

Министерство образования и науки Российской Федерации  
Уфимский авиационный техникум  
федерального государственного бюджетного образовательного учреждения  
высшего образования  
«Уфимский государственный авиационный технический университет»

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ СТУДЕНТОВ ПО ПРОВЕДЕНИЮ  
ПРАКТИЧЕСКИХ ЗАНЯТИЙ**

**для специальности**

**09.02.05 – Прикладная информатика (по отраслям)**

по МДК 02.05 раздела 5 «Отладка и тестирование  
программного обеспечения отраслевой направленности»

Уфа 2016

«Рассмотрено»  
На заседании ЦК  
Протокол №\_\_ от\_\_\_\_\_  
\_\_\_\_\_ Н.Е.Карпова

«Утверждаю»  
Директор УАТ  
ФГБОУ ВО «УГАТУ»  
\_\_\_\_\_ Р. М. Киреев

Методические указания содержат материал для выполнения студентами практических работ по Разделу 5 «Отладка и тестирование программного обеспечения отраслевой направленности» профессионального модуля «Разработка, внедрение и адаптация отраслевого программного обеспечения» для специальности 09.02.05 «Прикладная информатика» (по отраслям)

Организация-разработчик: Уфимский авиационный техникум ФГБОУ ВПО «УГАТУ»

Разработчик:

Старцева И.В., преподаватель Уфимского авиационного техникума ФГБОУ ВПО «УГАТУ»

СОГЛАСОВАНО: Заместитель директора по УР

Р. М. Хузин

# Лабораторная работа №1. Разработка программного обеспечения с помощью языка программирования VBA

**Тема.** Работа в среде VisualBasic.

**Цель занятия.** Приобрести начальные навыки работы в среде VisualBasic. Изучить элементы среды программирования, порядок установки элементов на форму и управления размещением элементов.

## 1.1 Краткие теоретические сведения

Рабочее окно (рис. 1.1) представляет собой интегрированную среду разработки – интерфейс языка программирования VisualBasic. Эта среда может настраиваться с помощью диалогового окна, вызываемого командой **Tools\Options**. Рабочее окно предлагает целый набор инструментальных средств, которые можно использовать при создании программ.

Большинство элементов рабочего окна характерны для приложений Windows: заголовок, меню, панели инструментов. Интегрированная среда разработки проекта многооконная. Состав окон приведен на рис. 1.1.

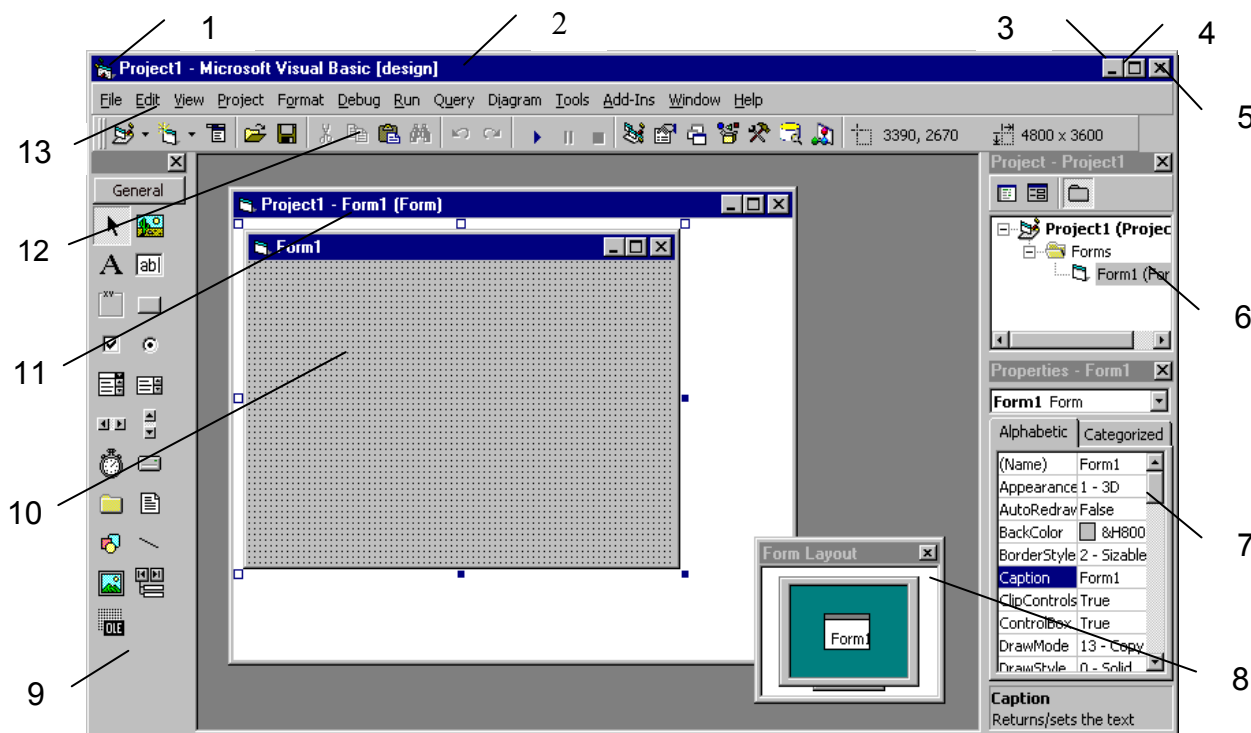


Рис. 1.1. Окно программы VisualBasic

1 – кнопка системного меню, 2 – заголовок, 3 – кнопка свертывания окна в пиктограмму, 4 – кнопка разворачивания окна, 5 – кнопка закрытия окна, 6 – окно проекта, 7 – окно свойств, 8 – окно позиционирования формы, 9 – панель элементов управления, 10 – шаблон формы, 11 – конструктор форм, 12 – стандартная панель инструментов, 13 – главное меню

## Меню (Menu)

Строка Меню содержит список команд, предназначенных для управления разработкой проекта, пункты меню могут иметь несколько уровней вложения:

**File (Файл)**– содержит команды для работы с файлами создаваемых приложений, загрузки, сохранения, вывода на печать.

**Edit (Правка)**– содержит команды редактирования, предназначенные для создания исходного текста программы, включая средства поиска и замены.

**View (Просмотр)**–обеспечивает доступ к различным частям приложения и среды разработки VB.

**Project (Проект)**- предназначен для добавления новых объектов VB к разрабатываемым проектам, добавления или удаления элементов управления на панель элементов управления, настройки свойств проекта.

**Format(Формат)** – дает доступ к различным настройкам элементов управления, размещенных на создаваемых программистом формах.

**Debug(Отладка)** – содержит средства, предназначенные для отладки программ или поиска ошибок.

**Run(Выполнение)**– служит для запуска и остановки программ непосредственно из среды разработки.

**Tools (Инструменты)**– обеспечивает доступ к работе с процедурами и меню внутри приложения. Этот пункт меню имеет важную команду **Options**, которая открывает одноименную диалоговую панель с закладками **Options**, где настраивается практически вся среда разработки VisualBasic.

**Add-Ins (Добавить в ...)** – дает доступ к инструментам, которые могут быть добавлены к окружению VB: мастера, ActiveX – элементы и другое.

**Diagram(Диаграммы)** – содержит средства для оформления диаграмм.

**Window (Окно)** – используется для работы с окнами в среде разработки.

**Query** – доступ к внешним базам данных.

**Help** – справочная система.

Выбор пунктов меню осуществляется мышью или клавишами. При управлении с помощью клавиатуры для входа в меню используется клавиша **Alt**. Выбор пунктов меню осуществляется с помощью клавиш управления курсором или с использованием горячих клавиш (подчеркнутые символы в командах меню): [**Alt-клавиша**].

### **Панели инструментов (Toolbars)**

VB имеет четыре стандартные панели инструментов: *Standard* – стандартная, *Edit* - редактирования, *Debug* – отладки и *FormEditor* – редактор форм.

**Стандартная панель** инструментов содержит команды, дублирующие основные команды меню.

**Панель редактирования** используется при вводе и редактировании текста программы. Содержит кнопки для вывода всплывающих списков свойств и методов объекта, констант, синтаксиса для процедур и методов, а также содержит кнопки для управления редактированием текста.

**Панель отладки** – служит для отладки программ в процессе ее выполнения и обеспечивает запуск программы на выполнение, временную остановку (пауза), выход из программы, установку точек останова, пошаговое выполнение в режиме пошагового выполнения, вычисление выделенного выражения.

**Панель редактора форм** – применяется при разработке форм. Она позволяет изменять порядок элементов управления, выравнивать их по горизонтали и вертикали, уравнивать размеры выделенных элементов управления по ширине и высоте, а также блокировать или разблокировать элементы управления в форме.

## **Панель инструментов элементов управления и компонентов пользователя (Toolbox)**

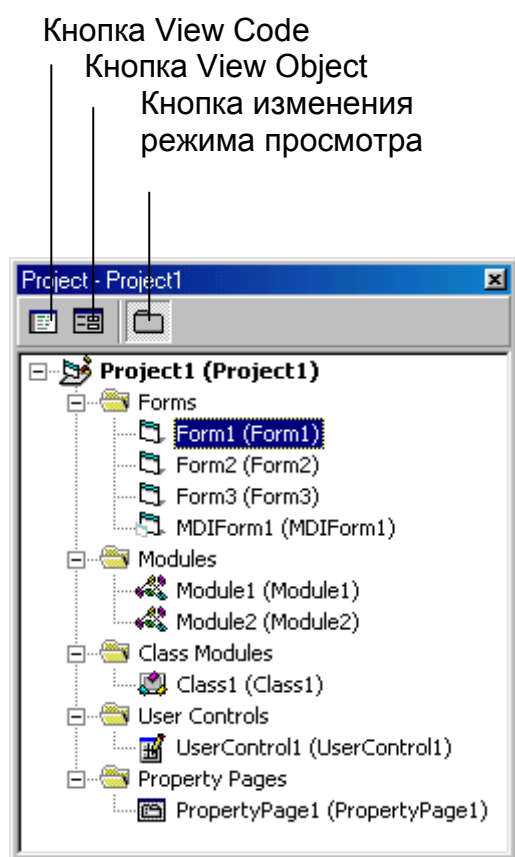


Рис.1.2. Окно Проект

Панель Toolbox содержит элементы управления. **Элементы управления** – это элементы, которые используются при разработке интерфейса создаваемых приложений. Общее количество доступных к использованию элементов управления зависит от того, какая версия VB используется.

Для добавления новых компонентов к панели инструментов Toolbox из числа зарегистрированных необходимо:

- ввести команду **Project\Components**, выбрать закладку **Controls**;
- найти в списке нужный элемент управления и установить напротив него флажок;
- выйти из окна диалога, щелкнув кнопку **Ok**.

### **Форма (Form)**

Создаваемые в VisualBasic окна называются **формами**. **Форма** – это основное окно интерфейса разрабатываемой программы, форма – это также основа для создания окон диалога.

### **Окно Проект (Project)**

В окне проекта (браузер проектов) (рис. 1.2.) отображаются все элементы приложения: формы, модули, классы и т.п., сгруппированные по категориям. В VB все разрабатываемые приложения называются **проектами**. Проект представляет группу связанных файлов и может содержать несколько групп компонентов (формы, модули и т.д.). Все проекты VB строятся по модульному принципу, поэтому и текст программы состоит не из одного большого файла, а из нескольких частей - процедур. Несколько проектов также могут объединяться в группы. Ниже заголовка окна проекта размещены три кнопки: кнопка просмотра текста программы; кнопка просмотра объекта; кнопка изменения режима просмотра: в виде списка компонентов или в виде дерева групп компонентов. В качестве объектов могут быть формы, MDI-формы, модули, классы, управляющие элементы, страницы свойств. Модули и классы не имеют визуальных компонентов.

### **Окно Свойства (Properties)**

В окне свойств (рис. 1.3.) задаются свойства выбранного элемента управления.

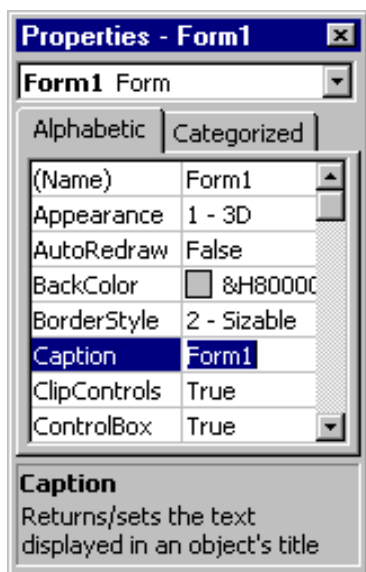


Рис. 1.3. Окно Свойства

В строке заголовка окна свойств рядом с текстом Properties указывается имя формы, которой принадлежит элемент управления. Поле со списком под строкой заголовка позволяет выбрать требуемый элемент управления. В списке, расположенном ниже, перечислены свойства этого элемента. Эти свойства могут быть упорядочены в алфавитном порядке (Alphabetic) или расположены по категориям (Categorized). Набор свойств зависит от типа элемента управления.

Как установить свойства объекта? Имеется три способа:

- ввести значение в поле справа от свойства;

- выбрать из списка, который открывается щелчком мыши по полю;

- установить с помощью окна диалога, при щелчке мышью по полю появляется кнопка (...) – троеточие или эллипсис. При щелчке по кнопке троеточие появляется окно

диалога для настройки соответствующего свойства.

### Окно Программа (Code)

Сразу после запуска окно Программа не отображается. Текст программы в VB разделяется на части – подпрограммы и записывается в процедуры обработки событий или процедуры пользователя. Поэтому текст программы, как правило, связан с определенным элементом управления. Это позволяет открыть окно диалога двойным щелчком по соответствующему элементу формы или по самой форме. Кроме того, окно программы можно открыть щелчком по кнопке *View Code* в окне *Project*.

В верхней строке окна программы (рис.1.4) располагается строка заголовка, в которой указано имя проекта и управляющие кнопки (системного меню, закрытия окна, свертывания и развертывания окна). Ниже строки заголовка расположены два раскрывающихся списка: список объектов (левый) и список процедур (правый).

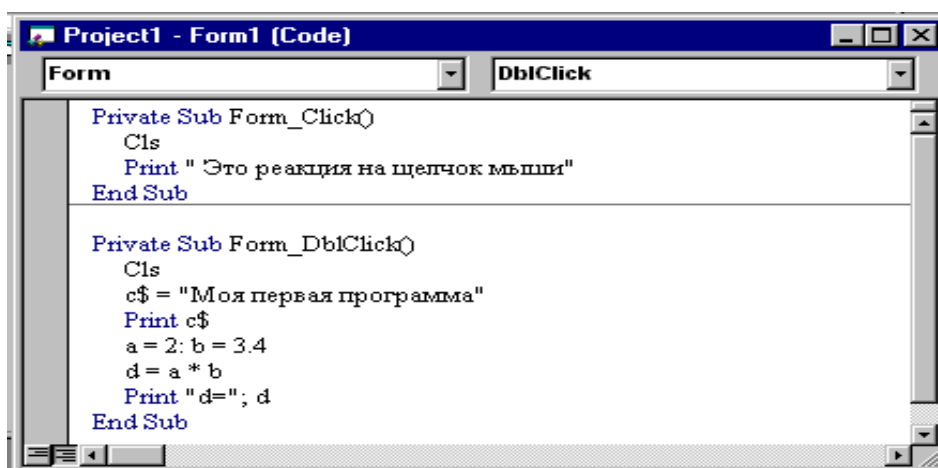


Рис. 1.4. Окно Программы (Code)

**Список объектов** содержит все объекты текущей формы. В их число входит и специальный объект *General*, содержащий общий код, используемый всеми процедурами формы.

**Список процедур** содержит список всех событий, распознаваемых текущим объектом. Если для объекта уже написаны процедуры обработки событий, то они выделяются здесь жирным шрифтом. Если выбрать любой пункт данного списка, то VB выведет соответствующую процедуру либо ее шаблон в окне программы и поместит в нее курсор.

### Окно позиционирования формы (FormLayout)

Данное окно позволяет установить место, где будет размещаться форма при запуске программы.

### Задание

1. Изучите среду разработки проекта.
2. Изучите приемы установки элементов управления на форму и управления размещением элементов управления на форме.

### Порядок выполнения задания

1. Запустите программу VisualBasic: введите команду **Пуск\Программы**, найдите в подменю команды Программы пиктограмму VisualBasic 6.0 и щелкните по ней дважды мышью. Если данной программы нет в меню, найдите ее в компьютере командой и **Пуск\Найти\Файлы и папки**. Имя исполняемого файла – Vb6.exe (могут быть и другие способы запуска Vb6 в зависимости от опыта пользователя и установленного программного обеспечения).

2. Щелкните в окне диалога *NewProject* дважды по пиктограмме *Standard.EXE* для создания стандартного приложения Windows (это окно может не появляться на экране, а сразу появится рабочее окно VisualBasic).

3. Изучите элементы рабочего окна VB (среда разработки программы) и команды меню. Измените размеры и положение формы в окне, перетаскивая его мышью.

4. Закройте окна Проект и Свойства кнопками закрытия окон.

5. Откройте окна Проект и Свойства командами **View\ProjectExplorer** и **View\PropertiesWindow**, соответственно. Найдите в стандартной панели инструментов одноименные кнопки.

6. Переместите окно Проект в нижний правый угол рабочего окна. Переместите окно Проект в первоначальное положение.

7. Закройте панель элементов управления (Toolbox). Выведите панель Toolbox в рабочее окно командой **View\Toolbox**.

8. Откройте новую форму. Присвойте имя, наименование, установите размеры формы и ее положение в проекте.

– Запустите программу командой **Run,Start**. Появится пустая форма.

– Закройте проект. Установите положение формы в проекте с помощью окна FormLayout. Снова запустите проект и проверьте положение формы в окне проекта.

9. Установите на форму пять меток и пять окон ввода (Label, TextBox). Опишите в таблице их свойства и присвойте значения этих свойств элементам управления:

Тип объекта	Имя (Name)	Наименование (Caption)	Положение верхнего левого угла		Размеры	
			сверху (Top)	слева (Left)	ширина (Width)	высота (Height)
Метка (Label)						

Тип объекта	Имя (Name)	Текст (Text)	Положение верхнего левого угла		Размеры	
			сверху (Top)	слева (Left)	ширина (Width)	высота (Height)
Окно ввода (TextBox)						

Можно также добавить такие свойства как цвет, шрифт и т.п.

10. Упорядочите элементы управления на форме с помощью опций команды главного меню **Format**.

11. Установите на форму рамку (Frame). Скопируйте в нее несколько элементов управления с формы, используя контекстное меню, и выровняйте их по вертикали.

12. Запишите в окне программы код, представленный на рисунке 1.4.

13. Добавьте на панель элементов управления сетку *MicrosoftFlexGrid*. Введите команду **Project\Components\Controls**, найдите в списке объект *MicrosoftFlexGridControl 6.0 (SP3)* и щелчком мыши установите напротив него флажок. Щелкните по кнопке Ok.

14. Добавьте к проекту еще одну форму командой **Project\AddForm**.

15. Установите на новой форме объект *MicrosoftFlexGrid*.

16. Создайте новую папку на рабочем диске и сохраните проект в этой папке командой **File\SaveAs**.

17. Откройте новый проект командой **File\NewProject**.

18. Откройте сохраненный ранее проект командой **File\OpenProject**.

19. Выйдите из программы VisualBasic.



### **Указание к выполнению задания**

Выполните последовательно все пункты задания. По ходу выполнения задания оформите отчет, отражая в нем порядок выполнения задания, вводимые команды и результаты выполнения команд. Вводимые команды описывать, например, следующим образом: *Файл\Сохранить*, указать диск, маршрутиция файла.

### **Требования к оформлению отчета**

Отчет должен содержать тему, цель занятия, описание порядка выполнения пунктов задания, ответы на контрольные вопросы.

### **Контрольные вопросы**

1. Как запустить программу VB?
2. Назовите основные элементы рабочего окна VB.
3. Как установить элементы управления на форму?
4. Какая команда используется для управления размещением элементов управления на форме?
5. Как добавить новые элементы управления на панель инструментов Toolbox?
6. Для чего предназначено окно Проект?
7. Каково назначение окна Свойства?
8. Для чего предназначено окно Программа?
9. Как вызвать окно Программа?
10. Где записывается текст программы объекта?
11. Как сохранить программу на диске?
12. Как загрузить существующую программу?

## Лабораторная работа №2. Составление простейших линейных программ с использованием форм

Для того чтобы открыть редактор VBA из приложения Microsoft Office, выберите команду **Сервис, Макрос, Редактор Visual Basic** или нажмите комбинацию клавиш <Alt>+<F11>.

Далее можно воспользоваться меню редактора: выбрать команды **Insert, Module**, еще раз войти в меню **Insert** и выбрать **Procedure**. Затем заполнить поле **Name** диалогового окна (ввести имя своей процедуры), и если программируется функция, то выбрать переключатель **Function** (рис. 2.1), нажать **Ok**.

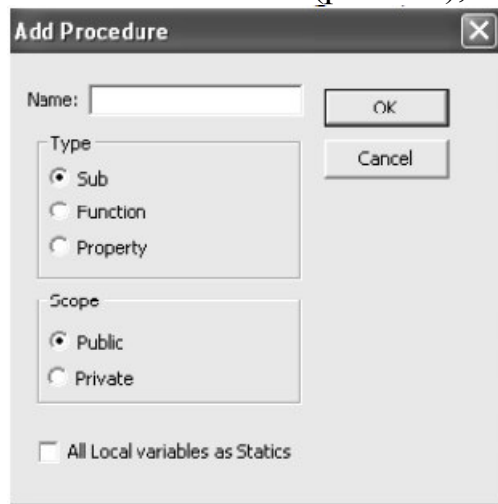



Рис. 2. Вид окна *Добавление процедуры*


На листе созданного модуля появятся стандартные строки начала и конца процедуры:

**Public Sub**ИмяПроцедуры()

**End Sub**

Между ними можно набрать код (инструкции) программы.

Для запуска программы можно выбрать в меню **Run** команду **Run Sub/UserForm** или нажать кнопку  на панели инструментов.

Если в программе обнаружены ошибки, то компилятор выдает сообщение и отладчик отмечает строку с ошибкой желтым цветом. Чтобы снять это выделение, можно нажать кнопку  на панели инструментов. После этого необходимо исправить ошибку и снова запустить программу на выполнение.

Программа (код программы) записывается в окне кода.

Окно кода используется при написании любой программы VBA, будь это код макроса, запуск которого осуществляется при нажатии кнопки в созданной пользователем форме, или подпрограмма.

Под строкой заголовка окна расположены два списка. В первом списке выводятся все объекты модуля, а во втором – список процедур, связанных с выбранным объектом.

Код программы вводится непосредственно в окно кода, так же как текст в любом текстовом редакторе.

Код программы может быть связан с формой – UserForm(программа пишется для соответствующей формы) так и не связан с ней (пример программ на Паскале). В последнем случае программу пишут в окне модуля.

Для того, чтобы получить окно модуля необходимо выполнить следующие действия:

*Вставка → Модуль*

Чаще в VBA имеют дело с формой (UserForm). Чтобы получить окно формы необходимо произвести такие действия:

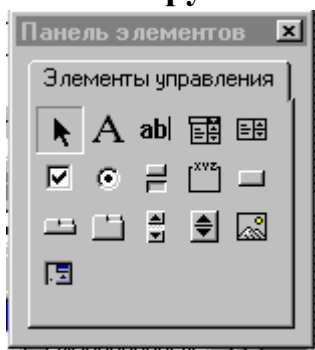
*Вставка → UserForm*

Если программа пишется под форму, то ее код будет включать ряд маленьких подпрограмм. Для каждого события, возникающего в форме необходимо написать процедуру (последовательность совместно выполняемых инструкций, имеющая имя) обработки.

В проекте VBA часто требуется создать собственную форму. Например, может понадобиться задать значения параметров перед выполнением некоторой операции.

В VBA, формы можно создать самостоятельно. Форма – это тоже самое, что и любое диалоговое окно. Панель элементов позволяет разместить ряд элементов управления в форме.

### Инструменты панели элементов



Допускается настройка панели элементов путем добавления к ней страниц или элементов управления с помощью команды *Дополнительные элементы...* из меню Сервис.

При добавлении страницы в нее автоматически вставляется инструмент Выбор объектов.

### Стандартные элементы управления панели элементов:



**Выбор объектов**

Это единственный инструмент на панели элементов, не создающий никаких элементов управления. Он служит для изменения размеров и положения элементов формы.



**Надпись**

Позволяет отобразить в форме неизменяемый текст, например подпись к рисунку.



## Поле

Содержит вводимый и изменяемый пользователем текст.



### Поле со списком

Вставляет объект, являющийся сочетанием списка и поля. Пользователь может либо выбрать нужное значение из списка, либо ввести его в поле.



### Список

Вставляет список выбираемых пользователем элементов. Допускается прокручивание списка, если не все его элементы видны одновременно.



### Флажок

Создает ячейку, которая может быть помечена пользователем, как имеющая значение истина или ложь, а также использующуюся для предоставления выбора нескольких вариантов.



### Переключатель

Используется для предоставления выбора одного варианта из многих.



### Выключатель

Создает кнопку, имеющую два состояния: включено и выключено.



### Группа

Позволяет установить графическую или функциональную группировку элементов управления. Для создания группы следует сначала создать ее рамку, а затем внутри нее создать необходимые элементы.



### Кнопка

Создает кнопку, при нажатии которой выполняется команда.



### Набор вкладок

Позволяет создать несколько страниц в одной и той же области окна или окна диалога.



### MultiPage

Служит для представления нескольких экранов информации в виде единого набора.



### Полоса прокрутки

Создает графический инструмент для быстрого перемещения по длинным спискам элементов или по большим документам, отображающий текущее положение.



### SpinButton

Прокручивающий элемент управления используется совместно с другими элементами для увеличения или уменьшения числовых значений. Допускается его использование для выбора объекта из диапазона значений или из списка элементов.



#### Рисунок

Отображает в форме точечный рисунок, значок или метафайл.

После размещения элементов управления на форме необходимо связать объект на форме с кодом.

В VBA очень просто связать объект с кодом. Для выполнения данной операции:

1. Дважды щелкните по элементу управления в форме. Появляется окно модуля для выбранного объекта. Выберите событие, для которого требуется создать процедуру обработки, в списке, расположенном в верхнем правом углу окна модуля. Введите текст процедуры.

2. Вызвать контекстное меню необходимого объекта правой клавишей мыши и нажать поле *Программа*.

Решение любой задачи имеет три части:

1. Ввод данных;
2. Обработка данных;
3. Вывод результата.

Под вводом данных понимается описание всех переменных, констант и массивов, используемых в программе, а также код, обеспечивающий присвоение этим переменным вводимых данных.

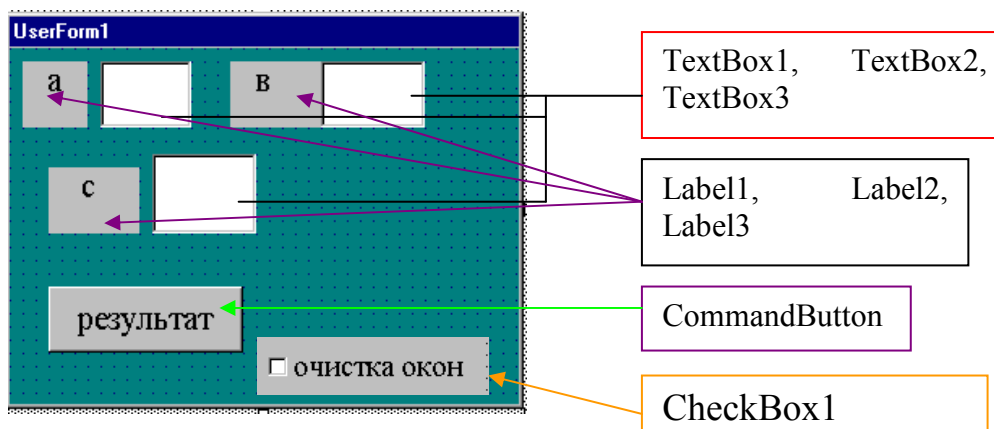
Под так называемой обработкой данных понимается код, состоящий из математических выражений, которые приводят к получению результата.

Вывод результата – это код программы, который позволяет отобразить полученный результат в необходимом виде: на экране (лист excel, форма), на принтере и т.д.

Решим задачу: найдем сумму:  $a + b = c$

#### Порядок выполнения работы:

1. Запустить Microsoft Office Excel.
2. На вкладке Разработчик выбрать Код → VBA.
3. Выполнить команду Insert/UserForm.
4. Поместить на форму элементы, требуемые для решения задачи, с панели элементов, и расположить их нужным образом.



4. Изменить свойства объектов на форме с помощью окна свойств.

Свойство	Значение
Label1.Caption	А
Label2.Caption	В
Label3.Caption	С
CommandButton1	Результат
CheckBox1.Caption	Очистка окон
Для всех объектов свойство .BackColor	По своему вкусу выбрать цвет Из палитры цветов
Для Label1, Label2 ,Label3 Свойство <b>Font</b>	В диалоговом окне “Шрифт”, которое появится после щелчка по Кнопке с изображением трех маленьких точек, расположенной напротив свойства Font в окне свойств, выбрать размер <b>16</b>

5. Написать программный код. Для этого рекомендуется выполнить двойной щелчок по кнопке *результат* и перейти в окно программы, где набрать текст процедуры обработки события Click() для кнопки и для флажка(CheckBox1):

```
Private Sub CheckBox1_Click()
    TextBox1.Text = ""
    TextBox2.Text = ""
    TextBox3.Text = ""
    TextBox3.Visible = False
    TextBox1.SetFocus
    CheckBox1.Value = False
End Sub
```

```
Private Sub CommandButton1_Click()
    Dim a As Integer
    Dim b As Integer
    Dim c As Integer
```

```

a = CInt(TextBox1.Text)
b = CInt(TextBox2.Text)
c = a + b
MsgBox "результатсмотри в TextBox3"
TextBox3.Visible = True
TextBox3.Text = c
EndSub

```

#### Пояснения к программе:

##### 1) **Dim a As Integer**

Эта инструкция описывает переменные как Integer – целые числа от -32768 и до 32767. При попытке присвоить а число, выходящее за пределы этого диапазона, возникает ошибка. При присваивании а дробного числа, выполняется округление.

*Инструкция Dim* – Описывает переменные и выделяет для них память.

2) **CInt** – функция преобразования типов данных (преобразовывает выражение в скобках к типу Integer).

Синтаксис **CInt(выражение)**

##### 3) **c=a+b**

Оператор присваивания ( = ) – вычисляется значение выражения, стоящего справа от знака присваивания, и присваивается переменной, стоящей слева от знака присваивания.

##### 4) **MsgBox "результат смотри в TextBox3"**

Появляется на экране окно сообщений MsgBox, в котором отображается сообщение, записанное в кавычках, и выполнение программы останавливается до тех пор пока не будет нажата кнопка "ОК".

##### 5) **TextBox3.Text = c**

Результат выполнения программы (c) выводится на форму в TextBox3

##### 6) **TextBox1.Text = "", TextBox2.Text = "", TextBox3.Text = ""**

Производится очистка полей TextBox1, TextBox2, TextBox3.

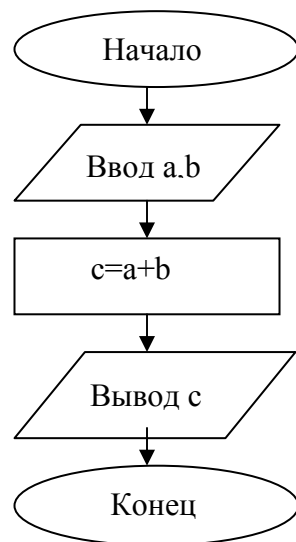
##### 7) **TextBox1.SetFocus**

Устанавливается фокус (курсор) в TextBox1.

##### 8) **CheckBox1.Value = False**

Исчезает галочка у флажка CheckBox1.

**Блок-схема к программе**



### **ЗАДАНИЯ ДЛЯ ВЫПОЛНЕНИЯ**

**Примечание:** используйте для ввода тип данных Double и функцию преобразования типов данных CDBl.

1.  $f(x,y,z) = (x^2 - y^2) / (1 + z + x^2)$ .
2.  $f(x,y,z) = (x + y + z) / (x^2 + y^2 + z^2)$ .
3.  $f(x,y) = x / (1 + y) + y / (1 + x) + 1 / (x + y)$ .
4.  $f(x,y,z) = (x + y + z) / (x * y * z)$ .
5.  $f(a,b,c,x) = a * x^2 + b * x + c$ .
6. Вычисления процентного отношения двух чисел (сколько процентов составляет величина первого от величины второго).
7.  $f(x,y) = (x + y)(x^2 + y^2)(x^3 + y^3)$ .
8.  $f(x,y,z) = (x * y * z) / (x + y^2 + z^3)$ .
9.  $f(x,y,z) = x / (y + z) + y / (x + z) + z / (x + y)$ .
10.  $f(x,y,z) = x^3 + y^4 + z^5$ .
11.  $f(x,y,z) = x * y / z + y * z / x + z * x / y$ .
12.  $f(x,y,z) = (x^3 + 1) / (y + z^2)$ .
13.  $f(x,y) = (x^2 + y^2) / (5 * x * y)$ .
14.  $f(x,y) = 3 * (x + y) / x * (x + y)$ .



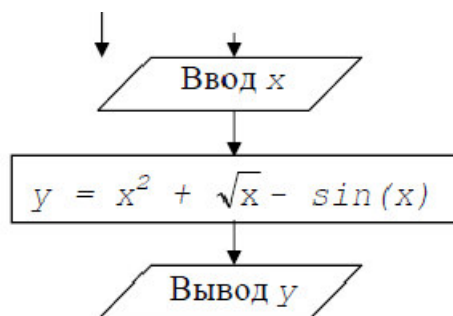
## Лабораторная работа №3. Составление простейших линейных программ с использованием форм

**Задача 1.** Найти значения переменной  $y$ , заданные формулой

$$y = x^2 + x - \sin(x)$$

для трех различных значений  $x$ .

**Решение.** Вход алгоритма: значение переменной  $x$ , выход: значение переменной  $y$ . Алгоритм состоит из трех шагов: 1) ввод значения  $x$ , 2) вычисление  $y$  для этого значения, 3) вывод полученного результата. Эти три шага следует также выполнить трижды – для каждого нового значения  $x$ . Блок-схема алгоритма для одного значения  $x$ :



Для иллюстрации возможных методов ввода будем задавать  $x$  различными способами: присваиванием, вводом с клавиатуры, вводом из ячейки листа Excel.

Определяем переменные, участвующие в программе: их имена и типы. Пусть это будут  $x$  и  $y$  вещественного типа `Single` (целый тип `Integer` для убрать нельзя, т.к. операция извлечения корня и функция  $\sin(x)$  могут дать в качестве результата вещественные числа). В начале программы следует поставить описания этих переменных.

Вывод реализуем оператором `MsgBox`. Его используем в двух режимах – просто вывод числа и вывод числа с текстом. Текст в этом случае нужно взять в двойные кавычки и соединить его с числом операцией `&` склейки строк. Аргумент оператора `MsgBox` имеет строковый тип (`String`), при выводе числового значения происходит автоматическое преобразование типов (здесь – из `Single` в `String`).

В программе участвуют встроенные функции – извлечение корня и вычисление синуса. В VBA это `Sqr(x)` и `Sin(x)`. Отметим, что извлечение корня из числа эквивалентно операции возведения этого числа в дробную степень (здесь – в степень  $1/2$ ).

```
Public Sub Task1()  
    Dim x As Single, y As Single 'описываем переменные  
    x = 2.4 'вводим x присваиванием значения  
    y = x^2 + Sqr(x) - Sin(x) 'вычисляем y, используя функцию sqr  
    MsgBox y 'выводим значение y без текста  
    x = InputBox("Введите x") 'вводим x с клавиатуры  
    y = x^2 + x^(1/2) - Sin(x) 'используем возведение в дробную степень  
    MsgBox "y=" & y 'выводим значение y с текстом "y="   
    x = Cells(1, 1) 'вводим x из ячейки A1 листа Excel  
    y = x ^ 2 + Sqr(x) - Sin(x)
```

```
MsgBox "y=" & y
EndSub
```

Перед тем, как запустить эту программу на счет, необходимо ввести какое-нибудь значение в ячейку A1 активного листа, иначе оператор  $x=Cells(1,1)$  присвоит  $x$  значение 0.

Функция  $Cells(i,j)$ , где  $i$  – номер строки,  $j$  – номер столбца.

В этой программе есть недостаток: она не проверяет, что введенное значение положительно ( $x > 0$ ). Если пользователь программы введет отрицательное значение, то операции  $x^{(1/2)}$  и  $Sqr(x)$  приведут к ошибке, система выдаст сообщение о неверном аргументе этих операций. Программа будет гораздо технологичней, если после каждого ввода поставить защиту, например: проверку значения, выдачу сообщения и выход из процедуры:

```
if x < 0 Then
MsgBox "введено отрицательное значение переменной x"
Exit Sub
End if
```

### ***Задача 2. Обменять значения двух переменных.***

**Решение.** Пусть имена переменных  $x$  и  $y$ . Задача состоит в том, чтобы переменная  $x$  получила значение переменной  $y$  и наоборот:  $y$  получил значение переменной  $x$ . Типичное *неправильное* решение задач с обменом значений:

```
x = y 'неверно!
y = x
```

В этом случае переменная  $x$  получит значение  $y$ , но переменная  $y$  в присваивании  $y=x$  получит уже новое значение  $x$ , приобретенное в результате первого оператора  $x=y$ . Таким образом,  $y$  приобретет свое прежнее значение. Необходимо помнить, что в любых операциях обмена значений переменных *требуется посредник* – дополнительная переменная для хранения промежуточного результата (старого значения одной из исходных переменных). Пусть такой переменной-посредником будет  $rab$ . Тогда правильное решение задачи:

```
rab = x
x = y
y = rab
```

Программа целиком:

```
Sub Обмен()
Dim x As Integer, y As Integer, rab As Integer
x = 10
y = 7
rab = x
x = y
y = rab
MsgBox (x)
MsgBox (y)
End Sub
```

### ***Задачи для самостоятельной работы:***

1. Найти значение переменной  $y$  при различных значениях  $A, B, C$ , вводимых с листа Excel, и значениях  $x, z$ , вводимых с клавиатуры:

$$\text{a) } y = \frac{A - 3B}{A + B}(A + C)\cos^2 x - z; \quad \text{b) } y = \frac{x A}{B} \sqrt{x^C + 1} \frac{1}{A^x + 1};$$

2. Задать длину ребра куба. Найти объём куба и площадь его поверхности.

3. Задать три действительных положительных числа. Найти среднееарифметическое и среднее геометрическое этих чисел.

