

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

И.А. Зикратов, В.Ю. Петров

**ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ
В УПРАВЛЕНИИ**

Учебное пособие



Санкт-Петербург

2010

Зикратов И.А., Петров В.Ю. Информационные технологии в управлении.
Учебное пособие. - СПб: СПбГУ ИТМО, 2010. -64 с.

Учебное пособие преследует цель - практическое усвоение студентами лекционного материала по курсу «Информационные технологии» и самостоятельное овладение навыками использования современных средств информационных технологий для разработки пользовательских приложений в среде Office.

Для студентов специальностей 080801 «Прикладная информатика в экономике».

Рекомендовано к печати на заседании совета Гуманитарного факультета, протокол № 8 от 22.03.2010.



В 2009 году Университет стал победителем многоэтапного конкурса, в результате которого определены 12 ведущих университетов России, которым присвоена категория «Национальный исследовательский университет». Министерством образования и науки Российской Федерации была утверждена Программа развития государственного образовательного учреждения высшего профессионального образования «Санкт-Петербургский государственный университет информационных технологий, механики и оптики» на 2009–2018 годы.

©Санкт-Петербургский государственный университет
информационных технологий, механики и оптики, 2010

© И.А.Зикратов, В.Ю. Петров, 2010

ОГЛАВЛЕНИЕ

Введение	5
1. Ввод и вывод информации. Встроенные диалоговые окна	6
1. Ввод и вывод информации. Функции MsgBox() и InputBox().....	6
1.1. Поиск ошибок в процедуре.....	6
1.2. Создание собственных процедур.....	7
2. Встроенные диалоговые окна для получения пути и имени файла.....	7
3. Вызов диалоговыми окнами приложения и работа с ними.....	9
2. Использование элементов управления, встраиваемых на рабочие листы	11
1. Использование кнопок.....	11
2. Использование выключателей, полей ввода.....	15
3. Полоса прокрутки как элемент управления.....	15
3. Создание диалоговых окон пользователей. Исследование работы элементов управления. Процедуры обработки событий	19
1. Основные положения.....	19
2. Страницы, флажки, переключатели в диалоговых окнах.....	19
3. Метки (надписи) и поля ввода.....	22
4. Список и выключатель.....	22
5. Комбинированный список.....	24
6. Полоса прокрутки и счетчик.....	26
7. Процедуры обработки событий.....	27
4. Обработка ошибок в программах VBA	28
1. Основные положения.....	28
2. Перехват ошибок. Терминология.....	29
3. Система перехвата ошибок.....	29
4. Компоненты системы перехвата ошибок.....	30
4.1. Инструкция On Error.....	30
4.2. Инструкция Resume.....	30
4.3. Инструкция Exit.....	31
4.4. Объект Err.....	31
5. Средства создания пользовательского интерфейса. Меню и панели инструментов	35
1. Основные положения.....	35
2. Изменение интерфейса на этапе разработки.....	36
3. Изменение интерфейса на этапе выполнения.....	37
3.1. Основные объекты.....	38
3.2. Создание строк меню (панелей инструментов). Объект CommandBar.....	38
3.3. Свойства объекта CommandBar, представляющего отдельную панель (строку).....	40
3.4. Методы объекта CommsandBar.....	41

3.5. Программирование элементов строк меню (панелей) CommandBar.....	45
4. Программирование контекстных меню.....	44
6. Создание приложения Excel для автоматизации расчетов	47
7. Графические элементы в OFFICE.....	50
1. Основные положения.....	50
2. Объект Shapes.....	50
2.1. Методы объекта Shapes.....	50
2.2. Свойства объекта Shapes.....	53
3. Объект Shape.....	53
3.1. Методы объекта Shape.....	53
3.2. Свойства объекта Shape.....	53
8. Построение диаграмм.....	55
9. Создание приложения для автоматизации заполнения бланка «Расчета структуры цены».....	58
Литература.....	61

ВВЕДЕНИЕ

Современный этап развития человеческого общества характеризуется переходом к всеобщей информатизации, внедрению компьютерных и информационных технологий. Зарубежный и российский опыт внедрения информационных технологий подтверждает их высокую экономическую эффективность для многих сфер применения: начиная с обеспечения выполнения рутинных операций по формированию документов и организации служебной переписки и заканчивая системным анализом, поддержкой принятия решений, построением систем дистанционного обучения, автоматизацией проектирования сложных информационных систем и др.

Динамическое развитие информационных технологий требует от специалистов различных областей деятельности знаний и умения для адаптации к процессам управления в сфере использования автоматизированных информационных технологий.

Многообразие реального мира не позволяет произвести «линейную» классификацию информационных технологий по какому-то одному классификационному признаку. Поэтому одну из информационных технологий, как правило, можно отнести к различным классам. Например, по пользовательскому интерфейсу - к диалоговым ИТ, по принципу построения – к объектно-ориентированным ИТ и т.д. В связи с этим, данное пособие не претендует на изучение всех видов и классов ИТ. В пособии преследуется цель познакомить и научить студентов применять в своей практике информационные технологии конечного пользователя, технологии электронного офиса, которые по классификационным признакам можно отнести к различным классам ИТ, но, тем не менее, без которых работа студента, инженера, руководителя и других специалистов немыслима.

Все примеры, задачи в пособии ориентированы на использование Ms.Office 2007. Успех создания пользовательских приложений зависит, в конечном итоге, от того насколько широко программисты и пользователи будут использовать платформу Office для поиска своих решений. Поэтому важным моментом является знание основ офисного программирования.

Основное внимание в пособии уделено использованию ИТ для создания автоматизированных систем и различных пользовательских приложений для табличного и текстового процессоров, имеющим более зрелые модели объектов и наиболее востребованным с точки зрения офисного программирования. Уделено место и обработке ошибок, что позволяет обойти программные сбои и сделать любую программу работоспособной. Особенно это важно в тех случаях, когда пользователь вводит неверные данные или ошибается в некоторых действиях.

Решения отдельных вопросов, которые могут быть полезными при изучении данного пособия рассмотрены в работе [1].

1. ВВОД И ВЫВОД ИНФОРМАЦИИ. ВСТРОЕННЫЕ ДИАЛОГОВЫЕ ОКНА

1. ВВОД И ВЫВОД ИНФОРМАЦИИ. ФУНКЦИИ *MsgBox()* и *InPutBox()*

Используя всего две функции - *MsgBox()* и *InPutBox()*, можно создать простой пользовательский интерфейс для различных приложений. С их помощью можно указывать на ошибки пользователя и исправлять их, вводить информацию и др. Удобнее всего работать с этими функциями в интерактивном режиме.

Перед выполнением первого задания следует вспомнить синтаксис этих функций и допущения, накладываемые на их параметры. Для этого следует либо обратиться к лекциям либо к соответствующей литературе.

1.1 Поиск ошибок в процедуре

Задание № 1.

Наберите в окне модуля исходный текст процедуры *ZapPrem()*. Процедура содержит множество ошибок. Разберитесь в программе, составьте алгоритм ее работы и исправьте ошибки. Около каждого оператора введите текст, поясняющий его работу. Запустите процедуру и убедитесь в том, что:

- ✓ она работает;
- ✓ сумму премии невозможно ввести символами вместо цифр, и нельзя ошибаться при вводе десятичного знака;
- ✓ невозможно оставить все без изменения, т.е. не вводить сумму премии вообще.

Измените программу так, чтобы при вводе пустой строки процедура не заканчивалась, а требовала у пользователя повторного ввода данных.

