6. Выполните действия, которые следует включить в макрос

*Примечание. При записи фиксируются лишь выполняемые операции, но не затраченное время, поэтому торопиться нет необходимости. Старайтесь действовать аккуратно, поскольку, как ошибка, так и операции по ее исправлению будут воспроизведены при запуске макроса.*

7. После выполнения всех шагов по вводу макроса щелкните на кнопке «Остановить запись» .

## 5 Запуск макроса

Для запуска макроса необходимо:

1. На вкладке Код меню Разработчик выбрать пункт Макросы.
2. Выберите в списке нужный макрос
3. Щелкните на кнопке Выполнить

Пример. Очень часто пользователь несколько раз в день передает распоряжения в различные инстанции. Каждое распоряжение должно заканчиваться, например, строками вида: 

Ответственный исп. Иванова С.А.
тел. 8(495)-234-67-34

. Необходимо создать при помощи макрорекордера макрос “ОтвИсп”, который бы автоматически подставлял информацию об ответственном исполнителе в активную ячейку, а информацию о телефоне - в ячейку ниже (вместо

"Иванова С.А." подставьте вашу фамилию). Макрос должен быть доступен для всех создаваемых документов. Макрос должен запускаться по нажатию клавиш Ctrl+Shift+M.

1. Откройте Excel и выделите на листе любую пустую ячейку.
2. Перейдите на вкладку Вид и в списке кнопки Макросы выберите пункт Относительные ссылки.
3. Запустите запись макроса. Откроется окно диалога Запись макроса.
4. В окне Запись макроса в поле Имя макроса введите имя: Подписано.
5. Установите указатель в поле Сочетание клавиш и нажмите Shift+M (значение в итоге должно выглядеть как Ctrl + Shift + M).
6. В поле Сохранить в выберите "Эта книга" (если макрос понадобится вам не только в одном документе укажите опцию "Личная книга макросов")
7. Нажмите на кнопку ОК.
8. Введите в текущую ячейку на листе Excel текст "Ответственный исполнитель ваши\_ФИО", например, "Ответственный исполнитель Иванова С.А.". Перейдите на ячейку ниже и введите текст " тел. 8(495)-234-67-34".
9. На вкладке Вид в списке кнопки Макросы выберите пункт Остановить запись.
10. Перейдите на другой лист и на вкладке Вид в списке кнопки Макросы выберите пункт Макросы. В открывшемся окне укажите требуемый макрос и нажмите на кнопку Выполнить. В текущую ячейку и ячейку ниже будет вставлен записанный вами текст.
11. В других ячейках проверьте работу комбинации Ctrl + Shift + M.
12. Настройте запуск макроса с помощью кнопки на панели быстрого запуска.
13. Сохраняя книгу Microsoft Excel, содержащую макросы, убедитесь в том, что сохраняете ее в формате "Книга Excel с поддержкой макросов", то есть итоговый файл имеет расширение .xlsm.



## **6 Редактирование макроса**

Изменить макрос Подписано так, чтобы вместо инициалов выводилось имя полностью, отображался бы новый номер телефона 777-54-55

1. Откройте книгу Excel из задания 1.2.
2. Перейдите на вкладку Вид и в списке кнопки Макросы выберите пункт Макросы. Откроется окно Макрос.
3. В списке доступных макросов укажите макрос Подписано и нажмите кнопку Изменить. Откроется редактор VBA, который позволяет изменять макросы и создавать собственные программы.
4. Найдите в коде строку, в которой активной ячейке присваивается символьная строка “Фамилия И.О.” и измените инициалы на полное имя.
5. Измените номер телефона на новый: 777-54-55.
6. Закройте редактор, выбрав File - Close and Return to Microsoft Excel.
7. Запустите макрос и проверьте его работу.

### **Задания для самостоятельного выполнения**

1. Создайте книгу Excel с названием Смета.xlsx.
2. Создайте в специальной книге Personal.xlsb макрос TestOpenWorkbook(), который бы проверял, открыта или нет книга с именем Смета.xlsx. Если книга открыта, этот макрос должен вывести в стандартном окне сообщения ее имя. Если книга закрыта, этот макрос должен ее открыть и также вывести ее имя в стандартном окне сообщения.
3. Создайте в специальной книге Personal.xlsb макрос Calendar(), который бы:
  - создавал пустую рабочую книгу;
  - переименовывал бы в ней лист "Лист1" в "Календарь";
  - удалял бы все остальные листы.
4. При этом сообщения с просьбой подтвердить удаление листа выводиться не должны.

5. Измените созданный вами макрос Calendar() таким образом, чтобы он дополнительно:
- внес в ячейки с A3 по A367 включительно даты с 01 января 2020 г. по 31 декабря 2020 г.
  - все ячейки, для которых даты попадают на субботы и воскресенье, были бы помечены красным цветом.
6. Создать макрос, который объединяет выделенные ячейки и форматирует текст в этой объединенной ячейке по центру по горизонтали и по вертикали.
7. Создать макрос, который защищает лист с паролем.
8. Создать макрос, который снимает защиту листа с паролем.
9. Создать макрос, производящий автозаполнение ячеек строки названиями месяцев года и устанавливает в этих ячейках шрифт полужирный курсив

### **Контрольные вопросы**

1. Что такое макрос в языке VBA? Виды макросов.
2. Где сохраняется код макроса VBA?
3. Как запустить макрос на выполнение?
4. Способы создания макросов.

## Практическая работа №6

### РАЗРАБОТКА ПОЛЬЗОВАТЕЛЬСКИХ ФОРМ

Цель работы. Получить навыки разработки графического интерфейса приложений.

#### 1. Теоретические сведения. Выполнение работы

##### 1.1 Пользовательские формы, свойства и методы

Для предоставления пользователю графического интерфейса используются формы VBA. Пользовательские диалоговые окна создаются на основе технологии пользовательских форм, к которым можно получить доступ из редактора Visual Basic VBE.

Пользовательская форма – это диалоговое окно, в котором располагаются различные элементы управления. Стандартная последовательность действий при создании пользовательской формы:

1. Вставьте новую форму UserForm в проект VBA рабочей книги.
2. Добавьте элементы управления в форму UserForm.
3. Настройте свойства добавленных элементов управления.
4. Создайте процедуры обработки событий для элементов управления. Эти процедуры добавляются в модуль кода UserForm и выполняются при возникновении различных событий, например, при щелчке на кнопке.
5. Разработайте процедуру, которая отображает форму UserForm. Эта процедура находится в модуле VBA, а не в модуле кода для формы UserForm.
6. Определите способ вызова процедуры, созданной в п. 5. Можно поместить кнопку на рабочий лист, команду ленты и т.д.

Форма\Form представляет собой окно, на котором размещаются управляющие элементы. VBA обладает встроенным набором элементов управления. К ним относятся:

- командные кнопки;
- метки или надпись;

- текстовые окна и поля;
- простые списки и др.

Для создания формы достаточно в редакторе Visual Basic щелкнуть правой кнопкой мыши на проекте (т. е. на имени документа) в окне Project Explorer и в контекстном меню выбрать Insert - UserForm. Откроется окно дизайнера форм (Form designer), в котором будет представлено пустое серое окно формы и рядом Toolbox — панель с набором элементов управления.

Если у вас включен показ окна свойств Properties (он включается по клавише <F4>), то в этом окне будут представлены свойства формы. Переход к редактору кода для этой формы (по умолчанию открывается событие Click) выполняется по клавише <F7>, возврат обратно в окно дизайнера форм - по <Shift>+<F7>.

Очень удобно, что для форм и элементов управления можно настраивать свойства при помощи графического интерфейса окна свойств - резко уменьшается количество программного кода, которое нужно писать вручную.

#### Самые важные свойства форм:

- Name - это свойство определяет имя формы. Это имя не видно пользователю программы, оно используется только программистом в коде для этой формы (и в окнах редактора Visual Basic). После создания формы ее имя, предлагаемое по умолчанию (UserForm1), рекомендуется заменить на какое-то значимое, чтобы было проще ориентироваться в программе (это относится ко всем элементам управления).
- Caption - определяет заголовок формы (по умолчанию совпадает с именем формы). Рекомендуется ввести строку, которая будет напоминать пользователю о назначении формы (например, "Выбор типа отчета").
- Enabled - если это свойство установлено в False, пользователь не сможет работать с формой. Используется для временного отключения формы,

например, пока пользователь не обеспечит какие-то условия для ее работы.

- **ShowModal** - если свойство установлено в True (по умолчанию), то пользователь не может перейти к другим формам или вернуться в документ, пока не закроет эту форму (так называемый "модальный" режим работы).

В процессе редактирования формы (из окна редактора Visual Basic) ее можно запускать по нажатию клавиши <F5>. После того, как форма будет готова, вы должны обеспечить ее запуск в документе.

#### Самые важные методы форм:

- Для запуска формы существует метод **Show(): UserForm1.Show**

Если форма уже была загружена в память, она просто станет видимой, если нет - то будет автоматически загружена (произойдет событие Load).

#### Способы вызова этого метода:

- из обычного макроса, привязанного к кнопке или клавиатурной комбинации;
- из автозапускаемого макроса (макроса с названием AutoExec для Word);
- из кода для элемента управления, расположенного в самом документе (например, CommandButton) или на другой форме (для перехода между формами);
- поместить его в обработчик события Open для документа Word или книги Excel, чтобы форма открывалась автоматически при открытии документа.