## Лабораторная работа №4. Алгоритм формирования источника информации для поля со списком

### Постановка задачи

Заполнить лист “Сотрудники” с помощью пользовательской формы “Карточка сотрудника”. В пользовательской форме спроектировать поле со списком специальностей, которые находятся на листе “Профессии”.

Эта задача реализует алгоритм заполнения поля со списком в пользовательской форме и запись на лист Excel данных, введенных в пользовательскую форму.

**Поле со списком (ComboBox)** – это элемент управления, который применяется для хранения списка значений. Список значений заранее создается на листе Excel и программно формируется для использования его в пользовательской форме.

Особенность нашей задачи заключается в том, что список специальностей, занесенных на лист Excel, может все время дополняться. Поэтому, чтобы узнать диапазон ячеек, заполненных специальностями, и программно сформировать поле со списком в пользовательской форме, следует использовать определенный алгоритм.

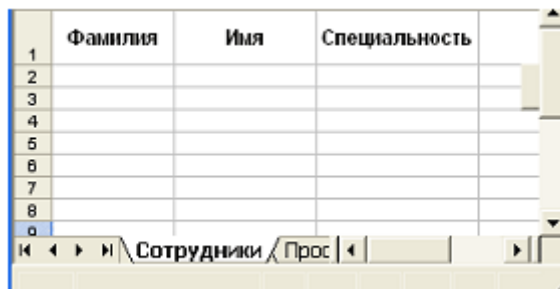
Решение задачи состоит из двух этапов:

- Вызов формы на экран и формирование поля со списком специальностей;
- Запись на лист Excel введенной информации (при нажатии на кнопку “ОК”).

### Порядок работы

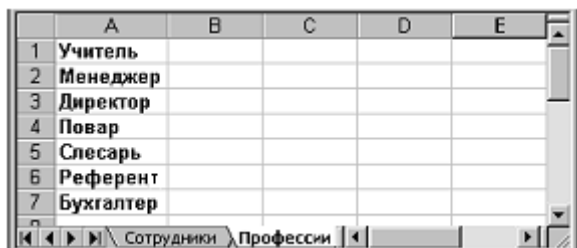
1. Переименуйте Лист1 в лист “Сотрудники”, а Лист2 — в “Профессии”.

2. Подготовьте на листе “Сотрудники” шапку таблицы, в которую через пользовательскую форму будет заноситься информация.



	Фамилия	Имя	Специальность
1			
2			
3			
4			
5			
6			
7			
8			
9			

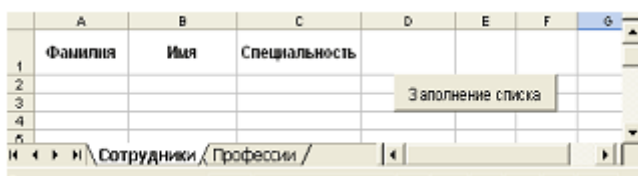
3. Составьте на листе “Профессии” список специальностей.



	A	B	C	D	E
1	Учитель				
2	Менеджер				
3	Директор				
4	Повар				
5	Слесарь				
6	Референт				
7	Бухгалтер				

4. Спроектируйте форму UserForm1 и назовите ее “Карточка сотрудника”.

5. Подготовьте на листе “Сотрудники” кнопку “Заполнение списка”. Для этого на вкладке Разработчик перейдите в Режим конструктора, нажмите кнопку Вставить → Элементы ActiveX → Кнопка. При нажатии на эту кнопку должна появляться подготовленная пользовательская форма, позволяющая создавать поле со списком специальностей. Создадим событийную процедуру нажатия на кнопку.



Алгоритм формирования источника информации для поля со списком в пользовательской форме:

**A.** Определите количество строк, заполненных специальностями, на листе “Профессии”.

Количество непустых ячеек в указанном диапазоне подсчитывает функция рабочего листа **CountA**. В качестве диапазона мы будем рассматривать весь столбец A:

*Range("A:A").*

Так как это функция, то необходимо указать, какой переменной мы присвоим вычисленное значение. В нашей задаче мы назвали эту переменную “список”.

*Список =*

*Application.CountA(Sheets("Профессии").Range("A:A"))*

В результате работы функции наша переменная “список” примет значение 7.

**B.** Сформируйте диапазон списка специальностей.

Мы знаем, что адрес начальной ячейки диапазона на листе “Профессии” – A1. Адрес конечной ячейки диапазона получится путем “склеивания” имени столбца A и значения, которое хранится в переменной “список”. Но переменная “список” объявлена как тип Integer, а “склеивание” можно применять только к строковым переменным. Поэтому мы должны к переменной “список” применить функцию **CStr**: эта функция меняет тип данных у переменных. После этого можно провести “склеивание” и присвоить полученный диапазон новой переменной.

*Д\_списка = "A1:A" & CStr(cсписок)*

В результате работы новой переменной Д\_списка будет присвоен диапазон A1:A7.

**C.** Присвойте сформированному диапазону имя “Специальности”.

Используем свойство **Name** объекта **Range**, расположенного на листе “Профессии”.

```
Sheets("Профессии").Range(Д_списка).Name = "Специальности"
```

В результате работы диапазону A1:A7 на листе “Профессии” будет присвоено имя “Специальности”.

**D.** Присоедините к полю в пользовательской форме список специальностей.

**ComboBox1** – объект “поле со списком” в пользовательской форме. **RowSource** – свойство объекта “источник-строка”. То есть источником для формирования поля со списком является диапазон ячеек с именем “Специальности”.

```
.ComboBox1.RowSource = "Специальности"
```

Программа формирования источника информации для поля со списком в пользовательской форме получится такая:

```
Private Sub CommandButton1_Click()  
Dim список As Integer  
список = Application.CountA(Sheets("Профессии").Range("A:A"))  
'Определим диапазон списка специальностей  
Д_списка = "A1:A" & CStr(список)  
'Присвоим имя диапазону списка специальностей  
Sheets("Профессии").Range(Д_списка).Name = "Специальности"  
'Очистим ячейки  
With UserForm1  
.TextBox1.Text = ""  
.ComboBox1.Text = ""  
'Введем список для поля со списком  
.ComboBox1.RowSource = "Специальности"  
'Выведем пользовательскую форму на экран  
.Show  
EndWith  
EndSub
```

6. Создайте процедуру для записи (при нажатии на кнопку “ОК”) данных о сотруднике из пользовательской формы на лист “Сотрудники”.

```
Private Sub CommandButton1_Click()  
Dim фамилия As String, специальность As String  
Dim строка As Integer, имя As String  
'строка - номер последней заполненной строки на листе "Сотрудники"  
'Определим номер последней заполненной строки  
строка = Application.CountA(Sheets("Сотрудники").Range("A:A"))  
With UserForm1  
'переменным "фамилия" и "имя" присвоим содержимое поля TextBox1 и TextBox2  
фамилия = .TextBox1.Text  
имя = .TextBox2.Text  
'переменной "специальность" присвоим содержимое поля ComboBox1  
специальность = .ComboBox1.Text  
End With  
With Sheets("Сотрудники")  
.Cells(строка + 1, 1) = фамилия  
.Cells(строка + 1, 2) = имя  
.Cells(строка + 1, 3) = специальность
```

EndWith

'Очистим содержимое ComboBox1.Text, TextBox1.Text и TextBox2.Text

UserForm1.ComboBox1.Text = ""

UserForm1.TextBox1.Text = ""

UserForm1.TextBox2.Text = ""

**End Sub**

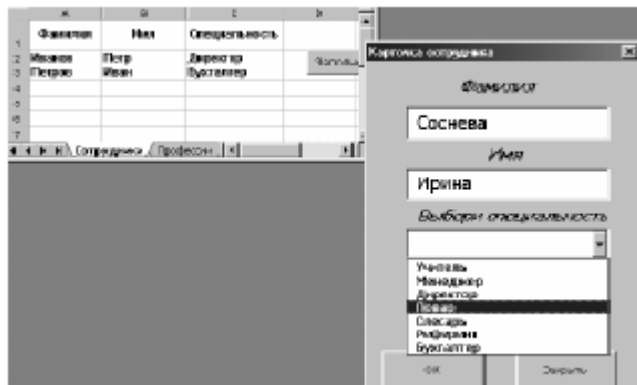
7. Запишите процедуру закрытия формы.

**Private Sub** CommandButton2\_Click()

UserForm1.Hide

**End Sub**

8. Проверьте вашу работу и сохраните ее.



## Лабораторная работа №5. Работа с матрицей. Формирование размерности матрицы с помощью запроса

### Постановка задачи

На листе Excel подготовить матрицу, размерность которой формируется с помощью запроса о количестве строк и столбцов. При этом количество строк и столбцов не должно превышать десяти. Элементы матрицы вводятся с применением датчика случайных чисел в диапазоне от 0 до 9. Спроектировать пользовательскую форму “Операции с матрицей”. В форме предусмотреть кнопки “Ввод матрицы”, “Вычислить”, “Очистить лист”, “Выход”.

При нажатии на кнопку “Ввод матрицы” формируется матрица на листе Excel. При нажатии на кнопку “Вычислить” должна подсчитываться сумма элементов матрицы, производиться поиск наибольшего элемента или происходить замена четных элементов на нечетные. Какая из этих операций будет выполняться, необходимо указать с помощью элемента **Переключатель (OptionButton)**.

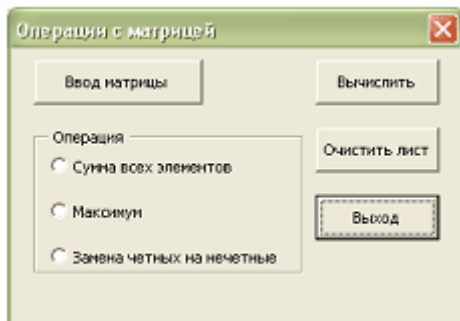
При нажатии на кнопку “Очистить лист” с листа “Матрицы” должно происходить удаление самой матрицы и всех результатов работы с ней.

Кнопка “Выход” закрывает форму.

Вызывать пользовательскую форму “Операции с матрицей” должна кнопка “Работа с матрицей” на листе Excel.

### Порядок работы

1. Спроектируйте пользовательскую форму UserForm1 “Операции с матрицей”.



2. Объявите глобальные переменные для пользовательской формы:

```
' Включаем проверку переменных VBA
```

```
OptionExplicit
```

```
' Глобальные переменные
```

```
Dim m As Integer, n As Integer, i As Integer, j As Integer
```

3. Напишите программы для кнопок.

#### Кнопка “Очистить лист”

```
Private Sub CommandButton2_Click()
```

```
For i = 1 To 30
```

```
For j = 1 To 30
```

```
Worksheets("Матрица").Cells(i, j).Value = ""
```

```
Next j
```

```
Next i
```



**End Sub**

### **Кнопка “Выход”**

**Private Sub** CommandButton1\_Click()

Userform1.Hide

**EndSub**

### **Кнопка “Вводматрицы”**

**PrivateSub** CommandButton4\_Click()

m = InputBox("Введитеколичествостолбцовматрицы", "Ввод")

If m > 10 Then

MsgBox "Количество столбцов более 10не обрабатываю!", 48, "Ошибка!"

GoTo metka

End If

n = InputBox("Введите количество строк", "Ввод")

If n > 10 Then

MsgBox "Количество строк более 10 не обрабатываю!", 48, "Ошибка!"

GoTo metka

End If

For i = 1 To m

For j = 1 To n

Randomize

Cells(i, j) = Int(10 \* Rnd)

Next j

Next i

Worksheets("Матрица").Cells(1, 1).Value ="Матрица"

For i = 1 To m

For j = 1 To n

Worksheets("Матрица").Cells(i + 1, j).Value = Cells(i, j)

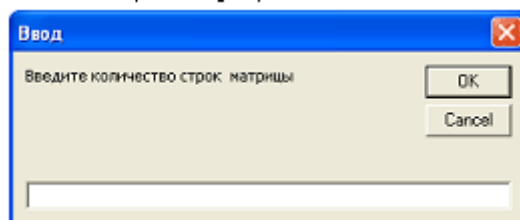
Next j

Next i

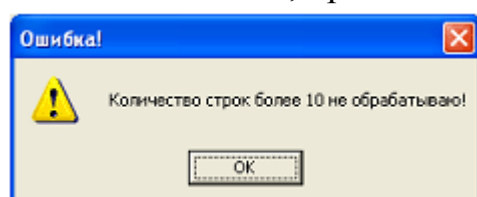
metka:

**EndSub**

При нажатии на кнопку “Ввод матрицы” должно появляться окно сообщения для ввода количества строк и столбцов матрицы.



Если ввести число, превышающее 10, появится окно сообщения:



## Кнопка “Вычислить”

'выполнение действий над матрицей

**Private Sub** CommandButton6\_Click()

Dim MAX As Integer

Dim d As Integer

Dim SUM As Integer

Dim flag As Integer

'определение суммы

SUM = 0

flag = 0

If OptionButton1.Value = True Then

For i = 1 To m

For j = 1 To n

SUM = SUM + Worksheets("Матрица").Cells(i, j)

Next j

Next i

Worksheets("Матрица").Cells(11, 1).Value = "Суммаэлементов ="

Worksheets("Матрица").Cells(12, 1).Value = SUM

MsgBox "=" & SUM, 48, " Суммаэлементов"

GoTo metka

End If

' определение максимума

If OptionButton2.Value = True Then

MAX = Worksheets("Матрица").Cells(2, 1).Value

For i = 1 To m

For j = 1 To n

If MAX < Worksheets("Матрица").Cells(i, j) Then

MAX = Worksheets("Матрица").Cells(i, j)

End If

Next j

Next i

Worksheets("Матрица").Cells(11, 1).Value = "Максимальныйэлемент"

Worksheets("Матрица").Cells(12, 1).Value = MAX

MsgBox "=" & MAX, 48, "Максимальныйэлемент"

GoTo metka

End If

'замена четных

If OptionButton3.Value = True Then

For i = 1 To m

For j = 1 To n

d = Worksheets("Матрица").Cells(i, j)

If d / 2 = Int(d / 2) Then

Worksheets("Матрица").Cells(i, j) = d + 1

flag = 1

End If

Next j

Next i

If flag = 1 Then

MsgBox "Замена произошла", 64, "Замена"

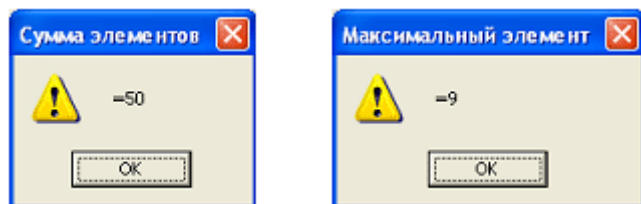
Else

MsgBox "Замены нет, все элементы нечетные", 64, "Замена"

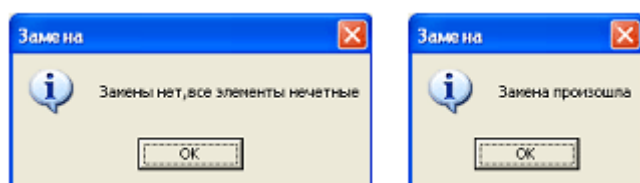
End If

```
End If  
metka:  
EndSub
```

Результат подсчета суммы элементов или поиска максимального элемента матрицы выводится в ячейку A12, дополнительно появляется окно сообщения:



При выборе операции замены элементов окно сообщения может быть таким:



4. Спроектируйте на листе “Матрица” кнопку “Работа с матрицей”, которая вызывает пользовательскую форму “Операция с матрицей”. Для этого выберите вкладку Разработчик → Вставить → Элементы управления формы → Кнопка.

```
Private Sub Запуск_Click()  
Userform1.Show  
End Sub
```

5. Сохраните свою работу.

## Лабораторная работа №6. Процедуры и модули. Разнесение чисел по листам

**Модуль**— это совокупность объявлений (описательная часть) и процедур, хранящихся как единое целое.

**Процедура**— любая совокупность кода VBA, рассматриваемая как единое целое. Как правило, процедура состоит из операторов, выполняющих какую-либо задачу или вычисляющих значение. Каждая процедура идентифицируется своим уникальным именем. Часто выполнение процедуры является реакцией на какое-либо *событие*. В этом случае говорят, что процедура *обрабатывает событие*.

### Создание процедур

Создание первой процедуры требует выполнения двух последовательных шагов:

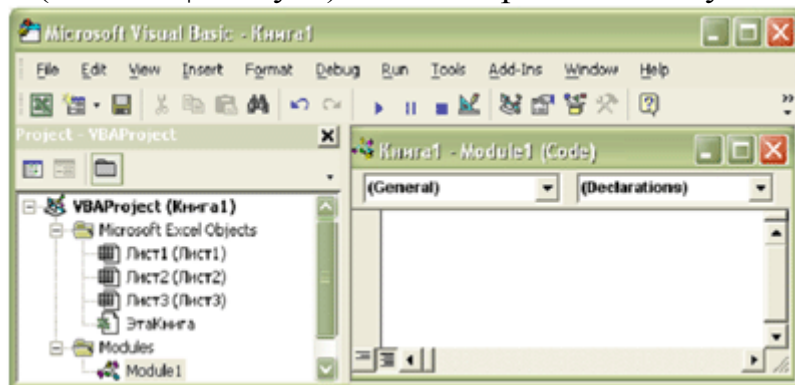
- сначала надо вставить модуль в рабочую книгу;
- затем в этом модуле создать процедуру.

Для любого создаваемого приложения надо создать свой модуль. Приложение может содержать несколько модулей, но это не обязательно.

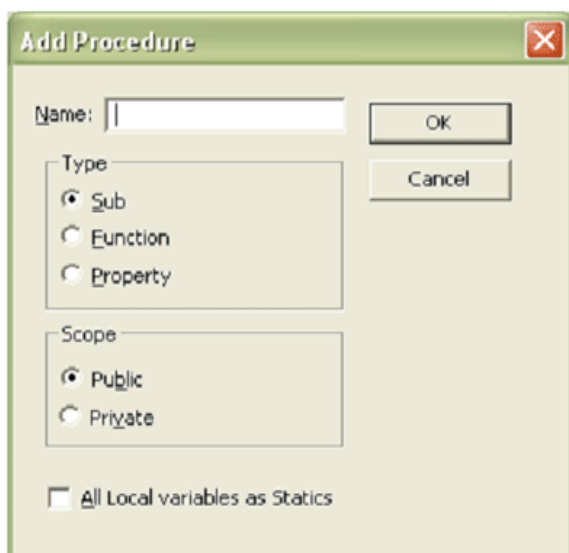
1. Откройте новую рабочую книгу.

2. Выполните команду **Сервис | Макрос | Редактор VisualBasic**. Откроется окно редактора VisualBasic.

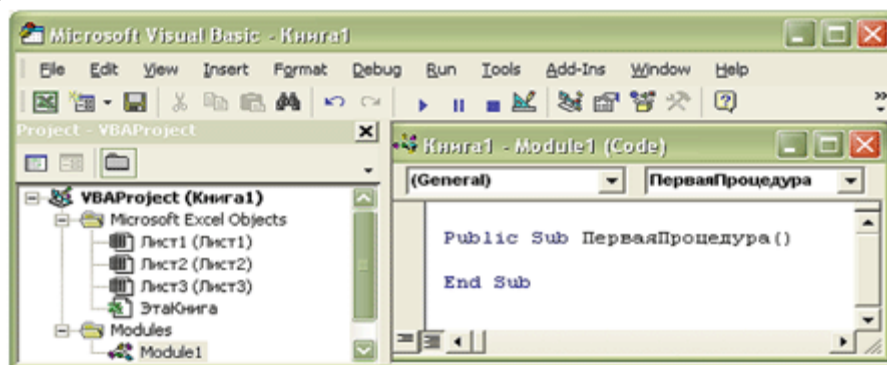
3. Откройте проект VBAProject (Книга1). В меню выберите команду **Insert | Module** (Вставка | Модуль). В ваше приложение будет добавлен модуль 1.



4. Выполните команду **Insert | Procedure** (Вставка | Процедура). Откроется диалоговое окно AddProcedure (Добавить процедуру).



5. Введите имя процедуры. В секции Type (Тип) установите переключатель Sub (Подпрограмма). Щелкните на кнопке “ОК”. Новая процедура будет добавлена в модуль.



В пустой строке тела процедуры находится текстовый курсор, предлагая начать ввод кода процедуры.

VBA требует соблюдения определенных правил при создании имен процедур.

- Первые три символа имени должны быть буквами.
- Имя может содержать буквы, цифры и знаки подчеркивания.
- Имя не может содержать пробелы, точки, запятые, восклицательные знаки и символы “@”, “&”, “\$”, “#”.
- Имя не должно содержать более 255 символов.

### Выполнение процедуры

После создания процедуры вы можете ее сразу выполнить. Для этого существует несколько путей:

1. Выбрать команду меню **RunSub | UserForm** (Выполнить подпрограмму | экранную форму) из меню Run (Выполнить) или щелкнуть на кнопке **RunSub | UserForm** стандартной панели инструментов.

2. Нажать клавишу **F5**.

3. Создать на любом листе Excel автофигуру и привязать к ней подготовленную процедуру.

### Переменные

Переменная – это место для хранения значений. Переменная содержит данные, которые могут изменяться в процессе выполнения программы.

### **Описание данных начинается с оператора DIM**

Имя переменной должно начинаться с буквы и может содержать буквы, цифры и другие символы. Имя не может содержать пробелы, точки, запятые, восклицательный знак и символы “@”, “&”, “\$”, “#”. Имя не должно содержать более 255 символов.

Тип переменных в операторе **Dim** можно не указывать. Тогда VBA применит тип данных по умолчанию (Variant). Кажется, что это удобно, но все же лучше тип данных определять. По нескольким причинам. Во-первых – в целях экономии ресурсов памяти. Ни один тип не требует 16 или 22 байтов для сохранения значений переменных. Во-вторых, VBA по-разному обрабатывает данные разных типов. Поэтому, не объявляя тип данных, вы можете получить результат, отличающийся от желаемого. И, наконец, от типов данных в VBA зависит время выполнения процедур.

Пример определения переменных:

***Dim A As Integer, B As Byte, C As String***

Переменная A определена как целое число (не больше 32 767 и не меньше –32 768); переменная B определена как целое неотрицательное число (не больше 255), а в переменной C может храниться текстовая информация.

### **Использование констант**

Как мы уже узнали, переменные используются для хранения данных, которые могут изменяться в процессе выполнения процедуры. Если надо хранить постоянную информацию, не изменяющуюся при выполнении процедуры, то применяются константы. Для объявления констант и их значений используется оператор **Const**, имеющий следующий синтаксис:

***Const Имя\_константы As тип\_данных = значение***

При объявлении констант используются те же типы данных, что и при объявлении переменных.

Пример объявления константы:

***Const Годы\_учебы As Byte = 11***

### **Программа на VBA — это последовательность операторов**

Для того чтобы сделать программу легкочитаемой, используют **оператор комментариев**. В языке VBA существуют два способа ввода комментариев: применение апострофа ('), который можно поставить в любом месте строки, и зарезервированное слово **Rem** вместо апострофа. При этом комментируется текст до конца строки.

**Оператор With/Endwith** избавляет программиста от большого количества повторений имени одного и того же объекта.

Синтаксис:

***With объект***

***Оператор 1***

***Оператор 2***

.....

***Оператор N***

### ***Endwith***

Например, вместо последовательности операторов

```
UserForm1.TextBox1.Text = Date
```

```
UserForm1.TextBox2.Text = " "
```

```
UserForm1.Label1.Caption = " "
```

```
UserForm1.Label2.Caption = "Название"
```

можно записать так:

***With*** UserForm1

```
.TextBox1.Text = Date
```

```
.TextBox2.Text = " "
```

```
.Label1.Caption = " "
```

```
.Label2.Caption = "Название"
```

### ***Endwith***

## **Управляющие структуры VBA**

Управляющие структуры определяют последовательность выполнения программы. Без них все операторы программы будут выполняться слева направо и сверху вниз. Однако иногда требуется многократно выполнить некоторый набор инструкций либо решить задачу по-другому, в зависимости от значения переменных или параметров, заданных пользователем во время выполнения. В этих случаях помогают конструкции управления и циклы.

VBA поддерживает следующие конструкции принятия решений:

***If ... Then***

***If ... Then ... Else***

***Select Case***

### **Конструкция *If ... Then***

Конструкция ***If ... Then*** применяется, когда необходимо выполнить один или группу операторов в зависимости от некоторого условия. Синтаксис этой конструкции позволяет задавать ее в одной строке или в нескольких строках программы:

**1-й способ:**

***If*** условие ***Then*** выражение

**2-й способ:**

***If*** условие ***Then***

выражение

***EndIf***

Обычно условие является простым сравнением, но оно может быть любым выражением с вычисляемым значением. Если условие истинно, то выполняются все выражения, стоящие после ключевого слова ***Then***.

Следующие два оператора эквивалентны:

***If***  $A \geq B$  ***Then***  $A = A * 2$

***If***  $A \geq B$  ***Then***

$A = A * 2$

***EndIf***

Заметим, что синтаксис оператора **If ... Then** для одной строки не использует оператор **EndIf**. Чтобы в случае истинности условия выполнить последовательность операторов, следует использовать блоковую конструкцию **If ... Then ... End If**.

```
If A >= B Then
```

```
A = A * 2
```

```
Sheets("Задача").Range("B2") = A
```

```
Rem На листе Задача в ячейку B2 поместить значение A
```

```
EndIf
```

Если условие ложно, то операторы после ключевого слова **Then** не выполняются, а управление передается на следующую строку (или строку после оператора **EndIf** в блочной конструкции).

**Конструкции If ... Then ... Else и If ... Then ... ElseIf**

Конструкция **If ... Then ... Else** определяет несколько блоков операторов, один из которых будет выполняться в зависимости от условия:

```
If условие Then
```

```
Выражение 1
```

```
...
```

```
Else
```

```
Выражение 2
```

```
...
```

```
EndIf
```

Сначала проверяется условие. Если оно истинно, VBA выполняет соответствующий блок операторов и затем передает управление инструкции, следующей за оператором **EndIf**. В противном случае выполняется блок оператора **Else**.

Конструкция **If ... Then ... ElseIf** в действительности всего лишь специальный случай конструкции **If ... Then ... Else**, который применяется в случае вложенности таких конструкций. Рассмотрим пример сравнения двух чисел

```
Sub Задача1()
```

```
Dim Rez As String
```

```
a = (InputBox("Введи a", "Ввод данных", 0))
```

```
Dim a As Single, b As Single
```

```
Ввод данных", 0))
```

```
b = (InputBox("Введи b", "Ввод данных", 0))
```

```
If a < b Then
```

```
Rez = "a < b"
```

```
ElseIf a = b Then
```

```
Rez = "a = b"
```

```
Else Rez = "a > b"
```

```
End If
```

```
Msgbox Rez, 64, "Информация"
```

```
End Sub
```

В конструкцию **If ... Then** можно добавить любое число блоков **ElseIf**. Однако при большом количестве блоков **ElseIf** конструкция **If ... Then** станет очень громоздкой и неудобной. В подобной ситуации следует применять другую конструкцию принятия решения – **SelectCase**.

**Конструкция SelectCase**



Конструкция **SelectCase** является альтернативой конструкции **If ... Then ... Else**: она делает код легче читаемым при наличии нескольких вариантов выбора.

Конструкция **SelectCase** работает с единственным проверяемым выражением, которое вычисляется один раз при входе в эту конструкцию. Затем VBA сравнивает полученный результат со значениями, задаваемыми в операторах **Case**, до совпадения.

```
SelectCase проверяемое_выражение  
[Case список_выражений 1  
[блок_операторов 1]]  
[Case список_выражений 2  
[блок_операторов 2]]  
...  
[CaseElse  
[блок_операторовn]]  
EndSelect
```

Каждый *список выражений* может содержать одно или более значений. В этом случае они отделяются запятыми. Каждый *блок операторов* может содержать несколько операторов или ни одного. Если окажется, что *проверяемое выражение* соответствует значениям из нескольких операторов **Case**, то выполняются операторы, совпадающие с первым оператором **Case** из всех найденных соответствий. VBA выполняет блок операторов **CaseElse** (заметим, что он необязателен), если не найдено ни одного соответствия проверяемого значения выражения и значений из всех списков операторов **Case**.

Еще раз отметим, что конструкция **SelectCase** вычисляет выражение только один раз при входе в нее, тогда как в конструкции **If ... Then ... Else** вычисляются различные выражения для каждого оператора **ElseIf**. Конструкцию **If ... Then ... Else** можно заменить конструкцией **SelectCase**, только если оператор **If** и каждый оператор **ElseIf** вычисляют одно и то же выражение.

### Операторы цикла

Операторы цикла предназначены для программирования повторяющихся фрагментов, т.е. для реализации циклических алгоритмов.

Существуют две разновидности операторов цикла: оператор цикла с фиксированным числом повторений и операторы цикла с переменным числом повторений, зависящим от условий.

#### Оператор цикла **For** (фиксированное число повторений)

Синтаксис:

```
For переменная = M1 To M2 [Step M3]  
операторы  
Next
```

– где M1, M2, M3 – выражения. Оператор цикла повторяет выполнение группы операторов, пока переменная (счетчик) изменяется от начального значения M1 до конечного M2 с указанным шагом M3. Если шаг не указан, то он полагается равным 1.

**Пример:**

Sum = 0

```

For I = 1 To 31
Sum = Sum + Sheets("Температура").Cells(I,1)
Next
ST = Sum/31

```

В приведенном примере предполагается, что на листе “Температура” в первом столбце записаны показатели температур за июль месяц. Надо рассчитать среднюю температуру за месяц.

**Оператор цикла While** (переменное число повторений)

*Синтаксис:*

**Do While** условие

операторы

**Loop**

Все операторы будут выполняться между **Do While** и **Loop** до тех пор, пока условие будет истинным. Если при входе в цикл условие ложно, то операторы выполняться не будут.

**Пример:**

**Rem** Удвоение числовых переменных массива А с четными номерами индексов и вывод на лист "Цикл".

```

Dim A(10) As Byte, i As Byte, j As Byte

```

**Rem** Массив надо заполнить

```

i = 0

```

```

j = 0

```

```

Do While i < 10

```

```

j = j + 1

```

```

i = i + 2

```

```

A(i) = A(i) * 2

```

```

Sheets("Цикл").Cells(j, 1) = A(i)

```

```

Loop

```

## Алгоритмы обработки информации на листе Excel

Рассмотрим основные алгоритмы обработки информации: нахождение суммы значений диапазона ячеек, определение количества элементов в диапазоне ячеек, обладающих заданными свойствами (счетчик), определение максимального и минимального значений в диапазоне ячеек.

### Задание

1. На Листе1 (Числа) в ячейки A1–A20 занести случайным образом значения из интервала (–50; 50).

2. На Лист1 (Числа) в ячейку C1 записать “Количество +”, а в ячейку D1 поместить подсчитанное значение с количеством положительных чисел.

3. На Лист1 (Числа) в ячейку C2 записать “Количество –”, а в ячейку D2 поместить подсчитанное значение с количеством отрицательных чисел.

4. На Лист1 (Числа) в ячейку C3 записать “Количество 0”, а в ячейку D3 поместить подсчитанное значение с количеством чисел, равных нулю.

5. На Лист2 (Положительные) в ячейку B1 записать “Положительные” и, начиная с ячейки B2, в столбик поместить все положительные числа.

6. На Лист3 (Отрицательные) в ячейку C1 записать “Отрицательные” и, начиная с ячейки D1, в строку поместить все отрицательные числа.
7. Создать кнопку “Количество” на листе “Числа”.
8. Создать кнопку “Перенос” на листе “Числа”.
9. Создать кнопку “Очистить” на листе “Положительные”.
10. Создать кнопку “Очистить” на листе “Отрицательные”.

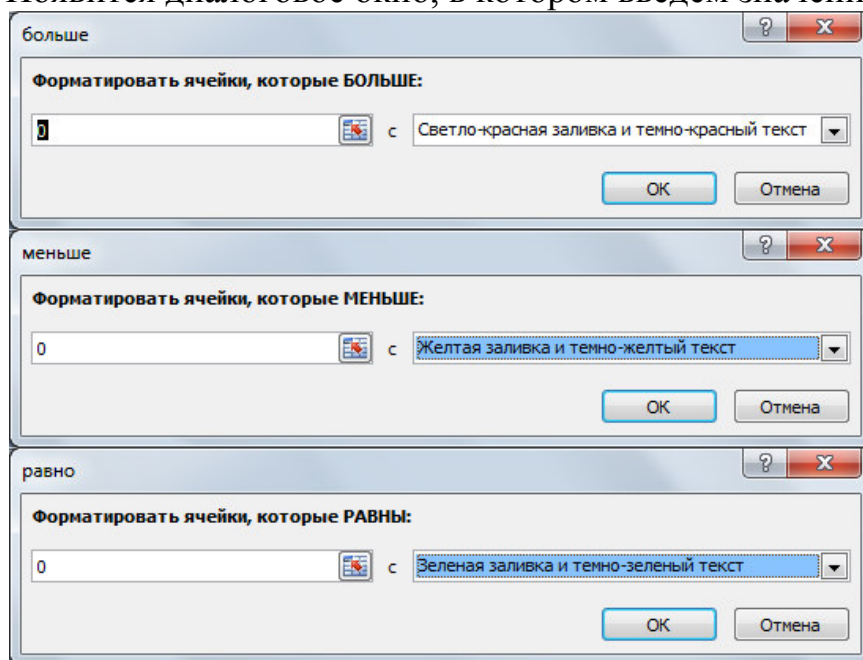
### План работы

1. Переименуйте: Лист1 в “Числа”, Лист2 в “Положительные”, Лист3 в “Отрицательные”.

2. Примените к столбцу А (лист “Числа”) условное форматирование. После заполнения диапазона ячеек числами к положительным числам будет применяться такой формат: полужирный курсив, красный цвет; к отрицательным – полужирный курсив, синий цвет; нулевые значения — полужирный курсив, зеленый цвет. Для этого:

- Выделите столбец А;
- Выполните команду **Формат | Условное форматирование**.

Появится диалоговое окно, в котором введем значения по образцу.



3. Перейдите в редактор VBA.

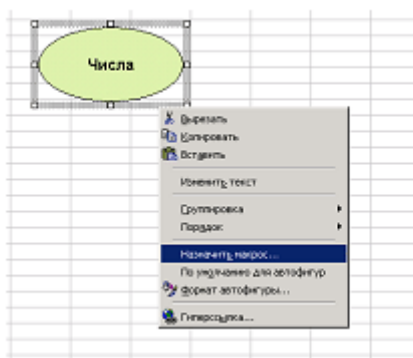
4. Создайте модуль с помощью команды **Insert | Module**.

5. Создайте в модуле процедуру с помощью команды **Insert | Procedure**. Присвойте имя процедуре “Числа”.

6. Напишите текст программы для занесения чисел на лист.

```
Public Sub Числа()
Dim I As Integer
Randomize Timer
For I = 1 To 20
Sheets("Числа").Cells(I, 1) = Int(Rnd * 100) - 50
```

**Next I**  
**End Sub**



7. Нарисуйте на листе “Числа” автофигуру. Назначьте ей процедуру выполнения программы **Числа**. Для этого:

- Выделите фигуру.
- Вызовите контекстно-зависимое меню.
- Выполните команду “Назначить макрос”.
- Выберите в открывшемся диалоговом окне только что созданную программу “Числа”.

8. Проверьте работоспособность программы.

9. Создайте макрос “Очистка\_Чисел” для очистки диапазона ячеек A1:D20 листа “Числа”.

10. Подготовьте автофигуру и привяжите к ней макрос.

11. Создайте в этом же модуле еще одну процедуру **KolPolOtr** для подсчета количества положительных, отрицательных и нулевых значений.

```
Public Sub KolPolOtr()
```

```
Rem Объявление переменных
```

```
Rem Pol — переменная для подсчета количества положительных чисел
```

```
Rem Otr — переменная для подсчета количества отрицательных чисел
```

```
Rem Nul — переменная для подсчета нулевых значений
```

```
Dim I As Integer, Pol As Integer, Otr As Integer, Nul As Integer
```

```
RemОбнуление переменных
```

```
Pol = 0
```

```
Otr = 0
```

```
Nul = 0
```

```
Rem Открытие цикла для проверки чисел
```

```
For I = 1 To 20
```

```
If Sheets("Числа").Cells(I, 1) > 0 Then
```

```
Pol = Pol + 1
```

```
ElseIf Sheets("Числа").Cells(I, 1) < 0 Then
```

```
Otr = Otr + 1
```

```
Else
```

```
Nul = Nul + 1
```

```
End If
```

```
Next I
```

```
Rem Вывод на лист "Числа" результатов подсчета
```

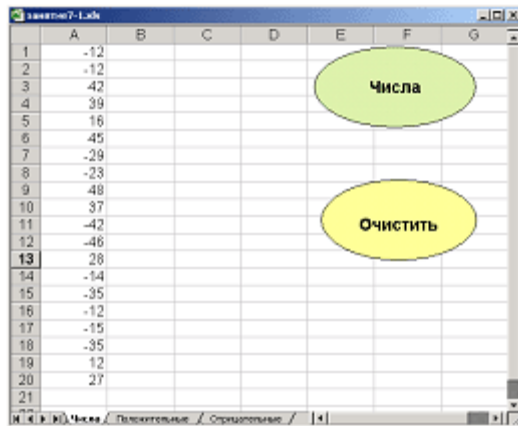
```
With Sheets("Числа")
```

```
.Range("C1") = "Количество +"
```

```
.Range("D1") = Pol
```

```
.Range("C2") = "Количество -"
```

```
.Range("D2") = Otr
.Range("C3") = "Количество 0"
.Range("D3") = Nul
End With
End Sub
```



12. Создайте автофигуру с именем “Количество” на листе “Числа” и привяжите к ней программу KolPolOtr.

13. Создайте в этом же модуле еще одну процедуру **Perenos** для переноса положительных и отрицательных чисел по разным листам.

```
Public Sub Perenos()
Rem Объявление переменных
Dim I As Integer, IndPol As Integer, IndOtr As Integer
Rem переменная-индекс для формирования адреса ячейки положительных элементов
IndPol = 2
Rem переменная-индекс для формирования адреса ячейки отрицательных элементов
IndOtr = 4
Rem Занесение в ячейку B1 слова "Положительные"
Sheets("Положительные").Range("B1") = "Положительные"
Rem Занесение в ячейку B1 слова "Отрицательные"
Sheets("Отрицательные").Range("C1") = "Отрицательные"
For I = 1 To 20
If Sheets("Числа").Cells(I, 1) > 0 Then
Sheets("Положительные").Cells(IndPol, 2) = Sheets("Числа").Cells(I, 1)
IndPol = IndPol + 1
ElseIf Sheets("Числа").Cells(I, 1) < 0 Then
Sheets("Отрицательные").Cells(1, IndOtr) = Sheets("Числа").Cells(I, 1)
IndOtr = IndOtr + 1
End If
Next I
End Sub
```

14. Создайте автофигуру с именем “Перенос” на листе “Числа” и привяжите к ней программу **Perenos**.