Sub ZapPrem()

```
'=====
Dim theDefault As String, thePrompt As String, theTitle As String
Dim theReply As String, OKFlag As Boolean, theB As Single

thePrompt = " Hello, гражданин ! " & _
    " Введите пожалуйста сумму премии за квартал с учетом копеек ."
theDefault = "123456789"
theTitle = "Ввод суммы премии за квартал"

Do
    theReply = InputBox(thePrompt, theTitle, theDefault)

    If (theReply) = Then Exit Sub
    theReply = Trim(theReply)
    theB = Val(theReply)
```

```

IfNot IsNumeric(theReply) Then
  MsgBox " Повторите ввод еще раз, пожалуйста." & _
    " Но введите, все таки, число, гражданин !", , theTitle
  OKFlag = False
  ElseIf theReply = theDefault Then
    MsgBox " Вы ничего не ввели. Повторите попытку, сделайте " & _
      " хоть что-нибудь сами.", , " Граждане ждут и волнуются !"
    OKFlag = False
  Else
    OKFlag = False
  End If
Loop Until OKFlag

```

```

Sheets("Лист1").Range("A1").Value = "Премия=" & theB
End Sub

```

1.2 Создание собственных процедур.

Задание № 2. Создайте процедуру, в которой бы использовались встроенные окна ввода/вывода и был бы организован диалог с пользователем. В ней необходимо проверить вводимые данные на подлинность, т.е. отфильтровать истинные данные (например, возраст не может быть более 100 лет).

Не забывайте сохранять свой материал на дискету.

Примерный алгоритм.

1. Спросить имя пользователя.
2. Если оно введено и соответствует разрешенному имени, то поздороваться с пользователем, введя соответствующий текст в окно диалога. После третьего неверного ответа выйти из процедуры.
3. Спросить дату рождения пользователя, и если она является недопустимой, указать ему на это и попросить ввести дату снова.
4. Посчитать и вывести количество дней, прожитых пользователем.

2. ВСТРОЕННЫЕ ДИАЛОГОВЫЕ ОКНА ДЛЯ ПОЛУЧЕНИЯ ПУТИ И ИМЕНИ ФАЙЛА

Таких окон в Excel два: *GetOpenFilename* и *GetSaveAsFilename*. Команды, их создающие, являются методами объекта Application, а не операторами VB. Файлов они не открывают и не сохраняют. Их задача – вернуть имя файла. Команды имеют следующий синтаксис:

```

theFileName=Application.GetSaveAsFilename(
  InitialFilename:=начальное имя файла;
  FileFilter:=файловый фильтр;
  FilterIndex:=индекс фильтра; Title:=заголовок)

```

```

theFileName = Application.GetOpenFilename(FileFilter:=файловый фильтр;
  FilterIndex:=индекс фильтра; Title:=заголовок)

```

Задание № 3. Наберите текст программы, представленный ниже. Запустите процедуру. Объясните и подпишите комментарии к каждому оператору. Проверьте:

- какие действия производят и что возвращают функции *GetSaveAsFilename* и *GetOpenFilename* ?

- как действия функций зависят от значений их фактических параметров и активизации соответствующих кнопок в рабочих окнах, вызванных этими функциями ?

Sub qaz_00

```
Dim theFileName As String; Answer As String; theDot As Integer
Dim theFilter As String; theFilter_1 As String; theFilter_2 As String
Dim theFilter_3 As String
```

```
theFileName = "qaz"
```

```
theFilter_1 = "Все файлы (*.*), *.* "
```

```
theFilter_2 = "Рабочие книги Excel (*.xls), *.xls"
```

```
theFilter_3 = "Все файлы (*.*), *.* , Рабочие книги Excel (*.xls), *.xls"
```

```
theFilter = theFilter_1 & ", " & theFilter_2
```

```
Answer = Application.GetSaveAsFilename(InitialFilename:=theFileName; _
    FileFilter:=theFilter_1; FilterIndex:=1; Title:="Проба пера")
```

```
Answer = Application.GetSaveAsFilename(theFileName; theFilter_1; 1; _
    "Другая проба пера")
```

```
Answer = Application.GetSaveAsFilename(theFileName; theFilter_2; 1; _
    "3333333333333333")
```

```
Answer = Application.GetSaveAsFilename(theFileName; theFilter_3; 1; _
    "4444444444444444")
```

```
Answer = Application.GetSaveAsFilename(theFileName; theFilter; _
    FilterIndex:=2; Title:="5555555")
```

```
Answer = Application.GetSaveAsFilename(theFileName; theFilter)
```

```
Answer = Application.GetOpenFilename(FileFilter:="Все файлы (*.*), _
    *.* , Рабочие книги Excel (*.xls), *.xls"; _
    FilterIndex:=2; Title:="Это уже другое окно!")
```

```
Answer = Application.GetOpenFilename(FileFilter:=theFilter; _
    FilterIndex:=2; Title:="Это уже другое окно!")
```

End Sub

Задание № 4. Наберите текст программы, представленный ниже, и запустите программу. Объясните, что делает эта программа. Можно ли использовать в операторе *IF* при сравнении значение *False* вместо "*Ложь*", а если нельзя - то почему? Являются ли кавычки при задании соответствующих значений необходимым фактором? Почему и в каких случаях?

Sub Test()

```

Dim theFileName As String, Answer As String
Dim theFilter As String, theDot As Integer

theFileName = "qaz": theFilter = "Все файлы (*.*), *.*"

Answer = Application.GetSaveAsFilename(theFileName, theFilter, 1)
If (Answer = "Ложь") Then Exit Sub 'False
theFileName = Trim(Answer)
theDot = InStr(theFileName, ".")
theFileName = theFileName & "." & "xls"
Else
theFileName = Left(theFileName, theDot - 1) & "xls"
End If

```

End Sub

3. ВЫЗОВ ДИАЛОГОВЫХ ОКОН ПРИЛОЖЕНИЯ И РАБОТА С НИМИ

В любом приложении Office существует огромное количество рабочих окон, которые можно вызвать, используя и команды меню и инструкции VBA.

Задание № 5. Наберите текст приведенной ниже программы. Объясните работу всех операторов и напишите комментарии в каждой строке процедуры. Исследуйте процедуру в пошаговом режиме, используя окно отладки. Если какие-то операторы не работают, то исправьте их. Добавьте в процедуру несколько своих подобных операторов.

Sub qq_1()

```

Workbooks.Add
Workbooks.Open ("C:/VUP/Teh/Лит-98/Кн2")
Worksheets("Лист_xxx").Select
ActiveWorkbook.Worksheets.Add
ActiveSheet.Name = "Январь"
Worksheets(5).Name = "Февраль"
ActiveWorkbook.SaveAs "Расплата"
Workbooks("Расплата").Close
Workbooks.Close

```

End Sub

Задание № 6. Наберите текст процедуры, приведенной ниже. Заставьте ее работать. Установите, используя эту процедуру, для области ячеек **A3:C7** денежный формат. Убедитесь, что это так !

Sub qq()

```
Application.Dialogs(xlDialogFormatNumber).Show
```

End Sub

Задание № 7. Используя процедуру приведенную ниже, откройте какой-нибудь файл и запомните его с другим именем. Объясните, куда и как попадают Ваши файлы при их записи и чтении.

Sub qq()

```
Application.Dialogs(xlDialogOpen)
```

```
Application.Dialogs(xlDialogSaveAs).Show
```

```
Workbooks.Close
```

End Sub**Задание № 8.**

Создайте максимально возможное для Вас число диалоговых окон приложения. Покажите как их можно использовать.