После того, как пользователь введет или выберет нужные данные на форме и нажмет требуемую кнопку, форму необходимо убрать. Для этого можно воспользоваться двумя способами:

- спрятать форму (использовать метод **Hide()**), например:

UserForm1.Hide

Форма будет убрана с экрана, но останется в памяти. При помощи метода **Show()** можно будет опять ее вызвать в том же состоянии, в каком она была до

скрытия. Пока форма невидима, можно программно изменять ее и расположенные на ней элементы управления. Окончательно форма удалится из памяти при закрытии документа. Удалить форму из памяти можно при помощи команды `Unload: Unload UserForm1`

Самые важные события форм:

- `Initialize` - происходит при подготовке формы к открытию (появлению перед пользователем). Обычно в обработчик для этого события помещается код, связанный с открытием соединений с базой данных, настройкой элементов управления на форме, присвоением значений по умолчанию и т. п.
- `Click` (выбирается по умолчанию) и `DbClick` — реакция на одиночный и двойной щелчок мыши соответственно. Для формы эти события используются не так часто. Обычно обработчики щелчков применяются для кнопок (элементов управления `CommandButton`).
- `Error` — это событие используется при возникновении ошибки в форме, предоставляя пользователю возможность исправить сделанную им ошибку.
- `Terminate` — используется при нормальном завершении работы формы и выгрузке ее из памяти (например, по команде `Unload`). Обычно применяется для разрыва открытых соединений с базой данных, освобождения ресурсов

## 1.2 Элементы управления: общие сведения

Элемент управления - это специализированный объект, который можно размещать на формах VBA (или непосредственно в документах) и который используется для организации взаимодействия с пользователем

Элементы управления являются объектами. Поэтому, как любые объекты, они обладают свойствами, методами и событиями. Создание элементов управления на рабочем листе или в форме как правило происходит на начальном этапе конструирования приложения. Добавление элементов

управления на форму чаще всего производится из дизайнера форм при помощи панели Toolbox. Для этого необходимо выбрать элемент управления на Toolbox и перетащить его на форму или, что более удобно, выделить элемент управления в Toolbox, а затем на форме выделить ту область экрана, которую будет занимать этот элемент управления. Назначение элементов представлены в таблице.

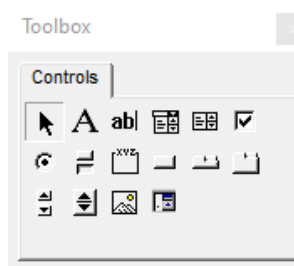


Рисунок 1. Панель инструментов

Таблица 1

Назначение кнопок панели инструментов Control Toolbox

Кнопка	Элемент управления	Назначение
	Поле(TextBox)	Добавляет в форму поле ввода
	Надпись(Label)	Добавляет
	Кнопка (CommandButton)	Добавляет в форму командную кнопку
	Список (ListBox)	Добавляет в форму поле обычного списка
	Поле со списком ComboBox	Добавляет в форму поле ввода с раскрывающимся списком
	Полоса прокрутки ScrollBar	Добавляет в форму полосу прокрутки
	Счетчик SpinButton	Добавляет в форму поле счетчика
	Переключатель OptionButton	Добавляет в форму отдельную кнопку переключателя
	Флажок CheckBox	Добавляет в форму флажок опции
	Рамка Frame	Добавляет в форму рамку (блок для других объектов)
	Выключатель ToggleButton	Добавляет в форму кнопку выключателя
	Рисунок Image	Добавляет в форму элемент для размещения изображения
	Набор вкладок TabStrip	Добавляет в форму набор вкладок
	Набор страниц MultiPage	Добавляет в форму многостраничный элемент

### 1.3 Пример разработки пользовательской формы

#### 1. Создать 3 таблицы на разных рабочих листах:

- таблица - информация о работающих (Табельный номер, Фамилия, Разряд);
- таблица – справочник по разрядам (Разряд, Оклад);
- таблица - сведения о начислении зарплаты (Табельный номер, Фамилия, Коэффициент отработанного времени, Начислено).

Ввод информации в исходные таблицы должен быть организован с использованием пользовательских диалоговых окон (форм).

2. Открыть MS Excel - Редактор VBA (лента Разработчик - Макросы)
3. В диалоговом окне Макрос ввести новое имя макроса, например, *Расчет*
4. Добавить 3 формы с названиями (используйте окно свойств каждой формы) **Главная, Сведения, Справочник** в проект при помощи команды Insert - UserForm.
5. На форме "**Главная**" разместить четыре кнопки: *Name*-cm\_1, *Caption* - "Сведения", *Name* -cm\_2, *Caption* -"Справочник", *Name*- cm\_3, *Caption* "Обработка", *Name* - cm\_4, *Caption* "Выход". Для переименования кнопок использовать окно свойств каждой из этих кнопок.
6. На форме "**Сведения**" добавить 3 поля с именами TextBox\_тн, TextBox\_фм, TextBox\_рз и с надписями: "Табельный номер", "Фамилия", "Разряд" и две кнопки: *Name* - cm\_ok, *Caption* - "ОК" *Name* - cm\_exit, *Caption* - "Выход".
7. В форму "**Справочник**" добавить 2 поля с именами TextBox\_рз и TextBox\_ок. Рядом с этими полями соответственно добавить две надписи: "Разряд", "Оклад" две кнопки: *Name* - cm\_ok, *Caption* -"ОК" *Name* -cm\_exit, *Caption* -"Выход".
8. Проверить вид каждой формы (Run\Run Sub/UserForm)
9. Назначить каждой кнопке соответствующую процедуру (VBA-код). Для этого дважды щелкнуть по каждой из вставленных в формы кнопок в



режиме конструктора и ввести в окне редактора VBA текст соответствующей процедуры:

- Кнопке cm\_1 ("Сведения") на форме "Главная" назначить процедуру:

```
Private Sub cm_1_Click()  
    Sheets(1).Activate  
    Range("A:D").Clear  
    ActiveSheet.Cells(1, 1) = "Таб.номер"  
    ActiveSheet.Cells(1, 2) = "Фамилия"  
    ActiveSheet.Cells(1, 3) = "Разряд"  
    Cells(1, 1).Activate  
    Сведения.Show  
End Sub
```

- Кнопке cm\_2 ("Справочник") на форме "Главная" назначить процедуру:

```
Private Sub cm_2_Click()  
    Sheets(2).Activate  
    Range("A:D").Clear  
    ActiveSheet.Cells(1, 1) = "Разряд"  
    ActiveSheet.Cells(1, 2) = "Оклад"  
    Cells(1, 1).Activate  
    Справочник.Show  
End Sub
```

- Кнопке cm\_3 ("Обработка") на форме "Главная" назначить процедуру:

```
Private Sub cm_3_Click()  
    Calculation  
End Sub
```

- Кнопке cm\_4 ("Выход") на форме "Главная" назначить процедуру:

```
Private Sub cm_4_Click()  
    End  
End Sub
```

- Кнопке cm\_OK ("OK") на форме "Сведения" назначить процедуру:

```
Private Sub cm_OK_Click()  
    Dim i As Integer  
    Sheets(1).Activate
```

```

ActiveCell.Offset(1, 0).Activate
i = ActiveCell.Row
' вместо предыдущих двух строк для записи данных в
' ту же книгу можно использовать оператор:
' i = Range("A1").CurrentRegion.Rows.Count+1
ActiveSheet.Cells(i, 1) = TextBox_тн
ActiveSheet.Cells(i, 2) = TextBox_фм
ActiveSheet.Cells(i, 3) = TextBox_рз
i = ActiveCell.Row + 1
End Sub

```

- Кнопке cm\_exit ("OK") на форме "Сведения" назначить процедуру:

```

Private Sub cm_exit_Click()
Hide
End Sub

```

- Кнопке cm\_OK ("OK") на форме "Справочник" назначить процедуру:

```

Private Sub cm_OK_Click()
Dim i As Integer
Sheets(2).Activate
ActiveCell.Offset(1, 0).Activate
i = ActiveCell.Row
ActiveSheet.Cells(i, 1) = TextBox_рз
ActiveSheet.Cells(i, 2) = TextBox_ок
i = ActiveCell.Row + 1
End Sub

```

- Кнопке cm\_exit ("Выход") на форме "Справочник" назначить процедуру:

```

Private Sub cm_exit_Click()
Hide
End Sub

```

10. Вставить в модуль (Модуль1) процедуру Calculation, для этого в окне проекта (Insert\Module) предварительно выделить Модуль1 и ввести текст:

```

Sub Calculation()
Sheets(3).Activate
Range("A:D").Clear
ActiveSheet.Cells(1, 1) = "Таб.номер"
ActiveSheet.Cells(1, 2) = "Фамилия"
ActiveSheet.Cells(1, 3) = "Коэффициент"

```

```

ActiveSheet.Cells(1, 4) = "Начислено"
i = 2
j = 2
Do While Sheets(1).Cells(j, 1) <> ""
ActiveSheet.Cells(i, 1) = Sheets(1).Cells(j, 1)
ActiveSheet.Cells(i, 2) = Sheets(1).Cells(j, 2)
ActiveSheet.Cells(i, 3) = InputBox("Коэффициент <=1", , "1")
k = 2
Do While Sheets(2).Cells(k, 1) <> ""
If Sheets(1).Cells(j, 3) = Sheets(2).Cells(k, 1) Then
ok = Sheets(2).Cells(k, 2)
End If
k = k + 1
Loop
ActiveSheet.Cells(i, 4) = ActiveSheet.Cells(i, 3) * ok
j = j + 1
i = i + 1
Loop
End Sub

```

11. Добавить на рабочий лист кнопку: Name - cm, Caption - "Расчет зарплаты" и назначить ей в режиме конструктора процедуру:

```

Private Sub cm_Click()
Главная.Show
End Sub

```

12. Используя кнопку "Расчет зарплаты" на рабочем листе, выполнить построенное приложение.