	A	B	C	D
1	-45		Количество +	10
2	-17		Количество -	10
3	38		Количество 0	0
4	-5			
5	29			
6	6			
7	-47			
8	25			
9	-41			
10	-5			
11	-43			
12	8			
13	-44			
14	-21			
15	28			
16	-34			
17	-32			
18	9			
19	-39			
20	21			
21				
22				
23				
24				
25				

15. Создайте автофигуры для очистки перенесенных значений на листах “Положительные” и “Отрицательные”.

16. Сохраните работу.

## Лабораторная работа №7. Создание и выполнение макросов Excel.

Термином *макрос* обычно называют файл, хранящий последовательность действий, заданных пользователем системы. Каждый макрос должен иметь собственное имя. С помощью макроса можно автоматизировать типовые технологические этапы при работе с системой. Если макрос создан, то после его запуска хранящаяся в нем последовательность действий (команд) будет автоматически исполнена. По своей сути макрос представляет собой программу и может быть создан автоматически в специальном режиме работы программной системы (в том числе и *Excel*) или как результат программирования в терминах языка системы. Если пользователь владеет языком задания макроса, то созданный любым способом макрос может быть подвергнут редактированию с целью изменения его возможностей или устранения ошибок. В пакете MicrosoftOffice таким языком является язык VBA.

При работе с Excel, как, впрочем, и с другими программами пакета MicrosoftOffice, для создания макроса легче всего использовать автоматический режим его создания, вызываемый из главного меню системы командами **СЕРВИС, Макрос**. При первоначальном запуске системы макросы отсутствуют, поэтому диалоговое окно «Макрос», вызываемое пунктом **Макросы...** показывает пустой список. Пункт меню **Безопасность...** открывает дополнительное меню, позволяющее задавать уровень безопасности при использовании макросов. Известен ряд компьютерных вирусов, маскирующихся под макросы, в связи с чем разработчиками Excel предпринят ряд дополнительных мер защиты. Так, например, может быть задан высокий, средний и низкий уровни безопасности при работе с макросами (по умолчанию средний и рекомендуемый уровень безопасности). Если он используется, то при загрузке файла с диска система попросит разрешение на подключение макросов к программе. Если такое разрешение будет дано, то макрос будет доступен в загружаемой таблице. Пункты меню **Редактор VisualBasic** и **Редактор сценариев** вызывают соответствующие программы (они должны быть установлены на компьютер отдельно с инсталляционных дискет и подключены к операционной системе).

Если в меню **СЕРВИС, Макрос** выбрать пункт **Начать запись...**, то откроется диалоговое окно, позволяющее задать имя макроса и, при желании, комбинацию клавиш, с помощью которой он также может вызван в обход пункта меню **Макросы....** По умолчанию система предлагает стандартное имя **Макрос#**. Во избежание недоразумений старайтесь задавать собственные имена макросов, отличные от стандартных. Начиная с этого момента, все действия с рабочей книгой дополнительно записываются в файл макроса. Остановить запись макроса можно кнопкой **Остановить запись** дополнительно открывшейся панели инструментов или через аналогичный пункт главного меню **СЕРВИС, Макрос**. Записанный макрос может быть сохранен в текущей рабочей книге и тогда он доступен в ней и других книгах в том случае, когда она открыта или в личной книге макросов. В последнем случае он может быть доступен в любой открытой книге.

Удалить макрос, созданный в текущей рабочей книге, можно кнопкой <Удалить> диалогового окна <<Макросы>>. Если макрос создан в личной книге макросов, то для его удаления потребуются более сложные действия. Поэтому старайтесь в первое время не пользоваться макросами личной книги.

Если макрос создан в личной книге макросов, то для его удаления необходимо запустить **Редактор VisualBasic**. В запустившейся оболочке надо открыть окно проектов командами **VIEW, ProjectExplorer** (если оно не открылось автоматически). После этого надо раскрыть содержимое проекта VBAProject (PERSONAL.XLS) и раскрыть ветвь Modules. В ответ на эти действия откроется список модулей проекта. Активируя каждый модуль двойным щелчком, просматривается его содержимое в окне редактора VBA. После того, как интересующий макрос найден, его текст выделяется в окне и удаляется. При необходимости можно удалить весь модуль, щелкнув его правой клавишей мышки, и воспользовавшись пунктом открывшегося меню, например, <RemoveModule1>.

Необходимо принять во внимание существование двух возможных типов записи ссылок на ячейки в Excel: A1 и R1C1. По умолчанию при программировании формул используется стиль A1, для которого адрес каждой ячейки представляет собой строку символов, содержащую имя столбца и номер строки. Использование этого стиля позволяет организовать относительную и абсолютную адресацию к ячейкам таблицы (за счет введения в строку символа \$). Тем не менее, при записи макросов Excel использует тип ссылки R1C1. В обозначении типа присутствуют первые буквы английских слов Row (строка) и Column (колонка). В первую очередь обратите внимание на то, что, в отличие от типа A1, при использовании типа ссылок R1C1 сначала записывается строка, а потом столбец. При использовании абсолютной адресации после символов R и C указывается собственно номер строки и столбца. Так, например, ячейка \$B\$3 имеет адрес R3C2. При использовании относительной адресации в стиле R1C1 после обозначения строки или колонки в квадратных скобках указывается смещение по отношению к текущей ячейке. Так, например, если данные находятся в ячейке B3, а ссылка на нее программируется в ячейке A5, то в формуле она запишется как R[-2]C[1]. Эта запись может интерпретироваться как обращение к ячейке, находящейся на две строки выше и одну колонку правее текущей. Соответственно запись R[2]C[-1] означает обращение к ячейке на две строки ниже и одну колонку левее (по отношению к активной ячейке A5 такая ячейка не существует).

**Пример 1.** Рассмотрим таблицу, показанную на рис. 1. В ней необходимо рассчитать сумму подоходного налога (с учетом используемой ставки налога), сумму к выдаче для каждого сотрудника, а также общие суммы уплачиваемых налогов и выплаченной заработной платы. Записывался макрос с именем Расчет\_заработной\_платы. Текст макроса имеет вид:

```
Sub Расчет_заработной_платы() ' Расчет_заработной_платы Макрос
' Макрос записан 01.12.2005 (Администратор)
Range("C2").Select
ActiveCell.FormulaR1C1 = "=RC[-1]*R7C3"
Range("D2").Select
ActiveCell.FormulaR1C1 = "=RC[-2]-RC[-1]"
```



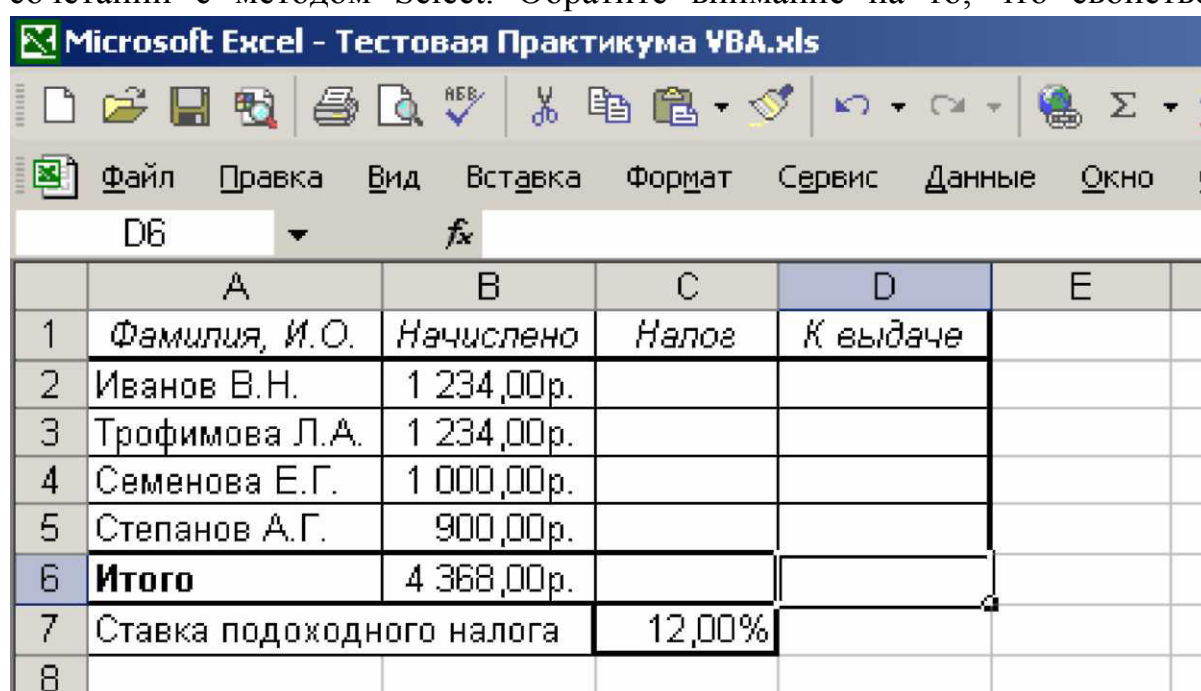
```

Range("C2:D2").Select
Selection.AutoFill Destination:=Range("C2:D5"), Type:=xlFillDefault
Range("C6").Select
ActiveCell.FormulaR1C1 = "=SUM(R[-4]C:R[-1]C)"
Range("D6").Select
ActiveCell.FormulaR1C1 = "=SUM(R[-4]C:R[-1]C)"
EndSub

```

В рассматриваемом примере первый оператор представляет собой заголовок процедуры. Имя процедуры совпадает с именем макроса. Следующие шесть строчек созданы системой в виде автоматически вставляемого комментария.

Первый исполняемый оператор программы `Range("C2").Select` создан системой в виде выражения, которое содержит в терминологии VBA свойство `Range` в сочетании с методом `Select`. Обратите внимание на то, что свойство имеет



	A	B	C	D	E
1	Фамилия, И.О.	Начислено	Налог	К выдаче	
2	Иванов В.Н.	1 234,00р.			
3	Трофимова Л.А.	1 234,00р.			
4	Семенова Е.Г.	1 000,00р.			
5	Степанов А.Г.	900,00р.			
6	<b>Итого</b>	4 368,00р.			
7	Ставка подоходного налога		12,00%		
8					

Рис. 1. Пример таблицы

записанный в круглых скобках аргумент в виде строки символов и отделяется от метода точкой. В нашем примере аргумент свойства представляет собой ссылку на ячейку в стиле A1, с которой началось программирование макроса.

С помощью Help-системы разберитесь с назначением свойства `Range`. Для этого установите в окне модуля маркер на текст `Range` и нажмите клавишу F1. Если вы испытываете затруднения с чтением текста на английском языке, который используется Help-системой, воспользуйтесь дополнительной русскоязычной литературой, посвященной описанию языка VBA. В этом случае удобно составлять собственное описание встречающихся англоязычных терминов и хранить его в удобном месте (например, в виде отдельного файла Excel). Аналогично изучите назначение метода `Select`.

Фактически анализируемая строка программы представляет собой набор действий по активизации ячейки C3 рабочего листа Excel. Система всегда одинаково

интерпретирует действия пользователя Excel, поэтому в случае затруднений с анализом результатов ее работы удобно создать новый дополнительный макрос как результат конкретного короткого действия и изучить его содержимое. Наконец, в особо сложных случаях можно скопировать текст созданного макроса, изменить его имя и запустить его из Excel для того, чтобы увидеть результат действий интересующего вас оператора.

Продолжите изучение операторов созданного макроса и убедитесь в том, что вы понимаете смысл и результат действия каждого оператора. Так следующий оператор рассматриваемого примера заносит в активную ячейку формулу для вычисления величины подоходного налога. В формуле используется стиль ссылок R1C1, причем ее первый операнд задан в относительной адресации, а второй в абсолютной.

Два следующих оператора программы задают другую активную ячейку и заносят в нее формулу для вычисления суммы к выдаче.

Следующий оператор программы выделяет диапазон ячеек листа Excel, после чего выделенные ячейки копируются во все содержащие фамилии сотрудников строчки таблицы.

Для расчета суммы уплачиваемых налогов делается активной предназначенная для этого ячейка рабочего листа и в нее заносится формула, содержащая функцию суммирования данных выделенных ячеек. Система использовала относительную адресацию в формате R1C1. Аналогичная операция проводится и с ячейкой, предназначенной для хранения общей суммы к выдаче.

### Задание

Из таблицы 1 возьмите вариант, соответствующий вашему номеру по списку в журнале. Разработайте и заполните таблицу и запрограммируйте в ней необходимые вычисления. При необходимости воспользуйтесь функциями. Убедитесь в правильности вычислений. Оформите таблицу, задайте шрифты, границы и т.п. В качестве примера будет рассматриваться таблица, предназначенная для расчета налогов и определения суммы заработной платы.

Используя копию созданной таблицы, создайте и изучите макросы, позволяющие программировать вычисления в таблице.

**Таблица 1. Варианты заданий для выполнения лабораторной работы**

Номер варианта	Вид таблицы
1	Ведомость складских остатков (наименование, цена, количество, отпускная цена, оптовая скидка)
2	Ведомость операций квартплаты (плательщик, вид услуги, полный тариф, начислено, льгота, пени, к оплате, задолженность, оплачено).
3	Ведомость операций оплаты за электроэнергию (плательщик, начальное показание, конечное показание, израсходовано, полный тариф, начислено, льгота, пени, к оплате, задолженность, оплачено)
4	Журнал учета выполнения лабораторных работ (фамилия и инициалы студента, названия лабораторных работ, для каждой работы дата, оценка защиты и рейтинг, средний балл, итоговый рейтинг, дата получения зачета).
5	Журнал учета экзаменационных оценок (перечень дисциплин, для каждой дисциплины дата, оценка, рейтинг по итогам семестра и сессии, общий рейтинг, средний балл).

6	Расписание занятий преподавателей кафедры (фамилии преподавателей, должность, ученое звание, ученая степень, для каждого дня нечетной и четной недели и каждой учебной пары название или код дисциплины, вид занятия, номера учебных групп, номер аудитории, объем учебной нагрузки).
7	Индивидуальная выписка для преподавателя по проведенным занятиям для представления на оплату (дата проведения, время проведения, номер аудитории, номера групп, вид занятия, источник финансирования (государственный бюджет или договор на оплату образовательных услуг), количество часов, количество оплачиваемых часов, часовая ставка, сумма к оплате).
8	Ведомость командировок (фамилия, город, страна, цель поездки, источник финансирования, дата убытия, дата прибытия, срок командировки, стоимость проезда туда, стоимость проезда обратно, суточные, сумма затрат).
9	Ведомость операций туристического агентства (фамилия, страна, город, вид транспорта туда, вид транспорта обратно, транспортные расходы туда, транспортные расходы обратно, отель, стоимость проживания в сутки, дата заезда, дата убытия, срок проживания, затраты на проживание, общие затраты).
10	Ведомость операций риэлтерского агентства (адрес, район, метро, этаж, жилая площадь, количество комнат, вспомогательная площадь, удобства, стоимость квадратного метра, цена помещения, затраты на ремонт и переоборудование помещения, общая стоимость).
11	Ведомость операций обменного пункта валюты (валюта прихода, сумма прихода, курс к рублю, комиссия вид валюты, курс валюты комиссии к рублю, комиссия в рублях, валюта расхода, сумма расхода, курс к рублю).
12	Ведомость операций авиакассы (фамилия, направление, рейс, дата вылета, время вылета, тариф авиакомпании, валюта тарифа, тариф в рублях, аэропортовый сбор пункта отправления, валюта сбора пункта отправления, сумма в рублях, аэропортовый сбор пункта прибытия, валюта сбора пункта прибытия, сумма в рублях, стоимость трансфера, валюта трансфера, сумма трансфера в рублях, комиссия).
13	Ведомость продаж универсама (вид товара, единица измерения, имеющееся количество, цена складская, цена отпускная, объем продажи в единицах измерения, остаток, стоимость продажи, скидка, льгота, сумма к оплате, вид оплаты, комиссия банка).
14	Смета затрат на ремонт (номер операции, операция, материалы, единица измерения, цена, стоимость, нормочасы, тариф, зарплата, наценка, стоимость, скидка, к оплате).
15	Ведомость операций телефонной компании (абонент, тарифный план, вид операции, тариф, время, цена операции, наценка, стоимость, скидка, льгота, к оплате).
16	Ведомость комплектации изделия (наименование комплектующего, количество, цена, количество на складе, стоимость складского остатка, затраты, наценка, стоимость).
17	Таблица футбольного чемпионата (команда, страна, город, игр, побед, ничьих, поражений, технических поражений, забито голов, пропущено голов, очков).
18	Ведомость операций типографии (автор, название, издательство, машинописных страниц, печатных листов, рисунков, таблиц, тираж, тип бумаги, цена печатного листа, цена печати, тип переплета, цена переплета, затраты на материалы, затраты на амортизацию оборудования, заработная плата, накладные расходы, стоимость).
19	Список трудов (номер, название, место опубликования, дата опубликования, вид публикации, номер страницы начала, номер страницы конца, всего страниц, формат страницы, машинописных листов, печатных листов, соавторы, доля автора, машинописных страниц автора, печатных листов автора).
20	Ведомость операций отделения связи (адрес назначения, адрес отправителя, вид отправления, вес отправления, тариф, дата отправления, упаковка, цена упаковки, страховка, общая цена отправления).
21	Ведомость операций страхового агентства (фамилия страхуемого, объект страхования, вид страхования, дата страхования, дата начала действия страховки, дата окончания действия страховки, срок страхования, тариф, цена полиса, скидка, льгота, к оплате).
22	Ведомость операций библиотеки (автор, название, издательство, год издания, объем, цена, дата выдачи, контрольная дата возврата, планируемый срок пользования, фактическая дата возврата, фактический срок пользования, ставка штрафных санкций, штраф).
23	Ведомость операций фотоателье (фамилия заказчика, вид операции, дата заказа, дата исполнения, общее время исполнения, тариф, срочность, количество, стоимость, скидки, льготы).
24	Ведомость банковских операций (фамилия, дата, вид операции, валюта операции, сумма операции, сумма операции в рублях, комиссия операции, валюта комиссии операции, комиссия операции в рублях).

25	Ведомость операций диспетчерской такси (клиент, адрес подачи машины, адрес назначения, дата поездки, время начала поездки, время окончания поездки, километраж, тип машины, расчетное время выполнения заказа, время на подачу машины, тариф, стоимость, скидка, льгота, к оплате).
26	Ведомость судейства соревнований по фигурному катанию (участник, город, страна, вид программы, оценки судей, каждая из которых включает оценку за технику исполнения, оценку за художественное впечатление, место в общем зачете, итоговая оценка за технику исполнения, итоговая оценка за художественное впечатление, суммарное место в общем зачете).

## Порядок выполнения работы

1. Создайте новую рабочую книгу Excel. Сделайте ее настройку:

■ выполните команду **Параметры Excel** в диалоговом окне выберите вкладку *Формулы*, установив следующие параметры:

Стиль ссылок R1C1: выключено.

На вкладке *Основные*:

Число листов: 3.

Стандартный шрифт: Arial, размер 10.

■ выберите вкладку *Дополнительно*, установив флажки следующих параметров:

Экран: показывать строку формул, окна на панели задач.

Примечания: только индикатор.

Параметры для листа: заголовки строк и столбцов, горизонтальная полоса прокрутки, вертикальная полоса прокрутки, сетка, нулевые значения, ярлычки листов.

При пересчете этой книги: обновить ссылки на другие документы, сохранять значения внешних связей.

2. Переименуйте рабочий лист, выполнив следующие действия:

■ установите указатель мыши на вкладку с именем листа (Лист 1) и вызовите контекстное меню, щелкнув правой клавишей мыши;

■ выберите в текстовом меню параметр **Переименовать**;

■ введите в диалоговом меню новое имя листа, придуманное вами.

3. Сохраните созданную рабочую книгу с новым, придуманным вами именем, выполнив команду **ФАЙЛ, Сохранить как...**

4. Создайте шаблон придуманной вами пользовательской таблицы.

5. Задайте наименования полей шапки таблицы. При необходимости укажите в них единицы измерения.

6. Заполните таблицу данными и запрограммируйте в ней необходимые вычисления. Убедитесь в правильности вычислений.

7. Скопируйте созданную таблицу на новый рабочий лист. Удалите в ней все формулы.

8. В меню **СЕРВИС, Макрос** выберите пункт **Начать запись....** Задайте имя макроса.

9. Повторно запрограммируйте формулы таблицы Excel и остановите запись макроса.

10. Командой **Сервис, Макрос, Редактор VisualBasic** запустите редактор VisualBasic. В окне проектов (**Project-VBAProject**) (рис. 2) раскройте содержимое проекта **VBAProject (PERSONAL.XLS)** и ветвь **Modules**. В ее составе должен быть

один (например, Module1) или несколько модулей. Дважды щелкните левой клавишей мышки по имени модуля. В ответ в правом верхнем окне должен появиться его текст. Просмотрите содержимое модулей и найдите записанный вами макрос.

11. Изучите текст макроса.

12. Удалите формулы из таблицы рабочего листа Excel и выполните макрос командой **Сервис, Макрос, Макросы**. Убедитесь, что в результате его работы содержимое таблицы восстанавливается.

13. Снова удалите формулы из таблицы рабочего листа Excel. Перейдите в окно VBA, установите маркер на первом операторе макроса. Выберите пункт **Run, RunSub/UserForm** и запустите модуль на выполнение. Перейдите в таблицу Excel и убедитесь, что в результате работы макроса формулы в ней восстановились.

14. Окончательно оформите созданную таблицу для представления ее в отчетной документации. Воспользуйтесь возможностями задания шрифтов, границ, заливок. Обеспечьте компактность отображения таблицы за счет минимизации ширины строк и столбцов в соответствии с имеющимися данными.

### Контрольные вопросы

1. В каком случае используется стиль ссылок Excel :A1, а в каком R1C1?
2. В чем разница абсолютной и относительной адресации ссылок в Excel?
3. Когда целесообразно использовать абсолютную адресацию в Excel?
4. Если написать макрос вручную, то какие обязательные операторы он должен содержать?
5. Каково назначение свойства Range?
6. Каково назначение метода Select?
7. Как вызвать систему помощи и получить справку по конкретному выражению макроса?
8. Как можно запустить макрос на выполнение?
9. Что такое построчный комментарий и как он оформляется?
10. Какие существуют возможности для оформления внешнего вида таблицы Excel перед ее публикацией в отчетной документации?

### Отчет о работе

Подготовьте отчет о выполненной лабораторной работе. Он должен содержать титульный лист, формулировку задания, пример созданной таблицы, содержание ее программирования и текст созданного вами макроса с включенными в него построчными комментариями действий системы, созданными вами как результат анализа текста макроса. Сформулируйте выводы, которые можно сделать по результатам выполненной работы.

## Лабораторная работа №8. Отладка программного обеспечения и исправление ошибок

Мы уже просматривали макрос, созданный в процессе выполнения предыдущей лабораторной работы средствами VBA. Рассмотрим назначение интегрированной среды разработки приложений VBA (рис. 2) более подробно.

Верхние строки представляют собой главное меню программы и набор пиктограмм часто используемых операций. Как обычно, этот набор может настраиваться в зависимости от потребностей пользователя. Ниже в левой части экрана находится уже знакомое нам окно проектов **Project-VBAProject**.

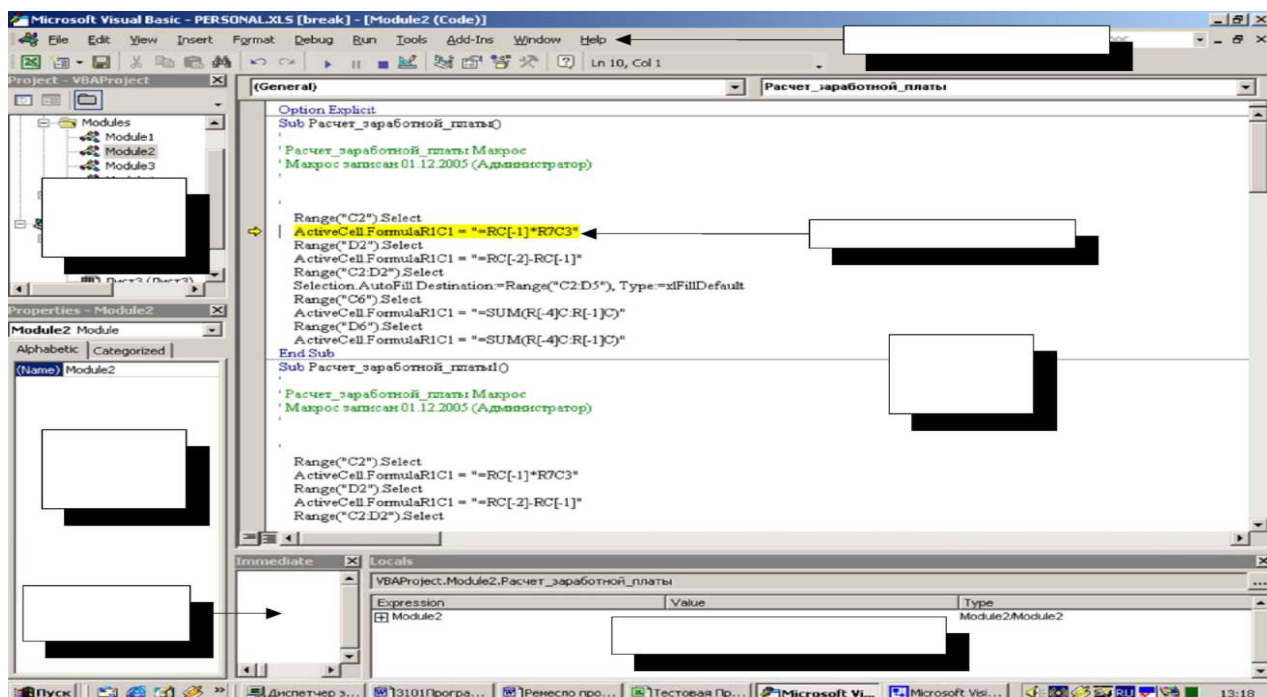


Рис. 2. Интегрированная среда разработки VBA

Содержимое выбранного в этом окне модуля Module2 раскрыто в окне редактора кодов, находящемся правее. На рисунке зафиксирован момент отладки программы, поэтому одна из строк кода выделена специальным цветовым маркером. Его положение указывает на следующий выполняемый оператор программы. Обратите внимание на окно локальных переменных **Locals** в нижней части экрана. В нем отображается содержимое ячеек памяти. В выполняемом макросе нет собственных переменных, поэтому ветвь Module2 в настоящий момент пустая. Окно тестирования **Immediate** позволяет изменять значения переменных программы в момент ее выполнения и даже вводить дополнительные операторы. На настоящий момент оно также пустое. Остальные элементы (кнопки управления, полосы прокрутки, раскрывающиеся списки и т.п.) являются стандартными для Windows и должны быть вам хорошо знакомы. Наконец дополнительно в левой нижней части экрана командой **View, Properties Window** открыто окно свойств.

Если программа еще не запущена, то в ответ на выбор этого пункта активируется маркер отладчика (рис. 2). Дальнейший выбор **Step Into** приведет к выполнению следующей строки программы. Результаты выполнения могут, например, проявиться в виде изменения переменных программы и быть проконтролированы в окне

локальных переменных **Locals**(если такие переменные существуют) или непосредственно на листе Excel, если выполняется макрос, созданный средствами Excel.

Готовая программа, которой, в частности, является созданный нами макрос, может быть запущена и выполнена с использованием интегрированной среды VBA. Мы можем запустить конкретный модуль, предварительно указав его маркером в окне редактора кодов. Для этого достаточно выбрать пункт **Run, RunSub/UserForm** главного меню. Операторы программы будут выполняться автоматически один за другим, а после завершения ее выполнения мы можем только контролировать результаты ее работы. Этот режим мы уже использовали при выполнении предыдущей лабораторной работы. Очень часто, особенно при создании новой программы, разработчика интересуют не только итоговые, но и промежуточные результаты ее выполнения. Для этого удобно воспользоваться режимом отладки программы (пункт **Debug** главного меню). Выбор строки **Step Into** позволяет выполнить текущий оператор программы.

**Примечание.** Меню отладчика предлагает еще две возможности пошагового выполнения программы: **Step Over** и **Step Out**. Они представляют интерес при работе с программами, содержащими вызываемые функции или процедуры. Режим **Step Over** позволяет автоматически выполнить вызываемую функцию (процедуру) и перейти к следующему оператору основной программы, а режим **Step Out** закончит выполнение текущей функции или процедуры. Способы их использования будут обсуждены в лабораторной работе, посвященной функциям и процедурам.

В качестве дополнительных возможностей отладчика отметим следующее. Можно автоматически выполнять операторы программы до оператора, на котором установлен курсор в окне редактора кодов. Этот режим вызывается строкой **Run To Cursor** пункта **Debug** главного меню. Программа запускается в автоматическом режиме из своего текущего состояния, а после остановки на отмеченном курсором операторе он выделится маркером отладчика. В текст программы можно вводить так называемые точки останова (строка **Toggle Breakpoint**). В окне редактора кодов такой оператор помечается специальным маркером. После запуска программы любыми средствами она автоматически выполняется до точки останова. Кроме этого, существует возможность наблюдать значения избранных вами переменных программы в окне наблюдаемых выражений **Watches** (на рис. 2 это окно не показано). Если вы захотите воспользоваться этим режимом, то командой **View/Watch Window** главного меню включите его, выберите соответствующую строку пункта **Debug** главного меню и задайте наблюдаемое выражение в открывшемся меню.

### Задание

Используйте вариант задания из таблицы 1 лабораторной работы №7, таблицу Excel, выполненную на его основе, и созданный вами макрос. Научитесь выполнять макрос в режиме отладчика и проверьте правильность его работы.

### Порядок выполнения работы

1. Откройте созданную вами рабочую книгу Excel. Скопируйте свою таблицу на новый лист. Удалите в ней все формулы.

2. Проверьте работоспособность ранее созданного макроса, для чего воспользуйтесь командой **Сервис, Макрос, Макросы**. Убедитесь в правильности вычислений.

3. Командой **Сервис, Макрос, Редактор VisualBasic** запустите редактор VisualBasic. Сделайте настройку интегрированной среды разработки VBA:

■ выполните команду **Tools, Options** и в диалоговом окне выберите вкладку *Editor*, установив следующие параметры:

Code Settings: Auto Syntax Check, Require Variable Declaration, Auto List members, Auto Quick Info, Auto Data Tips –включено.

Window Settings: Drag-and-Drop Text Editing, Default to Full Module View, Procedure Separator –включено. Auto Indent –включено. Tab – 4.

В этом случае редактор автоматически находит синтаксические ошибки в программе, любые переменные программы могут использоваться только после их явного предварительного объявления, разрешается автоматический вывод списка свойств и методов, разрешается автоматический вывод информации о функции, на экране отображается значение переменной, на которую установлен указатель мыши. Кроме этого, разрешено перетаскивание фрагментов программы мышью, в окне редактора кодов отображаются все процедуры текущего модуля, а между текстами процедур и функций модуля присутствует разделительная линия. Наконец, В программе автоматически устанавливаются отступы в тексте, а при нажатии клавиши Tab в текст вставляется 4 пробела.

■ Выберите вкладки *Editor Format, General* и *Docking*. Изучите их содержимое. Изменять их настройку, задаваемую по умолчанию, не рекомендуется.

4. Снова удалите формулы из таблицы. Выполните созданный вами макрос в пошаговом режиме. Для этого установите маркер в окне редактора кодов в тексте созданного вами макроса. Выполните команду **Debug, Step Into** и убедитесь, что маркер отладчика активировался на заголовке выполняемого макроса. Последовательно используя команду **Debug, Step Into** переключаясь на рабочий лист Excel, контролируйте процесс заполнения таблицы. Убедитесь в правильности вычислений.

5. Снова удалите формулы из таблицы. Установите маркер в окне редактора кодов в тексте созданного вами макроса на один из выполняемых операторов программы. Выполните команду **Debug, Run To Cursor** и убедитесь, что маркер отладчика активировался на выбранном вами операторе. Проверьте, что в результате выполнения начального фрагмента программы заполнилась соответствующая им часть таблицы Excel. Убедитесь в дальнейшей работоспособности программы за счет выполнения команд **Debug, Step Into**. Снова установите маркер на один из следующих операторов программы и убедитесь в правильности выполнения команды **Debug, Run To Cursor**.

6. Снова удалите формулы из таблицы. Установите маркер в окне редактора кодов в тексте созданного вами макроса на один из выполняемых операторов программы. Командой **Debug, Toggle Breakpoint** задайте точку останова. Выполните команду **Run, Run Sub/User Form** и убедитесь, что маркер отладчика активировался



на выбранном вами операторе. Убедитесь в дальнейшей работоспособности программы за счет выполнения команд **Debug, StepInto** и **Debug, RunToCursor**.

7. Снова удалите формулы из таблицы. Установите маркер в окне редактора кодов в тексте созданного вами макроса на точку останова. Повторно выполните команду **Debug, ToggleBreakpoint** и удалите точку останова. Убедитесь в правильности выполнения программы, запустив ее одним из возможных способов. Проверьте возможность задания нескольких точек останова в программе.

### Контрольные вопросы

1. Каково назначение окна локальных переменных?
2. Каково назначение окна редактора кодов?
3. Как выглядит маркер отладчика?
4. Каково назначение окна проектов?
5. Как выполнить программу по шагам?
6. Какие команды существуют для выполнения программы и в чем их отличие?
7. Что такое точка останова?
8. Как при выполнении программы по шагам можно автоматически выполнить ее определенную часть?
9. Как запустить программу на выполнение из Excel?
10. Как можно наблюдать результаты выполнения программы в пошаговом режиме в таблице Excel?

### Отчет о работе

Подготовьте отчет о выполненной лабораторной работе. Он должен содержать титульный лист и описание последовательности ваших действий с отладчиком, а также результаты выполнения программы. Дайте письменные ответы на контрольные вопросы. Сформулируйте выводы, которые можно сделать по результатам выполненной работы.

## **Лабораторная работа №9. Обмен данными между Excel и VBA**

Процессор ЭВМ манипулирует с данными, имеющимися в памяти машины. В зависимости от конкретной задачи эти данные могут принимать самые разнообразные значения, но они должны быть занесены в стандартные физические ячейки памяти, размер которых определяется конструкцией конкретного устройства. Поскольку для хранения различных данных может потребоваться различный объем памяти, используется метод последовательного размещения их в памяти. В этом случае одна единица данных может занимать одну или несколько последовательных физических ячеек памяти машины. Адресация к данным производится по адресу первой ячейки, но при этом общее число используемых ячеек должно быть точно известно. Так как только программист в состоянии предусмотреть возможные значения данных, используемых в программах, вопрос о распределении памяти для хранения информации ложится на его плечи. Конкретная организация памяти в задаче осуществляется за счет выбора программистом конкретного типа данных для хранения его информации.

Тип данных – способ внутреннего представления данных в памяти машины, учитывающий метод их кодирования в одной или нескольких ячейках памяти и предусматривающий возможности их расшифровки или преобразования.

Первые языки программирования содержали всего два типа данных – целые (Integer) и дробные (Real или Float или Single). С развитием языков программирования и расширением круга решаемых задач число используемых типов данных непрерывно росло. Так, для обеспечения требуемой точности и диапазона вычислений были введены соответственно для целых и дробных чисел типы Long и Double. Потребность в хранении текстовой информации привела к появлению типа данных Byte (в терминологии VBA), позволяющего наиболее экономно расходовать память ЭВМ (современные таблицы кодировки символов используют диапазон кодов от 0 до 255). Задачи, связанные с анализом и составлением текстовых сообщений, стали поддерживаться типом данных String. Для обеспечения возможностей ссылки на различные участки памяти был предложен специальный тип данных, называемый указателем (Object). В случае использования указателя в памяти хранится адрес ячейки памяти, содержащей интересующие нас данные или коды программы. Кроме этого, программисту предоставили возможность самому создавать интересующий его тип данных. Необходимость выполнения вычислений с датами и временем породила свой специальный тип данных Date. Особые условия выполнения вычислений с деньгами заставили добавить в перечень специальный тип Currency. Наконец, для упрощения начального ввода данных в клетки электронных таблиц Excel был разработан специальный тип данных Variant, позволяющий автоматически распознавать и обрабатывать числа и строки.

Готовясь к написанию программы, программист обязан задуматься над вопросом: какие значения могут принимать данные его программы? Ответив на этот вопрос, программист подбирает удобный ему тип данных из числа стандартных или создает свой. При этом приходится принимать во внимание следующее обстоятельство: использование стандартных типов данных существенно упрощает процесс создания программы, поскольку в языке программирования заложены

возможности действий с этими данными и их преобразование из типа в тип. Мы предполагаем, что вы знакомы с типами данных Excel, задаваемыми ячейкам командой **Формат, Ячейки** вкладка **Число** окно **Числовые форматы**. В языке VBA существуют типы данных, приведенные в табл. 2.

**Пример 2.** В программе, предназначенной для расчета начисления заработной платы рис. 1 для хранения номера в списке (если он будет добавлен в таблицу) можно выбрать тип данных Integer, для хранения фамилий сотрудников тип данных String. Ставка заработной платы и величина начисленного налога может быть описана типом данных Currency, а ставка налога типом данных Single. Кроме этого, например можно создать свой тип данных (Type), в который входят фамилия, начисленная сумма, сумма уплачиваемого налога и сумма к выдаче как самостоятельная единица хранимых в памяти данных.

Поскольку физически данные программы оказываются содержимым конкретных ячеек памяти машины, для их отыскания достаточно знать адрес первой ячейки, связанной с данными, и по типу данных определить общее число используемых для хранения элементарных ячеек. Такой подход имел место на самой ранней стадии программирования и оказался крайне неудобным из-за отсутствия наглядности в записи программы. Действительно, если память современной машины содержит несколько десятков, а то и сотен миллионов ячеек памяти, то обращение к ним по номерам было бы крайне неразумным. Уже первые трансляторы использовали прием, основанный на использовании так называемых идентификаторов.

Идентификатором называется символическое имя ячейки памяти. Каждый язык программирования содержит свои правила составления таких имен, общим является то, что программист вправе сам придумать имя, что позволяет ему сохранить в нем смысловое значение. В языке VBA имеются следующие ограничения на имена:

- Длина имени не должна превышать 255 символов.
- Имя должно начинаться с буквы.
- Имя не может содержать точек и символов %, &, !, #, @, \$.
- Буквы рассматриваются инвариантно по отношению к регистру, то есть имя Aa и aA есть одно и то же имя.
- Совпадения имен идентификаторов с так называемыми ключевыми словами не допускается.

Ключевые слова – набор специальных слов, написанных символами латыни и имеющих определенный смысл с точки зрения конструкций языка программирования. Ключевыми словами обозначаются, в частности, операторы языка и встроенные функции языка.

**Пример 3.** Возможные варианты идентификаторов языка VBA: I, j, Name, Переменная, Результатвычислений. Еще варианты записи идентификаторов: A%, B&, C!, D#, E@, F\$. В этом случае символы %, &, !, #, @, \$ не входят в состав идентификатора и используются в качестве специального признака типа данных (смотри табл.2).