Для вызова окон используйте набор *Dialogs* объекта *Application*, с аргументами в виде *числа-индекса*.

Эти *индексы* соответствуют константам Excel, которые можно выбрать, используя просмотр объектов. Посмотреть объекты можно после выполнения команды меню *View/Object Browser* (выбирая объекты класса *XlBuiltInDialog* библиотеки *Excel*).

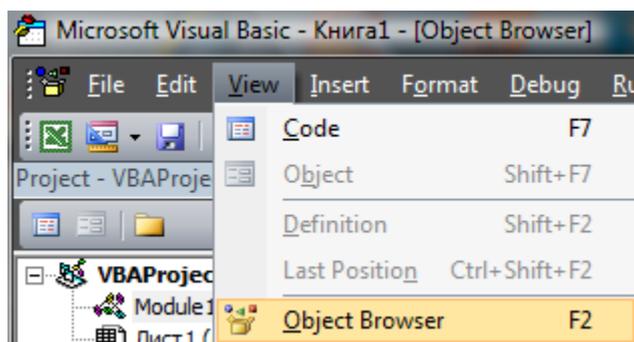


Рисунок 1.1 – Вызов команды меню *Object Browser*

Все результаты попробуйте объяснить. Сколько у Вас получилось окон?

2 ИСПОЛЬЗОВАНИЕ ЭЛЕМЕНТОВ УПРАВЛЕНИЯ, ВСТРАИВАЕМЫХ НА РАБОЧИЕ ЛИСТЫ

1. ИСПОЛЬЗОВАНИЕ КНОПОК

Во всех случаях жизни от кнопки можно ждать только одного из действий: включить или выключить что-то, запустить или остановить какой-нибудь процесс.

Задание № 1. Исследуйте процесс размещения, форматирования и использования кнопок на рабочем листе. В принципе, точно также как кнопки на лист вставляют и используют и другие элементы управления. Для конкретности, с помощью кнопок оцените время между началом процесса и его отдельными фазами. Для этого выполните действия, приведенные ниже.

1. Дайте рабочему листу книги любое название, например «*Test*».
2. Создайте на листе таблицу для расчета произвольных интервалов времени между началом и концом отдельных событий, а также между временем начала текущего события и началом работы (упражнения). В ее отдельные ячейки вставьте формулы и поясняющий текст так, как на рисунке 2.1.

	А	В	С	Д	Е
1					
2			Время начала/конца	Время между соседними упражнениями	Время от начала теста до конца текущего упражнения
3			Начало теста (упр-я №1)	16:20:40	
4			Конец упражнения №1	16:20:40	=C4-C3
5			Конец упражнения №2	16:20:40	=C5-C4
6			Конец упражнения №3	16:20:40	=C6-C5
7			Конец упражнения №4	16:20:40	=C7-C6
8					
9	Начать тест (упражнение №1)	Выполнено упражнение №1	Выполнено упражнение №2	Выполнено упражнение №3	Выполнено упражнение №4
10					
11					
12					

Рисунок 2.1 - Вид рабочего листа

3. Поставьте (перетащите мышью) на этот лист пять кнопок, используя панель инструментов «*Элементы управления*».

Панель «*Элементы управления*» появится после щелчка по кнопке «*Вставить*», расположенной в окне «*Элементы управления*» на вкладке ленты «*Разработчик*». Ее вид представлен на рисунке 2.2. Все элементы с панели управления доступны для использования на рабочем листе.

4. Группа элементов «*ActiveX*» доступна при включенном «*Режиме конструктора*» (см. рисунок 2.2).

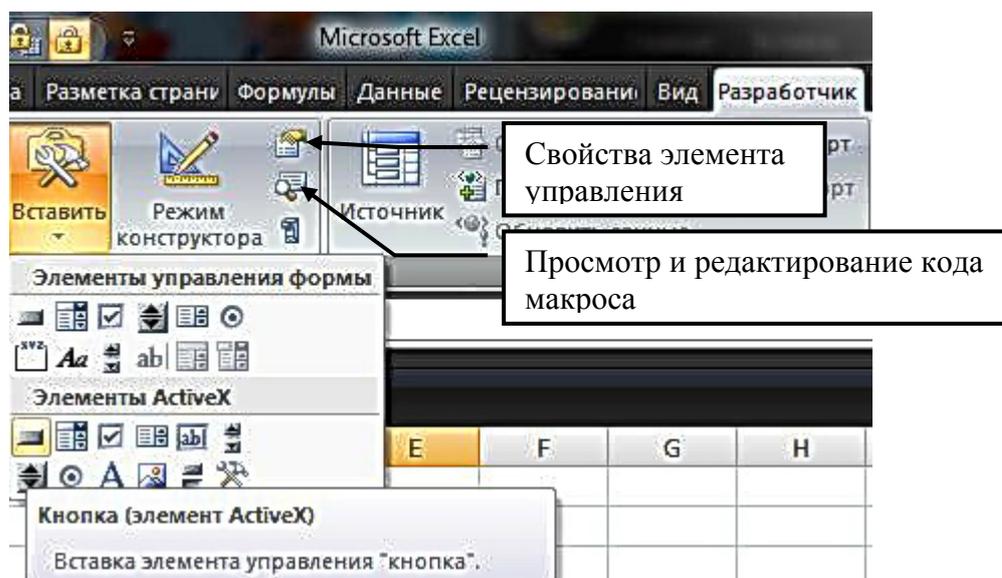


Рисунок 2.2 – Доступные элементы управления

Чтобы отформатировать кнопку (либо любой другой элемент управления), размещенную на листе (изменить ее размеры, цвет, определить доступность, передвинуть в другое место и т.д.), или написать для этого элемента управления программный код-макрос, должен быть включен **режим конструктора**. После отключения *режима конструктора* установленные элементы переходят в *рабочий режим*. Форматирование таких элементов становится невозможным.

Управляющие кнопки для этих действий представлены на рисунке 2.2. Кроме того, действие этих кнопок задублировано соответствующими командами контекстного меню, появляющегося при щелчке по элементу правой кнопкой мыши.

Вид окна для просмотра кода, в случае использования элемента *ActiveX* «Кнопка», представлен на рисунке 2.3.

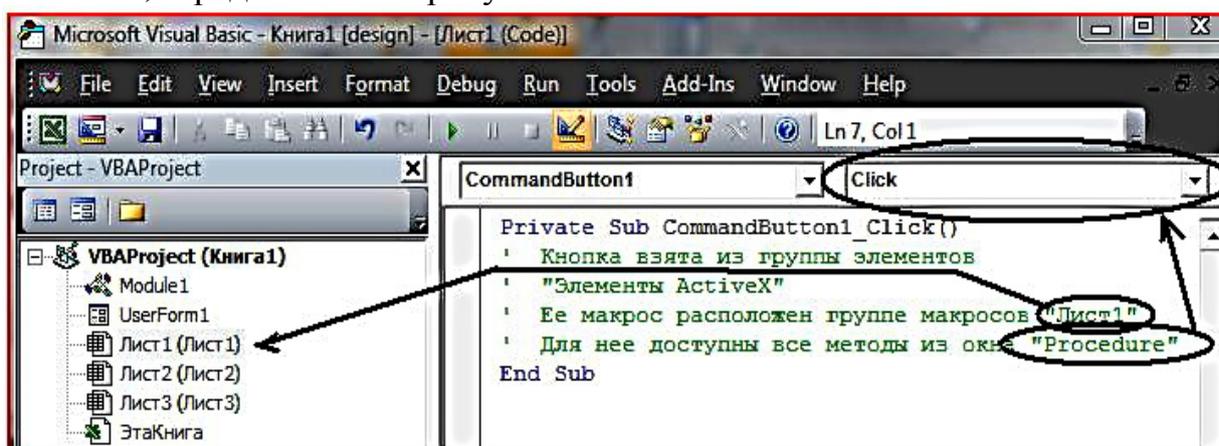
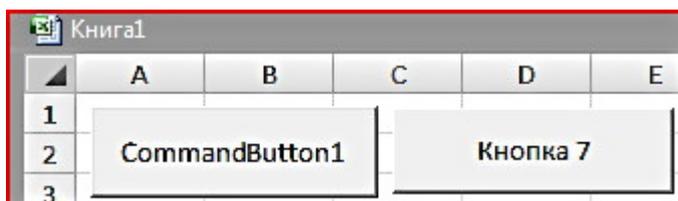