## 2. Задание для самостоятельного выполнения

1. Создать простое приложение-калькулятор. Даны два случайных числа, которые выводятся в соответствующие текстовые поля на форме. Пользователь с клавиатуры задает название арифметической операции, которую надо выполнить над числами: сложение, вычитание, умножение, деление
2. Выбрать любую предметную область на свое усмотрение (например, библиотека, гостиница, спортивный клуб и др.). Разработать на рабочих

листах таблицы для ввода и хранения данных. Создать главную форму приложения и формы для заполнения таблиц исходными данными и вычисляемыми.

### **3. Контрольные вопросы**

1. Что представляет собой форма пользователя?
2. Какие наиболее важные свойства формы?
3. Какой элемент создает поле ввода\вывода данных?
4. Как открыть форму в Excel?

## Практическая работа №7

### ФОРМЫ ДЛЯ РАБОТЫ С БАЗОЙ ДАННЫХ

Цель работы. Приобрести навыки формирования и программирования диалоговых форм для создания баз данных и управления ими средствами Excel и VBA.

#### Выполнение работы

##### 1. Проектирование формы. Типы элементов управления

Существуют два типа элементов управления: Элементы управления формой и Элементы управления ActiveX.

Одним из основных отличий, которые важно знать, является то, что элементы управления ActiveX отображаются как объекты, которые можно использовать в коде. ActiveX можно вставить в рабочий лист. Открыв редактор VBA, можно получить доступ к элементу управления программно. Это невозможно сделать с помощью элементов управления Form (макросы должны быть явно назначены каждому элементу управления), но элементы управления Form немного проще в использовании. Если вы просто делаете что-то простое, не имеет значения, что вы используете но для более сложных скриптов ActiveX имеет более широкие возможности. ActiveX также более настраиваемый.

С помощью VBA в Excel можно формировать табличные базы данных и пользовательский интерфейс управления ими.

➤ Создадим базу данных в виде таблицы по учету оплаты за электроэнергию. Таблица содержит поля:

- Фамилия;
- Имя;
- Адрес плательщика;
- Дата платежа;
- Тариф
- Показания счетчика (текущее и предыдущее);
- Расход электроэнергии;
- Сумма к оплате.

1. В новой рабочей книге Excel и перейдите в редактор VBA - меню Вставка - UserForm. В редакторе появится заготовка диалоговой формы. Измените свойство заголовков формы Caption UserForm1 на Учет оплаты за электроэнергию .
2. Создадим на форме две рамки (Frame). Первой рамке в свойстве Caption присвоим заголовок - Информация о клиенте, а второй рамке - Информация о платеже. В этих рамках добавим элементы управления Label и TextBox.
3. Элементам TextBox зададим имена:
  - Фамилия – *Famely*
  - Имя – *Name*
  - Адрес – *Adres*
  - Тариф – *Tariff*
  - Текущее показание счетчика – *TekPok*
  - Предыдущее показание счетчика – *PredPok*
  - Расход электроэнергии - *Rashod*
4. Разместим на форме три командные кнопки CommandButton с заголовками: Принять, Отмена, Выход, Диаграмма (Рисунок1).

В рабочей книге Лист2 переименуем на Учет, а Лист1 на Форма. На листе Форма поместим элемент ActiveX Кнопка, заголовок которой и зменим на Прием платежа , а имя кнопки на ПриемПлатежа.

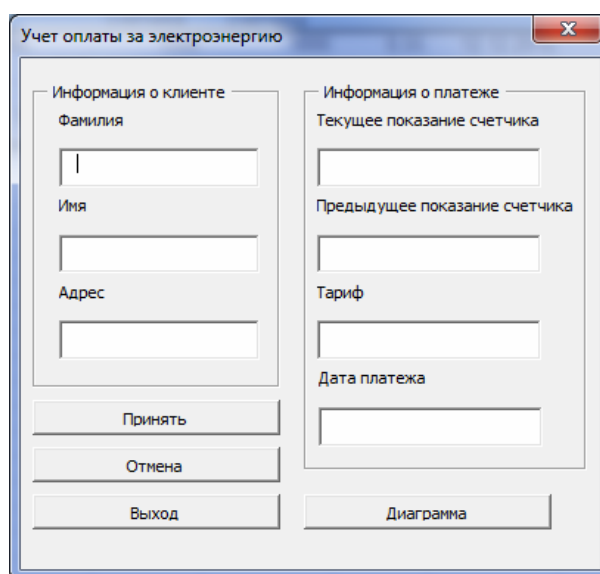


Рисунок 1 — Форма с размещенными на ней объектами интерфейса

Перейдем в редакторе на лист Форма и создадим процедуру отклика кнопки на щелчок:

```
Private Sub ПриемПлатежа_Click()  
    'Активируем процедуру для создания структуры таблицы базы данных  
    ЗаголовокРабочегоЛиста  
End Sub
```

Для создания структуры таблицы базы данных создадим процедуру СтруктураТаблицы, которая в свою очередь будет вызываться из процедуры обработки кнопки рабочего листа «Прием платежа».

Создадим новую процедуру с именем СтруктураТаблицы. Алгоритм действий данной процедуры:

- по значению ячейки *A1* проверяется, заполнена ли первая строка, которая отводится под заголовки столбцов; если заполнена, то работа процедуры завершается и передается управление в точку вызова;
- если первая строка пустая, то последовательно, в ячейки первой строки рабочего листа записываются названия полей базы данных, закрепляется первая строка и завершается работа - управление передается в точку вызова.

В окне редактирования кода введем текст этой процедуры :

```
Private Sub СтруктураТаблицы()  
    'Активируем рабочий лист  
    Application.Worksheets("Учет").Activate  
    'Проверка наличия заголовков столбцов.Если их нет, то заполняются  
    With ActiveSheet  
        If .Range("A1").Value = " Фамилия" Then  
            .Range("A2").Select  
        Else  
            'Очищаем рабочий лист  
            ActiveSheet.Cells.Clear  
            'Заполняем названия столбцов  
            Application.Worksheets("Учет").Range("A1:I1").Select  
            With Selection  
                .Value= Array("Фамилия", " Имя", " Адрес", " Текущее показание счетчика", " Предыдущее  
показание счетчика ", " Тариф", " Дата платежа ", " Расход электроэне ргии ", " Сумма")  
                .Interior.ColorIndex = 8  
                .Font.Bold = True  
            End With  
            'Добавляем комментарии  
            .Range("A1").AddComment  
            .Range("A1").Comment.Visible = False
```

```

.Range("A1").Comment.Text Text:= " Фамилия потребителя "
.Range("B1").AddComment
.Range("B1").Comment.Visible = False
.Range("B1").Comment.Text Text:= " Имя потребителя "
.Range("C1").AddComment
.Range("C1").Comment.Visible = False
.Range("C1").Comment.Text Text:= " Адрес потребителя "
.Range("D1").AddComment
.Range("D1").Comment.Visible = False
.Range("D1").Comment.Text Text := " Текущее показание счетчика "
.Range("E1").AddComment
.Range("E1").Comment.Visible = False
.Range("E1").Comment.Text Text:= " Предыдущее показание счетчика "
.Range("F1").AddComment
.Range("F1").Comment.Visible = False
.Range("F1").Comment.Text Text:= " Тариф"
.Range("G1").AddComment
.Range("G1").Comment.Visible = False
.Range("G1").Comment.Text Text:= " Дата платежа "
.Range("H1").AddComment
.Range("H1").Comment.Visible = False
.Range("H1").Comment.Text Text:= " Расход электроэнергии "
.Range("I1").AddComment
.Range("I1").Comment.Visible = False
.Range("I1").Comment.Text Text:= " Сумма"
End If
End With
'Форматирование табличных ячеек
Worksheets("База").Range("A:I").Select
With Selection
.HorizontalAlignment = xlCenter
.VerticalAlignment = xlCenter
.WrapText = True
.Orientation = 0
.AddIndent = False
.ShrinkToFit = False
End With
'Вызвать на экран форму
UserForm1.Show
End Sub

```

Составим программные коды процедур для кнопок Принять, Отмена и Выход. Для этого, щелкнем по UserForm1 в окне Project-VBAProject. В открывшейся форме два раза щелкнем по кнопке «Принять», тем самым перейдя в процедуру обработки события Click. Запишем следующий код :

```

Private Sub CommandButton1_Click()
'Объявление переменных
Dim famely, name, adres As String
Dim tariff, prpok, tekpok, rashod, summa As Single
Dim nomer As Integer
Dim data As Date