Программист может вводить переменные в текст программы на VBA по мере их необходимости, применяя явное или неявное (по умолчанию) их объявление.

В последнем случае переменная просто начинает использоваться в тексте. При первом ее появлении компилятор (интерпретатор) заносит новое имя в таблицу и закрепляет за ним определенный адрес и тип данных (в VBA—Variant).

Хотя возможность объявления переменных по умолчанию предусмотрена разработчиками языка, она представляется крайне нежелательной. Текст программы сам по себе представляет документ, в котором содержится исчерпывающая информация о ее работе, в том числе и о типах используемых данных. Введение переменных по умолчанию приводит к затруднениям при изучении программы и, как следствие, к ошибкам. Поэтому рекомендуется всегда явно определять переменные с помощью оператора Dim с указанием типа и задавать специальный режим принудительного объявления переменных программы помещенной в начале текста модуля инструкцией OptionExplicit.

**Пример 4.** Явноеобъявлениепеременной:

Dim I As Integer, Name, j As Integer, Переменная As Integer, GGG As Integer

Обратите внимание на то, что если вы не указываете явно тип переменной, то по умолчанию она имеет тип Variant. Так, в рассмотренном выше примере такой тип имеет переменная Name.

**Примечание.** Интегрированная среда разработки VBA в окне редактора кодов предлагает в качестве сервиса возможность выбора одного из существующих типов данных из автоматически раскрывающегося списка. Так, после набора ключевого слова Dim, указания идентификатора переменной и набора ключевого слова As автоматически открывается список возможных значений (в данном случае типов данных). Перемещение по списку может осуществляться с помощью маркеров или путем ввода символов с клавиатуры. После того, как требуемое значение в списке установлено, оно может быть перенесено в текст программы клавишей Tab или в результате двойного клика клавишей мышки. Этой возможностью удобно пользоваться для избегания грамматических ошибок при наборе текста программы.

Рассмотренные выше примеры объявления переменных предусматривали создание одиночных констант или переменных, обращение к которым осуществляется только по имени. Практика программирования широко использует переменные, обращение к которым ведется как по имени, так и по номеру. В этом случае можно говорить о создании переменных табличного типа, когда обращение к данным ведется по имени и номеру (индексу) внутри этого имени. Такие переменные обычно называются массивами. Массив — последовательно упорядоченные в памяти данные одного типа.

**Таблица 2. Типы данных языка VBA**

Тип данных	Размер (байт)	Служебный символ	Диапазон значений
Byte (байт)	1		От 0 до 255
Boolean (логический)	2		True или False
Integer (целые)	2	%	От -32768 до 32767
Long (длинное целое)	4	&	От -2147483648 до 2147483647
Single (плавающее обычной точности)	4	!	От -3,402823E38 до -1,401298E-45 и от 1,401298E-45 до 3,402823E38



массива может выступать не только константа, но и другая переменная, заданная своим идентификатором. Заметим, что недостатком рассмотренного приема является относительно высокая вероятность возникновения ошибки программирования связанной с выходом индекса (номера элемента) за границы массива. Программная среда VBA автоматически локализует такую ситуацию, выдавая соответствующее диагностическое сообщение.

**Пример 6.** Обращение к элементу массива в тексте программы с явным указанием номеров элементов: SS(-2,5).

Если переменная Name содержит число -2, а ячейка Переменная число 5, то обращение SS(Name, Переменная) полностью эквивалентно предыдущему.

Если в процессе предыдущих вычислений переменная Name примет значение -4, а мы попытаемся выполнить SS(Name, Переменная), то произойдет обращение к несуществующему элементу массива и возникнет ошибка выхода индекса за границы массива.

Массивы удобно использовать при программировании одностипных действий с ячейками памяти. В качестве примера рассмотрим задачу расчета начисления заработной платы (рис. 1). Поскольку исходные данные и результаты промежуточных вычислений должны храниться в памяти ЭВМ, в процессе программирования решения задачи на VBA приходится использовать идентификаторы. Заметим, что обычный идентификатор в этом случае не очень удобен. Действительно, хотя возможно введение в текст программы обычной переменной вида Налог\_Трофимова\_Л\_А, создаваемая программа может быть в этом случае использована только для расчетов налога, уплачиваемого именно Л.А. Трофимовой. Если мы хотим запрограммировать вычисления для другого лица, то нам придется вводить другой идентификатор. Подобные действия ведут к изменению текста исходной программы и крайне нежелательны на практике. Конечно, мы можем ввести идентификаторы обычных переменных вида Налог\_запись\_2, однако и в этом случае мы должны будем индивидуально описать последовательность манипуляций с ячейками памяти для каждого сотрудника, включенного в список. Для нашего примера это вполне возможно, но реальный список может состоять, например, из 100 фамилий.

Кроме всего прочего, каждый раз при изменении количества сотрудников мы должны корректировать объявления переменных и, возможно, делать добавления в текст программы. Программирование существенно упростится, если ввести в рассмотрение массивы данных, имеющие смысл Начислено(1 To 4), Налог(1 To 4), К\_выдаче(1 To 4) и рассматривать их элементы с одинаковыми номерами как записи, относящиеся к сотруднику, имеющему соответствующий идентификационный номер. На первый взгляд этот способ ничем существенным не отличается от использования идентификаторов одиночных переменных с номерами, однако если вспомнить, что существует возможность обращения к элементу массива с использованием идентификатора другой переменной, то можно рассматриваемую задачу попытаться описать и в общем виде.

**Пример 7.** В общем виде выражение для вычисления величины суммы к выдаче для каждого сотрудника может быть записано как:

К\_выдаче(i)= Начислено(i)– Налог(i)

Здесь символом = обозначена операция присваивания результата вычислений в правой части оператора ячейке, указанной в левой части. Во время выполнения этой операции старое содержимое ячейки K\_выдаче(i) теряется и она получает новое значение. В то же время символ – есть символ операции вычитания.

Если организовать повторения вычислений по этой формуле столько раз, сколько сотрудников имеется в списке для последовательно изменяющихся значений индекса i, то рассматриваемая задача может быть решена заметно проще, чем в случае объявления одиночных переменных.

Иногда приходится создавать массивы, размер которых невозможно определить на этапе компиляции программы. В нашем примере нам может быть неизвестно общее число сотрудников, для которых должна быть начислена зарплата. Конечно, можно объявить массивы с запасом, так, чтобы номер максимального элемента массива был заведомо большим максимально возможного числа сотрудников, допустим 100 человек. Однако такой прием приводит к нерациональному распределению памяти. Альтернативой является метод динамического объявления размера массива. В этом случае конкретный размер массива вычисляется в процессе выполнения программы и память для хранения данных отводится тоже во время выполнения. Чтобы воспользоваться этим методом, необходимо первоначально объявить массив без указания его размеров, а затем воспользоваться инструкцией ReDim. Менять границы изменения индекса массива можно сколь угодно много раз. Если массив больше не требуется в программе, память, занимаемая им, может быть освобождена с помощью инструкции Erase. Начислено.

#### **Пример 8.**

Dim Начислено() As Currency, i As Integer

i = 10

ReDim Начислено(1 To i)

Массив Начислено() первоначально был объявлен как массив неопределенной длины. Инструкция ReDim изменила массив, причем память под него была отведена в момент выполнения программы.

Очень часто при программировании возникает необходимость создания новых типов данных, вид которых определяется конкретной задачей. Так, например, программируя задачу, представленную на рис. 1, обратим внимание на то обстоятельство, что информация, размещенная в этой таблице, имеет одинаковую структуру по строкам. Более того, даже программируя соответствующую колонку таблицы в виде массива, программист обязан следить за тем, чтобы номера элементов разных массивов, относящихся к одному сотруднику, не отличались бы один от другого. Из соображений надежности программирования оказывается удобным рассматривать все, относящееся к одному сотруднику, в виде целой неделимой записи, содержащей соответственно фамилию, начисленную сумму, рассчитанный налог и сумму к выдаче. На самом деле речь идет о создании нового типа данных, определенного пользователем и включающего в себя относящиеся к записи поля. Структура данных - объединение под одним именем различных компонентов с индивидуальными именами и типами, называемых членами структуры.

Признаком структуры данных, как правило, является символ точки в ее идентификаторе, причем имя структуры записывается до точки, а имя ее компонента (члена) после точки. В языке VBA структуры данных можно создавать на основе типов данных, определяемым пользователем. Задание типа данных только описывает структуру, информация о которой размещается в общей области программы VBA. Для ее непосредственного объявления и резервирования ячеек памяти под хранение данных требуется явно объявить переменную в конкретном модуле.

**Пример 9.** Создание пользовательского типа данных, представляющего собой одну строку записи рис. 1.

```
Type ЗаписьВедомости
    Фамилия_И_О AsString
    НачисленоВедомость AsCurrency
    НалогВедомостьAsCurrency
    К_выдаче_ВедомостьAsCurrency
EndType
```

Объявление переменной:

```
Dim Запись1 As ЗаписьВедомости
```

Запись значений в элементы структуры с использованием оператора присваивания:

```
Запись1.Фамилия_И_О = "Иванов В.Н."
```

```
Запись1.Начислено_Ведомость = 1234
```

```
Запись1.Налог_Ведомость = Запись1.Начислено_Ведомость * 0.12
```

```
Запись1.К_выдаче_Ведомость=Запись1.Начислено_Ведомость-Запись1.Налог_Ведомость
```

Здесь символом \* обозначена операция умножения.

Объявление массива структур:

```
Dim Ведомость(1 To 4) As Запись_Ведомости
```

Соответствующие обращения к элементам массива и членам структуры будут иметь вид:

```
Ведомость(1).Фамилия_И_О = "Иванов В.Н."
```

```
Ведомость(1).Начислено_Ведомость = 1234
```

```
Ведомость(2).Фамилия_И_О = "Трофимова Л.А."
```

```
Ведомость(2).Начислено_Ведомость = 1234
```

**Примечание.** Интегрированная среда разработки VBA в окне редактора кодов предлагает в качестве сервиса возможность конкретного выбора типа данных, определенных пользователем, из автоматически раскрывающегося списка. Если структура данных ранее была объявлена и выполнена компиляция проекта, после набора символа точки автоматически открывается список возможных имен полей структуры. Этой возможностью удобно пользоваться для избегания синтаксических ошибок при наборе текста программы.

Отдельную проблему представляет прямая и обратная передача данных из таблицы Excel в ячейки памяти, объявленные в программе, написанной на VBA. Автоматически созданный макрос непосредственно манипулирует с ячейками таблицы, используя стили ссылки на ячейки в Excel: A1 и R1C1. Конечно, такой прием может быть использован и в рабочей программе, однако в этом случае ее



модификация и использование существенно затруднены. Гораздо предпочтительнее использовать свойство Cells() стандартного объекта ExcelRange. Сам объект представляет собой ячейку, столбец, строку или выделенный диапазон листа Excel. Свойство Cells() позволяет непосредственно обратиться к объекту Excel по номеру строки и колонки. Поскольку это свойство установлено по умолчанию для рабочего листа Excel, то его можно использовать без дополнительных указаний.

Свойство Cells() позволяет обратиться к ячейке рабочего листа задав номер строки и колонки. Если запись свойства стоит слева от символа равенства (оператор присваивания), то производится запись данных в ячейку таблицы, если справа, то считывание значения из ячейки таблицы. Кроме собственно записи данных свойство Cells() в сочетании со свойствами других объектов (Font, Color и т.п.) позволяет задавать параметры шрифта, его цвет, фон и так далее. Для изучения этих возможностей целесообразно ознакомиться с описанием соответствующих свойств и объектов в литературе, воспользоваться Help-системой или, что вероятно проще всего, запустить режим записи макроса в Excel, выполнить, например, установку цвета и изучить текст полученного макроса.

**Пример 10.** Использование свойства Cells() для считывания данных в переменную VBA и возврата значения в Excel и установки нового цвета шрифта. Используется тот факт, что положение и количество ячеек в таблице рис. 1 известно. Дополнительно в программе используется символ комментария ' и комбинация символов «пробел»\_ ( \_ ) для обозначения продолжения длинной строки.

```
Sub Расчет_заработной_платы2()  
Dim Начислено(1 To 4) As Currency, Налог(1 To 4) As Currency, _  
K_Выдаче(1 To 4) As Currency, i As Integer  
i = 1 'Задание начального номера массива  
Do While (i <= 4)  
    Начислено(i) = Cells(i + 1, 2) 'В первую ячейку массива Начислено записывается  
    'содержимое второй строки и второй колонки исходной таблицы Excel  
    Cells(i + 1, 2).Font.ColorIndex = 5 'В ячейке устанавливается новый цвет шрифта  
    Налог(i) = Начислено(i) * 0.12 'Рассчитывается значение налога и запоминается  
    'в соответствующей ячейке  
    Cells(i + 1, 3) = Налог(i) 'Значение налога возвращается в таблицу Excel  
    K_Выдаче(i) = Начислено(i) - Налог(i) 'Рассчитывается значение к выдаче  
    'и запоминается в соответствующей ячейке  
    Cells(i + 1, 4) = K_Выдаче(i) 'Значение к выдаче возвращается в таблицу Excel  
    i = i + 1 'Модификация счетчика строк  
Loop  
EndSub
```

### Задание

Используйте согласованный с преподавателем вариант задания (табл.1 лаб. раб. №7), выполненную на его основе таблицу Excel и созданный вами макрос. Модифицируйте созданный вами макрос и напишите новую программу так, чтобы ее основные вычисления производились с переменными VBA. При этом исходные данные первоначально должны быть считаны из таблицы, а результаты вычислений возвращены в нее.

### Порядок выполнения работы

1. Откройте созданную вами рабочую книгу Excel. Скопируйте свою таблицу на новый лист. Удалите в ней все формулы.
2. Определите и, при необходимости, задайте формат ячеек таблицы в соответствии с требованиями вашей задачи.
3. Скопируйте созданный вами макрос в окне редактора кода, удалите все внутренние операторы, оставив только его заголовок и последний оператор EndSub. Измените название процедуры (макроса).
4. Задайте режим обязательного объявления переменных, для чего выше заголовка вставьте строку OptionExplicit.
5. Напишите коды объявления внутренних переменных своей программы и задайтесь их типом данных. Прокомментируйте их в тексте программы.
6. Напишите коды программы считывания исходных данных из таблицы Excel во внутренние переменные программы VBA. Прокомментируйте их в тексте программы.
7. Напишите коды вычислений результирующих значений так, чтобы их результаты оказались во внутренних переменных программы VBA. Прокомментируйте их в тексте программы.
8. Напишите коды программы записи рассчитанных значений в соответствующие ячейки таблицы Excel. Прокомментируйте их в тексте программы.
9. Запустите созданную программу в режиме отладки командами **Debug, StepInto**. На каждом шаге выполнения контролируйте изменение внутренних переменных программы в окне локальных переменных Locals. Убедитесь в правильности выполнения расчетов. При выполнении фрагментов программы, обеспечивающих запись рассчитанных значений в ячейки Excel, дополнительно убедитесь в правильности выполнения этих действий.
10. Удалите в таблице Excel результаты вычислений и проверьте работоспособность программы в режиме **Debug, RunToCursor**. Проконтролируйте значения в окне локальных переменных Locals. Введите точки останова командой **Debug, ToggleBreakpoint** и убедитесь в правильности работы программы.
11. Проверьте правильность комментариев с учетом изменений в тексте программы, дополните и, при необходимости, скорректируйте их.

### Контрольные вопросы

1. Как связаны между собой типы данных Excel и VBA?
2. В чем необходимость использования данных типа Long и Double?
3. Как можно использовать тип данных Type?
4. Как явно объявить переменную в тексте программы?
5. Какие слова в языке программирования называются ключевыми?
6. В чем недостатки метода объявления переменной по умолчанию?
7. Как включить режим обязательного явного объявления переменных?
8. Чем массив отличается от обычной переменной?
9. Как можно использовать возможности динамического объявления размера массива?

10. Как можно использовать свойство Cells() для организации обмена данными между Excel и VBA?

### **Отчет о работе**

Подготовьте отчет о выполненной лабораторной работе. Он должен содержать титульный лист и текст написанной вами программы с построчным комментарием ее действий. Сформулируйте выводы, которые можно сделать по результатам выполненной работы.

## **Лабораторная работа №10. Функции и процедуры**

Составление программ для ЭВМ требует большого объема трудозатрат. Вполне очевидно, что один раз созданные и проверенные программы представляют собой самостоятельную ценность. Программисты стремятся использовать свои разработки в новых программных проектах и, как следствие, создают методы, позволяющие относительно несложно включать ранее разработанные коды в новые программные изделия. Достаточно быстро стало понятно, что обычное механическое копирование кодов в новую программу чревато серьезными ошибками. Для уменьшения вероятности появления ошибок при использовании ранее созданных программ в языках программирования высокого уровня была внедрена концепция так называемых функций и процедур. Для её практической реализации потребовались существенные доработки в системе команд процессора, результатом которых явилось появление специальных команд вызова функции (процедуры) и возврата в точку вызова.

Ключевая идея создания функций и процедур заключалась в обеспечении возможности многократного обращения к одной и той же последовательности кодов из разных мест программы. По своей сути термины функция и процедура в языках высокого уровня взаимозаменяемы. Отличие одного от другого сводится к не принципиальной разнице в способах их оформления в теле программы и, что более важно, в способах оформления вызова. Некоторые языки программирования, например Си, вообще рассматривают только функции. В языке VBA сохранились описатели Function для обозначения функций и Sub для обозначения процедур.

Функцией или процедурой называется самостоятельная программа, предназначенная для решения определенной задачи. Поскольку любая работоспособная программа попадает под это определение, можно сделать вывод, что функцией может быть любая последовательность кодов. На самом деле это действительно так. Написанная нами программа на языке высокого уровня, оформленная в соответствии с правилами языка, оттранслированная и запущенная на выполнение представляет собой функцию, запускаемую, например, операционной системой. В данном случае правила оформления такой программы представляют собой ничто иное, как правила оформления функций операционной системы. Самостоятельный интерес представляют собой правила оформления функций, написанных на языке высокого уровня и вызываемых из других программ, написанных также на том же или другом языке высокого уровня или на ассемблере. Наиболее строго определены правила создания функций в том случае, когда для написания вызываемой программы и собственно функции используется один и тот же язык программирования.

Практический смысл использования функций (процедур) в программировании определяется следующими обстоятельствами. Появляется возможность разбиения большой программы на отдельные составляющие, разработка которых гораздо проще разработки всей программы. Сокращается объем кодов программы за счет удаления повторяющихся действий и замены их вызовами. Повышается надежность программного обеспечения, поскольку программа использует уже многократно проверенные последовательности кодов. Все это, в конечном итоге, ведет к росту производительности труда программиста.

Изложенные соображения играют важную роль при проектировании сложных программ. Так, например, взявшись за выполнение задачи, программист разбивает ее на набор возможно менее связанных между собой функций или процедур, каждая из которых решает некую самостоятельную задачу. Подобный прием называется декомпозицией и широко используется на практике. Очевидно, что для любой более или менее сложной задачи можно найти чрезвычайно большое, если даже не бесконечное количество вариантов декомпозиции. Поэтому выбор конкретного варианта разбиения задачи во многом определяется используемой методологией ее проектирования, а также опытом и пристрастиями разработчика.

При изучении способов создания функций (процедур) следует принимать во внимание следующие моменты:

- Каждая функция (процедура) имеет имя. Это имя является идентификатором и должно быть тем или иным способом объявлено.
- Каждая функция (процедура) имеет свои коды, которые должны быть оформлены заданным языком программирования способом. Эти коды называются определением функции.
- Для решения задачи функция (процедура) может потребовать набор аргументов (исходные данные), которые передаются ей в момент вызова.
- Функция (процедура) может возвращать результаты своих вычислений (возвращаемые данные) в вызывающую программу. Возврат значений может, в частности, производиться через список аргументов.
- Каждая функция (процедура) должна быть вызвана по имени. Если вызов отсутствует, то функция выполняться не будет.

Имя функции (процедуры) рассматривается как ее идентификатор и составляется исходя из правил составления идентификаторов конкретного языка программирования.

Можно выделить два вида функций, которые используются в программах. С одной стороны, это функции, которые созданы программистом для решения своей собственной задачи. В этом случае программист создает коды необходимой ему функции и оформляет их в соответствии с правилами языка программирования. С другой стороны, в программах могут использоваться так называемые библиотечные функции. Обычно с их помощью выполняются некие часто встречающиеся действия: некоторые математические вычисления, операции проверки типов, преобразования форматов, обработки строк, работы со временем и датами и тому подобное. Полный список имеющихся в языке функций можно получить, воспользовавшись, например, системой помощи, которая стандартно вызывается нажатием клавиши F1 при запущенной системе программирования.

При использовании библиотечных функций, обращаясь к ним в своей программе, программист использует написанные другими неизвестными ему программистами коды для решения собственной задачи. Для обеспечения работы всей системы программисту необходимо позаботиться о подключении к программе кодов библиотечных функций. Обычно это происходит на этапе редактирования связей. Заметим, что в большинстве случаев программист не имеет доступа к исходным кодам библиотечных функций и пользуется только описанием их действий (назначением функции) и описанием списка аргументов.

Под определением функции или процедуры обычно понимают создаваемую программистом последовательность операторов программы, которая после трансляции может быть вызвана из другой программы. Если в программе используются встроенные процедуры или функции, то об их определении программисту беспокоиться не приходится. Их объектные коды заранее размещены в специальных библиотеках, используемых компилятором, и автоматически подключаются к итоговой программе. Задумываться об определениях программисту надо в том случае, когда он хочет создать собственные функции или процедуры.

Формально функция в VBA может быть описана так:

```
[Public или Private] [Static] Function Имя [(СписокАргументов)] [As Тип]
[Операторы]
[Имя=Выражение]
[ExitFunction]
[Операторы]
[Имя=Выражение]
EndFunction
```

Если указано ключевое слово **Public** (используется по умолчанию), процедура может быть вызвана из других процедур любых модулей. Ключевое слово **Private** означает, что процедура может быть вызвана только из того модуля, в котором она описана. Из соображений повышения надежности программирования рекомендуется, как правило, использовать ключ **Private**.

Установленный ключ **Static** означает, что локальные переменные процедуры сохраняют свои значения между вызовами и могут быть использованы в последующих вычислениях при следующем вызове процедуры (время жизни переменной). Из соображений повышения надежности программирования использовать ключ **Static** не рекомендуется.

Имя функции – это обычный идентификатор языка VBA.

СписокАргументов представляет собой перечисление аргументов функции. Он имеет еще одно название: список формальных параметров. Функция может иметь один аргумент (формальный параметр) или несколько. Как и обычные переменные, формальные параметры имеют определенный тип. Их основным отличием от обычных переменных является то обстоятельство, что под их хранение не выделяется память машины, а сами они используются в определении функции только для указания последовательности действий с аргументами функции. Каждый элемент списка формальных параметров имеет следующий формат:

```
[Optional] [ByVal или ByRef] [ParamArray] ИмяПеременной[()] [As Тип] [=поУмолчанию]
```

Ключевое слово **Optional** означает, что элемент списка является необязательным аргументом и должен иметь тип **Variant**. Все последующие элементы списка должны иметь такой же ключ и тип. Необязательные аргументы могут отсутствовать в списке переменных функции при записи оператора ее вызова.

Ключ **ByVal** означает, что параметр передается по значению. Если задан этот ключ, то вызывающая программа может передать в функцию значение аргумента, однако изменить это значение функция никаким способом не может. Этот прием призван защитить данные вызывающей программы, и может использоваться как основной при односторонней передаче данных от вызывающей программы к функции.

Ключ ByRef (используется по умолчанию) указывает, что параметр передается по ссылке. Это означает, что функции известен физический адрес памяти формального параметра. При необходимости функция может произвести запись по этому адресу (например, оператором присваивания). Подобный прием оказывается удобным для возврата результатов вычислений функции в вызывающую программу через список формальных параметров в том случае, когда оказывается необходимым вернуть больше одного параметра. При использовании процедур это вообще единственный способ возврата результатов вычислений.

Ключевое слово ParamArray может быть использовано только с последним элементом списка формальных параметров и позволяет передавать динамически объявляемый массив.

Ключ Тип представляет собой тип передаваемого параметра, а значение по умолчанию может использоваться только с ключом Optional и задает значение переменной.

После заголовка функции следует конечное число обычных операторов языка VBA, представляющих собой тело определения функции. Если в их состав входит оператор объявления переменных Dim, то имеет место объявление собственных локальных переменных функции. Если в заголовке функции не указан ключ Static, то эти переменные не сохраняют свои значения между вызовами, и каждый раз значения в них должны записываться заново. Кроме операторов объявления, в состав тела определения могут входить операторы присваивания, цикла и другие. В качестве их аргументов могут выступать как локальные переменные, константы, так и формальные параметры. Последние выступают как полноправные участники любых операций и операторов с той лишь оговоркой, что свое конкретное значение они получают только в момент вызова.

Результатом работы функции является некое значение, например число, которое вычисляется в теле функции. Возвращаемое значение должно иметь некий тип, указанный в заголовке функции как As Тип, соответствующий типу возвращаемого функцией значения. Для указания того, что все-таки является результатом вычислений функции и должно быть возвращено в вызывающую программу, в определении функции записывается отдельный оператор присваивания. В его левой части указывается Имя функции (из ее заголовка), а в правой – возвращаемое значение.

Формальное описание процедуры в VBA похоже на формальное описание функции и имеет вид:

```
[PublicилиPrivate] [Static] SubИмя [(СписокАргументов)]  
    [Операторы]  
EndSub
```

Формат элементов списка формальных параметров процедуры аналогичен формату формальных параметров функции. Таким образом, кроме ключевых слов заголовка и окончания, единственным принципиальным отличием определения функции от определения процедуры является наличие ее в тексте определения функции оператора [Имя=Выражение], указывающего возвращаемое в точку вызова значение.

Встречаются задачи, в которых в точку вызова необходимо возвращать не одно, а несколько значений. Поскольку функция может вернуть в вызывающую программу через оператор присваивания только одно значение, похожие задачи приходится решать способом передачи данных через список формальных параметров по ссылке. Если формальный параметр описан с ключом ByRef, то это означает, что вызываемая функция получает в свое распоряжение не копию данных, а адрес ячейки памяти, в которой эти данные находятся. Как следствие, у вызываемой функции появляется возможность изменить содержимое ячейки памяти вызывающей программы. Для этого в определении функции оператором присваивания задаются необходимые значения формальному параметру. В момент вызова процедуры (функции) формальному параметру ставится в соответствие фактическая ячейка памяти вызывающей программы. Именно в ней и произойдут указанные в определении изменения.

Вызов процедуры в языке VBA производится из любого места основной (вызывающей) программы за счет включения в ее текст специального оператора вызова. Вызов процедуры записывается как отдельный оператор с использованием ключевого слова Call. После него должно стоять имя процедуры и список ее фактических параметров, записанный в круглых скобках. Под фактическими параметрами понимаются имена ячеек памяти, объявленных в вызывающей программе. Очевидно, что если процедуре должно быть передано некоторое значение в виде аргумента, то вызывающая программа предварительно должна занести это значение в свою ячейку с использованием, например, оператора присваивания. Далее эта ячейка должна быть указана на соответствующем месте в списке формальных параметров.

**Примечание.** Альтернативным и часто используемым вариантом вызова процедур в VBA является просто запись имени процедуры с перечислением ее аргументов (фактических параметров) без заключения их в круглые скобки.

В отличие от процедуры, функция возвращает некоторое значение в точку вызова. Поэтому вызов функции производится с оператором присваивания. В левой части оператора указывается имя переменной, куда должен быть записан результат вычислений функции, а в правой ее имя и в круглых скобках аргументы (фактические параметры). Фактические аргументы функции выбираются из числа ячеек вызывающей программы.

Имя функции (процедуры) заносится компилятором в таблицу идентификаторов при первом вызове или при компиляции ее кодов, оформленных в виде текста программы и соответствующих заголовков с окончаниями (Sub Имя ([Аргументы]) [Операторы тела функции] EndSub или Function Имя ([Аргументы]) As Тип [Операторы тела функции] EndFunction).

**Пример 11.** Пример программы, реализующей задачу рис. 1 лаб. раб. №7 с использованием процедуры задания цвета шрифта в ячейке Excel.

```
Sub Расчет_заработной_платы3()  
Dim Начислено(1 To 4) As Currency, Налог(1 To 4) As Currency, _  
K_Выдаче(1 To 4) As Currency, i As Integer  
For i = 1 To 4  
    Начислено(i) = Cells(i + 1, 2) 'В первой ячейке массива Начислено записывается  
    'содержимое второй строки и второй колонки исходной таблицы Excel
```



```

If Начислено(i) > 1000000 Then
    Начислено(i) = 1000000
    Call Изменение_цвета_шрифта_в_ячейке(i + 1, 2, "Желтый") 'Вызов процедуры с
    'ключевым словом Call. Параметры процедуры заключены в круглые скобки
Else
    End If
If Начислено(i) < 0 Then
    Начислено(i) = 0
    Изменение_цвета_шрифта_в_ячейке(i + 1, 2, "Красный") 'Вызов процедуры без
    'ключевого слова Call. Параметры процедуры в круглые скобки не заключаются
Else
    Call Изменение_цвета_шрифта_в_ячейке(i + 1, 2, "Сброс")
End If
Налог(i) = Начислено(i) * 0.12 'Рассчитывается значение налога и запоминается
'в соответствующей ячейке
Cells(i + 1, 3) = Налог(i) 'Значение налога возвращается в таблицу Excel
К_Выдаче(i) = Начислено(i) - Налог(i) 'Рассчитывается значение к выдаче
'и запоминается в соответствующей ячейке
Cells(i + 1, 4) = К_Выдаче(i) 'Значение к выдаче возвращается в таблицу Excel
Next i
End Sub
Sub Изменение_цвета_шрифта_в_ячейке(Строка As Integer, Столбец As Integer, Цвет As String)
    Dim C As Integer
    Select Case Цвет
        Case "Красный": C = 3
        Case "Желтый": C = 6
        Case "Зеленый": C = 10
        Case Else: C = 0 'Автоматический выбор (Авто)
    End Select
    Cells(Строка, Столбец).Font.ColorIndex = C
End Sub

```

**Примечание.** Интегрированная среда разработки VBA в окне редактора кодов предлагает в качестве сервиса возможность указания имени переменной и типа данных при наборе операторов вызова функции или процедуры. Если функция или процедура ранее была объявлена и была выполнена компиляция проекта, после набора ее имени всплывает перечень ее аргументов.

При программировании в Excel иногда возникает необходимость создания дополнительных по отношению к стандартному библиотечному набору функций. Такие функции могут быть запрограммированы в ячейках таблицы обычным для Excel способом (**Вставка, Функция...**). Далее программист выбирает нужную ему *Категорию* функции, в соответствии с которой произведена их классификация. Если создать в модуле проекта VBA функцию, имеющую атрибут Public, используемый по умолчанию, то эта функция появится в списке библиотечных функций Excel в категории *Определенные пользователем*. Этот прием оказывается удобным для введения в систему Excel дополнительных возможностей программирования нестандартных, но многократно используемых действий. Так с его помощью легко реализовать возможности операторов ветвления и цикла, которые в обычной конфигурации оказываются недоступными.

**Пример 12.** Пример создания пользовательской функции Excel на VBA.

```
'Функция, определенная пользователем
PublicFunctionРасчет_налога(НачисленоAsInteger)
    Расчет_налога = Начислено * 0.12
EndFunction
```

### Задание

Используйте согласованный с преподавателем вариант задания (табл.1 лаб. раб. №7), выполненную на его основе таблицу Excel, написанную программу вычислений в таблице с использованием переменных VBA. Модифицируйте созданную вами программу так, чтобы перенести часть вычислений в процедуру или функцию. При этом исходные данные первоначально должны быть считаны из таблицы Excel, а результаты вычислений возвращены в нее.

### Порядок выполнения работы

1. Откройте созданную вами рабочую книгу Excel. Скопируйте свою таблицу на новый лист. Удалите в ней все формулы. Запустите интегрированную среду разработки VBA.

2. Выберите фрагменты кодов вашей программы, которые будут реализованы в виде процедуры или функции.

3. Создайте определение процедуры на основе выбранного вами фрагмента. Определите перечень формальных параметров процедуры и задайте их тип.

4. Замените выбранные фрагменты кодов программы на вызовы процедуры. Объявите и задайте фактические параметры процедуры.

5. Создайте определение функции на основе выбранного вами фрагмента. Определите перечень формальных параметров функции и задайте их тип.

6. Замените выбранные фрагменты кодов программы на вызовы функции. Объявите и задайте фактические параметры функции.

7. Проверьте работоспособность всей программы в режиме отладчика с использованием команды **StepInto**.

8. Ознакомьтесь с принципом отладки программы командой **StepOver**, автоматически выполняющим вызываемую процедуру или функцию и командой **StepOut**автоматически завершающей выполнение текущей процедуры или функции.

9. Введите в процедуру или функцию проверку значений аргументов с помощью оператора **IfThenElseEndIf**.

10. Воспользуйтесь оператором **SelectCaseEndSelect** в процедуре или функции для демонстрации возможностей его работы.

11. Отметьте в таблице Excel факт выхода значения данных за пределы диапазона. Для этого, например, измените цвет шрифта в ячейке.

12. Проверьте правильность комментариев с учетом возможных изменений в тексте программы и, при необходимости, измените и их.

13. Создайте функцию, определенную пользователем.

14. Перейдите на лист Excel. Выполните команду **Вставка, Функция...** Выберите из категории *Определенные пользователем* запрограммированную вами функцию. Задайте данные и убедитесь в правильности ее выполнения.

15. Проверьте работоспособность функции в режиме отладчика с использованием команды **StepInto**.

16. Проверьте правильность комментариев с учетом изменений в тексте программы, дополните и, при необходимости, скорректируйте их.

### **Контрольные вопросы**

1. Чем отличается объявление функции от ее определения?
2. В чем заключается практический смысл использования функций или процедур?
3. Что такое список формальных параметров и чем формальные параметры отличаются от фактических?
4. В каких случаях формальный параметр целесообразно передавать по ссылке, а в каких по значению?
5. Как вызвать библиотечную функцию VBA?
6. Какой смысл имеет задание типа функции в ее определении?
7. Чем отличается вызов функции от вызова процедуры?
8. Как создать определенную пользователем функцию Excel?
9. Каковы особенности отладки программы, использующей функции или процедуры?
10. Как результаты работы функции или процедуры могут быть получены в вызывающей программе?

### **Отчет о работе**

Подготовьте отчет о выполненной лабораторной работе. Он должен содержать титульный лист, текст написанной вами процедуры (функции). Приведите алгоритм и текст созданной вами функции пользователя. Сформулируйте выводы, которые можно сделать по результатам выполненной работы.

## **Лабораторная работа №11. Классы и объекты**

Хотя применение функций и процедур существенно упрощает создание программ (повышает производительность труда программиста), их использование в сложных программных системах наталкивается на ряд принципиальных ограничений. По своей сути обычная функция (процедура) представляет собой так называемый автомат без памяти. Это означает, что ее реакция на входное воздействие однозначно определена в момент разработки и никак не зависит от текущей ситуации.

Некоторые языки программирования (в том числе и VBA) допускают использование так называемых глобальных переменных. Написанная с использованием таких переменных функция может иметь существенно больший набор реакций на входное воздействие, поскольку в этом случае отклик функции зависит не только от текущих аргументов, но и от состояния ее глобальных переменных. Поскольку эти переменные существуют все время работы программы, функция, входящая в состав программы, может использовать их, в частности, для сохранения результатов вычислений от вызова к вызову. Аналогично можно использовать и статические переменные. Отметим, что в отличие от обычной функции, такая функция представляет собой автомат с памятью.

Исследования в области надежности программного обеспечения показали, что использование глобальных и статических переменных существенно увеличивает вероятность программной ошибки. Их основной причиной является возникающая неопределенность момента изменения состояния переменной, так как доступ к глобальной переменной имеет, в том числе и ошибочно, любая другая процедура или функция программы. Это обстоятельство в конечном итоге привело к появлению целой методологии программирования – так называемому структурному программированию. В его основе лежит концепция проектирования программы сверху вниз, модульное программирование и структурное кодирование. Предполагается, что модуль в структурном программировании представляет собой законченную конструкцию с одним входом и одним выходом, что, в частности, запрещает использование в нем глобальных переменных, которые по своей сути являются средством дополнительного воздействия на поведение модуля. Поэтому при проектировании программы сверху вниз заранее оговаривается перечень всех аргументов модулей, причем они обязательно передаются через список формальных параметров.

С другой стороны, глобальные и статические переменные позволяют существенно упростить межмодульные связи и сократить количество аргументов функции (процедуры). Поэтому стремление ряда руководителей административно внедрить методы структурного программирования наталкивалось на явное или скрытое сопротивление программистов, для которых подобные действия приводили, в конечном итоге, к усложнению межмодульных интерфейсов.

Попытки найти компромисс между потребностями практики программирования с одной стороны и требованиями обеспечения надежности программирования с другой привели к созданию специфических типов функций и процедур, называемых объектами. В отличие от обычных функций и процедур, объекты имеют переменные, значения которых сохраняются от обращения к

обращению. В то же время доступ к этим переменным возможен только через сам объект за счет использования его собственных свойств и методов. Это обстоятельство существенно снижает вероятность ошибки программирования связанной с несанкционированным изменением значения глобальной или статической переменной.

Функции и процедуры в программировании создавались, в первую очередь, для обеспечения возможности их многократного вызова из различных точек программы. Поскольку реакция функции как автомата без памяти на одинаковое воздействие всегда одинакова, различные по функциональному назначению, но одинаковые по алгоритму фрагменты программы могут реализовываться одним и тем же программным кодом. Так, например, все функции печати имеют один и тот же алгоритм. Поэтому было бы заманчиво написать универсальную функцию печати, не зависящую от вида, типа и состояния устройства. С другой стороны, при печати данных на разные устройства, приходится принимать во внимание их текущие настройки, состояние и историю работы. Если эти данные брать не из аргументов функции и не из глобальных или статических переменных, то приходится создавать механизм их хранения. Одним из вариантов такого механизма является создание собственных функции для каждого устройства, имеющих одинаковый алгоритм работы и программный код, но разные для каждого устройства ячейки данных, хранящие информацию об их состоянии. В конечном итоге такие функции получили название объектов.

Объект – это комбинация кода и данных. Код объекта фактически представляет собой набор функций или процедур одинаковый для всех схожих объектов. С каждым объектом связывается свой набор данных, который может быть изменен средствами кода объекта. Этот набор появляется в памяти машины в момент создания объекта и исчезает вместе с его удалением.