Рисунок 2.3 – Окно редактирования программного кода

Имя у кнопки, установленной на рабочем листе, будет «*CommandButton1*». Потом его можно будет изменить.



5. При установке кнопки из группы «Элементы управления Формы» ей будет дано имя «Кнопка 1» и т.д.

Измениться расположение макроса в проекте (см. рисунок 2.4).

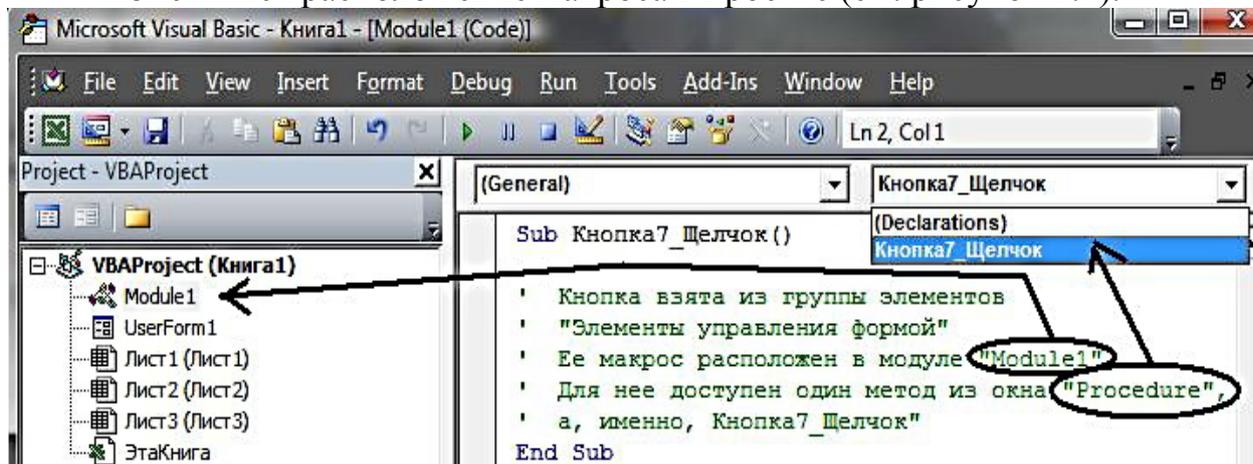


Рисунок 2.4 – Окно редактирования программного кода

Все перемещения и изменения параметров у кнопок доступны без изменений режима *Конструктора*. Достаточно щелкнуть по ним правой кнопкой мыши и выбрать команду.

Замечание. В дальнейшем в пособии использованы элементы управления из категории «Элементы ActiveX»

6. Дайте кнопкам имена и отформатируйте.

Активизировав установленную на листе кнопку, вызовите *Окно свойств* (*Properties* – см. рисунок 2.5) для каждой из них, щелкнув мышью по кнопке "Свойства элементов управления" на панели инструментов "Элементы управления", либо другим способом.



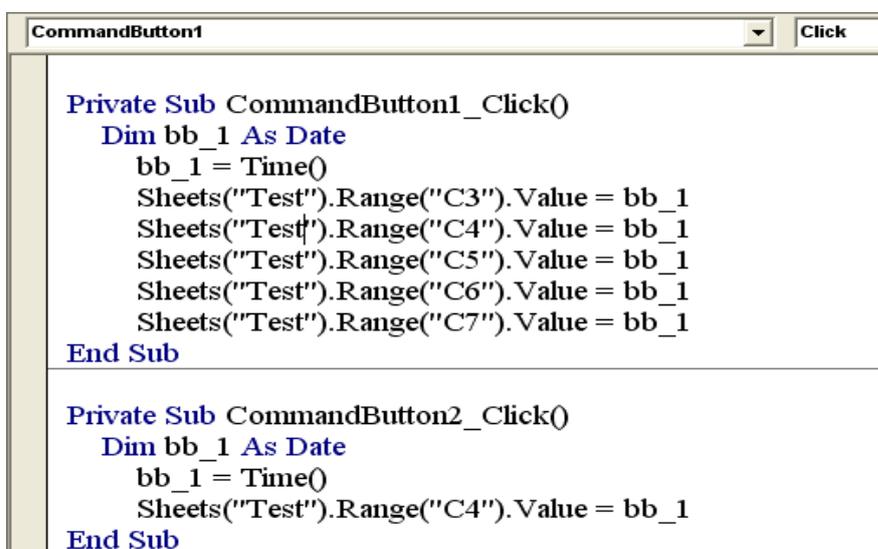
Рисунок 2.5 – Установка свойств для кнопки

Для ввода текста, появляющегося на кнопке, используйте свойство *Caption*. В этом же окне задайте цвет кнопки, цвет надписи на ней, откорректируйте другие свойства.

7. Определите те действия, которые будут происходить после щелчка по кнопкам, т.е. свяжите эти события с макросами. В нашем случае макросы - это процедуры (методы), которые должны считывать системное время и помещать его в необходимые ячейки на листе.

Для вызова окна программного кода (см. рисунок 2.3), в котором пишут текст программ, необходимо дважды щелкнуть по кнопке, установленной пользователем на листе, или щелкнуть по кнопке "Просмотр и редактирование кода", расположенной на панели инструментов "Элементы управления" (режим конструктора должен быть включен).

Примерный текст процедур, которые должны быть выполнены при активизации двух первых кнопок, приведен ниже. Остальное допишите самостоятельно.



```

Private Sub CommandButton1_Click()
    Dim bb_1 As Date
    bb_1 = Time()
    Sheets("Test").Range("C3").Value = bb_1
    Sheets("Test").Range("C4").Value = bb_1
    Sheets("Test").Range("C5").Value = bb_1
    Sheets("Test").Range("C6").Value = bb_1
    Sheets("Test").Range("C7").Value = bb_1
End Sub

Private Sub CommandButton2_Click()
    Dim bb_1 As Date
    bb_1 = Time()
    Sheets("Test").Range("C4").Value = bb_1
End Sub

```

Рисунок 2.6 - Примерный вариант программного кода

8. Сохраните книгу.

9. Убедитесь в работоспособности программного продукта.

10. Измените модуль с программой.

10.1. Вставьте оператор, требующий обязательного описания типов переменных.

10.2. Опишите переменную bb_1 так, чтобы избежать этого в каждой процедуре.

10.3. Измените программу так, чтобы при заполнении таблиц в ячейках не появлялись служебные символы ##### (если такие у вас получились).

10.4 Попробуйте вставить кнопки из группы элементов управления «Элементы управления Формы». Используя эти кнопки, добейтесь того же результата.