```



```

'Вычисление номера первой пустой строки в таблице
nomer = Application.CountA(ActiveSheet.Columns(1)) + 1
With UserForm1
'Проверяем, заполнена ли фамилия
If .txtFamil.Text = "" Then
MsgBox " Введите фамилию потребителя ", vbExclamation
Exit Sub 'Досрочный выход из процедуры
End If
'Проверяем, заполнено ли имя
If .txtName.Text = "" Then
MsgBox " Введите имя потребителя ", vbExclamation
Exit Sub
End If
'Проверяем, заполнен ли адрес
If .TxtAdres.Text = "" Then
MsgBox " Введите адрес потребителя ", vbExclamation
Exit Sub
End If
'Задаем значения переменным в элементах TextBox
family = .txtFamil.Text
name = .txtName.Text
adres = .TxtAdres.Text
'Проверяем наличие текущего показания счетчика
If IsNumeric(.txttekpok.Text) = False Then
MsgBox " Некорректное показание счетчика ",
vbExclamation
Exit Sub
End If
tekpok = CSng(.txttekpok.Text)
'Проверяем наличие предыдущего показания счетчика
If IsNumeric(.txtprpok.Text) = False Then
MsgBox " Некорректное показание счетчика ",
vbExclamation
Exit Sub
End If
prpok = CSng(.txtprpok.Text)
'Проверяем наличие тарифа
If IsNumeric(.txttarif.Text) = False Then
MsgBox " Некорректный тариф ", vbExclamation
Exit Sub
End If
tarif = CSng(.txttarif.Text)
If IsDate(.txtdata) = False Then
MsgBox " Введите корректно дату ", vbExclamation
Exit Sub
End If
data = .txtdata
If Val(.txtprpok.Text) > Val(.txttekpok.Text) Then
MsgBox "Текущие показания счетчика меньше предыдущего", vbExclamation
Exit Sub
End If
End With
'Вычисляем расход электроэнергии и сумму оплаты
rashod = tekpok - prpok

```

```

summa = rashod * tarif
'Запись данных в ячейки рабочего листа
With ActiveSheet
.Cells(nomer, 1).Value = famely
.Cells(nomer, 2).Value = name
.Cells(nomer, 3).Value = adres
.Cells(nomer, 4).Value = tekpok
.Cells(nomer, 5).Value = prpok
.Cells(nomer, 6).Value = tariff
.Cells(nomer, 7).Value = data
.Cells(nomer, 8).Value = rashod
.Cells(nomer, 9).Value = summa
End With
ClearForm
End Sub

```

Процедура ClearForm *очищает форму после внесения новой записи в базу данных:*

```

Private Sub ClearForm()
Unload UserForm1
UserForm1.Show
End Sub

```

Напишем процедуру обработки события Click кнопки Отмена:

```

Private Sub CommandButton2_Click()
Dim nomer As Integer
'Вычисляем номер последней строки
nomer = Application.CountA(ActiveSheet.Columns(1))
'Очищаем содержимое
With ActiveSheet
If nomer > 1 Then
.Cells(nomer, 1).Value = ""
.Cells(nomer, 2).Value = ""
.Cells(nomer, 3).Value = ""
.Cells(nomer, 4).Value = ""
.Cells(nomer, 5).Value = ""
.Cells(nomer, 6).Value = ""
.Cells(nomer, 7).Value = ""
.Cells(nomer, 8).Value = ""
End If
End With
End Sub

```

Напишем код для процедуры обработки события Click для кнопки Выход:

```

Private Sub CommandButton3_Click()
Sheets("Форма").Activate
End
End Sub

```

Проверим работу нашего приложения. В рабочей книге активируем лист Форма. При нажатии на кнопку «Прием платежа», появится пустая таблица с заголовками и форма для данных. Введем в нее значения (рисунок 2). При правильном вводе, при нажатии на кнопку «Принять» они появятся в таблице Excel, а форма очистится и будет готова к приему новых данных.

	A	B	C	D	E	F	G	H	I
1	Фамилия	Имя	Адрес плательщика	Дата платежа	Тариф	Текущее показание	Предыдущее показание	Расход	Сумма к оплате
2	Иванов	Павел	г. Тверь, ул. Ленина, 25	01.01.2018	2	1020	920	100	200
3	Петров	Иван	г. Тверь, ул. Ленина, 27	01.02.2018	2	10000	80000	20000	4000
4									

Рисунок 2 - Заполненные форма и таблица

## 2. Визуализация данных

Построить диаграмму на основе наших данных по учету электроэнергии. Напишем процедуру отклика кнопки **Диаграмма**, созданную на форме UserForm1. На основе данных листа Учет процедура должна строить на отдельном листе диаграмму. Дважды щелкните по кнопке **Диаграмма**, и запишите следующий код:

```

Private Sub CommandButton4_Click()
    'Активация листа с именем Диаграмма
    Sheets("Диаграмма").Activate
    'Удалить на листе все объекты
    For Each i In ActiveSheet.Shapes
        i.Delete
    Next i
    'Строим новую диаграмму
    ActiveSheet.ChartObjects.Add(25, 25, 500, 300).Select
    With ActiveChart
        'Выбираем тип объемная гистограмма
        .ChartType = xl3DBarClustered
        'Определим, сколько записей в таблице
        M = 2
        Do
            If Sheets("База").Cells(M, 1).Value = "" Then Exit Do
            M = M + 1
        Loop
        M = M - 1
        'Определяем диапазон данных для построения диаграммы :
        'с листа «Учет» от ячейки I2 до ячейки IM
        .SetSourceData
        Source:=Sheets("База").Range("I2:I" + Trim(Str(M))),
        PlotBy:=xlRows
        'Определяем подписи к данным из первого столбца таблицы
        For i = 2 To M
            .SeriesCollection(i - 1).Name = "= База!R" + Trim(Str(i)) + "C1"
        Next
        'Располагаем диаграмму на отдельном листе
        .Location Where:=xlLocationAsObject, Name:= " Диаграмма"
        With ActiveChart
            &apos; Заголовок
            .HasTitle = True
            .ChartTitle.Characters.Text = " Сумма оплаты за электроэнергию "
            'Легенда
            .HasLegend = True
            .Legend.Select
            Selection.Position = xlLeft
            .HasDataTable = False
            .Axes(xlCategory).MajorTickMark = xlNone
            .Axes(xlCategory).MinorTickMark = xlNone
            .Axes(xlCategory).TickLabelPosition = xlNone
        End With
    End With
End Sub

```

На форме UserForm1 дважды щелкните по кнопке Выход, и измените процедуру данной кнопки таким образом, чтобы при выходе активным оставался лист с диаграммой.

Процедура :

```
Private Sub CommandButton3_Click()  
&apos;'Активируем рабочий лист с именем «Форма»  
Sheets("Диаграмма").Activate  
'Выход из программы  
End  
End Sub
```

Проверим работу нашего макроса. При нажатии на кнопку **Диаграмма** на листе с именем **Диаграмма** появится гистограмма, построенная по табличным данным (Рисунок 3).

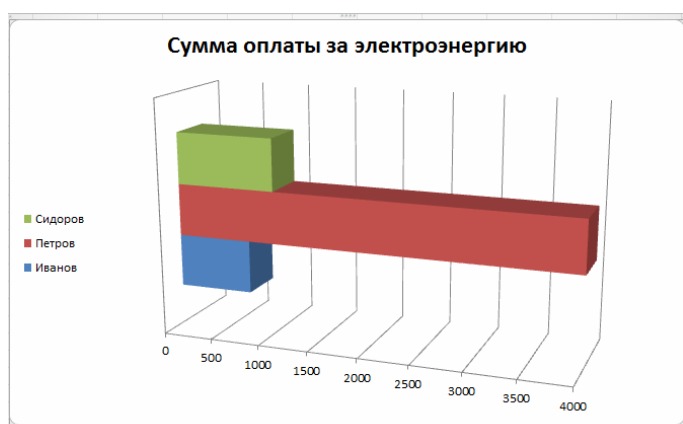


Рисунок 3 — Визуальное представление данных

### Задания для самостоятельного выполнения

Выбрать вариант и разработать программу с удобным диалоговым окном для создания на рабочем листе базы данных. Элементы управления выбрать самостоятельно для своего варианта. Написать процедуру графического представления данных.

1. Страховая компания. Страховая компания имеет филиалы, которые характеризуются наименованием, адресом и телефоном. В филиалы обращаются клиенты с целью заключения договора о страховании. В зависимости от принимаемых на страхование объектов и страхуемых рисков договор заключается по определенному виду страхования (страхование автотранспорта от угона, страхование домашнего имущества, добровольное медицинское страхование). При заключении договора фиксируются: дата

заклучения, страховая сумма, вид страхования, тарифная ставка и филиал, в котором заключался договор.

2. Гостиница. Гостиница предоставляет номера клиентам. Каждый номер характеризуется вместимостью, комфортностью (люкс, полу люкс, обычный) и ценой. О клиентах собирается определенная информация (фамилия, имя, отчество, паспортные данные, адрес жительства и некоторый комментарий). При заселении фиксируется дата заселения. При выезде из гостиницы для каждого места запоминается дата освобождения.

3. Ломбард. В ломбард обращаются различные лица с целью получения денежных средств под залог товаров. Клиент сообщает фамилию, имя, отчество и другие паспортные данные. После оценивания стоимости принесенного в качестве залога товара работник ломбарда определяет сумму, которую готов выдать на руки клиенту, а также свои комиссионные. Кроме того определяется срок возврата денег. Если клиент согласен, то договоренности фиксируются в виде документа, деньги выдаются клиенту, а товар остается в ломбарде. Если в указанный срок не происходит возврата денег, товар переходит в собственность ломбарда.