Для реализации подобного подхода к программированию функций, систематизации объектов и стандартизации принципов работы с ними, в языки программирования было введено понятие класс. Класс – это некоторое множество объектов, имеющих общую структуру и поведение. Фактически класс содержит набор функций и процедур, описывающих свойства и поведение объектов. Этот набор хранится в единственном экземпляре в виде программного кода и используется всеми объектами. Кроме этого, класс содержит описание структуры данных каждого объекта.

Как было показано в примере 9 (лаб. раб. №9), структура представляет собой специфический тип данных (в языке VBA тип данных, определяемый пользователем). Там же было отмечено, что для создания собственно переменной типа объявленной структуры, эта переменная должна быть явно описана в программе и иметь свое уникальное имя. Поскольку за каждым элементом такой переменной закреплены соответствующие ячейки памяти, можно говорить, что она обладает неким состоянием, определяемым содержимым закрепленных за ней ячеек памяти, и идентичностью, определяемой именем переменной в программе.

При введении в языки программирования понятия объект к описанным уже характеристикам состояния и идентичности добавили характеристику поведения. Под поведением обычно понимают реакцию объекта на внешнее

воздействие, сводящуюся к изменению его состояния. В отличие от функции с глобальными и статическими переменными, возможность изменения состояния объекта существенно ограничивается и определяется заранее, что позволяет сохранить надежность программирования на разумном уровне. Поведение объекта описывается набором функций и процедур класса. Наконец, как и в случае структуры, характеристика идентичности представляет собой свойство объекта, отличающее его от других объектов. Это имя задается в момент создания объекта.

Модуль класса содержит коды общих для всех объектов функций и процедур и описание структуры данных объекта. Для выделения памяти под хранение переменных объекта необходимо выполнить набор действий по его созданию. Далее каждый объект использует общие для всего класса процедуры и функции, но оперирует с собственными данными, которые хранятся в памяти до удаления объекта.

В языке VBA для изменения состояния объекта пользуются так называемыми свойствами. Все объекты одного класса имеют одинаковый набор свойств. Конкретный набор свойств объекта, возможность их считывания и изменения определяется при создании класса в виде набора функций специального вида. Поведение объекта в языке VBA задается методами и событиями. По своей сути метод представляет собой обычную процедуру. Возможные методы для объекта также описываются на этапе создания класса.

Событие представляет собой действие, распознаваемое объектом, для которого можно запрограммировать отклик. Событие вызывается действиями пользователя (например, щелчок мышью) или генерируются системой (например, деление на ноль).

Создание класса в языке VBA представляет собой типовую последовательность действий. Сначала командой **Insert, ClassModule** интегрированной системы отладки VBA создается так называемый модуль класса и ему присваивается имя, являющееся далее именем пользовательского класса. После этого описываются переменные класса. Обратите внимание на то, что при этом описании определяется только структура ячеек данных объектов класса. Сами переменные класса получают конкретные значения адресов в памяти машины только после того, когда на основе класса будут создаваться объекты, причем каждый объект будет иметь свой индивидуальный набор таких ячеек.

Затем определяется процедура инициализации класса `SubClass_Initialize()`. Эта процедура выполняется каждый раз, когда создается новый объект и может быть использована, например, для задания начальных значений переменным класса, динамического переобъявления размеров массивов в соответствии с требованиями конкретной задачи, чтения файлов и т.п. Далее может быть создана процедура `SubClass_Terminate()`. Она описывает действия, которые надо выполнить перед удалением объекта (например, печать результата). Если начальных или завершающих действий с объектом не требуется, то эти процедуры можно не создавать. Синтаксис определения процедур `SubClass_Initialize()` и `SubClass_Terminate()` имеет вид:

```
Private Sub Class_Initialize()
```

```
[Операторы]
End Sub
```

```
Private Sub Class_Terminate()
[Операторы]
EndSub
```

После этого создаются функции вида PropertyGet, PropertyLet и PropertySet (функция PropertySet позволяет устанавливать значения свойств объекта в составе класса), позволяющие читать и задавать значения свойств переменных класса. Их формальное описание имеет вид:

```
[Public или Private] [Static] PropertyGet Имя [(СписокАргументов)] [As Тип]
[Операторы]
[Имя=Выражение]
[ExitFunction]
[Операторы]
[Имя=Выражение]
EndProperty
```

```
[PublicилиPrivate] [Static] PropertyLetИмя [(СписокАргументов)]
[Операторы]
[Имя=Выражение]
[ExitFunction]
[Операторы]
End Property
```

```
[Public или Private] [Static] Property Set Имя [(СписокАргументов)]
[Операторы]
[Имя=Выражение]
[ExitFunction]
[Операторы]
EndProperty
```

Наконец, создаются методы класса, которые оформляются в виде обычных процедур Sub.

```
[PublicилиPrivate] [Static] SubИмя [(СписокАргументов)]
[Операторы]
EndSub
```

**Примечание.** Непосредственно запрограммировать события класса нельзя, однако свойства классов могут использовать события других стандартных объектов VBA.

**Пример 13.** Пример программы описания класса, реализующий задачу рис. 1 лаб. раб. №7. Дополнительно в класс введено свойство, позволяющее изменять количество строк таблицы.

```
Dim Фамилия() As String, Начислено() As Currency, Налог() As Currency, _
К_Выдаче() AsCurrency
```

Переменные класса представляют собой набор динамически объявляемых массивов. Необходимость динамического объявления связана с неопределенностью числа строк 'таблицы конкретного объекта

```
DimРазмер_таблицыAsInteger, Ставка_налогаAsSingle
```

```

Private Sub Class_Initialize()
Размер_таблицы = 4
Ставка_налога = 0.12
ReDim Фамилия(1 To Размер_таблицы)
ReDim Начислено(1 To Размер_таблицы)
ReDim Налог(1 To Размер_таблицы)
ReDim К_Выдаче(1 To Размер_таблицы)
EndSub

PrivateSubClass_Terminate() 'Действиянепредусматриваются
EndSub

Sub Расчет_заработной_платы()
For i = 1 To Размер_таблицы
    Начислено(i) = Cells(i + 1, 2) 'В первую ячейку массива Начислено записывается
    'содержимое второй строки и второй колонки исходной таблицы Excel
    If Начислено(i) > 1000000 Then
        Начислено(i) = 1000000
        Call Изменение_цвета_шрифта_в_ячейке(i + 1, 2, "Желтый") 'Вызов процедуры
        'с ключевым словом Call. Параметры процедуры заключены в круглые скобки
    Else
    EndIf
    If Начислено(i) < 0 Then
        Начислено(i) = 0
        Изменение_цвета_шрифта_в_ячейке i + 1, 2, "Красный" 'Вызов процедуры без
        'ключевого слова Call. Параметры процедуры в круглые скобки не заключаются
    Else
        Call Изменение_цвета_шрифта_в_ячейке(i + 1, 2, "Сброс")
    EndIf
    Налог(i) = Начислено(i) * Ставка_налога 'Рассчитывается значение налога и запоминается
    'в соответствующей ячейке
    Cells(i + 1, 3) = Налог(i) 'Значение налога возвращается в таблицу Excel
    К_Выдаче(i) = Начислено(i) - Налог(i) 'Рассчитывается значение к выдаче
    'и запоминается в соответствующей ячейке
    Cells(i + 1, 4) = К_Выдаче(i) 'Значение к выдаче возвращается в таблицу Excel
Next i
EndSub

SubИзменение_цвета_шрифта_в_ячейке(СтрокаAsInteger, СтолбецAsInteger, ЦветAsString)
DimCAsInteger
SelectCaseЦвет
    Case "Красный": C = 3
    Case "Желтый": C = 6
    Case "Зеленый": C = 10
    CaseElse: C = 0 'Автоматический выбор (Авто)
EndSelect
Cells(Строка, Столбец).Font.ColorIndex = C
EndSub

PropertyLetЧисло_строк_таблицы(РазмерAsInteger)
Размер_таблицы = Размер

```



```

ReDim Фамилия(1 To Размер_таблицы)
ReDim Начислено(1 To Размер_таблицы)
ReDim Налог(1 To Размер_таблицы)
ReDim К_Выдаче(1 To Размер_таблицы)
EndProperty

```

```

PropertyGetЧисло_строк_таблицы() AsInteger
Число_строк_таблицы = Размер_таблицы
EndProperty

```

Поскольку объекты как элементы языка программирования создавались на основе структур, процедур и функций, нотация, используемая для обращения к ним, сохранилась. Так символ точка, используемый в структурах для разделения общего имени переменной и ее составной части, используется при записи объектов для разделения его имени и свойства или метода. Как это принято в функциях, аргументы свойств объекта, если они требуются, заключаются в круглые скобки. При использовании методов аналогично правилам вызова процедур VBA список параметров следует после указания метода и в круглые скобки не заключается.

```

Объект.Свойство = Значение_свойства
Объект.Свойство1(параметр1, параметр2, ..., параметрN= Значение_свойства
Значение_свойства = Объект.Свойство
Значение_свойства = Объект.Свойство1(параметр1, параметр2, ..., параметрN)
Объект.Метод
Объект.Метод1 параметр1, параметр2, ..., параметрN

```

**Примечание.** Интегрированная среда разработки VBA в окне редактора кодов предлагает в качестве сервиса возможность конкретного выбора имени свойства или метода, допустимых для данного объекта, из автоматически раскрывающегося списка. Если свойства или методы ранее были включены в состав класса и выполнена компиляция проекта, после набора символа точки автоматически открывается список возможных имен. Этой возможностью необходимо пользоваться для избегания синтаксических и логических ошибок при наборе текста программы.

После того, как класс создан, можно на его основе создавать конкретные экземпляры, а также выполнять с ними различные действия. Для создания объекта на основе имеющегося класса необходимо выполнить следующую последовательность действий:

- объявить переменную оператором Dim и указать ее тип как имя используемого класса;
- создать объект оператором Set с именем ранее объявленной переменной, используя ключевое слово New и указание имени класса. Синтаксис оператора Set:

```
Set ОбъектнаяПеременная = [New] ОбъектноеВыражение
```

или

```
Set ОбъектнаяПеременная = Nothing
```

Ключевое слово New используется при создании нового экземпляра класса, а ключевое слово Nothing позволяет освободить системные ресурсы (память) от объекта, который в дальнейшем использоваться не будет.

После выполнения этих действий в памяти создается набор переменных, связанных с указанным объектом. С этого момента оказываются доступными свойства и методы класса в отношении созданного объекта. Аналогично можно создать еще несколько объектов используемого класса и выполнять с ними разнообразные действия. При этом значения переменных класса и, следовательно, свойств разных объектов могут быть различными, а при использовании одних и тех же методов будет получен разный результат.

Если работа с объектом завершена, то он может быть удален из памяти оператором Set с ключевым словом Nothing. После его выполнения занимаемая объектом память освобождается. Кроме этого, все созданные объекты автоматически удаляются из памяти в момент завершения работы программы.

**Пример 14.** Программа использования класса, реализующая задачу рис. 1 лаб. раб. №7.

```
Sub Расчет_зарботной_платы4()  
'Объявление переменной с типом созданного класса  
Dim Первый_объект As Ведомость_зарботной_платы, iAsInteger  
'Создание объекта  
Set Первый_объект = New Ведомость_зарботной_платы 'Использование метода объекта  
Первый_объект.Расчет_зарботной_платы  
'Использование свойства объекта. Задание нового числа строк таблицы  
Первый_объект.Число_строк_таблицы = 3  
'Чтение текущего числа строк таблицы  
i = Первый_объект.Число_строк_таблицы()  
'Удаление объекта  
Set Первый_объект = Nothing  
EndSub
```

Возможность программирования классов и создания на их основе необходимого количества однотипных объектов оказывает важное влияние на способ декомпозиции сложной программной системы при ее проектировании. Проектировщик создает описание некой сущности, являющейся предметом исследования и проектирования, в виде программной модели. Описание системы взаимодействия объектов между собой позволяет составлять совокупную модель описываемой сущности в виде множества взаимодействующих по определенным правилам объектов различных фрагментов сущности. В этом случае декомпозиция исходной задачи может рассматриваться как иерархия классов объектов с учетом их взаимодействия. Подобный прием называется объектной декомпозицией.

### Задание

Используйте согласованный с преподавателем вариант задания (табл.1 лаб. раб. №7), выполненную на его основе таблицу Excel, написанную программу вычислений в таблице. Создайте на ее основе класс, позволяющий производить требуемое количество таблиц и обработку данных в них.

## Порядок выполнения работы

1. Откройте созданную вами рабочую книгу Excel. Скопируйте свою таблицу на новый лист. Удалите в ней все формулы. Запустите интегрированную среду разработки VBA.

2. Воспользовавшись командой **Insert, ClassModule** создайте модуль класса. Если эти действия выполняются в первый раз, то в окне проекта появится папка ClassModules, а в ней запись Class1. Если в проекте классы уже создавались, то вставка нового модуля класса просто добавит запись в папку ClassModules. Далее командой **View, Properties Window** вызовите окно свойств создаваемого класса. В этом окне задайте имя класса вместо предлагаемого по умолчанию имени Class1. Выполните команду **Debug, Compile VBA Project**.

3. В окне редактора кодов скопируйте созданную вами в процессе выполнения предыдущей работы процедуру или функцию, предназначенную для вычислений в таблице, в модуль класса. Далее используйте ее в качестве основы для программирования свойств и методов создаваемого класса.

4. Определите перечень переменных класса. Скорее всего, в их числе будут внутренние переменные вашей процедуры и, возможно, некоторые дополнительные. Выделите из состава используемой процедуры (функции) объявления и перенесите их в начало модуля класса в виде самостоятельных строк. Добавьте необходимые дополнительные переменные. Обратите внимание на массивы переменных. Если они также являются переменными класса, их целесообразно объявлять как динамические массивы (пример 8). В этом случае надо предусмотреть переменную класса для хранения актуального размера массива, а также свойства, позволяющие задать ей новое значение и предусматривающее изменение размеров массивов (ReDim).

5. Создайте процедуру SubClass\_Initialize() и разработайте коды действий, выполняемых в момент создания объекта. В частности, в этой процедуре можно предусмотреть объявление динамических массивов в соответствии с неким начальным их размером.

6. Создайте процедуру SubClass\_Terminate() и разработайте коды действий, выполняемых в момент удаления объекта. В частном случае процедура может не выполнять никаких действий.

7. Создайте функции вида PropertyGet и PropertyLet, позволяющие читать и задавать значения переменных класса и выполнять на их основе обработку данных. Если в качестве основы программирования класса вы использовали функцию, то оформите ее в виде одной из процедур PropertyGet или PropertyLet.

8. Если для основы программирования класса вы использовали процедуру, то она будет являться одним из методов класса. При необходимости создайте еще методы класса, оформленные в виде обычных процедур Sub.

9. Напишите программу создания объекта на основе разработанного вами класса. Для этого объявите переменную с типом созданного вами класса. Далее создайте новый объект. Используйте методы класса. Используйте свойства класса.

10. Запустите созданную программу в режиме отладки командами **Debug, Step Into**. На каждом шаге выполнения контролируйте изменение внутренних переменных программы в окне локальных переменных **Locals**. Убедитесь в

правильности выполнения расчетов. При выполнении фрагментов программы, обеспечивающих запись рассчитанных значений в ячейки Excel, дополнительно убедитесь в правильности выполнения этих действий.

11. Проверьте правильность комментариев с учетом изменений в тексте программы, дополните и, при необходимости, скорректируйте их.

### **Контрольные вопросы**

1. Какие проблемы возникают при практическом использовании функций или процедур?
2. В чем заключаются основные идеи метода структурного программирования?
3. В чем отличие автомата с памятью от автомата без памяти?
4. Каковы преимущества и недостатки использования глобальных переменных в тексте программы?
5. Чем объекты отличаются от обычных функций и процедур?
6. Чем класс отличается от объекта?
7. Каким образом на этапе выполнения программы можно получить доступ к переменным класса?
8. Чем метод класса отличается от свойства класса?
9. Что надо сделать для создания объекта класса?
10. Что такое событие?

## **Лабораторная работа №12. Файлы с последовательным доступом. Файлы с произвольным доступом**

Данные в памяти ЭВМ хранятся в виде двоичных чисел. Единственное, что может сделать процессор с данными – это извлечь содержимое некой ячейки памяти, выполнить над ним некоторое заранее оговоренное и выбранное из перечня возможных действие и занести число (результат) назад в память в ту же или другую ячейку. Вполне естественным является следующий вопрос, каким способом числа попали в ячейку памяти первоначально?

Существует всего четыре варианта ответа. Во-первых, это данные могли остаться в ячейке памяти от предыдущей программы или, если программа загружается в память сразу после включения машины, в ячейке памяти осталась случайная комбинация установок триггеров, возникающая после подачи напряжения на ОЗУ. Часто такие данные называют мусором. Во-вторых, данные могут быть размещены в ячейке вместе с программой, то есть сама программа при компиляции предусматривает некое начальное значение в конкретной ячейке памяти. В-третьих, число могло попасть в ячейку в результате выполнения команды процессора на запись данных в ОЗУ, например, при выполнении оператора присваивания. Наконец, в-четвертых, число могло быть занесено в ячейку памяти в результате выполнения команды ввода.

На первый взгляд существенных различий между двумя последними вариантами нет. Тем не менее, следует принимать во внимание следующее обстоятельство: в третьем варианте заносимое число является результатом вполне конкретных действий над данными, которые при необходимости могут быть повторены. В то же время, в четвертом варианте занесенное число представляет собой результат реального физического воздействия на устройство ввода в данный момент времени, которое может быть уникальным и никогда более не повторяющимся.

Вполне естественным было бы ожидать то, что любой язык программирования высокого уровня содержит в своем составе команды или операторы ввода вывода. Поскольку на первом этапе развития вычислительной техники способы подключения устройств ввода-вывода к процессору существенно рознились, языки программирования предусматривали отдельные операторы для вывода на печать, вывода на дисплей, ввода с клавиатуры, файловой работы. В настоящее время произошла унификация подобного рода операторов. Так, в языке VBA сохранились несколько операторов, смысл которых представлен в табл. 7. Основным назначением базовых операторов ввода вывода VBA является работа с файлами. Файл представляет собой единицу хранения данных, имеющих конкретный смысл. Так, файл может быть программой (исполняемыми кодами), исходным текстом, документом, просто хранилищем записей. Физически файл хранится, как правило, на накопителях на магнитных дисках, хотя операционная система машины рассматривает любое внешнее устройство как приемник или источник файлов.

В VBA существует три способа организации данных в файле. Эти типы определяют и тип доступа к файлу. Различают:

- последовательные файлы, предназначенные для чтения и записи последовательных блоков символьных данных, представляющих собой последовательность кодов символов, включая служебные;
- файлы произвольного доступа, предназначенные для записи и чтения данных, структурированных как записи фиксированной длины;
- двоичные файлы, предназначенные для записи и чтения числовых данных произвольной длины и представляющие собой частный случай файла произвольного доступа с длиной записи 1 байт.

Перед началом работы программы с файлом он должен быть открыт инструкцией `Open`, которая задает имя открываемого файла (включая указание пути к нему). При открытии указывается номер открываемого канала системы. Дополнительно может быть задан тип файла (последовательного доступа, произвольного доступа, двоичный), ключ записи и состояние файла (для чтения, записи или добавления). Кроме этого для файла может быть указан его задаваемый размер в байтах. Номер свободного канала может быть определен предварительно с помощью инструкции `FreeFile`.

Оператор `Open` –разрешает выполнение операций ввода/вывода при работе с файлом.

Синтаксис:

*Open* Путь *For* Режим [*Access* Доступ] [*Блокировка*] *As* [ # ]НомерФайла[*Len*=Длина]

Параметр	Описание
Путь	Строковое выражение, указывающее имя файла
Режим	Устанавливает режим работы с файлом. Допустимые значения: <i>Append</i> , <i>Binary</i> , <i>Input</i> , <i>Output</i> или <i>Random</i>
Доступ	Устанавливает операции, разрешенные с открытым файлом. Допустимые значения: <i>Read</i> , <i>Write</i> или <i>Read Write</i>
Блокировка	Устанавливает операции, разрешенные с открытым файлом другим процессам. Допустимые значения: <i>Shared</i> , <i>Lock Read</i> , <i>Lock Write</i> и <i>Lock Read Write</i>
НомерФайла	Допустимый номер файла. Число в интервале от 1 до 255. Обратите внимание на то, что параметру НомерФайла предшествует символ #. Значение НомерФайла нельзя изменять, пока файл открыт. Но при следующем открытии файла НомерФайла может быть другим числом
Длина	Число, меньшее либо равное 32 767 (байт). Для файлов, открытых в режиме <i>Random</i> , это значение является длиной записи. Для файлов с последовательным доступом это значение является числом буферизуемых символов. Про инструкцию <i>open</i> важно также знать, что во время ее работы VBA также резервирует файловый буфер в памяти компьютера для ускорения процесса записи и считывания (прямое записывание информации на диск может существенно замедлить выполнение программы, что особенно заметно при работе с большими файлами). Максимальное число файловых буферов устанавливается в системном файле <i>Config.sys</i>

**Пример 15.** Открытие для чтения несуществующего файла. Выполнение этого фрагмента программы приведет к выдаче системой сообщения об ошибке "Файл не найден" и прекращению работы программы.

```
Open "C:\test1.sss" For Input As #1
Close #1
```

**Пример 16.** Открытие для записи не существующего файла последовательного доступа. Выполнение этого фрагмента программы приведет к созданию в текущем каталоге нового файла `test1.hhh`. Поскольку вывод в файл не производился, размер созданного файла 0 байт.

```
Dim canal As Integer
'Определение номера свободного файлового канала ввода-вывода
canal = FreeFile()
Open " C:\test1.sss" For Output As #canal
Close #canal
```

Если файл уже существует, то при открытии его в состоянии Output старый файл удалится, а новый запишется на его место. Если предполагается внесение изменений в уже существующий файл, он должен быть открыт в состоянии Append.

**Пример 17.** Программа записи файла исходных данных для начисления зарплаты в соответствии с рис. 1 лаб. раб. №7.

```
Open "C:\Зарплата.sss" ForOutputAs #1
'Запись в файл
Print #1, "Иванов В.Н."
Print #1, 1234
Write #1, "ТрофимоваЛ.А."
Write #1, 1234
Write #1, "Семенова Е.Г.", 1000, "Степанов А.Г.", 900
Close #1
```

Файлы с произвольным доступом позволяют обращаться к записи в файле по ее номеру. Такая возможность обеспечивается за счет создания регулярной структуры записей определенного формата, которую легко обеспечить, например, за счет типов данных, определяемых пользователем.

**Пример 18.** Предполагается, что создана структура (тип данных, определяемый пользователем) следующего вида:

```
Type Запись_файла
Фамилия_И_О AsString
Начислено_ВедомостьAsCurrency
EndType
```

Тогда программа работы с файлом произвольного доступа может содержать следующие операторы:

```
DimrecordAsЗапись_файла
Dim number As Integer, j As Integer
Open " C:\Зарплата1.rtf" ForRandomAs #1
'Запись в файл произвольного доступа
record.Фамилия_И_О = "Иванов В.Н."
record.Начислено_Ведомость = 1234
Put #1, 1, record
'Чтение только что сделанной записи
Get #1, 1, record 'Считывается только что записанное значение из файла
record.Фамилия_И_О = "Трофимова Л.А."
'record.Начислено_Ведомость содержит считанное из файла число 1234
Put #1, 2, record
record.Фамилия_И_О = "Семенова Е.Г."
record.Начислено_Ведомость = 1000
'Запись в файл произвольного доступа по текущему номеру счетчика
Put #1, , record 'Действие с записью 3
j = Seek(1) 'Значение указателя записи равно 4
record.Фамилия_И_О = "Степанов А.Г."
record.Начислено_Ведомость = 900
Put #1, j, record
Close #1
```

**Таблица 3. Операторы ввода вывода VBA**

<b>Действие</b>	<b>Ключевые слова</b>
Создать или редактировать файл	Open
Закрыть файл	Close, Reset
Управление форматами записи	Format, Print, Print #, Spc, Tab, Width #
Копирование файлов	FileCopy
Чтение свойств файлов	EOF, FileAttr, FileDateTime, FileLen, FreeFile, GetAttr, Loc, LOF, Seek
Определение размера файла	FileLen
Установка или чтение атрибутов файлов	FileAttr, GetAttr, SetAttr
Управление свойствами файлов	Dir, Kill, Lock, Unlock, Name
Чтение последовательных файлов	Input #, Line Input #
Запись в последовательный файл	Print #, Write #
Чтение файлов произвольного доступа или двоичных	Get
Запись в файл произвольного доступа или двоичный	Put
Задание номера записи файла произвольного доступа	Seek

Файл произвольного доступа открывается с указанием типа Random и по умолчанию доступен для чтения и записи. Явное указание режима только чтения или только записи обеспечивается за счет добавления ключевых слов AccessRead или AccessWrite.

Для записи в файл произвольного доступа можно воспользоваться оператором Put. Его первый параметр есть номер канала, второй – номер записи в файле (может отсутствовать), третий – имя переменной, значение которой надо записать в файл. Чтение информации из файла может быть осуществлено оператором Get, параметры которого аналогичны.

В системе существует внутренний указатель текущего номера рабочей записи файла. Его начальное значение равно нулю. Каждый раз при выполнении операторов Put и Get значение указателя становится равным номеру записи, с которой выполнялся оператор, плюс единица. Если в операторах Put и Get номер записи явно не задан, то новое действие будет выполняться с записью, номер которой содержится в указателе. Текущее значение указателя номера рабочей записи может быть получено оператором Seek. Двоичные файлы представляют собой разновидность файлов с произвольным доступом с элементарными записями размером в один байт, в связи с чем указатель текущего номера рабочей записи двоичного файла имеет смысл счетчика байтов. Программирование действий с записями двоичных файлов проводится так же, как и в случае работы с файлами произвольного доступа.

В процессе работы с файлом изменение его содержимого происходит в буфере операционной системы, а не на диске. Это обстоятельство приходится принимать во



внимание, например, при отладке программы. Если при работе с отладчиком интегрированной системы отладки выполнить команду **Run, Reset**, то изменения файла будут потеряны. Непосредственное завершение работы с файлом и запись данных на диск происходит в момент выполнения инструкции **Close**. Кроме этого, запись информации на диск (закрытие файла) происходит в момент завершения работы всей пользовательской программы. Для снижения вероятности ошибок программирования целесообразно всегда после завершения действий с файлом принудительно закрывать его инструкцией **Close**.

### **Задание**

Используйте согласованный с преподавателем вариант задания (табл.1лаб. раб. №7), выполненную на его основе таблицу Excel, написанную программу вычислений в таблице и созданный на ее основе класс, позволяющий производить требуемое количество таблиц и обработку данных в них. Научитесь работать с последовательными файлами и файлами произвольного доступов. Напишите программу, позволяющую сохранять исходные данные вашей таблицы в виде файлов последовательного или произвольного доступов. Включите ее в состав методов класса. Используйте сохраненные файлы в следующем сеансе работы с таблицей для восстановления в ней исходных данных.

### **Порядок выполнения работы**

1. Напишите программу, открывающую для чтения несуществующий файл последовательного доступа. Ознакомьтесь с диагностикой системы.
2. Напишите программу, открывающую для записи несуществующий файл. Внесите в текст комментарии действий программы. Убедитесь, что после завершения ее работы создан новый файл.
3. Напишите программу, открывающую для записи существующий файл. С помощью оператора **Print #** выполните запись в файл некой информации. Закройте файл. Внесите в текст комментарии действий программы.
4. Объявите в программе некоторую переменную из числа исходных данных вашей таблицы, откройте ранее записанный файл, считайте в режиме отладки в объявленную переменную записанные в файл данные оператором **Input #**, контролируя считанные значения в окне локальных переменных.
5. Измените записываемую в файл информацию и при повторном считывании убедитесь в том, что информация в файле изменилась.
6. Откройте файл в режиме **Append**. Сделайте запись в файл новой информации и закройте файл.
7. Откройте файл только для чтения и убедитесь, что файл содержит как предыдущую, так и новую записи.
8. Создайте файл с произвольным доступом. Запишите в него информацию. Закройте файл.
9. Напишите программу чтения информации из файла произвольного доступа. Внесите в текст комментарии действий программы. Проконтролируйте правильность ее работы.

10. Выберите тип файла для хранения исходных данных вашей таблицы. Создайте его и запишите в файл все исходные данные вашей таблицы.

11. Удалите исходные данные из таблицы, считайте их из созданного вами файла заново и занесите их в таблицу. Проконтролируйте правильность считывания информации в вашу таблицу.

12. Измените содержимое исходных данных вашей таблицы и проконтролируйте изменение файла произвольного доступа.

13. Подключите созданную вами программу в качестве метода созданного вами класса.

14. Проверьте правильность комментариев с учетом изменений в тексте программы, дополните и, при необходимости, скорректируйте их.

### **Контрольные вопросы**

1. Как организован последовательный файл?
2. Как организован файл произвольного доступа?
3. Что нужно сделать для того, чтобы начать работу с файлом?
4. В чем заключается отличие в обращении к элементу данных последовательного файла и файла произвольного доступа?
5. Как определить факт достижения конца файла?
6. Как создать файл?
7. Чем отличается формат оператора Print от формата оператора Put?
8. Как указать, что файл открывается только для чтения?
9. Как определить номер свободного канала для открытия файла?
10. Как закрыть файл, и в какой момент данные оказываются на диске?

### **Отчет о работе**

Подготовьте отчет о выполненной лабораторной работе. Он должен содержать титульный лист, рисунок алгоритма и текст написанной вами программы работы с файлами последовательного и произвольного доступа. Сформулируйте выводы, которые можно сделать по результатам выполненной работы.

## Лабораторная работа №13.Создание таблиц и связей между ними

**Цель работы.** Построение инфологической модели предметной области, определение состава базы данных, выбор модели данных и СУБД, разработка базы данных: таблиц для хранения информации, связей между ними, ограничений на значения полей.

### Контрольный пример

Реализация общей задачи разбита на несколько этапов:

1. В результате анализа поставленной задачи составим инфологическую модель предметной области. Инфологическая модель (Рис.1) описывается тремя сущностями (**Фирма**, **Товар**, **Заказ**), атрибутами (обязательными), характеризующими каждую сущность, и связями между сущностями (фирма-заказ, товар-заказ).

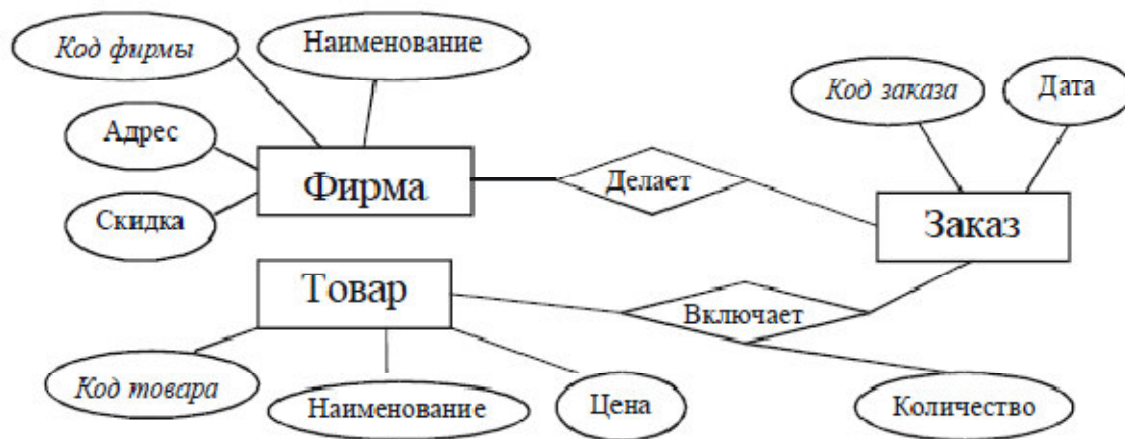


Рис. 1. Инфологическая модель предметной области

2. На основании этой модели, с учетом того, что конкретная реализация будет выполнена в среде СУБДAccess, разрабатываем схему данных для каждой будущей таблицы (базовый вариант):

Таблица **Фирма**

Наименование полей	Тип	Примечание
Код_фирмы	Счетчик	Ключевое поле
Наименование_ф	Текстовый	
Адрес	Текстовый	
Скидка	Числовой	

Таблица **Товар**

Наименование полей	Тип	Примечание
Код_товара	Счетчик	Ключевое поле
Наименование_т	Текстовый	
Цена	Числовой	

Таблица **Заказ\_Фирма**

Наименование полей	Тип	Примечание
Код_заказа	Счетчик	Ключевое поле
Код_фирмы	Числовой	
Дата	Дата/время	

Таблица **Заказ\_Товар**

Наименование полей	Тип	Примечание
Код_заказа	Числовой	
Код_товара	Числовой	
Количество	Числовой	

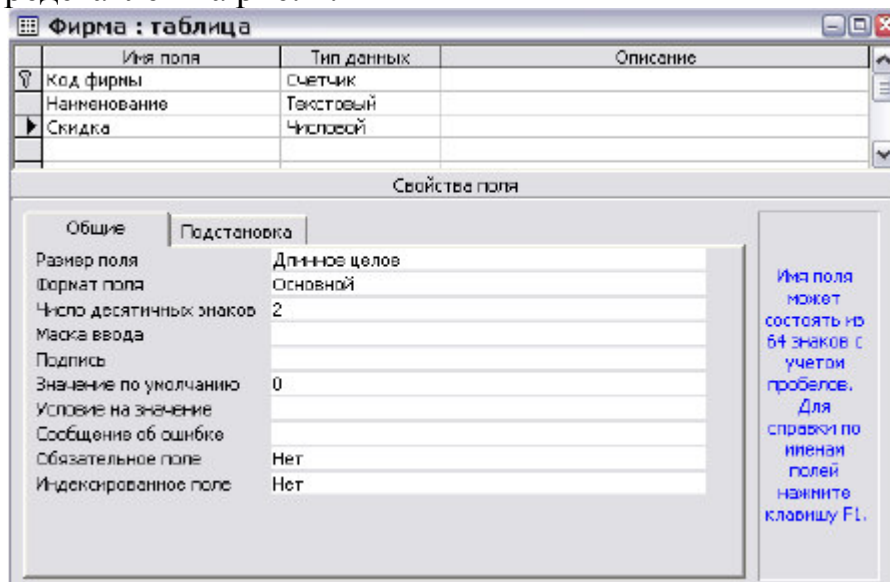
### 3. Создание таблиц в СУБД Access.

Для этого необходимо выбрать закладку *Таблицы* (пункт списка «Объекты») и нажать кнопку *Создать* и выбрать режим *Конструктор* или изначально выбрать пункт *Создание таблицы в режиме конструктора* рабочей области окна, после чего описать первую таблицу **Фирма** (описываются заголовки столбцов, указываются типы полей и ключевое поле). (Таблицы типа **Фирма** принято называть справочниками). Тип поля выбирается из списка. Для поля **Скидка** установим размер поля – *Одинарное с плавающей точкой* и формат поля – *Процентный*.

Для создания ключевого поля необходимо вызвать контекстное меню для нужной строки и выбрать пункт «Ключевое поле» или воспользоваться пиктограммой панели инструментов.

Перед закрытием таблицу необходимо сохранить и дать ей соответствующее имя.

Результат представлен на рис. 2.

Рис. 2. Описание структуры таблицы **Фирмы**

Аналогичным образом создаем остальные таблицы. Две последние таблицы **Заказ\_Фирма** и **Заказ\_Товар** представляют собой описание связей между объектами и помимо уникальных данных, которые описываются обычным

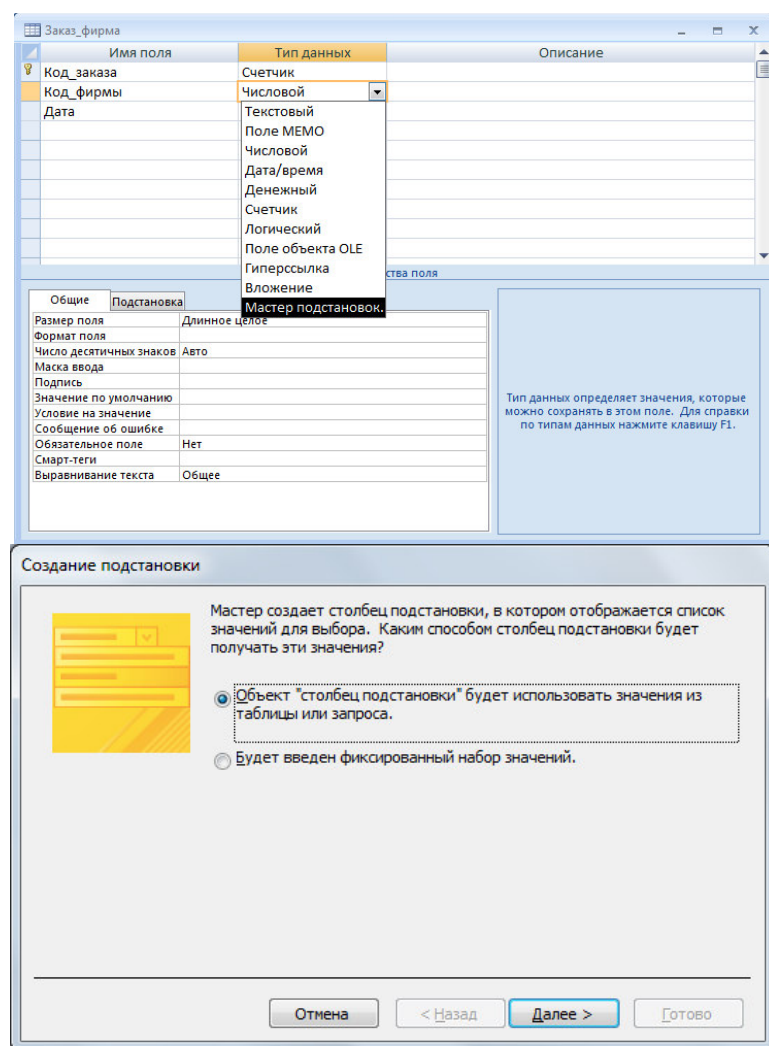
способом, содержат поля из таблиц справочников. Следовательно, при описании полей таблицы в режиме Конструктора можно воспользоваться закладкой *Подстановка* и выполнить следующие настройки: *Тип элемента управления* – Выбрать *Поле со списком*. *Тип источника строк* – Выбрать *Таблица или запрос*. *Источник строк* – указывается таблица, которая является источником данных. **Например**, для поля **Код\_Фирмы** из таблицы **Заказ\_Фирма**, источником является таблица **Фирма**.

Указать источник данных для полей: Код\_Фирмы – Таблица **Заказ\_Фирма** (источник – коды фирм из таблицы **Фирма**). Код\_Товара – Таблица **Заказ\_Товар** (источник – коды товаров из таблицы **Товар**).