11. Дополнительное сложное задание. Попробуйте.

Вставьте еще одну кнопку на лист. Напишите для нее такой макрос, чтобы он удалял одну кнопку из ранее пяти поставленных.

Здесь нужно учесть, что при установке кнопок был включен режим конструктора, а при их использовании он выключен.

2. ИСПОЛЬЗОВАНИЕ ВЫКЛЮЧАТЕЛЕЙ, ПОЛЕЙ ВВОДА

Задание № 2.

1. Откройте книгу, созданную при выполнении задания №1.
2. Вставьте на рабочий лист еще два элемента управления: *Выключатель* и *Текстовое поле* так, как на рисунке 2.7.



Рисунок 2.7 - Вид рабочего листа

3. Отформатируйте установленные элементы управления.
4. Событие *Выключателя* **Change** свяжите с макросом, который бы блокировал *Кнопку 3* и *Кнопку 5*, а также выводил в *текстовое поле* информацию о состоянии *Выключателя*: «Включено» или «Выключено».
5. Добейтесь работоспособности разработанных программ.

3. ПОЛОСА ПРОКРУТКИ - КАК ЭЛЕМЕНТ УПРАВЛЕНИЯ

А). Полоса прокрутки возвращает число (*свойство Value*), которое определяется положением ее движка и двумя числами соответствующими его начальному и конечному положению.

Эти данные можно задать программно, а можно и при форматировании данного объекта - конкретной полосы прокрутки. Для ее форматирования используют *окно свойств* точно также как и для кнопки (см. п. 1). Основные свойства, определяющие ее поведение, ограничения на возвращаемые значения, указаны на рисунке 2.8.

Практически все объекты управления могут быть отформатированы таким образом до момента их использования в приложениях

Б). Полосу прокрутки можно помещать и на рабочих листах и в окнах диалога (формах). В данной работе будем размещать ее на рабочих листах.

Обычно, положение движка полосы прокрутки определяет число, которое указывает на какой-то элемент списка (5-й, 10-й и т. д.). Всегда можно сделать еще и так, чтобы это положение движка не только определяло, но и указывало на конкретную строку или столбец заданной таблицы, т.е. движок располагался напротив искомой строки или столбца.

Property	Value	Description
(Name)	ScrollBar1	Имя элемента - "Полосы прокрутки"
Delay	50	Задержка в миллисекундах, после которой возникает событие Change, при щелчке по кнопкам прокрутки.
Height	317,25	Определяет величину изменения значения свойства Value, при щелчке по поверхности полосы прокрутки
LinkedCell	A1	Ячейка связи, отражающая значения свойства Value
Max	12	Числа соответствующие крайним положениям бегунка.
Min	1	Числа соответствующие крайним положениям бегунка.
ProportionalThumb	False	Свойство определяющее ширину бегунка. При значении свойства - False, ширина наименьшая, в наибольшей степени соответствующая стандартной ширине строки.
SmallChange	1	Определяет величину изменения значения свойства Value, при щелчке по кнопкам полосы прокрутки

Рисунок 2.8 - Свойства полосы прокрутки

	A	B	C	D	E	F
1	11					
2		Иванов	Петров	Сидоров		
3	Январь	10	30	1		<p>При форматировании полосы прокрутки установите следующие значения у ее свойств:</p> <p><i>SmallChange=1</i></p> <p><i>Max=12</i></p> <p><i>Min=1</i></p> <p><i>ProportionalThumb=False</i></p> <p>Такие значения свойства позволяют, перемещая движок полосы прокрутки на 1 шаг, переместится на 1 строку листа таблицы.</p>
4	Февраль	20	40	2		
5	Март	30	50	3		
6	Апрель	40	60	4		
7	Май	50	70	5		
8	Июнь	60	80	6		
9	Июль	70	90	7		
10	Август	80	100	8		
11	Сентябрь	90	110	9		
12	Октябрь	100	120	10		
13	Ноябрь	110	130	11		
14	Декабрь	120	140	12		
15						

Рисунок 2.9 - Рабочий лист с полосой прокрутки

Например, на рисунке 2.9 движок расположен напротив одиннадцатой строки таблицы. При этом может быть активизирована одна из ячеек, находящаяся напротив движка, т.е. в одиннадцатой строке, например, A13, B13, C13, D13.

В). Для этого необходимо:

во-первых, расположить полосу прокрутки рядом с таблицей и установить у нее размер поля для движения движка таким, чтобы в него попадали все строки (столбцы) таблицы (см. рисунок 2.9, строки с 3-й по 14-ю);

во-вторых, используя форматирование полосы прокрутки, установить соответствие между начальными и конечными крайними положениями движка в этой полосе прокрутки и соответствующими им числами (в данном случае - это 1 и 12), а также числом, определяющим величину шага между соседними положениями движка (в данном случае шаг должен быть равен 1);

в-третьих, связать полосу с макросом, который должен быть написан заранее, так, чтобы при каждой инициализации полосы прокрутки (щелчка по ней мышью) этот макрос выполнялся.

3.1 Задания

Задание № 3.

Создайте таблицу для отображения заработка трех сотрудников по месяцам, приведенную на рисунке 2.9. Используя полосу прокрутки, добейтесь того, чтобы при нахождении ее движка напротив ячейки с соответствующим названием месяца, эта ячейка была бы активизирована.

Для этого сделайте следующее.

1. Полосу прокрутки расположите так, чтобы поле для движения движка начиналось с верхней границы строки "*Январь*" и заканчивалось на нижней границе строки "*Декабрь*". В этом случае, с учетом выполнения условий форматирования, представленных на рисунке, каждый шаг движка в полосе прокрутки будет перемещать движок на один месяц (на одну строку) вверх или вниз.

2. Отформатируйте полосу прокрутки в соответствии указаниями, представленными на рисунке 2.9.

3. Для того чтобы выполнить заданные требования нужно написать макрос. Вызовите окно модуля и в нем для события ***Change*** (изменение) объекта прокрутка напишите процедуру, текст которой представлен ниже на рисунке 2.10.

4. Проверьте работоспособность созданного приложения.

5. Совершенно очевидно, что предложенный код, хоть и предельно понятен, но страдает безумными излишествами. Измените текст процедуры так, чтобы он занимал не более 3-х строк. Для этого целесообразно вспомнить, что у объекта *Range* есть замечательное свойство *Cells*.

6. Проверьте работоспособность измененного приложения.

```

ScrollBar1 Change
Private Sub ScrollBar1_Change()
    Dim N_Mes As Integer
        ' N_Mes - вспомогательная переменная, позволяющая избежать
        ' излишне длинной записи при проверке условных операторов.
    Sheets(2).Range("A1").Value = ScrollBar1.Value
        ' Размещаем на втором листе в ячейку A1 значение св-ва Value прокрутки
        ' это св-во должно соответствовать номеру месяца от 1 до 12
    N_Mes = Sheets(2).Range("A1").Value
        ' Присваиваем переменной N_Mes значение, равное числу из ячейки "A1",
        ' с рабочего листа "Лист2 "(т.е. номер месяца)
    Sheets("Лист2").Select      ' Явно указываем на лист "Лист! "
    If N_Mes = 1 Then          ' В том случае, если это 1-й месяц
        Range("A3").Select      ' Активируем ячейку "A3".
        ElseIf N_Mes = 2 Then Range("A4").Select:
        ElseIf N_Mes = 3 Then Range("A5").Select
        ElseIf N_Mes = 4 Then Range("A6").Select:
        ElseIf N_Mes = 5 Then Range("A7").Select
        ElseIf N_Mes = 6 Then Range("A8").Select:
        ElseIf N_Mes = 7 Then Range("A9").Select
        ElseIf N_Mes = 8 Then Range("A10").Select:
        ElseIf N_Mes = 9 Then Range("A11").Select
        ElseIf N_Mes = 10 Then Range("A12").Select:
        ElseIf N_Mes = 11 Then Range("A13").Select
        ElseIf N_Mes = 12 Then Range("A14").Select
    End If
End Sub

```

Рисунок 2.10 - Примерный программный код

Задание № 4

1. Таблицу на рисунке 2.9 дополните еще одним столбцом (столбцом *E* с аналогичными данными для Смирнова).