4. Оптово-розничная продажа товаров. Компания торгует товарами из определенного спектра. Каждый товар характеризуется наименованием, оптовой ценой, розничной ценой и справочной информацией. В компанию обращаются покупатели, для каждого из которых в базе данных фиксируются стандартные данные (наименование, адрес, телефон, контактное лицо). По каждой сделке составляется документ, в котором наряду с покупателем фиксируются количество купленного им товара и дата покупки.

5. Учет заказов. Компания занимается оптовой продажей различных товаров. Каждый из товаров характеризуется ценой, справочной информацией и признаком наличия или отсутствия доставки. В компанию обращаются заказчики. Для каждого из них в базе данных запоминаются стандартные данные (наименование, адрес, телефон, контактное лицо). По каждому заказу составляется документ, в котором наряду с заказчиком фиксируются количество купленного им товара и дата покупки.

6. Трудовая занятость. При обращении в бюро работодателя его стандартные данные (название, вид деятельности, адрес, телефон) фиксируются в базе данных. При обращении в бюро соискателя его стандартные данные (фамилия, имя, отчество, квали-

фикация, профессия, иные данные) также фиксируются в базе данных. По каждому факту удовлетворения интересов обеих сторон составляется документ.

7. Нотариус. Нотариальная контора готова предоставить клиенту определенный комплекс услуг. Услуги формализованы, т. е. составлен их список с описанием каждой услуги. При обращении клиента его стандартные данные (название, вид деятельности, адрес, телефон) фиксируются в базе данных. По каждому факту оказания услуги клиенту составляется документ, в котором указываются дата, услуга, сумма сделки, комиссионные (доход конторы), описание сделки.

8. Продажа запчастей. У компании есть определенный набор поставщиков, по которым известны название, адрес и телефон. У поставщиков приобретаются детали. Каждая деталь характеризуется названием, артикулом и ценой. Некоторые из поставщиков могут поставлять одинаковые детали (один артикул). Каждый факт покупки запчастей у поставщика фиксируется в базе данных, причем обязательными для запоминания являются дата покупки и количество приобретенных деталей. Цена детали может меняться от поставки к поставке.

9. Курсы по повышению квалификации. В учебном заведении организованы курсы повышения квалификации. Группы слушателей формируются в зависимости от специальности и отделения. В каждую из них включено определенное количество слушателей. Проведение занятий обеспечивает штат преподавателей, для каждого из которых в базе данных зарегистрированы стандартные анкетные данные (фамилия, имя, отчество, телефон) и стаж работы. В результате распределения нагрузки получена информация о том, сколько часов занятий проводит каждый преподаватель с соответствующими группами. Хранятся также сведения о виде занятий (лекция, практика), дисциплине и оплате за 1 час.

10. Определение факультативов для студентов. Преподаватели кафедры в высшем учебном заведении обеспечивают проведение факультативных занятий по некоторым предметам. Имеются сведения о студентах, включающие стандартные анкетные данные (фамилия, имя, отчество, группа). По каждому факультативу существует определенное количество часов и вид проводимых занятий (лекции, практика, лабораторные работы).

11. Учебная нагрузка. Необходимо распределять нагрузку между преподавателями кафедры. Имеются сведения о преподавателях, включающие наряду с анкетными данными сведения об их ученой степени, занимаемой должности и стаже работы.

Преподаватели кафедры должны обеспечить проведение занятий по некоторым дисциплинам. По каждой из них существует определенное количество часов. Все проводимые занятия делятся на лекционные и практические. По каждому виду занятий устанавливается свое количество часов.

12. Учет дополнительных работ. Для наведения порядка в этой сфере классифицированы все виды дополнительных работ и определена сумма оплаты по факту их выполнения. При возникновении дополнительной работы назначается ответственный, фиксируется дата начала. По факту окончания фиксируется дата и выплачивается дополнительная сумма к зарплате с учетом классификации.

13. Техобслуживание станков. Компания занимается ремонтом станков и другого оборудования. Клиентами компании являются промышленные предприятия. Ремонтные работы организованы следующим образом: все станки классифицированы по типам, странам-производителям, годам выпуска и маркам. Все виды ремонта отличаются названием, продолжительностью в днях, стоимостью. Исходя из этих данных, по каждому факту ремонта фиксируется вид станка, дата начала и дата окончания ремонта. Анализ показал, что нужно иметь информацию о том, сколько раз ремонтировался тот или иной станок.

14. Турфирма. Фирма продает путевки клиентам. У каждого клиента запрашиваются стандартные данные – фамилия, имя, отчество, адрес, телефон. После этого сотрудники компании выясняют у клиента, куда он хотел бы поехать отдыхать. Ему демонстрируются различные варианты, включающие страну проживания, особенности климата, отель. Обсуждается длительность пребывания и стоимость путевки. Если удалось найти приемлемый вариант, регистрируется факт продажи путевки (или путевок, если клиент покупает сразу несколько), фиксируется дата отправления. Иногда клиенту предоставляется скидка (скидки фиксированы и могут суммироваться). Фирма работает с несколькими отелями (название, категория, адрес) в нескольких странах. Путевки продаются на одну, две или четыре недели. Стоимость путевки зависит от длительности тура и отеля.

15. Грузовые перевозки. Компания осуществляет перевозки грузов по различным маршрутам. Необходимо отслеживать стоимость перевозок с учетом заработной платы водителей. Для каждого маршрута определено название, вычислено примерное расстояние и установлена некоторая оплата для водителя. Информация о водителях



включает фамилию, имя, отчество и стаж. Для проведения расчетов хранится полная информация о перевозках (маршрут, водитель, даты отправки и прибытия).

*16. Учет телефонных переговоров.* Телефонная компания предоставляет абонентам телефонные линии для междугородних переговоров. Абонентами компании являются юридические лица, имеющие телефонную точку, ИНН, расчетный счет в банке. Стоимость переговоров зависит от города, в который осуществляется звонок, и времени суток (день, ночь). Каждый звонок абонента оператор фиксирует в базе данных. При этом запоминаются город, дата, длительность разговора и время суток.

*17. Учет расходов.* Сотрудники частной фирмы могут осуществлять мелкие покупки для нужд фирмы, предоставляя в бухгалтерию товарный чек. Бухгалтерия отслеживает внутриофисные расходы. Каждый вид расходов имеет название, некоторое описание и предельную сумму средств, которые могут быть потрачены по данному виду расходов в месяц. При каждой покупке сотрудник оформляет документ, где указывает вид расхода, дату, сумму и отдел. Нужно хранить данные о расходах.

*18. Библиотека.* Библиотека решила зарабатывать деньги, выдавая напрокат книги, имеющиеся в небольшом количестве экземпляров. У каждой книги, выдаваемой в прокат, есть название, автор, жанр. В зависимости от ценности книги для каждой из них определена залоговая стоимость (сумма, вносимая клиентом при взятии книги напрокат) и стоимость проката (сумма, которую клиент платит при возврате книги, получая назад залог). Читатели регистрируются в картотеке, которая содержит стандартные анкетные данные (фамилия, имя, отчество, адрес, телефон). Все обращения читателей фиксируются, при этом по каждому факту выдачи книги запоминаются дата выдачи и ожидаемая дата возврата.

*19. Прокат транспорта.* Фирма, занимающаяся прокатом транспорта, имеет автопарк, содержащий некоторое количество автомобилей различных марок, стоимостей и типов. Каждый автомобиль имеет свою стоимость проката. Клиенты проходят обязательную регистрацию, в ходе которой о них собирается стандартная информация (фамилия, имя, отчество, адрес, телефон). Обращения клиентов фиксируются, при этом по каждой сделке запоминаются дата выдачи и ожидаемая дата возврата. Стоимость проката автомобиля должна зависеть не только от самого автомобиля, но и от срока его проката, а также от года выпуска.

20. Кредиты. Коммерческий банк выдает кредиты юрлицам. В зависимости от условий получения кредита, процентной ставки и срока возврата все кредитные операции делятся на несколько основных видов. Каждый из этих видов имеет свое название. Кредит может получить юридическое лицо (клиент), при регистрации предоставивший следующие сведения: название, вид собственности, адрес, телефон, контактное лицо. Каждый факт выдачи кредита регистрируется банком, при этом фиксируются сумма кредита, клиент и дата выдачи.

### **Контрольные вопросы**

1. Перечислите инструменты, используемые для разработки пользовательских форм.
2. В чем разница между элементами управления формами и элементами управления ActiveX?

## **Практическая работа №8**

### **АВТОМАТИЗАЦИЯ С VBA В MS ACCESS**

Цель работы. Ознакомится с приемами автоматизации работы с базой данных при помощи VBA

#### **Теоретическая часть. Выполнение работы**

##### **1. В чем отличия работы с базами данных в MS Excel и MS Access**

Программа СУБД MS Access предназначена для работы с базами данных офисного масштаба. Подобно рабочим книгам Excel, база данных Access позволяет работать с табличными данными, систематизированными каким-нибудь образом. При этом существуют два основных отличия. Так легко, наглядно и просто, как это выглядит на рабочем листе Excel, работать с таблицами Access сложнее по определенным правилам. Второе отличие заключается в том, рабочие инструменты Access обладают несопоставимо большими функциональными возможностями и позволяют решать абсолютно любые задачи обработки данных в масштабах офиса. Структура данных может быть при этом достаточно сложной, а также объем данных может быть большим.