**При задании полей с подстановкой рекомендуется использовать Мастер подстановок.**

Чтобы использовать Мастер подстановок, необходимо:

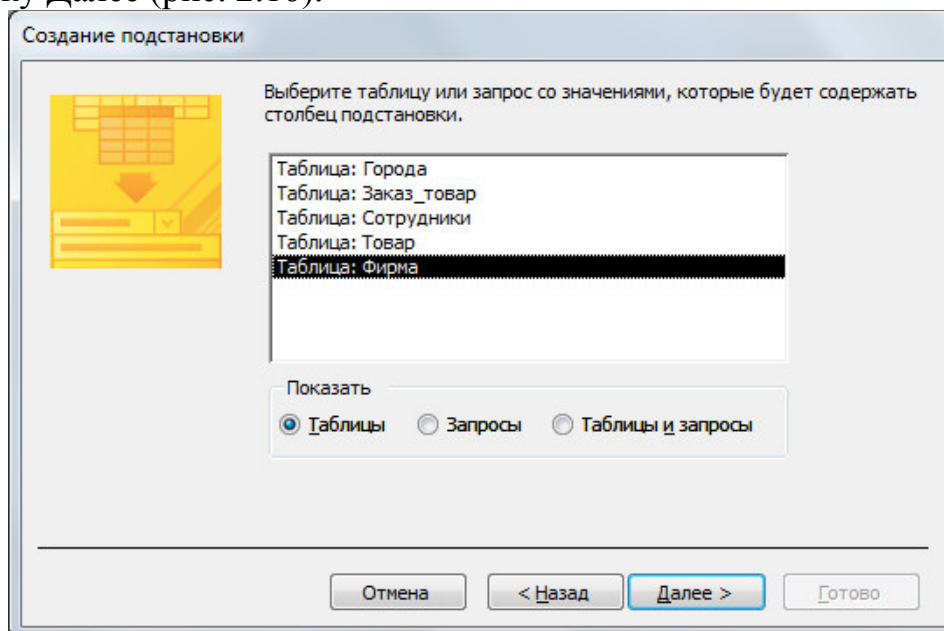
1. Открыть таблицу в режиме Конструктора.
2. Выделить поле "Код\_фирмы" и выбрать из списка в столбце **Тип данных** значение **Мастер подстановок** (рис. 2.15).



**Рис. 2.15.** Первое диалоговое окно Мастера подстановок

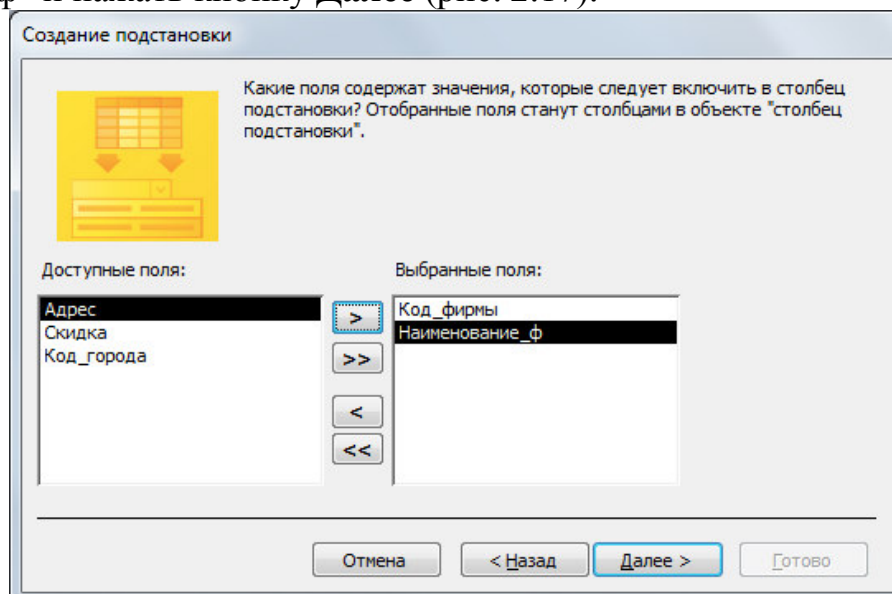
3. В открывшемся диалоговом окне **Мастер подстановок** выбрать способ задания значений: **Объект "столбец подстановки" будет использовать значения из таблицы или запроса**, т. к. в этом случае мы должны использовать данные из таблицы "Клиенты". Нажать кнопку **Далее**.

4. В следующем диалоговом окне можно выбрать из списка таблицу или запрос, из которого будет осуществляться подстановка. Выбрать таблицу "Фирма" и нажать кнопку **Далее** (рис. 2.16).



**Рис. 2.16.** Второе диалоговое окно Мастера подстановок

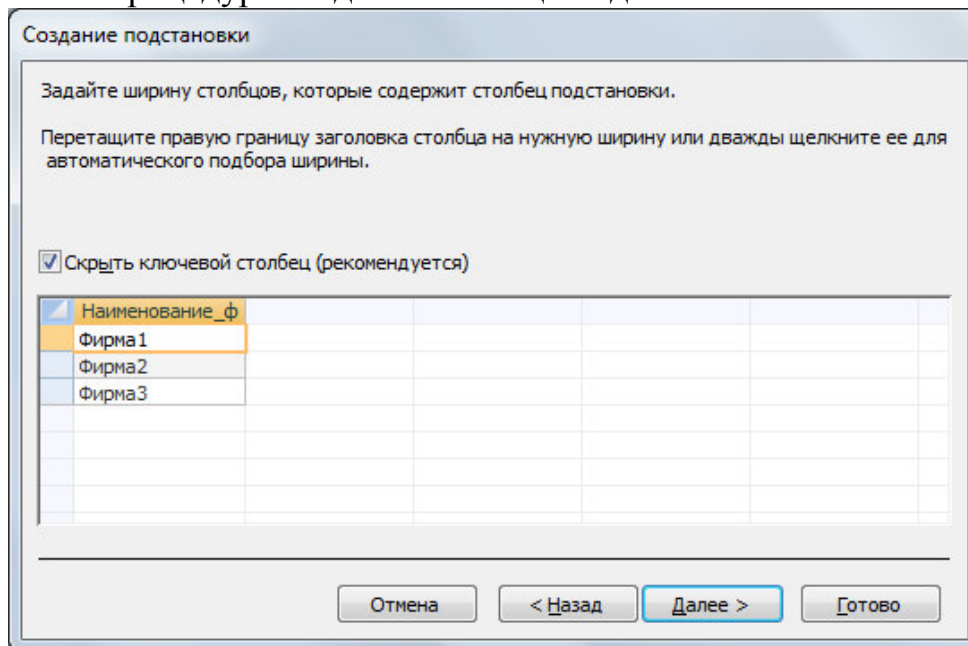
5. В списке **Доступные поля** выводятся все поля таблицы "Фирма". Переместить из списка доступных полей в список подстановки поля "Код\_фирмы" и "Наименование\_ф" и нажать кнопку **Далее** (рис. 2.17).



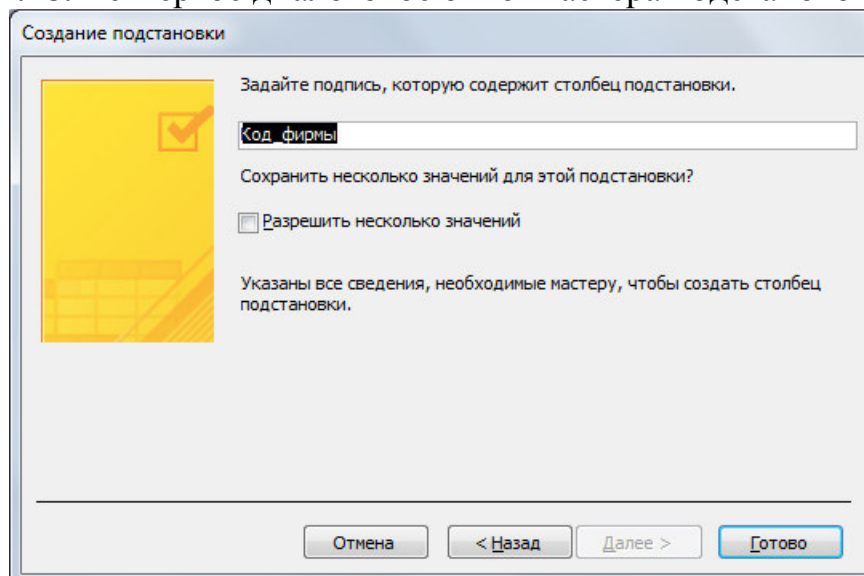
**Рис. 2.17.** Третье диалоговое окно Мастера подстановок

6. Просмотреть столбец подстановки, с помощью перетаскивания границы выбрать ширину столбца, а также оставить установленный по умолчанию флажок **Скрыть ключевое поле**. Действительно, нам не нужно в раскрывающемся списке видеть коды клиентов. Нажать кнопку **Далее** (рис. 2.18).

7. Ввести название столбца подстановок и нажать на кнопку **Готово** (рис. 2.19). При необходимости Мастер подстановок попросит сохранить те изменения, которые вы внесли в таблицу, прежде чем окончательно установит подстановки – ответьте Да. На этом процедура создания столбца подстановок заканчивается.



**Рис. 2.18.** Четвертое диалоговое окно Мастера подстановок



**Рис. 2.19.** Пятое диалоговое окно Мастера подстановок

4. Создание схемы данных. Для того чтобы описать связи между таблицами, используем пункт меню *AccessРабота с базами данных~Схема данных*. Связывание полей проведем способом перетаскивания поля из одной таблицы в другую. Например, поле из таблицы **Фирма** *Код\_фирмы* необходимо переместить в таблицу **Заказ\_Фирма** на аналогичное поле. *Следует отметить, что поля, по которым выполняется связь, должны быть одинаково описаны* (допускается связь полей с типами Счетчик и Числовой). В окне установки связей выполняют настройки рис. 4.



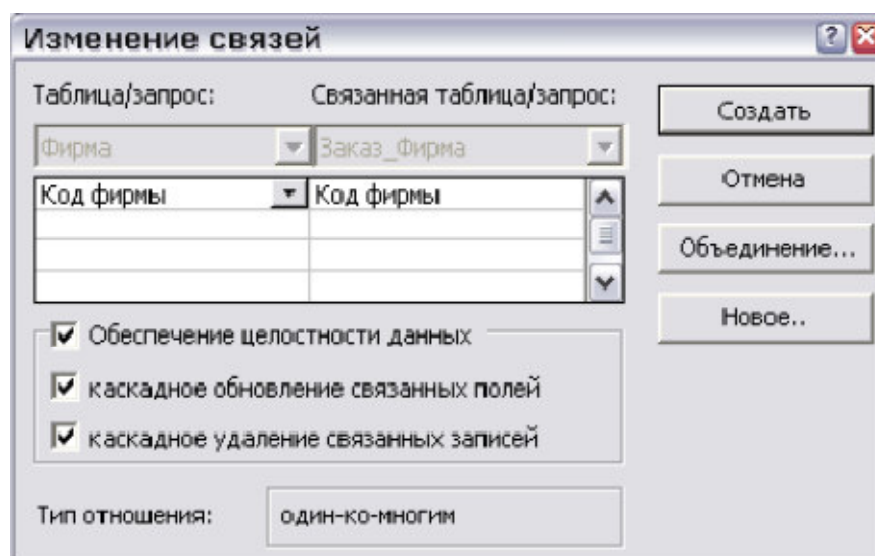


Рис. 4. Установка связей между таблицами

Результат установления связей в базе данных представлен на рис. 5.

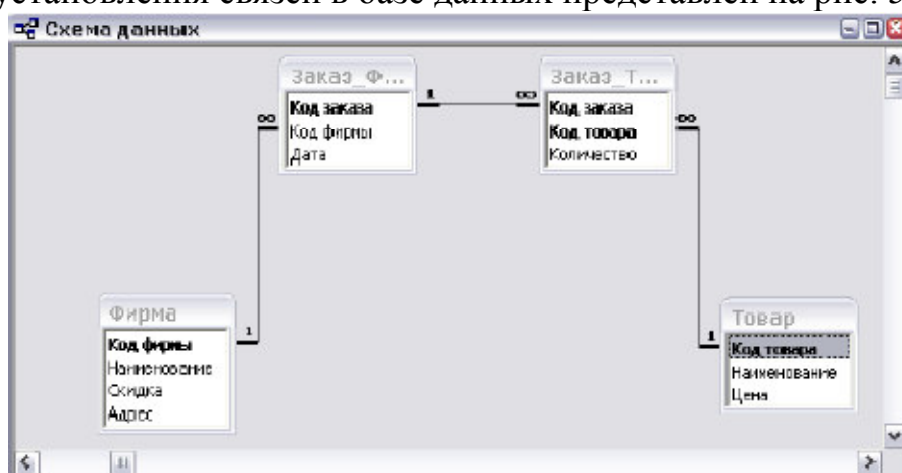


Рис. 5. Схема данных

### Индивидуальное задание

- 1) Определить список необходимых таблиц и дополнительных атрибутов, согласно своему варианту по прилагаемой ниже таблице.
- 2) Создать дополнительную таблицу или добавить новые атрибуты в модель. Если необходимо, откорректировать схему данных («привязать» новую таблицу).
- 3) Заполнить таблицы данными: не менее 5 строк для таблиц справочников и 10 – для таблиц заказов.

### Дополнительные таблицы

«Города»

Наименование поля	Тип (формат) поля
Код_города	Счетчик
Наименование	Текстовый

«Сотрудники»

Наименование поля	Тип (формат) поля
-------------------	-------------------



Код_сотрудника	Счетчик
ФИО	Текстовый
Телефон	Текстовый

### Дополнительные атрибуты

Номер	Наименование поля	Тип (формат) поля
1	2	3
Для таблицы «Фирма»		
1	Код_города	Числовой (связь с таблицей «Города»)
2	ФИО_директора	Текстовый
3	Телефон	Текстовый
4	E-mail	Текстовый
5	Расчетный_счет	Текстовый
Для таблицы «Товар»		
6	Дата_поставки	Дата/время
7	Годен_до	Дата/время
8	Измерение	Текстовый
9	Производитель	Текстовый
Для таблицы «Заказ_фирма»		
10	Дата_выполнения	Дата/время
11	Код_отв_сотрудника	Числовой (связь с таблицей «Сотрудники»)

### Таблица вариантов

№ вар.	Дополнительные атрибуты										
	1 (+создание таб. «Города»)	2	3	4	5	6	7	8	9	10	11 (+создание таб. «Сотр.»)
1	*										
2		*						*			
3			*						*		
4				*						*	
5					*	*					
6				*			*				
7			*					*			
8		*							*		
9					*					*	
10											*
11			*				*				
12				*				*			
13					*		*				
14		*								*	
15					*			*			

## Лабораторная работа №14. Создание форм и страниц доступа к данным

**Цель работы.** Необходимо разработать ряд пользовательских форм, для работы с приложением для созданной базы данных. Для размещения приложения по работе с БД в сети использовать страницы доступа к данным.

Форма – объект приложения БД, предназначенный для ввода и редактирования данных. Для работы с формами приложения служит закладка *Формы* списка *Объекты*.

### Контрольный пример

1. *Создание форм с помощью Мастера форм.* Для создания формы для таблицы **Фирмы** воспользуемся *Мастером* форм. Переходим на закладку **Формы**, выбираем кнопку *Создать*. В появившемся окне (рис. 6) указываем способ создания (Мастер форм) и таблицу, для которой создается форма (**Фирма**). В следующем окне выбираем поля таблицы, которые будут отображаться на форме. Поскольку поле *Код фирмы* имеет тип *Счетчик*, то значение его генерируется автоматически и на форме поле может не присутствовать.

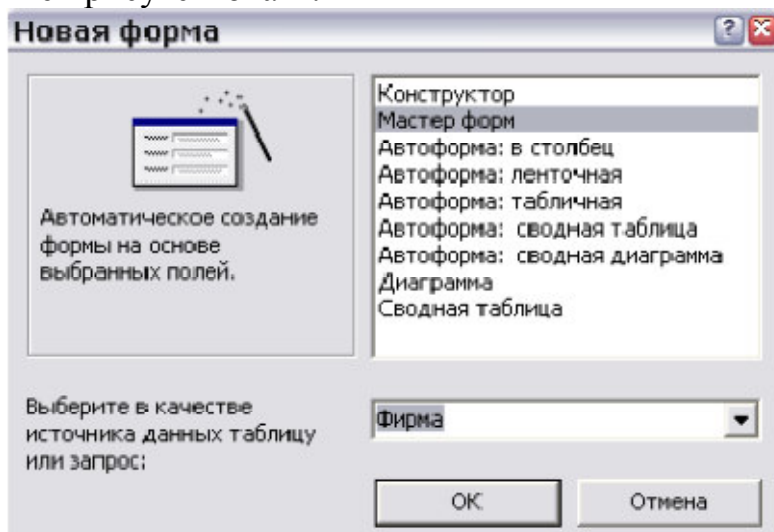


Рис. 6. Создание форм

В следующем окне укажем внешний вид формы по желанию, далее – зададим стиль оформления и, наконец, назовем форму и нажмем кнопку *«Готово»*. Результат представлен на рис. 7.

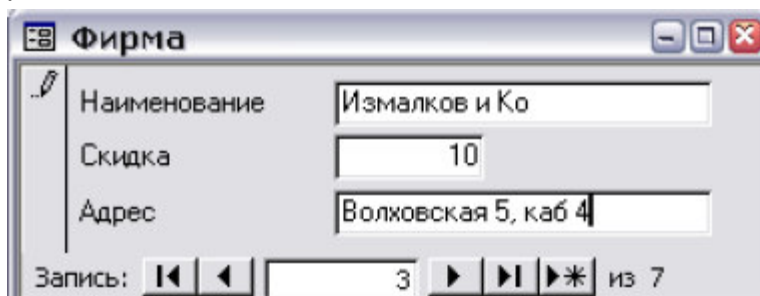


Рис. 7. Форма Фирма

2. *Создание форм в режиме Конструктора.*

Форму Товары создаем с помощью Конструктора: при указании способа создания выбираем пункт *Конструктор*, в качестве таблицы выберем **Товар**. С помощью контекстного меню добавляем области *заголовка и примечания* формы.

Используя *Панель элементов для конструирования* (рис. 8), разместим на новой форме необходимые компоненты.



Рис. 8. Панель элементов

1. Добавим заголовок с помощью элемента *Надпись*.

2. Из схемы данных таблицы **Товар** (дочернее окно на окне конструктора) перетащим на форму необходимые поля. Поле Код можно не добавлять, т.к. коды товара генерируются автоматически (тип поля – счетчик). При необходимости изменим надпись перед *Поле* для ввода данных, добавим примечание, откорректируем размеры элементов. Если первоначально при создании не была указана таблица-источник, это можно сделать, откорректировав свойства формы (пункт меню Вид ~ Свойства). В поле *Источник записей* закладки *Данные* нужно указать требуемую таблицу. В этом же окне можно откорректировать и другие свойства формы, например, изменить её цвет или убрать навигатор работы с записями (Макет ~ Кнопки перехода ~ Нет). В итоге получим форму, аналогичную рис. 9.

Рис. 9. Форма Товар

3. *Разработка многотабличных форм.*

Составная многотабличная форма создается для работы с несколькими взаимосвязанными таблицами. Многотабличная форма может состоять из основной части и одной или нескольких подчиненных включаемых форм, т.е. быть составной. Подчиненная форма может быть построена на основе, как подчиненной, так и главной таблицы относительно таблицы-источника основной части формы.

1. Для создания формы в окне базы данных выберем закладку *Формы* и нажмем кнопку *Создать*.

2. В окне *Новая форма* выберем режим создания *Мастер форм* и выберем в качестве источника данных основной части формы из списка таблицу **Заказ-фирма**.

3. В открывшемся диалоговом окне *Создание форм* сначала выбираем таблицу **Заказ\_фирма** и поля из неё, включаемые в форму. После этого в этом же окне выберем таблицу **Заказ-товар** и укажем необходимые поля (код товара и количество) рис. 10.

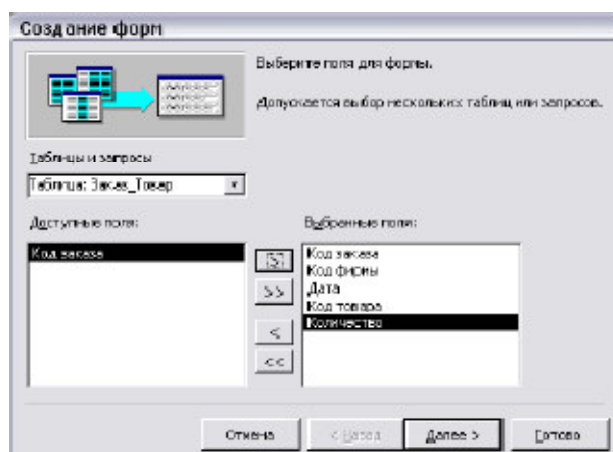


Рис. 10. Окно для выбора полей формы

4. Перейдите на следующее окно, нажав кнопку Далее.

5. В окне создания форм выделим таблицу **Заказ\_фирма**, которая будет являться источником основной части формы и выбираем один из вариантов подключения формы (в нашем случае лучше выбрать вариант Подчиненные формы).

6. В следующем окне выберем вариант оформления подчиненной формы.

7. Выберем стиль оформления формы на следующем окне.

8. В последнем диалоговом окне Создание форм отредактируем заголовки формы.

9. Доработаем форму в режиме конструктора:

9.1. Дадим форме заголовок;

9.2. Уберем автоматический счетчик записей на основной форме (см. Создание форм в режиме конструктора) и поместим на форму кнопки для перемещения по записям, редактирования записей. Для этого перетаскиваем элемент *Кнопка* с панели инструментов, в появившемся окне (рис. 11) определяем Категорию и Действие, которое будет выполняться при нажатии на кнопку (например, действие *Добавить запись* категории *Обработка записей*), выбираем рисунок кнопки (или подпись) и даем ей имя. Аналогично можно создать кнопки для обработки других событий, например перехода по записям.

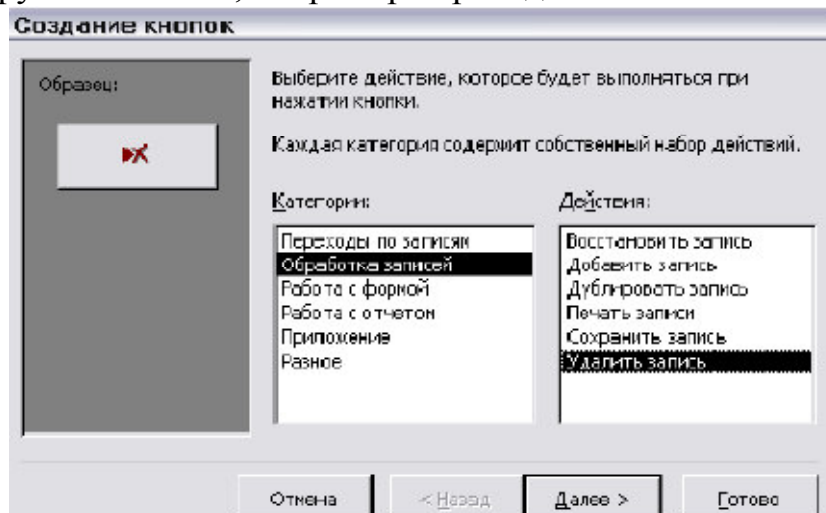


Рис. 11. Задание процедур кнопок

Результатом работы должна стать форма следующего вида рис. 12.

Рис. 12. Многотабличная форма

### Индивидуальное задание

1. Для вариантов 1 и 10 создать дополнительные формы по работе с таблицами городов или сотрудников. Все остальные варианты включают свои дополнительные поля на существующие формы.
2. Для четных вариантов добавить на формы со стандартными навигаторами кнопки для удаления записей (стандартный интерфейс их не содержит). Для нечетных вариантов – добавить кнопки поиска произвольной записи из таблицы.
3. Добавить в таблицы ещё по 5 записей с помощью созданных форм.

## Лабораторная работа №15.Обработка данных средствамиAccess

**Цель работы.** Разработать запросы к БД, используя редактор ABE (запросы по абзацу).

Одним из основных инструментов обработки данных в СУБД являются запросы. Запрос – это формализованное требование пользователя на отбор данных или на выполнение действий. Запрос строится на основе одной или нескольких таблиц. При этом могут использоваться таблицы базы данных, а также сохраненные таблицы, полученные в результате других запросов. Запрос позволяет выбрать необходимые данные из одной или нескольких взаимосвязанных таблиц, произвести вычисления и получить результат в виде таблицы. В Access может быть создано несколько видов запроса:

**Запрос на выборку** – выбирает данные из взаимосвязанных таблиц и других запросов. Результатом его является таблица, которая существует до закрытия запроса.

**Запрос на создание таблицы** – основан на запросе выборки, но в отличие от него результат запроса сохраняется в новой таблице.

**Запросы на обновление, удаление, добавление** – являются запросами действия, в результате выполнения которых изменяются данные в таблицах.

### Основы конструирования запросов

Основные принципы конструирования запроса заложены в технике конструирования запроса на выборку, являющегося основой всех видов запроса. Разработка запроса производится в режиме *Конструктор запросов*. Для создания запроса надо в окне базы данных выбрать закладку *Запросы* (пункт *Запросы* меню *Объекты*) и нажать кнопку *Создать*. В открывшемся окне *Новый запрос* выбрать *Конструктор*. В окне *Добавление таблицы* выбрать используемые в запросе таблицы и нажать кнопку *Добавить*. Затем кнопкой *Заккрыть* выйти из окна *Добавление таблицы*. В результате появится окно конструктора запросов. Окно конструктора разделено на две панели. Верхняя панель содержит схему данных запроса, а нижняя панель является бланком запроса по образцу. Схема данных запроса отображает выбранные таблицы и связи между ними, имеющиеся в схеме данных БД. Бланк запроса по образцу представлен в виде таблицы в нижней панели окна запроса. Каждый столбец бланка относится к одному полю, с которым нужно работать в запросе. При заполнении бланка необходимо:

- В строку *Поле* включить имена полей, используемые в запросе
  - В строке *Вывод на экран* отметить поля, которые должны быть включены в результирующую таблицу
  - В строке *Условие отбора* задать условия отбора записей
  - В строке *Сортировка* выбрать порядок сортировки записей результата
- Если в запрос включаются все поля таблицы, то в строке *Поля* ставится знак \*.

### Контрольный пример

Рассмотрим технологию создания *однотабличного запроса на выборку* на примере получения информации из таблицы **Товар** базы данных **Учет заказов**.

- Пусть необходимо получить информацию о *товаре* с названием *Коврик*.

- 1) Для создания запроса в окне базы данных выберем закладку *Запросы* и нажмем кнопку *Создать*.
- 2) В окне Новый запрос выберем Конструктор.
- 3) В окне Добавление таблицы выберем таблицу **Товар**, нажмем кнопку и закроем окно Добавление таблицы.
- 4) Заполним бланк запроса по образцу в соответствии с рис. 14.

Рис. 14. Бланк запроса по образцу

- 5) Закроем *Конструктор* и сохраним запрос с именем **Коврики**. Выполним сохраненный запрос нажатием кнопки *Открыть*. В результате имеем таблицу (рис. 15).

Наименование	Цена
коврик	3 000,00р.
	0,00р.

Рис. 15. Результат выполнения запроса Коврики

- Пусть необходимо получить информацию обо всех товарах из таблицы **Товар** с ценой больше 30р.

Для этого повторим пункты 1–3 предыдущего запроса (создание), затем заполним бланк запроса по образцу в соответствии с рис. 16.

Закроем *Конструктор* и сохраним запрос с именем **Цена товара**.

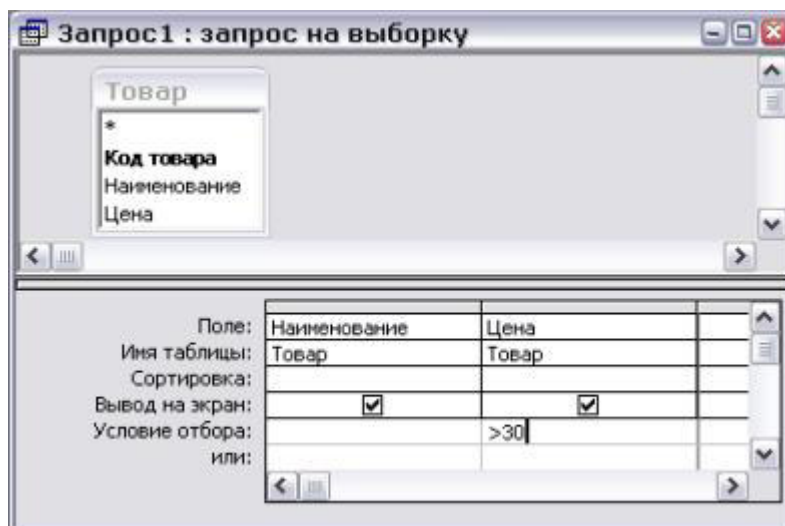


Рис. 16. Запрос на выборку товара с ценой >30

Выполним сохраненный запрос. В результате имеем таблицу (рис. 17.).

Цена товара : запрос...

Наименование	Цена
табуретка	152,02р.
веревка	45,00р.
коврик	3 000,00р.
	0,00р.

Запись: 4

Рис. 17. Результат выполнения запроса Цена товара

• Пусть необходимо выбрать фирмы, названия которых начинаются с буквы «А» из таблицы **Фирмы**.

- 1) Выполним пункты 1–3 предыдущих запросов (выбрать таблицу **Фирма**).
- 2) Заполним бланк запроса по образцу в соответствии с рис. 18.

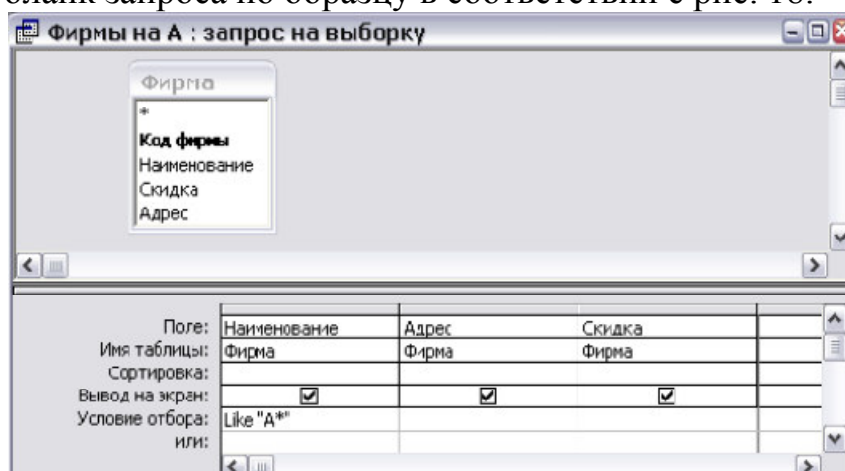


Рис. 18. Запрос на выборку фирм



3) Закроем *Конструктор* и сохраним запрос с именем **Фирмы на А**. Выполним сохраненный запрос нажатием кнопки Открыть. В результате имеем таблицу (рис. 19).

Наименование	Адрес	Скидка
Алертные слоники	11-я линия, 48	5
Адвентизм и Ко	Ленина, 46	8
		0

Рис. 19. Результат выполнения запроса на выборку фирм

2. Рассмотрим технологию конструирования *многотабличного запроса на выборку* на примере получения информации из таблиц **Заказ\_Фирма**, **Заказ\_Товар** и **Товар** базы данных **Учет заказов**.

- Пусть необходимо получить информацию о товарах, проданных в период с 1.01.2002 до 1.01.2003.

1) Повторим пункты 1–3 предыдущих запросов.

2) В окне *Добавление таблицы* выберем таблицы **Заказ\_Фирма**, **Заказ\_Товар** и **Товар** и закроем окно *Добавление таблицы*. Связи между таблицами установлены автоматически в соответствии со схемой данных базы данных.

3) Заполним бланк запроса по образцу в соответствии с рис. 20.

Поле:	Наименование	Цена	Дата
Имя таблицы:	Товар	Товар	Заказ_Фирма
Сортировка:			
Вывод на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Условие отбора:			>#01.01.2002# And <#01.01.2003#
или:			

Рис. 20. Запрос на выбор товара по датам

4) Закроем *Конструктор* и сохраним запрос с именем **Товары по дате**. Выполним сохраненный запрос нажатием кнопки *Открыть*. В результате имеем таблицу (рис. 21).

Наименование	Цена	Дата
▶ мыло	4,00р.	12.12.2002
веревка	45,00р.	13.07.2002
коврик	3 000,00р.	15.03.2002
табуретка	152,02р.	15.03.2002
сикатор	10,52р.	24.10.2002
*		

Запись: 1 из 5

Рис. 21. Результат выполнения запроса Товары по дате

3. Рассмотрим технологию конструирования многотабличного запроса на выборку с использованием вычисляемых полей.

- Пусть необходимо получить информацию о заказах, фирмах, сделавших заказ и суммах по оплате каждого товара.

1) Повторим пункты 1–3 предыдущих запросов.

2) В окне *Добавление таблицы* выберем таблицы **Заказ\_Фирма**, **Заказ\_Товар**, **Фирма** и **Товар** и закроем окно *Добавление таблицы*.

3) В рассматриваемом запросе поле **Сумма** является вычисляемым и в таблицах не присутствует. Для его создания в бланке запроса по выбору необходимо в строке *Поле* указать *Имя нового поля* – **Сумма** и через двоеточие выражение, с помощью которого будут вычисляться значения:

Сумма: Заказ\_товар!Количество\*Товар!Цена\*(1-Фирма!Скидка/100)

(деление на 100 необходимо, если скидка изначально не задана в процентном формате).

Для построения выражения можно воспользоваться построителем выражений (а можно ввести вручную). Для его вызова необходимо курсор установить в строке *Поле*, щелкнуть правой клавишей мыши и в контекстном меню выбрать пункт *Построить*. В появившемся окне (рис. 22.) строим выражение, выбирая таблицу в окошке слева, нужное поле в центральном списке и <значение> справа и нажимая кнопку *Вставить*. Потом добавляем скобки или знак операции с помощью кнопок или с клавиатуры, затем следующее выражение и т.д. Фиксируем выражение нажатием кнопки *ОК*.

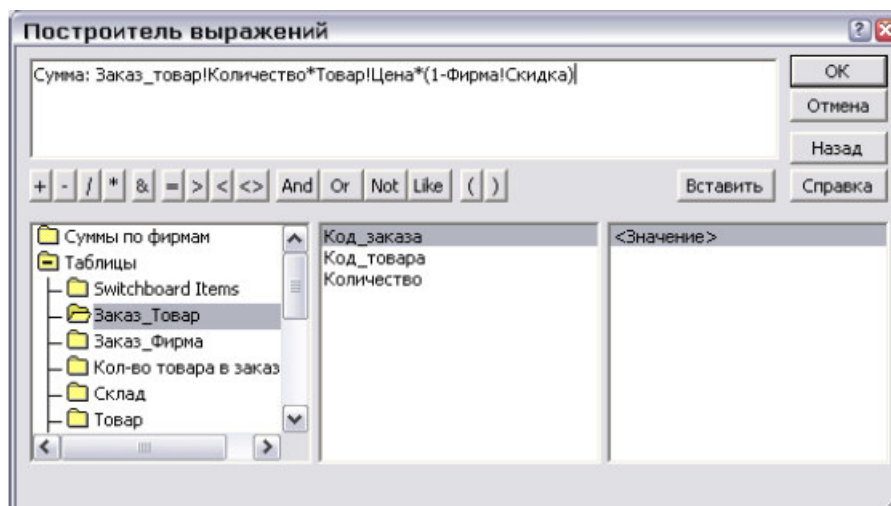


Рис. 22. Окно построителя выражений

4) Заполним бланк запроса по образцу в соответствии с рис. 23.

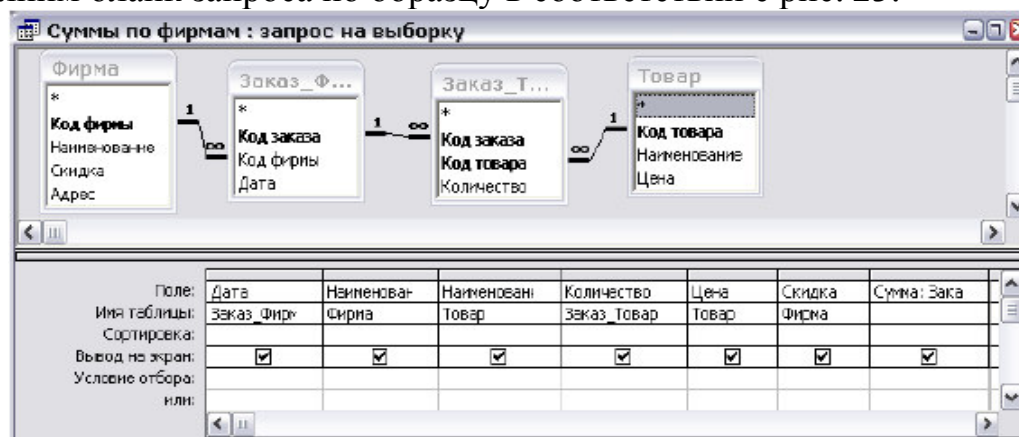


Рис. 23. Запрос с использованием вычисляемого поля

5) Закроем *Конструктор* и сохраним запрос с именем **Суммы по фирмам**. Выполним сохраненный запрос. В результате имеем таблицу (рис. 24).

Дата	Фирма.Наименов	Товар.Наименов	Количество	Цена	Скидка	Сумма
14.03.2003	Маня и Ко	лезвие бритвы	120	20,00р.	5	1192
12.12.2002	Варя и Ко	мыло	340	4,00р.	3	1319,2
14.03.2003	Маня и Ко	табуретка	50	152,02р.	5	1220,95
13.01.1998	Кикиморо+	сигары	50	10,52р.	2	515,48
12.03.2003	Кикиморо+	веревка	40	45,00р.	2	1764
13.07.2002	Маня и Ко	веревка	220	45,00р.	5	9405
12.03.2003	Кикиморо+	мыло	67	4,00р.	2	262,64
12.03.2003	Кикиморо+	лезвие бритвы	78	28,00р.	2	2140,32
15.03.2002	Измалков Inc.	коврик	2	3 000,00р.	10	5400
15.03.2002	Кикиморо+	табуретка	10	152,02р.	2	1489,796
24.10.2003	Маня и Ко	сигары	3	10,52р.	5	29,982

Рис. 24. Результат выполнения запроса Суммы по фирмам

### Индивидуальное задание

1. Для вариантов 1 и 10 создать запрос по образцу на выборку любой фирмы из таблицы **Фирма** (по наименованию) с точным совпадением значения. Для вариантов 2,8 и 14 запрос на выборку по значению ФИО директора, для вариантов

3,7,11 – по значению телефона фирмы, для №4,6,12 – по значению E-mail фирмы, для № 5,9,13,15 – по значению расчетного счета фирмы.

2. Для четных вариантов выполнить запрос на выборку фирм со скидкой не больше определенного значения, для нечетных – со скидкой меньшей определенного значения.