2. На этот же лист добавьте еще одну полосу прокрутки - на этот раз, *Горизонтальную*, для того чтобы можно было выбрать не только строку, но и столбец (*B,C,D,E*).

3. Напишите процедуры для обработки соответствующих событий таким образом, чтобы можно было активизировать любую ячейку таблицы (*B3:E14*), пользуясь горизонтальной и вертикальной полосами прокрутки.

При написании макроса используйте метод *Cells* объекта *Range*.

4. Текст процедур не должен занимать место более 3-х строк.

5. Проверьте работоспособность программного продукта.

6. !! *Добейтесь такой работы программ, чтобы одна ячейка таблицы (B3:E14) при любых положениях движков прокруток всегда была активна.* Как правило, этого сначала не получится. При смене положения движка у полос прокруток активизация ячеек будет проходить через раз. Один раз какая-то ячейка будет выделена, а при повторном изменении положения движка выделенных ячеек не будет. И т.д.

3. СОЗДАНИЕ ДИАЛоговых ОКОН ПОльзовАТЕЛЕЙ. ИССЛЕДОВАНИЕ РАБОТЫ ЭЛЕМЕНТОВ УПРАВЛЕНИЯ. ПРОЦЕДУРЫ ОБРАБОТКИ СОБЫТИЙ

1. ОСНОВНЫЕ ПОЛОЖЕНИЯ

В отличие от предыдущей работы элементы управления будут вставлены на экранные формы, а не рабочие листы. Для того, чтобы это сделать необходимо перейти в окно редактора VBA, выполнив команду меню *Разработчик/Код/Visual Basic*. Тот же результата можно добиться при использовании комбинации горячих клавиш $\langle Alt+F11 \rangle$.

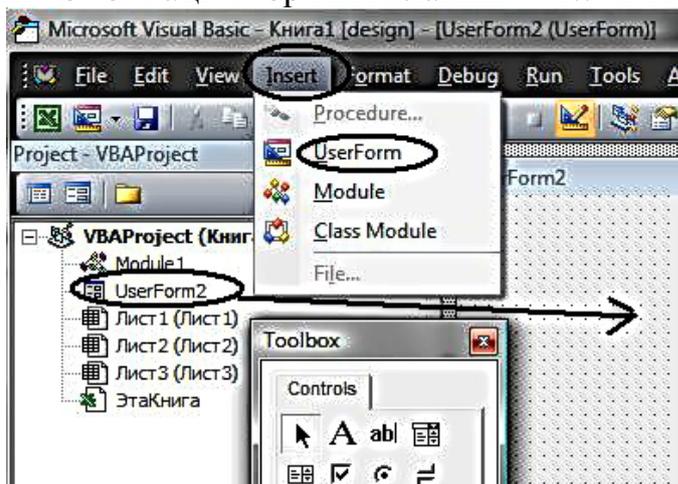


Рисунок 3.1 - Окно редактора VBA

После этого командой меню редактора VBA *Insert/UserForm* следует перейти в режим редактирования экранной формы (рисунок 3.1). В окне справа появится заготовка формы для размещения на ней элементов управления, а в окне проекта (*Project-VBAProject*) – значок, отражающий факт появления в проекте окна формы *UserForm2*. Одновременно с этим появится окно *“Toolbox”* с набором элементов управления,

которые можно перетащить на заготовку экранной формы.

Выполняя задания, вспомните о том, какие свойства имеют различные элементы с панели инструментов *«Элементы управления» (Toolbox)*, что они собой представляют, и как с ними обращаться.

В этой работе не указано с какими событиями и как следует связывать те или иные макросы. Эти вопросы достаточно полно рассмотрены выше, поэтому при необходимости решите их самостоятельно.

Номера и имена листов, флажков и переключателей, которые Вы будете создавать, могут быть любыми и не совпадать с теми, которые указаны в тексте пособия. Словосочетания: *диалоговое окно, экранная форма, пользовательская форма* будем считать синонимами.

2. СТРАНИЦЫ, ФЛАЖКИ И ПЕРЕКЛЮЧАТЕЛИ В ДИАЛОГОВЫХ ОКНАХ

✓ Страницы

Задание № 1. Создайте новую пользовательскую форму (диалоговое окно). На эту форму перетащите с панели элементов *“Элементы управления”* *“Набор страниц” (MultiPage)*. Используя свойства элементов, дайте названия окну диалога, страницам, так, как это сделано на рисунке 3.2. При

создании (добавлении) новых страниц элемента *MultiPage* используйте его контекстное меню.

✓ Флажки

Задание №2. Исследуйте работу *флажков (CheckBox)* и *переключателей (OptionButton)*, для этого:

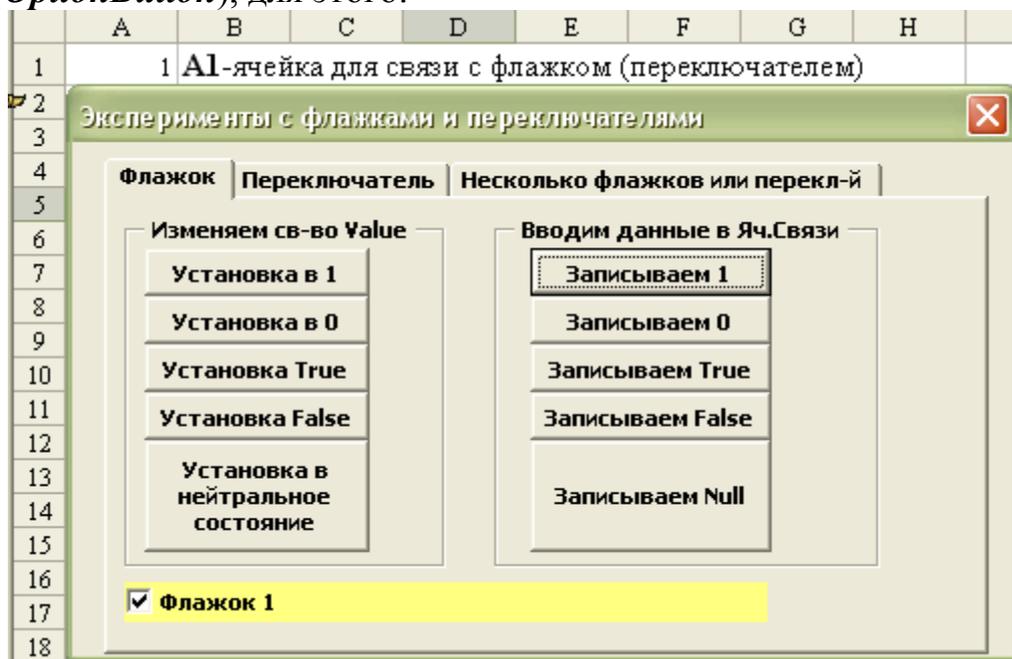


Рисунок 3.2 - Вид страницы «Флажок» на экранной форме

1. Разместите в окне созданного диалога на странице "*Флажок*" элементы управления, изображенные на рисунке 3.2.