##### **2. Основные сведения об объектах ACCESS, DAO и ADO**

К основным объектам Access Application относятся:

- Объект Application – это сама СУБД Access, а также среда для выполнения макрокоманд и процедур VBA. При установке характеристик объекта Application изменяются свойства всей среды Access.
- Объекты Form ссылаются на конкретную открытую форму и являются членами совокупности Forms. Добавить объект или удалить его из совокупности Forms можно, только открыв или закрыв форму. Существует около 30 свойств событий, распознаваемых формой.

- Объекты Report ссылаются на конкретный открытый отчет и являются членами совокупности Reports. Добавить объект или удалить его из совокупности Reports можно, только открыв или закрыв отчет. Отчет распознает семь свойств событий.
- Объект Control представляет элемент управления в форме или отчете и принадлежит к совокупности Controls данной формы или отчета. Каждый тип элемента управления имеет собственный набор свойств, включая характеристики распознаваемых им событий.
- Объект DoCmd - необычный объект, доступный только в VBA. Не представляя никакой физической сущности с заданными свойствами и поведением, он осуществляет важную связь между макропрограммированием и программированием на VBA. Его можно использовать при выполнении макродействий в Visual Basic.

### Использование объектов Access Application

В семействах Forms и Reports содержатся лишь объекты, которые были открыты при помощи оператора DoCmd:

Open тип\_объекта или его эквивалентов.

Объект Form представляет форму MS Access, открытую в режиме Design(Конструктор), Form(Режим формы) или DataSheet (Режим таблицы).

Чтобы начать работу с объектом Form, необходимо открыть форму, используя метод OpenForm объекта DoCmd. При неоднократном обращении к одному и тому же объекту Form лучше объявить объектную переменную, которая будет передавать этот объект.

Объект Report представляет собой отчет MS Access, открытый в режиме Design (Конструктор), Print Preview (Предварительный просмотр) или Layout Preview (Образец).

Чтобы в VB можно было работать с объектом Report, сначала надо открыть отчет методом OpenReport объекта DoCmd. Можно получить ссылку на объект Report и присвоить ее объектной переменной.

Метод объекта DoCmd, соответствующий макрокоманде, обычно выполняет точный эквивалент макродействия. Методы объекта не возвращают значения или объекты и поэтому имеют следующий синтаксис:

DoCmd.метод аргумент1, аргумент2,..., аргументN

Существует два типа методов DoCmd, работающих с объектом Application в Access. Согласно одному из них объект нужно задавать как аргумент метода.

#### Пример

DoCmd.OpenReport"Приказ"

Второй метод применяется к активному объекту, поэтому объект не нужно задавать в качестве аргумента.

#### Пример

DoCmd.Close

### **3. Объекты DAO**

Объекты DAO используются для управления базами данных из всех приложений, поддерживающих VBA.

Они позволяют:

- проектировать и создавать базы данных, изменять определения таблиц, запросов, индексов и связей.
- добавлять, удалять и изменять записи в базах данных.
- защищать и обеспечивать безопасность данных в базах данных.
- работать с различными форматами данных и присоединять таблицы из других баз данных.
- использовать информацию из удаленных баз данных и создавать приложения клиент-сервер.

К наиболее часто используемым объектам DAO относятся:

- Объект Database представляет открытую базу данных. Одновременно в одном рабочем пространстве можно открыть несколько баз данных. Существующая база данных открывается в процедуре VBA с помощью метода OpenDatabase объекта Workspace (рабочего пространства, создаваемого автоматически при запуске Access). Для ссылки на

текущую базу данных Access позволяет использовать функцию CurrentDb().

- Объект Recordset представляет набор записей в таблице базы данных либо набор записей, получаемых в результате выполнения запроса или оператора SQL. В непустом объекте Recordset можно указать одну текущую запись.

Каждый объект Recordset состоит из записей и полей. Создаются объекты Recordset при вызове метода OpenRecordset. Этим методом обладают разные классы объектов: Database, QueryDef, RecordSet, TableDef. Каждый объект этих классов может породить набор записей(таблицу), составляющую новый объект RecordSet.

#### Перемещение по записям

Методы MoveLast, MoveFirst, MoveNext и MovePrevious находят в указанном объекте Recordset последнюю, первую, следующую за текущей и предыдущую запись и делают ее текущей. Для перемещения на заданное число записей вызывается метод Move.

#### Пример

Рассмотрим автоматизацию работы с базой данных на примере несложной базы данных, состоящей из пяти таблиц Клиенты, Операции, ТипКлиента, Тип. Таблица Клиенты будет содержать сведения о клиентах фирмы. Вторая таблицы будет содержать сведения о приходных и расходных операциях, связанных с этими клиентами. Так как Клиенты подразделяются на несколько категорий, добавим таблицу ТипКлиента. КодТипа – первичный ключ.

#### Структура таблицы ТипКлиенты

Имя поля	Тип данных	Размер поля
<u>КодТипа</u>	Счетчик	
ТипКлиента	Текстовый	25

## Структура таблицы Клиенты

Имя поля	Тип данных	Размер поля
<b><u>КодКлиента</u></b>	Счетчик	
Фамилия	Текстовый	15
Имя	Текстовый	15
Отчество	Текстовый	15
ДатаРождения	Дата/Время	Краткий формат даты
Адрес	Текстовый	50
E-mail	Текстовый	20
скидка	Числовой, Процентный	Одинарное с плавающей точкой
КодТипа	Мастер подстановок из таблицы «ТипКлиента»	
КодОбращения	Мастер подстановок из таблицы «Обращение»	

## Структура таблицы ТипОперации

Имя поля	Тип данных	Размер поля
<b><u>КодОперации</u></b>	Счетчик	
Операция	Текстовый	25

## Структура таблицы Операции

Имя поля	Тип данных	Размер поля
Дата	Дата/Время	
<b><u>ПорядковыйНомер</u></b>	Счетчик	
КодОперации	Мастер подстановок из таблицы ТипОперации	
КодКлиента	Мастер подстановок из таблицы «Клиенты»	
Сумма Операции	Денежный	

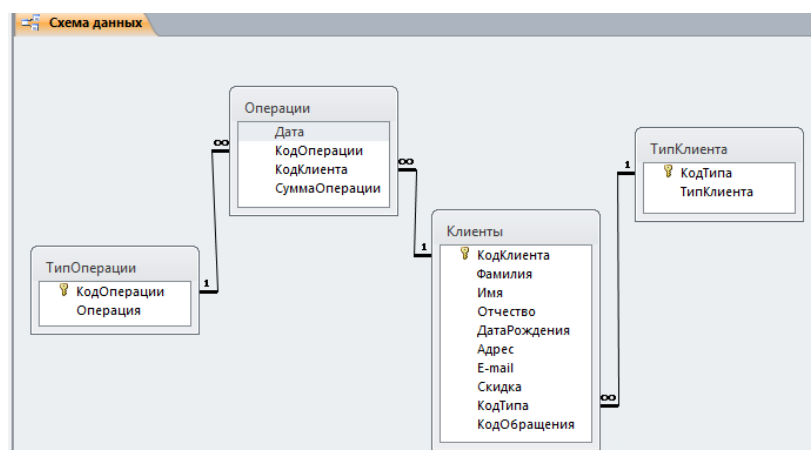


Рисунок 1 Схема базы данных по клиентам

## Ввод данных

На следующем этапе создадим экранные формы для ввода данных в таблицы “Клиенты” и “Операции”. Также создадим вспомогательную (подчиненную) форму для формы “Клиенты”, которая будет отображать операции только по текущему клиенту. Так как подчиненная форма служит для отображения данных, а не для их редактирования назначим соответствующее свойство. Для этого переходим в режим Конструктора, правой кнопкой на форме выбираем Свойства и делаем соответствующие назначения на вкладке Данные - выбрать значение Нет для полей «Разрешить удаление», «Разрешить изменения», «Разрешить добавления» и «Ввод данных» (рисунок 2).

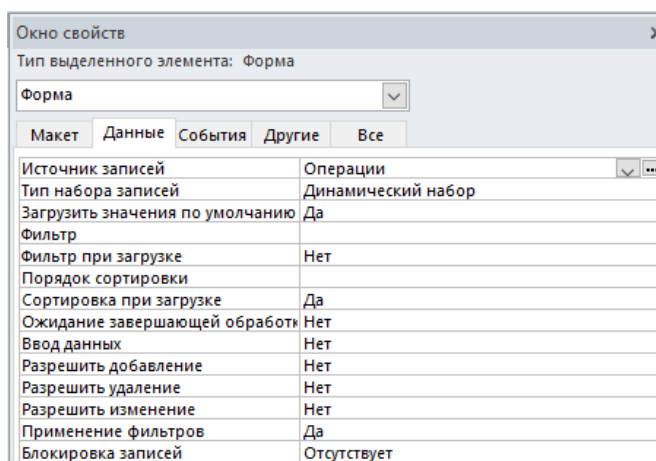


Рисунок 2. Назначение свойств подчиненной формы для запрета редактирования в ней данных

### **Пример автоматизации 1**

При помощи VBA усовершенствуем нашу форму – автоматизируем заполнение поля Скидка. Подобно любому объекту Office, форма генерирует ряд событий. И элементы управления, размещенные на форме, также генерируют события. Поле Скидка заполняется в зависимости от категории клиента, которая кодируется в поле КодТипа. Создадим процедуру обработки события для элемента экранной формы.