3. Для вариантов 1 и 10 создать запрос по образцу на выборку городов или сотрудников, название (имя) которых начинается на определенную букву. Для вариантов 2,3,7,8,12,15 – выполнить запрос на выборку товаров по первой букве производителя или единицы измерения в зависимости от имеющегося атрибута. Вариантам 4,5,6,9,11,13,14 – создать запрос на выборку товара, не начинающегося на определенную букву.

4. Для всех вариантов выполнить запрос на информацию о фирмах, приобретающих определенный вид товара (добавить в запрос 4 основные таблицы, в качестве условия записать конкретное наименование имеющегося (!) товара, это поле на экран можно не выводить).

5. Для всех вариантов на основе запроса Суммы по фирмам создать новый (а не модифицировать старый!) запрос, выбирая информацию по конкретной фирме. Совет: скопируйте запрос о суммах по фирмам стандартным образом в эту же БД и модифицируйте копию с учетом нового условия.

## Лабораторная работа №16. Создание запроса с произвольной выборкой

**Цель работы:** Научиться создавать запросы с условиями поиска, вводимыми пользователем, запросы с групповыми операциями, запросы на создание и обновление таблиц.

### Запросы с произвольной выборкой

Пусть необходимо получать информацию о начисленных суммах по различным фирмам. Для решения этой задачи можно использовать два пути:

- сконструировать запрос отдельно для каждой фирмы. Это не очень эффективно, т.к. количество фирм меняется, что заставляет постоянно добавлять новые запросы.
- создать один запрос на выборку, задавая условия поиска в специально созданной форме (запрос с произвольной выборкой).

*Создавая запрос с произвольной выборкой, необходимо заранее создать опорный запрос. В качестве опорного запроса в нашем случае можно выбрать запрос **Суммы по фирмам** (см. Лабораторную работу №15). Сделайте копию запроса суммы по фирмам с именем **Суммы по фирмам 1**.*

### Контрольный пример

1. Создадим *Форму* для задания критериев отбора записей в режиме *Конструктора* обычным способом. Используя *Панель элементов для конструирования*, разместим на новой форме необходимые компоненты: Переместим на форму элемент *Поле*, изменим надпись перед *Поле*: удалим условное название «Поле» и введем комментарий «Введите название фирмы». Выберем элемент *Кнопка* и переместим его на форму. В открывшемся окне *Создание кнопок* выберем категорию *Разное* и Действие **выполнить запрос**.

Нажмем кнопку *Далее* и выберем запрос, который откроется при нажатии на эту кнопку – **Суммы по фирмам**. Далее выберем оформление для кнопки (Текст или рисунок). В результате выполненных действий форма для запроса примет вид (рис. 25).

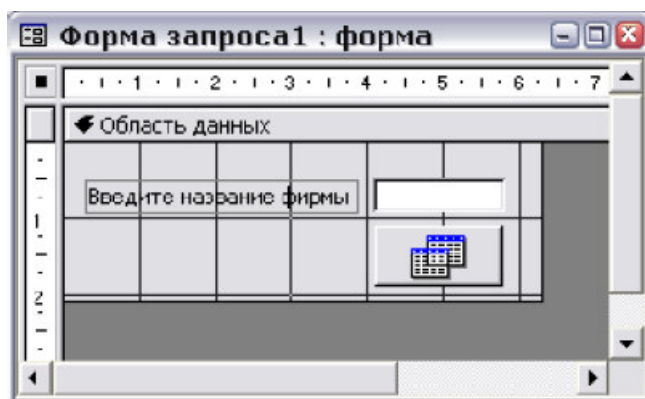


Рис. 25. Форма для запроса

Остается закрыть *Конструктор* и дать новой форме имя **Форма запроса1**.

2. В связи с тем, что условия для выборки будут задавать из формы, необходимо преобразовать Запрос **Суммы по фирмам**. Для этого откроем запрос

**Суммы по фирмам** в режиме *Конструктора*. В поле **Наименование. Фирма** (строку *Условие отбора*) ввести выражение:

**Like [Forms]![Форма запроса1]![Поле0]**

Выражение рекомендуется вводить с помощью построителя выражений (см. Лабораторную работу №15), выбирая операцию, потом тип операнда – формы, нужную форму в списке, и требуемый элемент (Поле) из имеющихся и нажимая кнопку *Вставить*. Закроем запрос, сохранив изменения. Откроем закладку *Формы* и откроем **Форму запроса1**. Введем условие для отбора записей (рис. 26).

Рис. 26. Форма для запроса с условием для отбора записей

Результат выполнения запроса представлен на рис. 27.

Дата	Фирма.Наименование	Товар.Наименование	Количество	Цена	Скидка	Сумма
14.03.2003	Маня и Ко	лезвие бритвы	120	28,00р.	5	3192
14.03.2003	Маня и Ко	табуретка	50	152,02р.	5	7220,95
13.07.2002	Маня и Ко	веревка	220	45,00р.	5	9405
24.10.2002	Маня и Ко	сикатор	3	10,52р.	5	29,982

Рис. 27. Результат выполнения запроса на выборку

### Корректировка данных средствами запроса

#### 1. Использование групповых операций в запросах

Групповые операции позволяют выделить группы записей с одинаковыми значениями в указанных полях и использовать для других полей этих групп определенную статистическую функцию. Пусть требуется создать запрос для таблицы **Заказ-Товар**, где количество для одинаково наименования товара суммируется.

1) Создадим запрос на выборку для таблицы **Заказ-Товар**. Из списка таблицы **Заказ-Товар** перенесем в бланк запроса поля **Код товара**, по которому должна производиться группировка и **Количество товара**, будем использовать функцию **Sum** для подсчета количества товара.

2) Нажмем на кнопку *Групповые операции* (при отсутствии строки *Групповые операции* выполнить команду *Вид – Групповые операции(Показать или скрыть – Итоги)*). Заменяем слово *Группировка* в столбце **Кол-во товара** на функцию **Sum** (рис. 28).



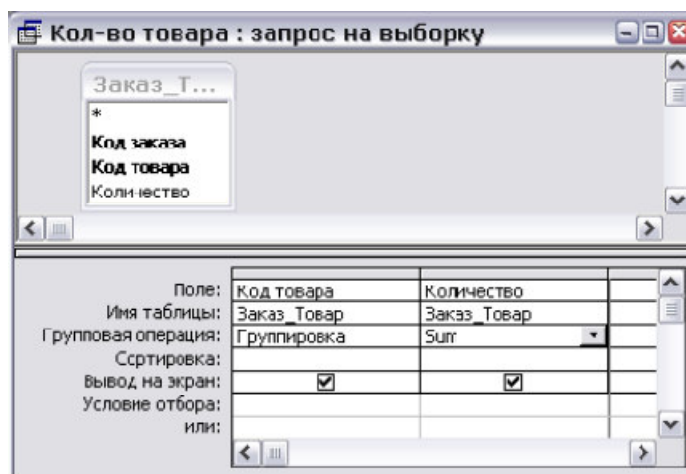


Рис. 28. Запрос с групповой операцией

Сохранить запрос с именем **Кол-во товара**.

3) Откроем запрос на выполнение. Убедитесь, что в таблице просуммированы данные по каждому из товаров в заказах.

## 2. Конструирование запроса на создание таблицы.

*Запрос на создание таблицы* используется для сохранения результата запроса. Этот вид запроса основан на *запросе на выборку*, но в отличие от него сохраняет таблицу с результатами запроса. Необходимость в сохранении результатов запроса возникает, например, когда невозможно построить запрос непосредственно на другом запросе. Сформируем запрос на создание таблицы, используя запрос **Кол-во товара**. Для этого выполним следующие действия в режиме *Конструктор*:

1) Преобразуем этот запрос в запрос на создание таблицы. Выбирается пункт меню **Запрос ~ Создание таблицы**.

2) В окне **Создания таблицы** введем имя создаваемой таблицы – **Кол-во товара в заказе**. Закроем запрос. Обратите внимание на изменение пиктограммы запроса **Количество товара**.

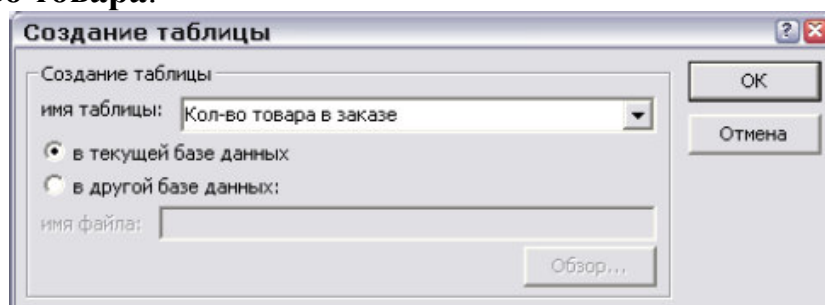


Рис. 29. Определение имени таблицы, создаваемой в запросе

3) Запустите модифицированный запрос **Количество товара**.

4) Перейдите на вкладку **Таблица** и убедитесь в создании новой таблицы «**Количество товара в заказе**»

## 3. Конструирование запроса на обновление.

При выполнении этого запроса изменения вносятся в группу записей, отбираемых с помощью указанных пользователем условий отбора. Значения для изменений в полях определяются в бланке запроса в строке *Обновление*.

1) Добавим в БД **Учет заказов** – таблицу **Склад**.

Наименование полей	Тип	Примечание
Код_склада	Счетчик	Ключевое поле

Код_товара	Числовой	
Получено	Числовой	
Остаток	Числовой	

Настроить связи, сделать подстановку нужных полей. Заполнить таблицу записями для всех товаров. При этом значение поля **Получено** должно быть больше значения поля **Остаток** и оба должны быть положительными.

2) Создадим запрос на обновление на примере обновления поля **Остаток** в таблице **Склад**. Остаток высчитывается по формуле: *Получено на складе – Кол-во товара в заказе*. Количество товара в заказе получено в запросе на выборку **Кол-во товара** с использованием функции *Sum*.

**!Запрос на обновление непосредственно на основе запроса построить нельзя!**

Поэтому используется для обновления не запрос, а таблица **Кол-во товара в заказе**, полученная по запросу на создание таблицы. Для формирования запроса на обновление сначала создается запрос на выборку на основе трех таблиц – обновляемой таблицы **Склад**, таблицы **Товар** и таблицы **Кол-во товара в заказе**. Для преобразования запроса на выборку в запрос на обновление выбирается пункт меню **Запрос ~ Обновление**. Перетащить поле **Остаток** из списка таблицы **Склад**. В строке **Обновление** ввести формулу:

**[Получено]-[Sum-Количество]**

Сохранить запрос под именем **Обновление Склада**.

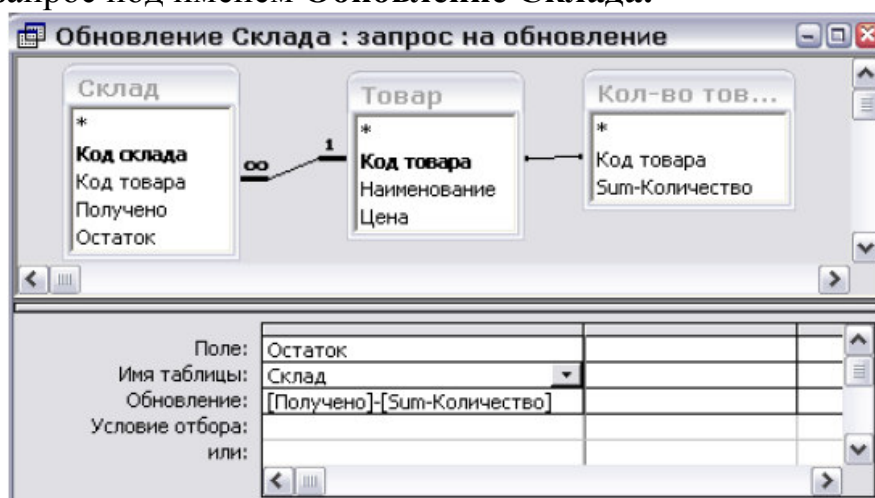


Рис. 30. Запрос на обновление таблицы Склад

Выполненные нами запросы на создание и обновление должны работать по следующей схеме: сначала вносятся данные в таблицы **Заказ\_Фирма** и **Заказ\_Товар**, т.е. добавляются товары в заказ. Следовательно, их необходимо со склада списать. Откроем таблицу **Склад**, посмотрим на столбец **Остаток**, запомним значения. Для списания товара вызываем запрос на создание таблицы **Количество товара**, если запрос уже выполнялся и таблица уже существует, подтвердим её замену. После этого запускаем запрос **Обновление склада**, при этом СУБД, учитывая данные только что измененные данные в таблице **Количество товара**, изменит значения остатков в таблице **Склад**. Убедиться в этом можно, открыв таблицу **Склад** и сравнив новые значения с запомненными старыми.



### **Индивидуальное задание**

1. Для четных вариантов выполнить запрос с произвольной выборкой на отбор товара по первой букве наименования (создать форму и запрос).
2. Для нечетных вариантов – запрос с произвольной выборкой из таблицы **Склад** товаров, значение остатка которых меньше вводимого пользователем (создать форму для ввода барьера остатка и запрос).

## Лабораторная работа №17. Редактирование базы данных средствами VBA

**Цель работы.** Разработать форму для работы с заказами, учитывая выбор клиента и заказываемых им товаров, внесение в базу текущей даты заказа, наличие товаров на складе и списание их при оформлении заказов. Реализованы эти функции должны быть в виде процедур обработки событий объектов формы, написанных на языке VBA (VisualBasicforApplication).

### Контрольный пример

#### 1. Создаем макет формы заказов.

Создаем новую форму в режиме *Конструктора* (Формы ~ Создать ~ Конструктор), с помощью контекстного меню добавляем заголовок и примечание формы. После этого перетаскиваем на форму *поле со списком*, при этом должны быть включены мастера (иконка «Волшебная палочка» панели инструментов). В окне создания списка (рис. 31.) указываем, что данные будут браться из таблицы, в следующем окне мастера выбираем эту таблицу.

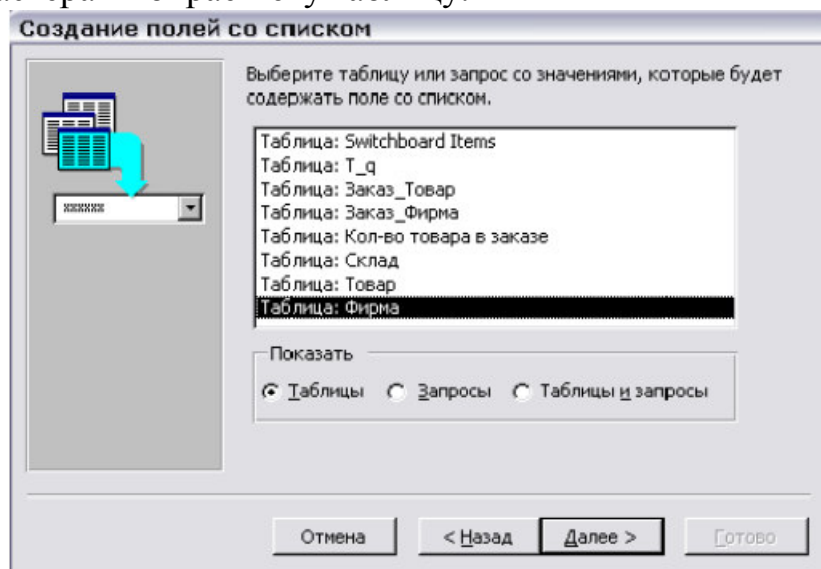


Рис. 31. Мастер создания полей со списком

Поскольку необходимо выбирать фирму-заказчика, то указываем таблицу **Фирма**. Затем выбираем нужное поле – **Наименование**, настраиваем ширину списка, даем ему имя и нажимаем на *Готово*. После этого на форме корректируем сопроводительную надпись списка. Аналогичным образом формируем список для наименований товаров (таблица **Товар**).

Кроме этого, необходимо добавить на форму пять полей с подписями и четыре кнопки, причем две из них можно сделать с помощью мастера (кнопки *Отмена* и *Добавить фирму*). Для первой кнопки после перетаскивания кнопки на форму в окне *Создания кнопок* указываем категорию *Работа с формой* и действие – *Заккрыть форму*. Для добавления фирмы необходимо просто открыть форму **Фирма**, созданную ранее, где и добавить новую запись (категория – *Работа с формой*,

действие – *Открыть форму*). Для остальных кнопок будут написаны процедуры обработки.

Добавим на форму группировочные рамки и линии оформления, в итоге получим форму рис. 32.

Рис. 32. Макет формы работы с заказами

## 2. Схема работы с формой.

Подразумевается, что работа пользователя будет проходить по следующему алгоритму:

1) Выбирается фирма из списка. При её отсутствии пользователь переходит на форму фирмы с помощью кнопки *Добавить фирму* и добавляет клиента.

2) Нажатием кнопки *Новый заказ* пользователь формирует новую запись в таблице заказов. Программа отображает номер заказа.

3) Выбирается товар из списка. Система отображает его цену и количество на складе. Если эти параметры удовлетворяют пользователя, он вносит необходимое для заказа количество в соответствующее поле и нажимает кнопку *Добавить в заказ*. При этом заказанное количество товара со склада списывается.

4) Повторяется шаг 3 до тех пор, пока все товары не будут добавлены в заказ. После добавления каждого из них система пересчитывает общую стоимость заказа в соответствующем поле.

## 3. Отображение информации на форме в соответствии с выбранным значением списка.

Очевидно, что при выборе товара из списка в поле *Остаток* должно высвечиваться имеющееся количество его на складе, а в поле *Цена* – цена из таблицы **Товар**. Для этого напишем процедуры обработки такого события, как *Изменение* поля со списком. Процедуры пишутся в редакторе VBA, вызов его осуществляется следующим образом. 1) Вызываем с помощью контекстного меню окно свойств для поля со списком товаров (щелкаем по полю правой кнопкой мыши, выбираем пункт *Свойства*). 2) Выбираем вкладку *События*, для пункта

Изменениевыбираем [Процедура обработки] и нажимаем на [...] рядом. В появившемся окне (основной части) представлены все процедуры нашей формы. Конкретно же процедура, связанная с изменением списка, должна выглядеть примерно следующим образом: **Не забудьте учесть, что номера кнопок, полей и полей со списком, названия переменных у вас могут быть другими! Это касается всех процедур приведенных ниже!**

```
Private Sub ПолеСоСписком29_Change()
Dim b As Integer ' заводим целочисленную переменную
Dim rs As Recordset ' заводим переменную типа запись
b = ПолеСоСписком29.Value ' присваиваем переменной значение из списка товаров
Set rs = CurrentDb.OpenRecordset("SELECT * FROM Товар WHERE Код_товара=" + Str(b))
' формируем динамический sql-запрос на выбор из таблицы товаров товара с нужным
кодом
Поле16 = rs!Цена ' считываем в поле значение цены, выбранного товара
Set rs = CurrentDb.OpenRecordset("SELECT * FROM Склад WHERE Код_товара=" + Str(b))
' аналогично для количества со склада
Поле14 = rs!Остаток
End Sub
```

Необходимо также учесть, что при открытии формы поля **Цена** и **Остаток** должны быть очищены. Для этого в окне свойств формы выбираем *События ~ Открытие ~ [Процедура обработки]* и формируем следующую или похожую процедуру:

```
Private Sub Form_Open(Cancel As Integer)
Поле14.SetFocus
Поле14.Text = ""
Поле16.SetFocus
Поле16.Text = ""
End Sub
```

#### 4. Добавление записей в таблицы программным способом.

Кнопка *Новый заказ* связана с процедурой добавления в таблицу **Заказ\_Фирма** новой записи. При этом код заказа генерируется автоматически (поле типа Счетчик), код клиента определяется по значению поля со списком фирм, а дата ставится текущая. Обработаться будет нажатие кнопки. Для организации этой процедуры в окне свойств кнопки *Новый заказ* выбираем *События ~ Нажатие кнопки ~ [Процедура обработки] ~ [...]*. В окне редактора вводим следующую процедуру:

```
Private Sub Кнопка6_Click()
Set rs = CurrentDb.OpenRecordset("SELECT * FROM Заказ_Фирма") 'с помощью
динамического sql-запроса открываем нужную таблицу
rs.AddNew ' добавляем новую запись
rs!Код_фирмы = ПолеСоСписком27 'в поле Код_Фирмы вносим значение из поля со
списком фирм
rs!Дата = Date 'в поле даты вводим текущую дату (функция Date())
rs.Update 'обновляем запись
rs.Bookmark = rs.LastModified
Поле31 = rs!Код_заказа 'отображаем код нового заказа сгенерированный
автоматический в поле формы.
End Sub
```

Кнопка *Добавить в заказ* добавляет записи в таблицу **Заказ\_Товар**, отвечает за списание заказанного количества со склада и подсчет общей стоимости заказа. Код заказа добавляется из поля, в которое предыдущая процедура внесла код текущего заказа.

```
Private Sub Кнопка18_Click()  
Dim b As Integer  
Dim rs As Recordset  
b = ПолеСоСписком29.Value 'присваиваем переменной значение поля со списком (код  
товара)  
Set rs = CurrentDb.OpenRecordset("SELECT * FROM Склад WHERE Код_товара=" + Str(b))  
' находим в таблице склад строку нужного товара с помощью SQL-запроса.  
rs.Edit 'включаем режим редактирования записи  
rs!Остаток = rs!Остаток - Поле10 'изменяем остаток товара на величину заказанного  
количества  
rs.Update 'обновляем запись  
Set rs2 = CurrentDb.OpenRecordset("SELECT * FROM Заказ_Товар") 'открываем таблицу  
Заказ_Товар  
rs2.AddNew 'добавляем запись  
rs2!Код_заказа = Поле31 'вносим в поле код заказа значение номера заказа,  
сгенерированного в результате предыдущей процедуры  
rs2!Код_товара = ПолеСоСписком29 'вносим код товара из поля со списком товаров  
rs2!Количество = Поле10 'вносим количество из поля формы  
rs2.Update 'обновляем запись  
rs2.Bookmark = rs.LastModified  
Sum = Sum + Val(Поле10.Value) * Val(Поле16.Value) 'считаем стоимость как произведение  
цены на количество товара, накапливая её для всех товаров (***)  
Поле33 = Sum 'выводим значение стоимости  
Поле10.SetFocus  
Поле10.Text = "" 'очищаем поле количества  
End Sub
```

(\*\*\*) *Примечание:* чтобы переменная Sum не обнулялась после каждого выполнения процедуры и стоимость накапливалась, необходимо объявить эту переменную как глобальную. Для этого в разделе (General) добавляем строку:

```
Public Sum As Integer
```

5. Сохраните полученную форму.

6. Запустите форму и сформируйте новый заказ.

### Индивидуальное задание

1. Для четных вариантов изменить код так, чтобы при попытке заказать количество товара большее имеющегося на складе система выдавала предупреждение, и данные не вносились.

2. Для нечетных вариантов добавить поле **Итого стоимость со скидкой** и рассчитать эту стоимость, т.е. для описанной функции Sum учесть скидку фирмы, сделавшей этот заказ. (Получить значение скидки из таблицы **Фирма** и умножить Sum на (1-Скидка/100), т.к. скидка выражена в %).

### Контрольные вопросы

1. Как описываются переменные в VBA?

2. Для чего используется тип переменной Recordset?
3. Какая команда служит для добавления записей в базу данных?
4. Как средствами VBA вывести на форме значения полей таблицы?
5. Как очистить поля формы командами VBA?

## Лабораторная работа №18.Разработка отчетов

**Цель работы.** Используя стандартные средства Access, разработать отчеты к базе данных.

Средства Access по разработке отчетов предназначены для создания макета отчета, по которому может быть осуществлен вывод данных из таблиц в виде выходного печатного документа. При необходимости вывода в отчете данных из нескольких таблиц в качестве основы для отчета может быть использован многотабличный запрос. Создавая отчет, можно группировать данные по нескольким уровням. Для каждого уровня может производиться вычисление итогов, определяться заголовки и примечания по каждой группировке. Отчет можно создавать с помощью *мастера* или в режиме *конструктора отчетов*.

### Контрольный пример

#### 1. Создание отчета с помощью Мастера

1) В окне базы данных щелкнем по закладке *Отчет*, а затем нажмём на кнопку *Создать*.

2) В окне диалога *Создание отчета* выберем из списка *Мастер отчетов*, а ниже выберем таблицу (например, таблица **Фирма**), раскрыв предлагаемый список (рис. 33), затем нажмём кнопку *OK*.

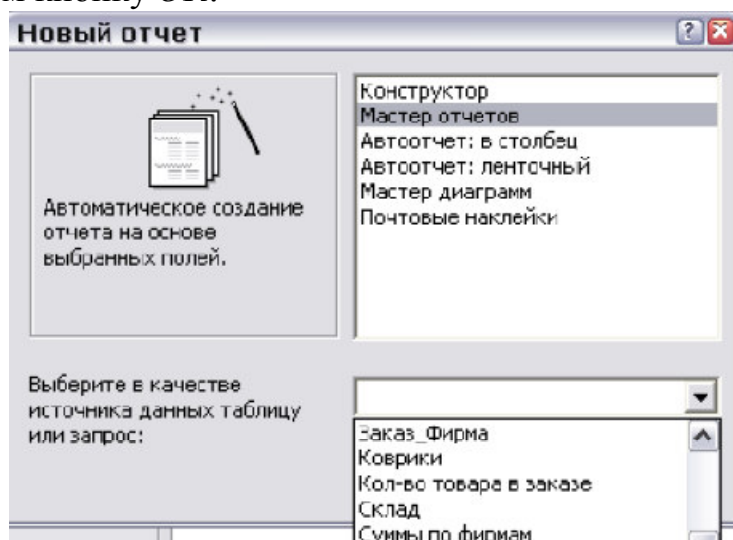


Рис. 33. Создание отчета в режиме мастера отчетов

3) В окне *Мастера по созданию отчетов* выберем поля для отчета (Код



фирмы указывать не обязательно), с помощью кнопок переместим поля в правую часть окна. Нажмём кнопку *Далее* (рис. 34).

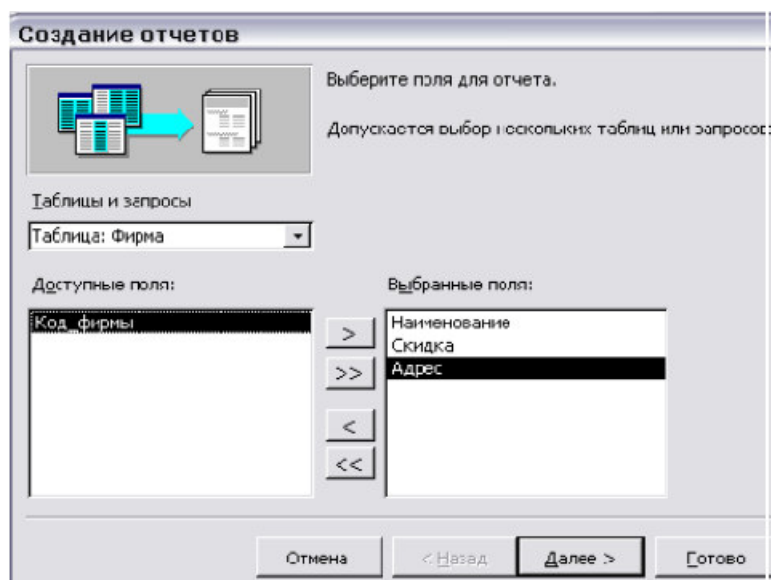


Рис. 34. Выбор полей для отчета

- 4) Выберем уровень группировки по Наименованию фирмы.
- 5) Порядок сортировки укажем – Отсутствует.
- 6) Выберем ступенчатый макет.
- 7) Выберем понравившийся стиль.
- 8) Укажем имя Отчета. Нажмём кнопку *Готово* (рис. 35).
- 9) Закроем документ отчета.

Фирмы		
Наименование	Скидка	Адрес
Атлет	4	7-я линия, 49
Адвентис и Ко	8	Ленина, 46
Апертые слоникси	5	11-я линия, 48
Бард и Ко	3	1-я 2
Глыба+	6	8-я линия, 46
Игнат	8	Школьная 2

Рис. 35. Внешний вид отчета по фирмам

## 2. Создание отчета с помощью **Конструктора**

Отчеты можно создавать и вручную – в режиме *Конструктора*. Для этого необходимо:

- 1) В окне базы данных выберем вкладку *Отчеты*.
- 2) Нажмём кнопку *Создать*.
- 3) В окне диалога *Новый отчет* выберем *Конструктор*.



4) Выберем имя таблицы или запроса, содержащих данные, по которым строится отчет. Для того чтобы использовать в отчете данные из нескольких таблиц, создавайте отчет на основе запроса.

5) После нажатия кнопки *ОК* новый отчет будет открыт в окне *Конструктора отчетов* (рис. 36).

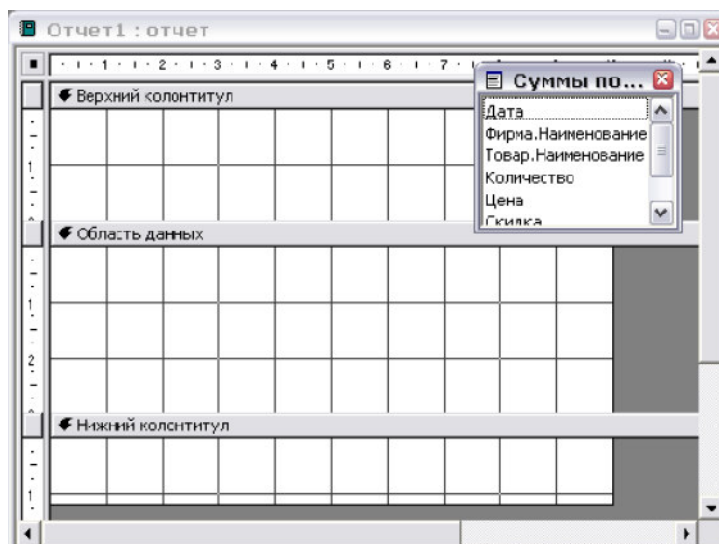


Рис. 36. Форма отчета

6) По умолчанию отчет состоит из 3-х частей:

- верхний колонтитул. Здесь указывается информация об отчете, т.е.: название отчета, реквизиты, наименования полей и др.;
- область данных. В этой части отчета выводятся данные из таблицы или запроса;
- нижний колонтитул. Здесь указывается сводная информация об отчете, т.е. дата, количество страниц, др.

Редактирование отчета происходит так же, как и экранных форм. Создадим по вышеописанной схеме прайс-лист компании. 1) Выберем в качестве источника данных таблицу Товар. 2) Перетаскиваем на лист в область данных поля Наименование и Цена. 3) Подписи полей переносим в область верхнего колонтитула, там же указываем название отчета и реквизиты фирмы. 4) В области нижнего колонтитула размещаем поля с информацией о текущей дате и номере страницы. Для этого в созданное поле вводится выражение с помощью *Построителя выражений* (контекстное меню ~ Свойства ~ закладка Данные ~ Данные (кнопка с [...])), где выбирается тип *Общие выражения*.

### 3. Отчеты с группировкой и вычислением итогов

Для сложных отчетов с группировкой и подведением итогов оптимально создание отчета с помощью *Мастера* с доработкой в *Конструкторе*. Так, при создании отчета **Суммы по фирмам** необходимо сгруппировать проданные товары по заказам (по датам), а заказы – по фирмам-клиентам и вычислять суммы по каждому клиенту и в целом по обороту. Для этого необходимо при создании отчета *Мастером* выполнить следующие действия: 1) В качестве источника данных выбрать запрос **Сумма по фирмам 1**. 2) Поместить на отчет все поля запроса. 3)

Указать уровни группировки сначала по названию фирмы, а потом по дате (рис. 37.) В окне *Интервалы группировки* (щелчок по кнопке *Группировка*) для даты поставить *обычный интервал*.

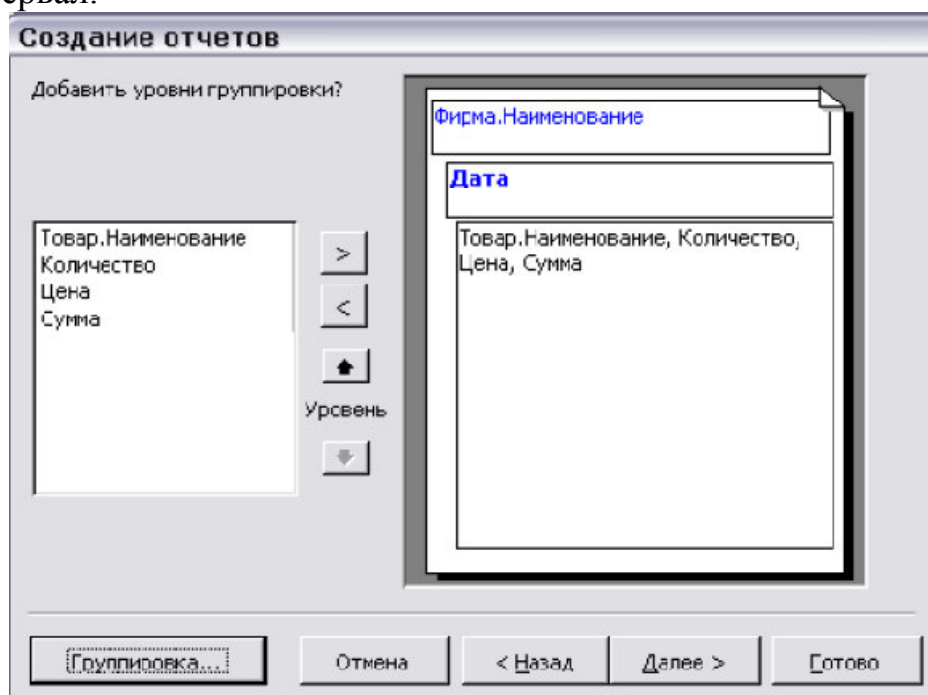


Рис. 37. Задание группировки в отчете

5) Для подведения итогов на следующем шаге создания отчета (окно сортировки) нужно нажать кнопку *Итоги* и в появившемся окне выбрать необходимые статистические формулы и поля, к которым они должны быть применены (рис. 38).

6) На следующих шагах мастера выбрать макет, стиль, имя отчета.

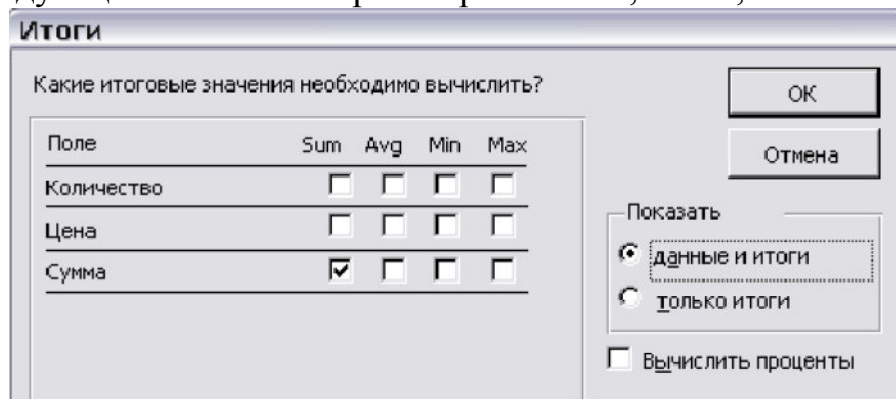


Рис. 38. Задание итоговых значений

7) После окончания работы *Мастера* отчет можно откорректировать в *Конструкторе*: убрать лишние итоговые значения (по умолчанию итоги считаются на каждом уровне группировки), отредактировать заголовки полей, оформление и пр. Пример созданного отчета с применением *Мастера* и доработанного в режиме *Конструктора* представлен на рис. 39, а его вид в режиме *Предварительного просмотра* на рис. 40.



## Лабораторная работа №19. Создание приложения пользователя

**Цель работы.** Для организации эффективной работы пользователя нужно создать целостное приложение данной предметной области, в которой все компоненты приложения должны быть сгруппированы по функциональному назначению.

Большое количество объектов БД, не сгруппированных по функциям приложения, затрудняет выполнение пользователем задач обработки данных в автоматизированной предметной области.

Для объединения объектов в едином диалоговом приложении создаются так называемые *кнопочные формы*. Кнопочная форма является панелью управления приложением. Кнопки такой формы обеспечивают вызов других кнопочных форм, а также отдельных объектов – отчетов, форм, макросов, модулей, с которых начинается решение задачи. Сюда же помещаются кнопки для возврата к кнопочным формам предыдущих уровней, выхода из Access.

В Access имеется средство автоматизированной разработки формы по управлению приложением, называемое *диспетчером кнопочных форм*.

Другим способом создания целостного приложения является организация пользовательской панели управления – *меню системы*. Ее пункты, в том числе и вложенные, выполняют те же функции, что и кнопки кнопочной формы.

### Контрольный пример

#### Создание панели управления приложения

Как было указано выше, на панели управления приложением размещаются кнопки для вызова таблиц, форм и отчетов. Вызов же запросов происходит через запуск макросов.

1. Создание кнопочной формы.

1) Создадим макросы для вызова всех созданных запросов.

Для создания макросов перейдем на закладку *Макросы* и выберем кнопку *Создать*.

В окне *Макроса* выберем макрокоманду *Открыть Запрос* (рис. 41). В поле *Имя запроса* указываем имя соответствующего запроса, например **Товары по дате**.

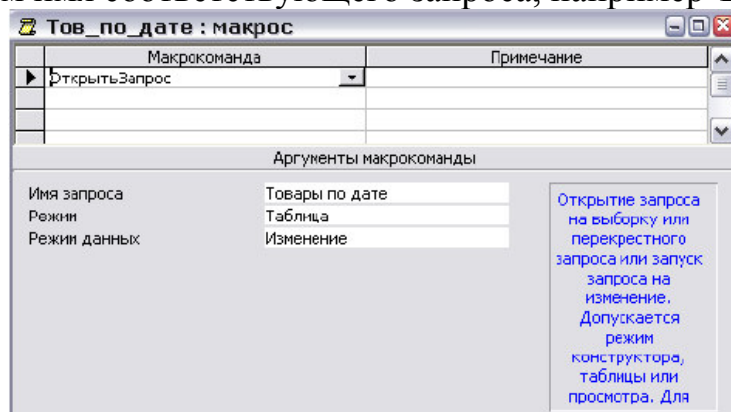


Рис. 41. Создание макроса для открытия запроса **Товар по дате**

Остается только закрыть запрос и дать ему имя (может совпадать с именем запроса). Повторите аналогичные действия для макросов всех запросов.

2) Создадим главную кнопочную форму приложения **Учет заказов**

Выберем пункт меню Сервис~Служебные программы~Диспетчер кнопочных форм.

В окне Диспетчера кнопочных форм подтвердим создание кнопочной формы.

В следующем окне Диспетчера кнопочных форм появится окно со строкой **Главная кнопочная форма** (рис. 42). В этом окне диспетчера кнопочных форм формируется список кнопочных форм разных уровней.