2. Используя свойства элемента, свяжите *Флажок 1* с ячейкой *A1* рабочего листа.

Установите свойство *TripleState = True*. При этом у флажка появится 3-е состояние -нейтральное (*Null*). По умолчанию у флажка 2 состояния, а свойство *TripleState = False*.

3. Поэкспериментируйте.

3.1. Активизируйте групповое окно "*Изменяем свойство Value*".

3.2. Создайте 5 отдельных процедур, устанавливающих свойство *Value* у флажка в *1, 0, True, False* и *Null*. Присвойте (назначьте) созданные макросы соответствующим кнопкам этого группового окна.

3.3. Расположив предварительно рядом два окна - одно с диалогом, другое с листом "*Лист1*", активизируйте окно диалога (щелкните по нему мышкой, а потом нажмите клавишу *F5* на клавиатуре). Добейтесь поднятия и опускания флажка щелкая по соответствующим кнопкам на форме, оценивая при этом содержимое ячейки связи *A1*.

3.4. Активизируйте групповое окно "*Вводим данные в Яч.Связи*".

Проделайте то же самое, что и п.п.3.2 - 3.3, создав предварительно процедуры, записывающие непосредственно в ячейку связи *1, 0, True, False* и *Null*.

✓ Переключатели

Задание № 3. На странице "*Переключатель*" элемента *MultiPage*, создайте интерфейс подобный представленному на рисунке 3.2, но для переключателя.

Произведите исследования одного переключателя аналогичные тем, которые были сделаны для элемента управления «*Флажок 1*», расположенного на странице "*Флажок*" (в п.п. 2-3). При этом учтите, что у переключателя только два состояния.

✓ Несколько флажков или несколько переключателей

Задание № 4. Несколько флажков или несколько переключателей могут быть объединены групповыми окнами (*Frame*).

На странице "*Несколько флажков или переключ-й*" расположите элементы так, как указано на рисунке 3.3. Примечания разместите на полях ввода (*TextVox*), предварительно задав для них режим ввода многостраничного текста.

Напишите 4 макроса:

✓ первый, для записи конкретного имени "*Иван*" в ячейку *A2* и свяжите его с флажком "*Флажок 2*";

✓ второй, для записи конкретного отчества "*Иванович*" в ячейку *B2* и свяжите его с флажком "*Флажок 3*";



Рисунок 3.3 - Вид страницы «Несколько флажков или переключ-й» на экранной форме

✓ третий, для записи конкретной имени "Сидоров" в ячейку **C2** и свяжите его с флажком "Флажок 4";

✓ четвертый, для стирания информации из ячеек **A2, B2, C2** и свяжите его с переключателем "Перекл.2".

Запустите диалог. Исследуйте работу флажков и переключателей, расположенных в этих групповых окнах.

Сделайте письменные выводы.

3. МЕТКИ (НАДПИСИ) и ПОЛЯ ВВОДА

Задание № 5. Исследуйте процесс размещения и работы с *метками (Label)* и *полями ввода (TextBox)*.

1. Создайте рабочую форму "Метки и поля ввода" так, как показано на рисунке 3.4.



Рисунок 3.4 - Вид экранной формы «Метки и поля ввода»

2. Присвойте имена меткам группового поля "*Результат ввода*": *MET1, MET2, MET3*, а рабочим полям ввода в окне "*Ввод*" соответственно: *Поле_Фам, Поле_Им, Поле_Отч*.

3. Напишите макрос, который бы изменял текст меток группового поля "*Результат ввода*", на текст, приведенный на рисунке, и присвойте его кнопке "*Исходное состояние*"

4. Напишите макрос, который бы изменял текст меток группового поля "*Результат ввода*" на текст, набираемый в соответствующих полях ввода группового окна "*Ввод*", и присвойте его кнопке "*Ввести новую информацию*".

5. Запустите диалог. Отладьте макросы. Сделайте выводы.

4. СПИСОК и ВЫКЛЮЧАТЕЛЬ

Задание № 6. Исследуйте свойства и методы элемента *список (ListBoxes)* и *выключателя (Togglebutton)*.

1. Рассмотрите рисунок 3.5. Он содержит список, 3 групповых окна, и расположенные отдельно 2 кнопки и выключатель. Создайте такую же свою экранную форму.

Рисунок 3.5 - Экранная форма для исследования элементов управления «Список» и «Выключатель»

2. Элемент **Выключатель** по своим свойствам и поведению практически не отличается от флажка. После его установки напишите макрос, который бы переключал режим множественного выбора записей в списке с обычного на множественный, используя свойство *MultiSelect*.

3. Для заполнения пустого списка используйте свойство *List*. Напишите макрос, который бы производил заполнение списка пятью фамилиями и свяжите его с кнопкой «**Заполнить список (Инициализация)**».

4. Создайте процедуру, которая при подсвечивании элемента списка, например - "Сидоров", определяла бы и записывала в текстовые поля группового поля "**Информация об активном элементе**" его номер по списку и имя, а также общее количество элементов – фамилий. Выберите событие, которое нужно связать с этим макросом.

5. Для группового поля «**Ввод нового элемента**» создайте процедуру для записи фамилии из текстового поля "**Поле_Имени**" в список. Сделайте так, чтобы вводимая в список фамилия расположилась после той, которая имеет порядковый номер в списке, введенный Вами в поле ввода "**Поле_номера**". Присвойте этот макрос кнопке "**Ввести в список**".

В создаваемой процедуре сделайте защиту от попытки ввода неправильного номера в поле ввода «**Поле_Номера**». Например, в списке всего 5 фамилий, а Вы ввели номер равный 10 и т.д.

6. Создайте процедуру для удаления нескольких записей из списка и присвойте ее кнопке "**Удалить из списка**". Запись в списке, начиная с которой будут удаляться элементы списка, и количество этих записей должны

вводиться в поля ввода "*Начальн_номер*" и "*Колич_элементов*", соответственно.

Для удаления нескольких записей вам понадобится цикл *For... Next*. При этом учтите, что первый элемент списка имеет номер =0, а при удалении каждого элемента список сдвигается вверх, заполняя пустое пространство. Т.е. место удаленного элемента займет нижерасположенный элемент.

7. Установив режим множественного выбора элементов списка, добейтесь того, чтобы подсвеченные элементы из списка можно было бы записать в столбец **B** любого рабочего листа. Разработанную процедуру свяжите с кнопкой «*Разместить выбранные эл-ты списка в столбце «B» рабочего листа*».

8. Проверьте и отладьте созданное Вами программное обеспечение.

5. КОМБИНИРОВАННЫЙ СПИСОК

Комбинируемый список (*ComboBox*) является таким элементом, который включает в себя одновременно и поле ввода и раскрывающийся список. Обращение к тексту, расположенному в поле ввода, осуществляется через свойство *Text*, а к выбранному элементу списка через свойство *Value*. Раскрывающийся список практически идентичен обычному списку - *ListBox*. Основное отличие – отсутствие у него множественного выбора.

Создайте комбинируемый список соблюдая следующие условия:

во-первых, поместите в него элементы 3 столбцов с рабочего листа, используя свойство *RowSource*;

во-вторых, свяжите ячейку рабочего листа A12 со списком, чтобы в ней дублировалось значение свойства *Value*;

При желании можно несколько изменить внешний вид списка, изменив свойство *ListStyle* (0 -по умолчанию).