Откройте форму Клиенты в режиме Конструктора.

1. Щелкните правой кнопкой мыши на поле КодТипа и в контекстном меню выберите Обработка событий. Появится окно Построитель. Выделите в списке пункт Программы, как показано на рисунке, и щелкните на кнопке ОК.

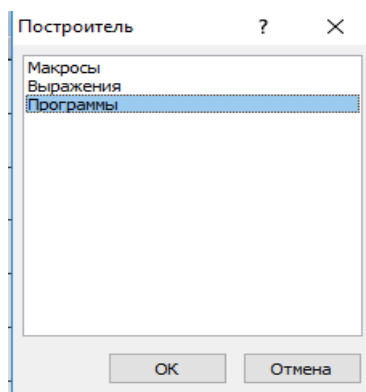


Рисунок 3. Окно Построитель

На экране отобразится окно редактора программного кода. Нужно обработать событие AfterUpdate. Введите текст процедуры обработки события «После обновления» для поля со списком КодТипа (рисунок 4).

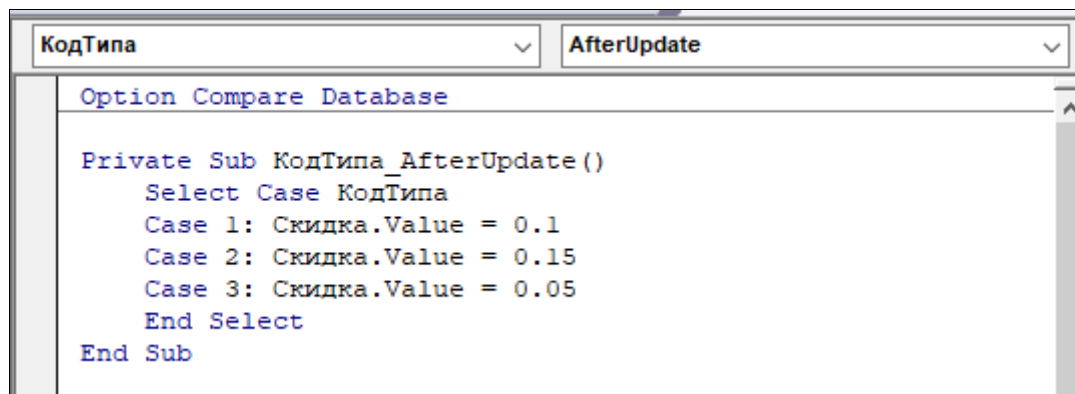


Рисунок 4. Процедура обработки события «После обновления»

Таким образом, в зависимости от того, какой тип выбран для текущего клиента, в поле Скидка помещаются соответствующие значения: 5% - для разовых клиентов, 10% - для реализаторов и 15% - для постоянных клиентов.

## Пример автоматизации 2

Событие самой формы Current происходит при смене текущей записи, при открытии формы, а также в других случаях. Часто программный код, который служит для автоматизации формы, помещают в обработчик этого события. Допустим, у клиента текущей записи сегодня день рождения, данный факт не должен остаться незамеченным.

Создадим процедуру обработки события Current формы «Клиенты».

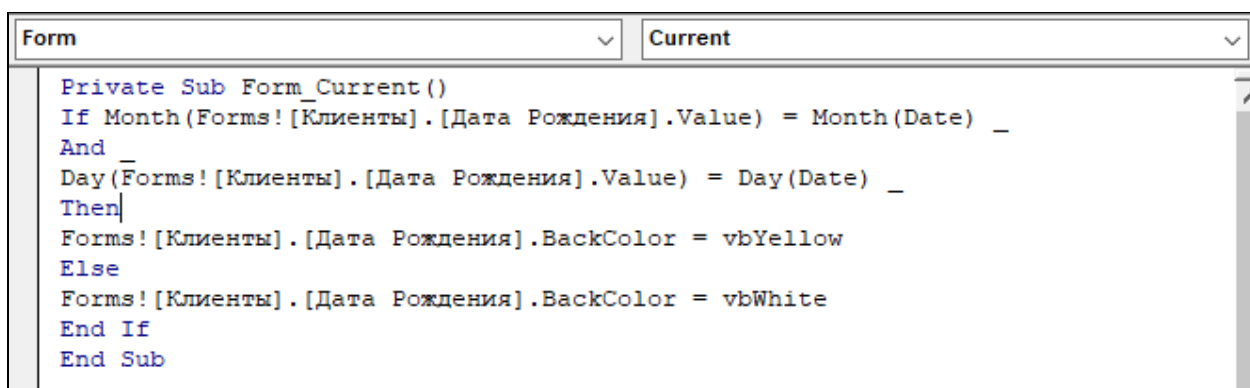


Рисунок 5. Процедура обработки события формы Клиенты

В данном макросе используются функции Month( ) и Day( ), которые возвращают номера месяца и дня в месяце из значения даты, переданного им в качестве параметра.

Смысл кода этого макроса такой: если месяц даты рождения совпадает с месяцем текущей системной даты и день месяца даты рождения совпадает с днем месяца системной даты, то для элемента управления ДатаРождения задать цвет фона - желтый. В противном случае задать для элемента управления ДатаРождения цвет фона – белый.

### Задания для самостоятельной работы

Решается задача обработки поступления товаров на склад. База данных содержит две таблицы. Входная информация для решения задачи содержится в таблице «Наличие», а выходная – в таблице «Накладные». При решении этой задачи корректируются записи входной таблицы

"Наличие" и формируются (добавляются) новые записи в выходную таблицу "Накладные".

Справочная информация о товарах, хранящихся на складе, имеется в таблице "Наличие", содержащей следующие данные:

- Код товара
- Название товара
- Остаток
- Дата
- Единицы измерения

В форму осуществляется ввод информации о каждой поступившей на склад партии товаров:

- номер накладной
- код товара
- дата поступления
- количество

В таблице "Наличие" осуществляется поиск записи с введенным кодом детали. Если искомая запись найдена, то программа должна предусмотреть выполнение следующих операций:

- 1) Произвести корректировку найденной записи в таблице "Наличие", заключающуюся в добавлении к значению поля "Остаток" введенного значения "количество" и в замещении значения поля "Дата" на введенное значение "дата поступления"
- 2) Добавить новую запись в таблицу "Накладные". Выходная таблица имеет следующий состав полей:

- Номер накладной
- Код товара
- Дата поступления
- Количество

### **Контрольные вопросы**

1. Перечислите основные объекты Access Application
2. Назначение объектов DAO. Перечислите их

## Практическая работа №9

### ПРОГРАММИРОВАНИЕ В MS WORD

Цель работы. Изучить объекты MS Word, их свойства и методы

#### Теоретические сведения. Выполнение работы

##### 1. Основные сведения об объектах Word, их свойствах и методах

Visual Basic for Applications поддерживает набор объектов, соответствующих элементам Word. Используя свойства и методы этих объектов можно автоматизировать все операции в Word. Однако целесообразно автоматизировать выполнение тех операций, для реализации которых нет стандартных средств в Word или их выполнение стандартными средствами является трудоемкой или рутинной работой.

Наиболее важными объектами являются:

Объект Document представляющий собой новый или созданный ранее открытый документ.

Основными свойствами объекта Document являются:

- Count - количество открытых в данный момент документов;
- ActiveDocument - активный документ.

Некоторые методы объекта Document и коллекции Documents:

- Open - открывает файл, содержащий существующий документ и автоматически добавляет его в коллекцию;
- Add - добавляет новый пустой документ;
- Save - сохраняет изменения в существующем документе без закрытия;
- Save As (только для объекта) - сохраняет активный вновь созданный документ в текущей папке;
- Item - позволяет получить доступ к элементу коллекции;

- Activate (только для объекта) - активизирует открытый документ;
- PrintOut (только для объекта) - печать документа;
- Close - закрывает документ.

Объекты, задающие структуризацию текста документа:

- Character (символ)
- Word (слово)
- Sentence (предложение)
- Paragraph (абзац)
- Section (раздел документа)

Все эти объекты имеют свойства:

- Count - свойство возвращает количество элементов в коллекции;
- First - свойство возвращает объект, являющийся первым элементом коллекции;
- Last - свойство возвращает объект, являющийся последним элементом.

Коллекции Characters, Words, Sentences имеют единственный метод Item(Index).

Коллекция Paragraphs имеет все вышеперечисленные для данной группы коллекций свойства и множество свойств, значения которых определяют формат абзаца. Формат абзаца может быть определен и с помощью методов.

Приведем некоторые методы коллекции Paragraphs:

- Item - определяет элемент коллекции;
- Add - добавление нового пустого абзаца (параметр метода указывает точку вставки, задается объектом Range);
- InsertParagraph, InsertParagraphAfter, InsertParagraphBefore - осуществляют вставку пустого абзаца вместо текста или

после, или перед текстом, задаваемым объектом Selection или Range;

- Reset - удаляет форматирование, сделанное вручную, применяя к абзацу формат, заданный стилем абзаца;
- Indent, Outdent - увеличивают, уменьшают отступ абзаца от края листа;
- TabHangingIndent(Count), TabIndent(Count) - увеличивают (Count>0), уменьшают (Count<0) выступ или отступ абзаца от края листа на заданное количество позиций;
- Space1, Space2, Space15 - устанавливают межстрочный интервал (одинарный, двойной, полуторный).