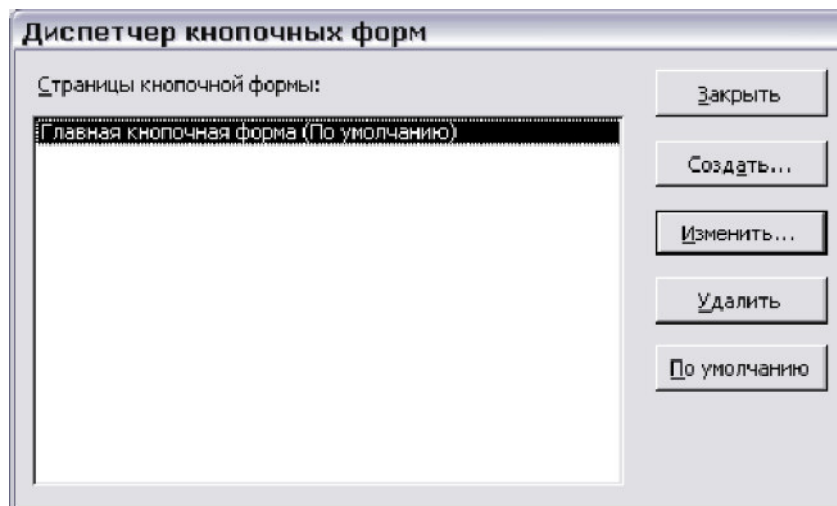


Рис. 42. Окно списка кнопочных форм приложения

3) Следующим шагом создадим другие кнопочные формы приложения:

Создадим три кнопочные формы для решения задач Ввода и редактирования данных, Поиска информации и Формирования отчетов.

Для создания кнопочной формы в окне диспетчера кнопочных форм нажмем кнопку *Создать*. В окне *Создание* в поле *Имя страницы кнопочной формы* (рис. 43) введем имя первой кнопочной формы **Ввод и редактирование данных**.

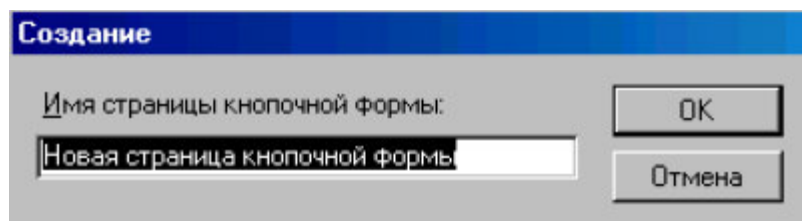


Рис. 43. Окно создания пустой кнопочной формы

После нажатия кнопки *Ок* в окне диспетчера кнопочных форм появится строка **Ввод и редактирование данных**.

Аналогичными действиями создаем кнопочные формы **Поиск данных** и **Отчетные формы**.

4) Отредактируем имена и сформируем элементы кнопочной формы.

Элементами кнопочных форм являются кнопки с подписями. Подпись задается пользователем и должна по возможности кратко и точно описывать действия, которые будут выполняться при нажатии кнопки. Для формирования элементов главной кнопочной формы в окне *Диспетчера кнопочных форм* выделим соответствующую строку и нажмем кнопку *Изменить*.

В окне диалога *Изменение страницы кнопочной формы* (рис. 44) переименуем **Главная кнопочная форма** в **Приложение «Учет заказов»**, введя это имя в поле *Название кнопочной формы*.

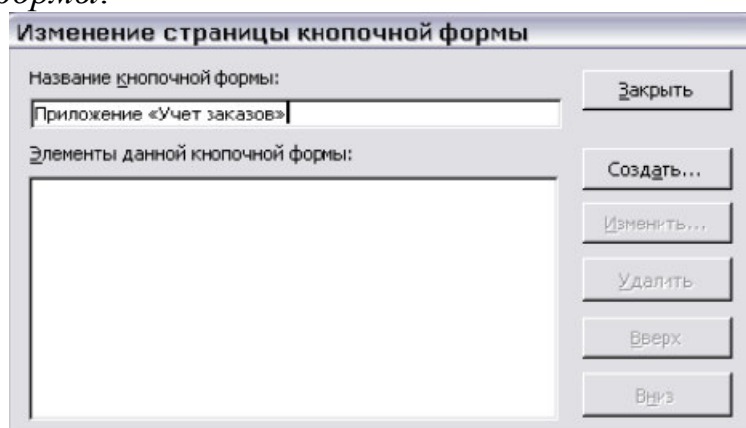


Рис. 44. Окно создания кнопочной формы и ее элементов

Элементы, создаваемые в кнопочной форме, могут быть подразделены на два типа:

- Элементы, предназначенные для организации вызова других кнопочных форм;
- Элементы, обеспечивающие такие действия как открытие формы, отчета, запуск макроса, программы и выход из приложения с закрытием базы данных.

#### 4.1. Сформируем элементы для вызова кнопочных форм:

Для формирования в главной кнопочной форме **Приложение «Учет заказов»** кнопки вызова подчиненной формы нажмем кнопку *Создать*.

В окне *Изменение элемента кнопочной формы* в строке *Команда* выберем **Переход к кнопочной форме**. В строке *Кнопочная форма* выбирается форма, к которой должна обеспечивать переход создаваемая кнопка текущей формы. Откроем список созданных для приложения кнопочных форм и выберем **Ввод и редактирование данных**. В строке *Текст* введем подпись для этой кнопки **Ввод и редактирование данных**. Результат действий представлен на рис. 45. Завершим операцию нажатием кнопки ОК.

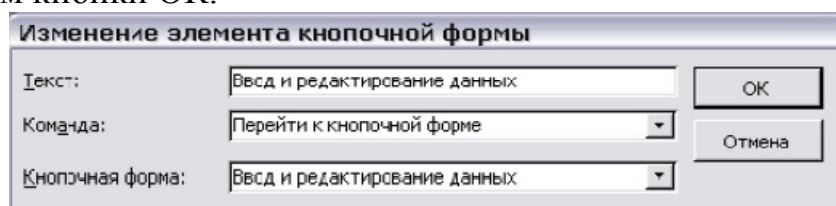


Рис. 45. Окно создания элемента кнопочной формы

Аналогичным образом создаем кнопки вызова других подчиненных форм. В поле элементы данных у вас должны появиться три созданных элемента.

По окончании этого процесса закрыть окно *Изменение страницы кнопочной формы* и перейти в окно *Диспетчера кнопочных форм*.

#### 4.2. Формируем элементы для открытия объектов приложения:

Для формирования кнопок, обеспечивающих выполнение пользователем своих задач, необходимо, чтобы они были связаны с одним из объектов приложения. Для связи создаваемой кнопки с нужным объектом надо в окне *Изменения объекта кнопочной формы* выбрать в строке *Команда* одну из команд:



- Открытие формы в режиме добавления
- Открытие формы в режиме редактирования
- Открытие отчета
- Запуск макроса
- Запуск программы

Третья строка этого же окна будет содержать наименование формы, отчета или макроса, соответствующее выбранной во второй строке команде.

1. Создадим в кнопочной форме **Ввод и редактирование данных** кнопку для работы с формой **Товары**.

В окне *Диспетчера кнопочных форм* выберем форму **Ввод и редактирование данных**. Для создания элементов в этой форме нажмем кнопку *Изменить*. В открывшемся окне *Изменение страницы кнопочной формы* нажмем кнопку *Создать*. В окне *Изменение элемента кнопочной формы* выберем команду **Открыть форму для изменения**. В появившейся строке *Форма* откроем список доступных форм и выберем форму **Товар**. В строку *Текст* введем подпись кнопки **Справочник Товаров**. Результат действий представлен на рис. 46.

Рис. 46. Окно создания элемента кнопочной формы

2. Аналогичным образом создаем кнопки для запуска остальных необходимых форм, запросов и отчетов, не забывая, что запросы открываются через макросы.

В каждой из подчиненных кнопочных форм необходимо создать *кнопку для возврата* к главной форме **Приложение «Учет заказов»**.

Для этого в окне *Изменение элемента кнопочной формы* для каждой из создаваемых форм (кроме формы по умолчанию) введем параметры, приведенные на рис. 47.

Рис. 47. Создание кнопки для возврата в главную форму

После создания всех необходимых кнопок закрываем *Диспетчер кнопочных форм*. В списке форм базы данных появляется форма с именем **Кнопочная форма**. Внешний вид данной формы можно отредактировать в режиме *Конструктора*. Для запуска созданного приложения необходимо эту форму открыть. Внешний вид такой формы представлен на рис. 48. Если вы случайно удалите кнопочную форму из приложения, то настройки её останутся и решением в этом случае будет создание

пустой кнопочной формы в новой базе данных и копирование её в исходное приложение. Все настройки будут перенесены на эту новую форму.

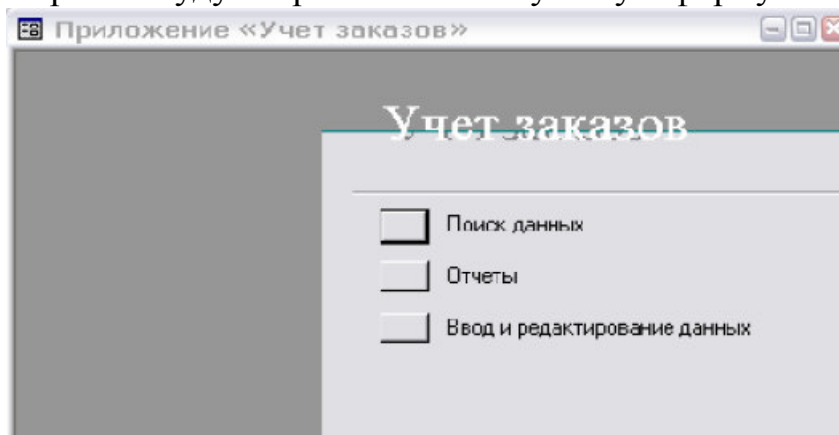


Рис. 48. Внешний вид главной страницы кнопочной формы

### **Индивидуальное задание**

1. Создать макросы ко всем запросам, построенным в режиме Конструктора (Л.р. №15). Учесть при этом, что для запроса с произвольной выборкой необходимо открытие формы, а не запроса.

2. Откорректировать кнопочную форму в соответствии с появившимися элементами.

3. Реализовать возможность изменения приложения (кнопочной формы) из самого приложения (добавить на форму определенную команду из стандартного списка). Добавить в меню системы команду, позволяющую автоматически конвертировать таблицы Access в таблицы Excel.

### **Контрольные вопросы**

1. Опишите технологию создания макросов в Access.
2. С какой целью создается приложение пользователя?
3. Для чего служит диспетчер кнопочных форм?
4. Как создать страницу кнопочной формы?
5. Как создать меню пользователя в Access?



## **Лабораторная работа №20. Автоматизация приложения с помощью VBA. Объекты базы данных VBA**

**Тема:** Автоматизация приложения с помощью VBA.

**Цель:** Научиться создавать модули.

### **Задание**

1. Создать процедуру для проверки правильности вводимой даты.
2. Создать функцию для автоматического установления скидки.
3. Создать процедуры для контроля количества товаров.
4. Создать процедуру для добавления записи в таблицу.
5. Создать процедуру для удаления записи из таблицы.
6. Создать процедуру поиска заказчика из Омска.
7. Создать функцию для копирования цены из поля таблицы в поле формы.
8. Создать процедуру подсчета числа заказов на определенную дату.
9. Создать процедуру возврата в окно базы данных.
10. Создать процедуру обработки события отсутствия в списке.
11. Создать процедуры обработки событий для отбора данных при открытии отчета.

### **Основные сведения**

#### **Основные понятия VisualBasic для приложений**

**VBA (VisualbasicforApplications)** – это общий язык программирования для приложений MicrosoftOffice, включая Word, Excel, PowerPoint, Access и MicrosoftVisualBasic.

Все программы хранятся в модулях. База данных может содержать два вида модулей: стандартные, являющиеся объектами базы данных, и модули форм и отчетов, являющиеся частью этих объектов.

Стандартные модули используются для создания и хранения процедур, выполняемых из запросов или нескольких форм или отчетов приложения. Процедуры, определенные в стандартном модуле, могут вызываться из любого места в приложении. Чтобы создать новый модуль, нажмите кнопку **Создать** во вкладке **Модули** окна базы данных. Разумно называть модули в соответствии с их назначением.

В языке VBA имеются три основных элемента : объекты, события и методы. Объекты – элементы, входящие в состав приложения. События – все, что происходит с объектами. Объект реагирует на событие путем выполнения некоторого действия, состоящего из одного или большего числа методов.

#### **Модули форм и отчетов**

Чтобы облегчить создание процедур обработки событий, возникающих в форме или отчете, Access предоставляет модуль класса, связанный с каждой формой или отчетом. В модуле класса формы или отчета содержатся специальным образом именованные процедуры обработки событий, личные процедуры, доступные только из модуля класса, и общие процедуры, которые можно вызвать как методы класса. Отредактировать модуль формы или отчета можно, открыв форму или отчет в режиме конструктора и выполнив команду **Вид → Программа**.

#### **Объекты базы данных в VBA**

Объекты базы данных (таблицы и запросы) и объекты приложений (формы и отчеты) также имеют соответствующие типы данных. В следующей таблице приведены наиболее часто встречающиеся типы объектных данных Access VBA и содержащие их объектные библиотеки.

Тип объекта	Библиотека	Тип объекта базы данных
Database	DAO 3.6	Базы данных, открытые при помощи ядра Jet 3.0
Form	Access	Формы, включая подчиненные
Report	Access	Отчеты, включая подчиненные
Control	Access	Элементы управления форм и отчетов
QueryDef	DAO 3.6	Объекты-запросы (эквиваленты операторов SQL)
TableDef	DAO 3.6	Объекты-таблицы (имеющие структуру, индексы)
RecordSet	DAO 3.6	Виртуальное представление таблицы или результирующий набор записей запроса

### Объект Recordset

Recordset – это объект, который используется для выполнения операций с записями. Объекты Database, QueryDef, TableDef включают метод OpenRecordset.

Существует три типа объекта Recordset: динамический, статический и таблица. Recordset типа таблицы представляет собой открытый буфер для таблицы. Динамический Recordset – множество записей, получаемое в результате выполнения запроса на выборку. Этот тип Recordset можно создать для выбора записей из локальной или присоединенной таблицы (таблицы из внешней базы данных).

Для динамического типа Recordset можно использовать методы Find при поиске записей. Для Recordset типа таблицы можно использовать метод Seek.

### Ссылки на библиотеки объектов

Чтобы сделать объекты доступными для использования из модулей, Access 2000 использует ссылки. Ссылками Access 2000 по умолчанию являются VBA, Access 2000, DAO 3.6 Object Library и текущая база данных. Чтобы увидеть список ссылок, откройте вкладку Модули окна базы данных и затем выберите опцию меню Tools → Reference . Необходимые ссылки пометьте флажком. Объекты, на которые есть ссылки, представлены в списке окна Object Browser (вкладка *Модуль* → **View** → **ObjectBrowser**).

### События

Запустить действия, сохраненные в процедуре, можно одним из двух способов: путем выбора в меню или с помощью свойств событий. Процедура запускается после того, как пользователь выполнил действие, вызывающее конкретное событие. Access распознает событие, используя специальные свойства события для форм, отчетов и элементов управления (полей).

### Основные события Access.

Событие	Описание
---------	----------

Текущая запись (On Current)	Запись становится текущей
До обновления (Before Update)	Изменение данных до обновления
После обновления (After Update)	Изменение данных после обновления
Вставка (On Insert)	Вставка новой записи
Удаление (On Delete)	Удаление существующей записи
Открытие (On Open)	Открытие формы или отчета
Закрытие (On Close)	Закрытие формы или отчета
Меню (On Menu)	Выбор из пользовательского меню
Вход (On Enter)	Выбор элемента управления
Выход (On Exit)	Освобождение элемента управления
Двойное нажатие кнопки (On Dbl Click)	Двойной щелчок на элементе управления
Нажатие кнопки (On Click)	Щелчок на командной кнопке
Форматирование (On Format)	Изменение расположения данных в отчете
Печать (On Print)	Вызов печати данных отчета

События являются свойствами объектов и элементов управления Access. Каждый тип объекта имеет определенный список свойств событий. Мастера элементов управления могут генерировать процедуры событий для большинства стандартных форм, отчетов или операций с наборами записей. Например, Мастер кнопок создает процедуры для обработки события **Нажатие кнопки**. Если процедура требует индивидуальной обработки, то можно использовать построитель кода. Для открытия построителя кода, нажмите кнопку **Построить** справа от свойства события и выберите пункт **Программы**. Откроется окно модуля, в котором уже находится первый и последний оператор.

### Методы

Каждый объект или семейство имеют определенный набор методов, которые могут быть выполнены. Примеры некоторых методов формы: Refresh – обновляет все поля в основном источнике записей с изменениями, сделанными в форме, Set Focus – активизирует указанную форму или элемент управления.

Многие методы имеют аргументы, позволяющие уточнить, как они должны действовать на объект. Например, при применении метода OpenRecordset к таблице вы можете создать динамический, или обновляемый, набор записей (Dynaset) либо статический набор записей, позволяющий только просматривать данные.

После открытия набора записей для перехода к нужной записи можно использовать методы Move (MoveFirst, MoveLast, MoveNext, MovePrevious).

Для перехода к определенной строке, удовлетворяющей заданным критериям, используется один из методов Find (FindFirst, FindLast, FindNext, FindPrevious).

### **Соглашения по синтаксису**

Область определения переменной (или константы) определяет, известна ли она только одной процедуре, всем процедурам в модуле или всем процедурам в базе данных. Для объявления общей переменной используйте инструкцию Public в разделе описаний модуля. Чтобы переменную можно было использовать во всех процедурах некоторого модуля, объявите ее в разделе описаний стандартного модуля или модуля формы или отчета. Переменная или константа, описанная в процедуре, доступна только в этой процедуре.

**Инструкция Const** используется для объявления констант.

[Public / Private] Const {имя константы [As тип данных]}=<константное выражение>}

**Инструкция Dim** используется для объявления переменной или массива переменных.

Dim [ WithEvents ] {имя переменной [( [ <индекс> ], ... )] [ As [ New ] тип данных ] }, ...

где <индекс> есть [ нижняя граница To ] верхняя граница.

Ключевое слово New указывает, то объявляемая объектная переменная является новым экземпляром объекта. Ключевое слово WithEvents применяется в модуле класса для указания объектной переменной, использующейся при отклике на события, гнерируемые объектом ActiveX.

Для объявления переменной как объекта типа Database введите

Dim mydb As Database

Следующая инструкция описывает массив, содержащий строки фиксированной длины по 20 символов, причем 50 элементов массива пронумерованы от 51 до 100

Dim mymassiv (51 To 100) As String \* 20

**Инструкция Set** используется для присваивания значения объектной переменной.

Set имя переменной = ссылка на объект

Чтобы создать переменную, ссылающуюся на текущую базу данных, введите

Dim mydb As Database

Set mydb = CurrentDb()

Чтобы создать переменную, ссылающуюся на таблицу Клиенты в текущей базе данных, используя только что определенную переменную mydb, введите

Dim myTab As TableDef

Set myTab = mydb![Клиенты]

Чтобы создать переменную, ссылающуюся на поле Название таблицы Клиенты, введите

Dim myPole As Field

Set myPole = myTab![Название]

### **Ссылки на формы, отчеты, их свойства и элементы управления**

Ссылки на формы или отчеты можно делать только когда они открыты. Access использует коллекции Forms и Reports (множества открытых форм и отчетов), чтобы сохранить путь, из которого открываются соответствующие формы и отчеты. Коллекцию необходимо указывать, т. к. Access позволяет использовать одно и то же

имя для формы и для отчета. Объекты Form и Report имеют свойства. Пример ссылки на свойство Источник записей формы Товары:

Forms![Товары].RecordSource

Чтобы установить значение свойства формы в собственном модуле класса формы, используйте идентификатор Me – ссылку на самого себя:

Me.RecordSource

Чтобы сослаться на элемент управления Код товара формы Товары, введите:

Forms![Товары]![КодТовара]

### **Ссылки на подчиненные формы и отчеты**

Ссылка на подчиненную форму Заказанный товар в форме Заказы будет иметь вид :

Forms![Заказы]![Заказанный товар]

Для ссылки на свойство Вывод на экран рассмотренной выше подчиненной формы введите выражение :

Forms![Заказы]![Заказанный товар].Visible

Для ссылки на элемент управления Код товара подчиненной формы Заказанный товар, внедренной в форму Заказы, будет иметь вид :

Forms![Заказы]![Заказанный товар].Form![Код товара]

### **Функции и подпрограммы**

В VBA имеется два типа процедур : функции (процедуры Function) и подпрограммы (процедуры Sub). Процедуры обоих типов могут иметь аргументы – переменные, с помощью которых передаются значения, определяющие работу процедуры при конкретном вызове. Функции можно выполнять из любого места в Access, в том числе из выражений в запросах, а подпрограмму можно выполнять только при вызове из функции, из другой подпрограммы или как процедуру обработки события в форме или отчете.

### **Объект DoCmd**

Пользуясь методами объекта *DoCmd*, можно выполнять макрокоманды и команды меню в процедурах VBA. К наиболее употребляемым макрокомандам, которые выполняются из среды VBA, относятся :ApplyFilter (Применить фильтр), Close (Заккрыть), FindNext (Следующая запись), FindRecord (Найти запись), Maximize (Развернуть), Minimize (Свернуть), OpenForm (Открыть форму), OpenQuery (Открыть запрос), OpenReport (Открыть отчет) и др.

Например, чтобы открыть форму Клиент для ввода данных в режиме формы, введите :

DoCmd.OpenForm “Клиент”, acNormal , , , acAdd

Заккрыть форму Клиент можно с помощью инструкции:

DoCmd.Close acForm, “Клиент”

### **Аргументы в процедурах**

Иногда имена форм и отчетов в процедурах удобно заменять строковыми переменными. Например, существует кнопочная форма, из которой вызываются

другие формы. Для вызова форм используется одна повторяющаяся процедура, в которую через аргументы передаются имена форм.

Стандартный модуль в этом случае выглядит:

```
Public Function Открыть(strFormName As String) As Integer
    Screen.ActiveForm.Visible=False
    DoCmd.OpenForm strFormName
End Sub
```

Объект Screen ссылается на конкретную форму, отчет или элемент управления. Применяя объект Screen, можно сослаться на активный объект, не зная его имени.

Можно заменить объект Screen объектом Me, если приведенная выше процедура используется только в одной форме (Me.Visible=False).

При вызове этой процедуры в свойстве события *Нажатие кнопки* нужно ввести имя процедуры с именем конкретной формы:

= Открыть(“Заказы”).

### Использование функции DLookup

Функция DLookup выполняет поиск информации в еще не открытых таблицах.

Синтаксис:

DLookup("[Поля, которые нужно отобразить]","[Таблица представления]", "<Критерий поиска>")

где Таблица представления – таблица, в которой находятся искомые поля.

Критерий поиска имеет следующий формат:

"[Поле в таблице]" = '<Пример данных>'.

'<Пример данных>' обычно представляется в виде символьной константы.

Если же данные являются полем в таблице, то необходимо использовать синтаксис "&[Поле в рассматриваемой таблице]&"

При вводе названий полей и критериев для функция DLookup() запрещено использовать пробелы. Эту функцию можно использовать в процедурах VBA и в запросах для отбора данных. Если поле таблицы является текстовым, то его нужно заключать в одинарные кавычки.

### Внутренние константы

Помимо констант, которые поставляются вместе с Access, можно использовать любые константы из библиотек DAO и VBA. Внутренние константы всегда доступны. Имена внутренних констант имеют приставку из двух букв, указывающую на их принадлежность к библиотеке. Константы Access начинаются с букв as, константы DAO - с букв db, а константы VBA - с букв vb. Например, константа dbOpenTable открывает таблицу.

Внутренние константы используются в качестве аргументов в операторах вида DoCmd.Команда.

### Технология работы

#### Создание процедуры для проверки правильности вводимой даты

1. Откройте форму **Заказы** в режиме конструктора.
2. Откройте свойства поля **ДатаИсполнения**.

3. Щелкните по кнопке справа от свойства **До обновления** и выберите **Процедуру обработки событий**.

4. Введите следующую процедуру, которая выводит сообщение о неверной дате,

```
Private Sub ДатаИсполнения_BeforeUpdate(Cancel As Integer)
    If [ДатаИсполнения] < [ДатаРазмещения] Then
        MsgBox "Дата исполнения не может быть раньше даты размещения"
        Me![ДатаИсполнения].Undo
        Cancel = True
    End If
End Sub
```

С помощью метода **Undo** элемента управления происходит отмена изменения в поле и аргумент **Cancel** устанавливается равным **True** для отмены изменений в буфере записи.

5. Проверьте работу процедуры, перейдя в режим формы. Введите неверную дату и нажмите **Enter**.

### Создать функцию для автоматического установления скидки.

1. Выберите вкладку **Модули** окна БД и нажмите **Создать**.

2. Введите следующий фрагмент текста программы:

```
Function Disc (intQuantity As Integer) As Single
    Select Case intQuantity
        Case Is >= 1000
            Disc = 0.5
        Case Is >= 500
            Disc = 0.4
        Case Is >= 100
            Disc = 0.3
        Case Is >= 50
            Disc = 0.2
        Case Is >= 10
            Disc = 0.1
        Case Else
            Disc = 0
    End Select
EndFunction
```

3. Выберите опцию меню **Debug->Compile**, чтобы убедиться, что функция не содержит синтаксических ошибок.

4. Выберите опцию меню **View->Immediate Window** (Окно отладки).

5. В окне отладки введите **? Disc()**, поставив в скобки конкретное количество.

6. Нажмите **Enter** и проверьте результат.

7. Сохраните модуль, помня, что его имя не должно совпадать с именем функции.

8. Откройте форму **ЗаказанныйТовар** в режиме конструктора.

9. Убедитесь, что в свойстве **Имя поля** **Количество** задано **Количество**.

10. В свойстве **Данные поля** **Скидка** напишите  
=Disc(NZ([Количество])).

11. Проверьте работу функции, перейдя в режим формы.

## Создать процедуры для контроля количества товаров

1. В запрос — источник записей формы **ЗаказанныйТовар** дополнительно включите поля **НаСкладе**, **МинимальноеКоличество** и **Ожидается**.

2. Добавьте эти же поля в область данных формы **Заказанный товар**.

3. Предусмотрите выдачу надписи **Недостаточный запас** и дополнительное поле **НеобходимоЗаказать**.

4. Для свойства формы **Текущая запись** напишите процедуру :

```
Private Sub Form_Current()  
If (Me![НаСкладе] + Me![Ожидается]) <= Me![МинимальныйЗапас] Then  
    Me![НаСкладе].ForeColor = 255  
    Me!НедостаточныйЗапас.Visible = True  
    Me!НедостаточныйЗапас.ForeColor=255  
Else  
    Me![НаСкладе].ForeColor = 0  
    Me!НедостаточныйЗапас.Visible = False  
EndIf  
End Sub
```

Если сумма значений в полях **НаСкладе** и **Ожидается** меньше, чем **МинимальныйЗапас**, то количество товара задается красным цветом в поле **НаСкладе** и выводится надпись **Недостаточный запас**. Если минимальный запас достаточен количество в поле **НаСкладе** выводится черным цветом, а надпись **Недостаточный запас** скрывается. Работа процедуры проверяется при курсоре, установленном в поле **НаСкладе**.

4. Добавьте при отключенном мастере элементов в форму **ЗаказанныйТовар** кнопку **Заказать**.

5. Для этой кнопки в списке свойств задайте имя и подпись **Заказать**.

6. В свойстве **Нажатие кнопки** напишите процедуру:

```
Private Sub Заказать_Click()  
    Me![НеобходимоЗаказать] = Me![Минимальный Запас] - (Me![НаСкладе] +  
        Me![Ожидается])  
End Sub
```

7. В процедуру **Текущая запись** добавьте строку после ELSE

```
Me![НеобходимоЗаказать]=0
```

## Создать процедуру для добавления записи в таблицу

1. Создайте форму, содержащую кнопку с именем **НовыйТип**.

2. В свойстве события кнопки **Нажатие кнопки** напишите следующую процедуру

```
Private Sub НовыйТип_Click()  
    Dim dbs As Database, rst As Recordset  
    Set dbs = CurrentDb  
    Set rst = dbs.OpenRecordset("Типы", dbOpenDynaset)  
    With rst  
        .AddNew  
        ![Тип] = "Йогурты"  
        .Update  
    End With
```



```

End With
dbs.Close
End Sub

```

3. Откройте форму и проверьте работу кнопки. Просмотрев таблицу Типы, убедитесь, что запись добавлена.
4. В форму добавьте поле для ввода типа товара с именем **Тип**, проверьте, если поле **КодТипа** таблицы **Типы** не является счетчиком, то нужно будет ввести еще одно поле в форму **НовыйТип** для ввода значений поля **КодТипа**
5. В процедуре исправьте строку `![Тип] = "Йогурты"` на `![Тип] = Me![Тип]`.
6. Проверьте работу формы.

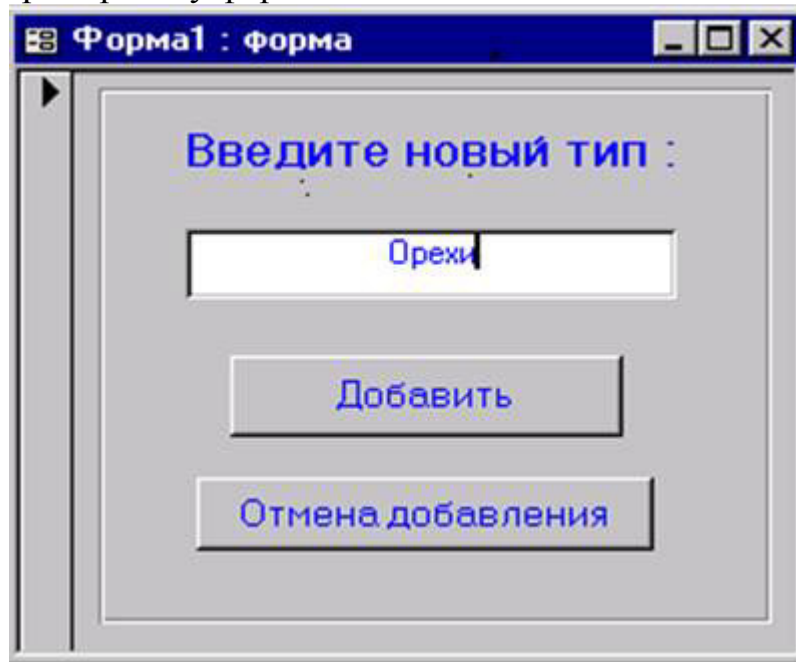


Рис.6.1 Форма для добавления нового типа.

### Создать процедуру для удаления записи из таблицы

В форме для ввода нового типа создайте кнопку **Отмена добавления** с процедурой следующего вида:

```

Private Sub Отмена_добавления_Click()
Dim dbs As Database, rst As Recordset
Set dbs = CurrentDb
Set rst = dbs.OpenRecordset("Типы", dbOpenDynaset)
    With rst
        .MoveLast
        .Delete
    End With
dbs.Close
End Sub

```

### Создание процедуры поиска заказчика из Омска

1. Создайте стандартный модуль:

```

Function Поиск()
Dim rst As Recordset
Set rst = CurrentDb.OpenRecordset("Клиенты", dbOpenDynaset)
    rst.FindFirst "Город = 'Омск'"

```

```

If rst.NoMatch Then
    MsgBox "Нет клиентов из Омска!"
Else
    MsgBox "Первый клиент Из Омска - " & rst![Название]
EndIf
EndFunction

```

## 2. Вокнеотладки введите ? Поиск().

### Создать функцию для копирования цены из поля таблицы в поле формы

1. Создайте функцию (в стандартном модуле):

```

Function Copir(tForm As Form, tPole As String)
Dim tDB As Database, tTB As Recordset
Set tDB = CurrentDb()
Set tTB = tDB.OpenRecordset("Товары", dbOpenTable)
tTB.Index = "КодТовара"
tTB.Seek "=", tPole
tForm.[Цена] = tTB![Цена]
tTB.Close
End Function

```

2. В ячейке свойства После обновления поля Код Товара формы Заказанный товар введите  
= Copir(Forms![Заказанный товар], Forms![Заказанный товар]![КодТовара]).

### Примечание.

Для передачи информации в функцию используются два параметра. Первый – название формы, в которую нужно скопировать поле. Это поле Цена в форме Заказанный товар. Второй параметр – поле формы, по которому производится поиск в таблице Товары. Во второй строке модуля объявляются переменные, соответствующие базе данных и таблице. Третья строка устанавливает переменную tTB на текущую открытую БД. В четвертой строке открывается таблица. В пятой строке задается индекс по полю Код товара. В шестой строке осуществляется поиск в таблице Товары поля с кодом товара, который выбран в форме. Седьмая строка производит копирование цены из таблицы в форму. В восьмой строке таблица закрывается.

Открытие объекта Recordset типа Table позволяет быстро найти запись в таблице при помощи метода Seek. Этот метод можно применять только к объекту Recordset типа Table, если поле таблицы, содержащее искомое значение, проиндексировано.

Еще один способ – использование функции DLookup. Процедура должна выглядеть следующим образом:

```

Private Sub КодТовара_AfterUpdate()
Dim strFilter As String
strFilter = "КодТовара=" & Me![КодТовара]
Me![Цена] = DLookup("Цена", "Товары", strFilter)
End Sub

```

### Создание процедуры подсчета числа заказов на определенную дату

Создайте функцию подсчета количества заказов, сделанных в определенный день. Данная функция осуществляет подсчет заказов, оформленных 26 марта 2000 года.

```
Function Zakaz ()
Dim tDB As Database, rs As Recordset, dc As Integer
Set tDB = CurrentDb()
Set rs = tDB.OpenRecordset("Заказы", dbOpenTable)
rs.Index = "ДатаРазмещения"
rs.Seek "=", "26.03.00"
If rs.NoMatch Then
    MsgBox "Нет заказов в этот день "
Else
    Do
        dc = dc + 1
        rs.MoveNext
        If rs.EOF Or rs![ДатаРазмещения] <> "26.03.00" Then
            Exit Do
        End If
    Loop
    MsgBox "Всего" & Str$(dc) & "заказов 26 марта."
End If
End Function
```

### Создание процедуры возврата в окно базы данных

1. В кнопочную форму, созданную ранее, в режиме конструктора при отключенном мастере добавьте кнопку **Окно базы данных**.

2. Процедура:

```
PrivateSubОкно_базы_данных_Click()
Dim dn As String
dn = "Заказы"
DoCmd.Close
DoCmd.SelectObject acTable, dn, True
EndSub
```

Данная процедура откроет окно базы данных и маркирует таблицу **Заказы**.

### Создание процедуры обработки события отсутствия в списке

1. Откройте форму **Заказы** в режиме конструктора.

2. В свойстве **Ограничиться списком** поля **КодКлиента** формы **Заказы** выберите значение **-Да**

3. В свойстве события **Отсутствие** в списке поля **КодКлиента** введите следующую процедуру:

```
Private Sub КодКлиента_NotInList(NewData As String, Response As Integer)
Dim intReturn As Integer
intReturn = MsgBox("Клиента с таким кодом в списке нет. Хотите добавить?", vbQuestion + vbYesNo)
If intReturn = vbYes Then
DoCmd.OpenForm FormName:="НовыйКлиент", DataMode:=acFormAdd, WindowMode:=acDialog
Response = acDataErrAdded
Exit Sub
```

```

End If
Response = acDataErrDisplay
End Sub

```

### Примечание.

Данная процедура выводит окно сообщения, в котором можно подтвердить, что вы хотите добавить информацию о новом клиенте.

Рис.6.2. Форма для ввода нового клиента, созданная в лабораторной работе №3

Если нажать **Да**, процедура откроет форму **Новый Клиент** в режиме диалога, используя метод DoCmd.OpenForm. Форма показана на рисунке 6.2.

Access передает два аргумента процедуре обработки события **NotInList**. Первый, **NewData**, содержит строку символов, введенную в поле со списком. Второй, **Response**, определяет реакцию Access на это событие.

После ввода новых данных нажмите кнопку **Сохранить**, чтобы они добавились в таблицу. Процедура присваивает аргументу Response константу acDataErrAdded. Access обновляет поле со списком и снова наличие введенного значения со списком. Если вы не сохранили введенные данные о новом клиенте, Access не найдет его в списке и выведет стандартное сообщение для этого события.

Если в окне сообщения была нажата кнопка **Нет**, аргументу Response присвоится константа acDataErrAdded, и Access отобразит стандартное сообщение.

4. Запустите форму **Заказы** и в поле **КодКлиента** введите код, которого нет в списке.

5. Перейдите к другому полю или нажмите **Enter** и проверьте работу процедуры.

**Создать процедуры обработки событий для отбора данных при открытии отчета**

1. Создайте простую модальную форму **Диалог** следующего содержания:

2. Для кнопок **Просмотр отчета** и **Отмена** напишите процедуры:

```
Private Sub Отмена_Click()  
    DoCmd.Close  
End Sub  
Private Sub Просмотр_отчета_Click()  
    Me.Visible = False  
End Sub
```

3. Откройте **ОтчетОКлиентах** в режиме конструктора.

4. Откройте запрос – источник записей отчета.

5. В строке **Условие отбора** поля **ДатаРазмещения** введите условие:

Between Forms![Диалог]![С какой даты] And Forms![Диалог]![По какую дату]

6. В заголовок отчета добавьте поля и надписи:

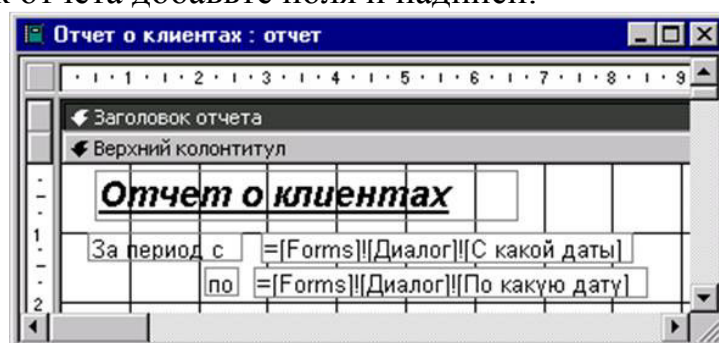


Рис.6.4. Отчет о клиентах.

7. Для открытия и закрытия отчета создайте процедуры:

```
Private Sub Report_Close()  
    Dim strDocName As String  
    strDocName = "Диалог"  
    DoCmd.Close acForm, strDocName  
End Sub  
Private Sub Report_Open()  
    Dim strDocName As String  
    strDocName = "Диалог"  
    DoCmd.OpenForm strDocName, , , , acDialog  
End Sub
```

8. Откройте отчет в режиме предварительного просмотра и проверьте правильность выбранных данных.