Объекты Range (диапазон) и Selection (выделение), представляющие части документа. Объект Range позволяет задать произвольный диапазон, представляющий собой последовательность индексированных элементов и может быть получен через метод Range или свойство Range других объектов.

Свойствами объекта Range являются:

- Start - начальная символьная позиция диапазона.
- End - конечная символьная позиция диапазона.
- Text - позволяет получить или изменить содержимое объекта.

Объект Selection задает выделенную в документе область (фрагмент). Выделенный в документе фрагмент задает непрерывную область элементов, но не является диапазоном, заданным своим началом и концом. Выделить можно только один фрагмент, поэтому один объект Selection может быть активен в данный момент времени. Объект Selection может быть получен с помощью свойства Selection или метода Select других объектов.

Ниже приведены некоторые методы присущие объектам Selection и Range:

- Move - метод перемещения точки вставки;

- MoveStart, MoveEND - методы изменения значения свойств Start и End;
- Collapse - сворачивает диапазон к его началу или концу.
- Next - метод получения ссылки на очередной элемент коллекции объектов в диапазоне или выделенном фрагменте;
- Delete - удаляет текст входящий в диапазон.
- InsertAfter, InsertBefore вставляет текст до или после текста, входящего в диапазон. После вставки текста диапазон расширяется, включая в себя вставленный текст.
- Copy - копирует объект в буфер обмена;
- Cut - перемещает объект в буфер обмена;
- Paste - позволяет поместить содержимое буфера в область, заданную объектом Range или Selection.

## **2. Запись макросов с помощью макрорекордера и способы выполнения макросов в приложении Microsoft Word**

В программе Microsoft Word создается много документов типа справка, расписка, докладная записка и так далее. Эти документы имеют постоянную и переменную части. Создание таких документов можно упростить, используя язык программирования VBA.

### **➤ Задание**

- создать макрос с текстом документа «Справка»;
- создать формы пользователя для ввода переменной части текста документа «Справка»;
- написать программного кода процедур обработки нажатия кнопок на форме пользователя;
- редактировать текст процедуры макроса в соответствии с текстом документа «Справка» с целью подключения текстовых полей и поля со списком формы пользователя.


Создадим документ «Справка» с помощью макрорекордера; создается макрос очистки окна документа Word; создается новая панель инструментов с кнопками для запуска макросов; создается меню с пунктами для запуска макросов.

1. Создать документ Word и сохранить его с именем Пр9.docx.
2. Создать макрос1 с помощью макрорекордера. Макрос1 создает текст документа «Справка».
3. Выполнить команду Разработка, Макрос, Запись макроса:
  - в поле Имя макроса оставить Макрос1. В поле Макрос доступен для: выбрать имя данного документа Пр9. Нажать кнопку ОК;
  - набрать текст документа «Справка» с нужными параметрами абзаца и шрифта;
  - остановить запись макроса.
4. Проверить работу макроса запустив его на исполнение командой: Разработчик, Макросы, Макрос1, Выполнить. Текст документа:

<p style="text-align: center;"><b>СПРАВКА</b></p> <p><i>дана настоящая в том, что Иванов Иван Иванович учится в Финансовом университете в качестве бакалавра. Справка выдана для предъявления по месту требования 16 марта. 2019 г.</i></p> <p>МП</p> <p><b>Директор</b> <span style="float: right;"><b>И.Н. Петрова</b></span></p>
---

5. Создать Макрос2 с помощью макрорекордера. Макрос2 выполняет очистку содержимого документа (Выделить все {Ctrl+A}, клавиша Delete).
6. Создать панель быстрого доступа для Пр9.docx с кнопками для запуска макросов:



- выбрать пункт меню  - Параметры Word - Настройка;
  - в окне “Настройка панели быстрого доступа и сочетаний клавиш” на вкладке “Выбрать команду из ” выбрать пункт Макросы, а на вкладке “Настройка панели быстрого доступа” выбрать пункт Для Пр9;
  - в том же диалоговом окне команд Макросы выделить команду Макрос1 и добавить в панель быстрого доступа для документа Пр9. Аналогично добавить команду Макрос2;
  - изменить надпись или значок на кнопке, назначенной макросу “Изменить...”, изменить кнопку и отображаемое имя (например, “Справка”), выбрать новый значок для кнопки, несколько изменить его и назначить макрос (Макрос1) этой кнопке;
  - аналогично изменить надпись и кнопку для вызова еще одного макроса (Макрос2), например “Очистка документа”.
7. Выполнить макросы, используя кнопки панели быстрого доступа для Lab.
- Сохранить документ на диске в личной папке в файле с именем lab.docm с типом файла «Документ Word с поддержкой макросов».
8. Закройте Word.

### **3. Создание формы пользователя для ввода переменной части текста**

1. Открыть документ Word с именем Пр9.docm, и сохранить с именем Пр9\_1.docm.
2. Открыть окно редактора VBA командой Alt+F11 или Разработчик, Код, Visual Basic.
3. В проекте Пр9\_1 создать форму пользователя командой Insert - UserForm
4. Элементы для формы брать на панели Элементы управления.

5. Изменить заголовок формы (свойство Caption). В заголовке должны быть Фамилия, инициалы и номер группы студента.
6. В процедуры нажатия кнопок ввести программный код. Для входа в процедуру выполнить двойной щелчок мышью на соответствующей кнопке формы в режиме конструктора.

```
' Нажатие кнопки Выполнить
Private Sub CommandButton1_Click()
Ма крос1
End Sub
```

```
' Нажатие кнопки Выход
Private Sub CommandButton2_Click()
UserForm1.Hide
End Sub
```

```
Private Sub CommandButton3_Click()
Макрос3
End Sub
```

7. Для объекта Поле со списком (ComboBox):

```
' Ввод элементов списка в поле со списком
Private Sub UserForm_Initialize ()
ComboBox1.AddItem "П.П. Петров"
ComboBox1.AddItem "С.С. Сидоров"
ComboBox1.AddItem "И.Н. Петрова"
ComboBox1.ListIndex = 0
End Sub
```

8. Отредактировать текст макроса Макрос1, заменив часть постоянного текста для документа «Справка», полями формы пользователя.

```

' Код макроса Макрос1
Sub Макрос1()
    Selection.Font.Size = 12
    Selection.Font.Bold = wdToggle
    Selection.ParagraphFormat.Alignment = wdAlignParagraphCenter
    Selection.TypeText Text:="СПРАВКА"
    Selection.TypeParagraph
    Selection.Font.Bold = wdToggle
    Selection.ParagraphFormat.Alignment = wdAlignParagraphLeft
    Selection.TypeText Text:="да на  на стояща я в том, что "
    Selection.Font.Italic = wdToggle
    Selection.TypeText Text:=UserForm1.TextBox1
    Selection.TypeParagraph
    Selection.TypeText Text:=UserForm1.TextBox2
    Selection.Font.Italic = wdToggle
    Selection.TypeParagraph
    Selection.TypeText Text:="работает "
    Selection.Font.Italic = wdToggle
    Selection.TypeText Text:=UserForm1.TextBox3
    Selection.Font.Italic = wdToggle
    Selection.TypeParagraph
    Selection.TypeText Text:="в должности "
    Selection.Font.Italic = wdToggle
    Selection.TypeText Text:=UserForm1.TextBox4 & ". "
    Selection.Font.Italic = wdToggle
    Selection.TypeParagraph
    Selection.TypeText Text:="Справка выдана для предъявления"
    Selection.TypeParagraph
    Selection.Font.Italic = wdToggle
    Selection.TypeText Text:=UserForm1.TextBox5
    Selection.TypeParagraph
    Selection.TypeText Text:=UserForm1.TextBox6
    Selection.Font.Italic = wdToggle
    Selection.TypeParagraph
    Selection.TypeParagraph
    Selection.TypeText Text:="МП"
    Selection.TypeParagraph
    Selection.TypeText Text:="Директор" & vbTab
    Selection.ParagraphFormat.TabStops.Add Position:=CentimetersToPoints(15.25 _
    ), Alignment:=wdAlignTabRight, Leader:=wdTabLeaderSpaces
    Selection.Font.Italic = wdToggle
    Selection.TypeText Text:=UserForm1.ComboBox1.Text
    Selection.Font.Italic = wdToggle
    Selection.TypeParagraph
End Sub

```

9. Создайте макрос Макрос2 и Макрос3:

```
' Код макроса Макрос2 (удаляет содержимое документа)
Sub Макрос2()
    Selection.WholeStory
    Selection.Delete Unit:=wdCharacter, Count:=1
End Sub

' Код макроса Макрос3 (открывает форму пользователя)
Sub Макрос3()
    UserForm1.Show
End Sub
```

10. Кнопку Справка на панели быстрого доступа Для Пр9\_1 подключить к макросу Макрос3.

11. Проверить работу кнопок на панели быстрого доступа Для Пр9\_1.

12. Сохранить документ.

### **Задание для самостоятельного выполнения**

Создать макрос и форму для автоматизации процесса создания документов-справок (на выбор из списка):

- с места жительства
- с места учебы
- о годовой заработной плате
- о состоянии здоровья
- об отсутствии (наличия) судимости
- об отсутствии (или имеющейся) академической задолженности.

### **Контрольные вопросы**

1. Перечислите важные объекты приложения MS Word, их свойства и методы.
2. Какие объекты задают структуру текста документа, какие у них свойства