	A	B	C	D	E	F
1	Петров	Вася	Сторож			
2	Иван	Васильевич	Сидоров			
3	Сидоров	Коля	Поэт			
4	Чертников	Ваня	Шофер			
5	Кашцев	Леня	Певец			
6	Вий	Жора	Шеф			
7	Лешой	Вова	Босс			
8	Ягов	Женя	Директор			
9	Бабов	Сима	Банщик			
10	Горьныгчев	Змей	Пожарник			
11						
12	Ваня					

← Это диапазон ячеек для формирования раскрывающегося списка. Три столбца

← Ячейка связи со списком

Рисунок 3.6 – Вид рабочего листа с исходными данными

В результате работы Вы должны получить диалоговое окно представленное на рисунке 3.6.

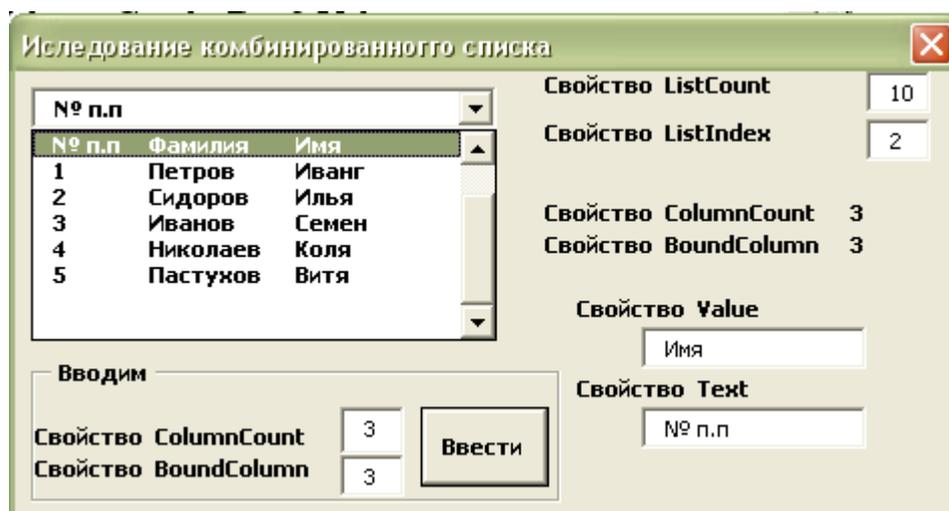


Рисунок 3.6. – Окно диалога для исследования элемента управления «Комбинированный список»

Задание № 7.

1. В групповом окне «*Вводим*» обеспечьте возможность ввода и изменения у раскрывающего списка двух его свойств.

ColumnCount – определяет число столбцов в списке, выводимом на экран. Может принимать следующие значения :

- =0 - столбцы не выводятся;
- =1, 2, 3 - выводится 1, 2, 3 столбца;
- = -1 - выводятся все столбцы.

BoundColumn – определяет столбец, в котором происходит выбор данных для установки значения свойства **Value**. Например, на рисунке 3.6 свойство **BoundColumn**=3, а это значит то, что значение свойства **Value** будет сформировано из третьего столбца списка.

2. Справа в окне диалога расположите текстовые поля и метки для вывода в них свойств комбинированного списка.

Напомним, что свойство **ListCount** – возвращает количество элементов в списке. Это свойство только читается, а **ListIndex** – задает номер выбранной строки. В том случае, если Вы не выбрали ни один элемент, **ListIndex**= -1, а **ListCount**=0.

3. Проведите эксперименты.

✓ Сделайте так, чтобы раскрывающийся список содержал один столбец, два, ни одного. Обеспечьте выбор данных из первого столбца.

✓ Используя свойство **ColumnWidths**, установите различную ширину у каждого столбца. Это можно сделать в *пунктах* (1/72 дюйма), например ... (46;31;120), или в *сантиметрах* - ... (2 см; 3 см; 4см).

✓ Используя обращение к элементам списка **Объект.List(строка, столбец)**, создайте процедуру для записи в какую-нибудь ячейку рабочего листа одного из элементов списка.

6. ПОЛОСА ПРОКРУТКИ И СЧЕТЧИК

Счетчик (*SpinButton*), как и полоса прокрутки (*ScrollBar*), изменяет значение свойства *Value*. Один щелчок мышью изменяет это свойство на величину, заданную свойством *SmallChange* в пределах, определенных свойствами *Min* и *Max*. Вертикальное или горизонтальное расположение счетчика определяется свойством *Orientation*.

Задание № 8. Исследуйте полосу прокрутки и счетчик

1. Создайте окно диалога (форму) "*Исследование прокрутки и спиннера*" представленное на рисунке 3.7. Если необходимо, то объектам присвойте имена. На рабочем листе "*Лист3*" создайте список, изображенный на том же рисунке. Этот список используйте для задания исходных данных при создании макросов и исследовании прокрутки и счетчика.



Рисунок 3.7 - Окно диалога и рабочий лист для исследования работы элемента управления «Полоса прокрутки» и «Счетчик»

2. Напишите процедуру, которая бы при изменении состояния (положения движка) полосы прокрутки заполняла бы соответствующие поля ввода в групповом окне "*Полоса прокрутки*". Эту процедуру свяжите с изменением полосы прокрутки.

3. Аналогичным образом поступите со счетчиком (спиннером).

Обратите внимание на то, что номер по порядку в списке, представленном на рисунке 3.7 справа не совпадает с номером строки на рабочем листе. Например, у Петрова номер п.п. – 1, а запись с его фамилией находится в 4 строке.

4. Напишите процедуру для того, чтобы задать диапазон значений, возвращаемых свойством *Value* прокрутки и счетчика, соответственно. Свяжите ее с кнопкой «*Установка диапазона*»

5. Отладьте процедуры и добейтесь работы окна диалога.

7. ПРОЦЕДУРЫ ОБРАБОТКИ СОБЫТИЙ

Задание № 9. Для вновь созданной книги выполните следующее:

1. Для событий:

- *открыть книгу,*
- *закреть книгу,*
- *добавить новый лист в книгу,*
- *активизировать лист в книге*

-напишите макросы. Макросы, используя окно ***MsgBox***, должны сообщать о том, что соответствующее событие наступило.

2. Создайте и отладьте две процедуры:

- ***Auto_Open,***
- ***Auto_Close.***

Текст процедур может быть любой, например, вызов окна ***MsgBox***.

4. ОБРАБОТКА ОШИБОК В ПРОГРАММАХ VBA

1. ОСНОВНЫЕ ПОЛОЖЕНИЯ

Ошибки возникают в связи с непредвиденными действиями пользователя. Большое количество ошибок не является фатальными, но тем не менее, программы из-за них будут работать непредсказуемо или же их выполнение будет прервано.

Данная работа предполагает изучение механизма устранения ошибок в программах VBA.

Общее задание на работу. Всю содержащуюся в работе информацию об ошибках, их возникновению, обработке, методикам отладки программ, полученную из приведенного ниже материала, а там где необходимо, из справочного материала VBA, законспектировать. Примеры отладить и записать на дискету.

Задание № 1. Наберите текст функции *File_1*, приведенной ниже, которая возвращает *True*, если файл существует и *False*, если нет. Функция *Dir*, используемая в подпрограмме-функции, возвращает файл, соответствующий заданной строке, который встретился первым; или строку нулевой длины, если такого файла нет.

Процедура *Prim* выведет результаты работы функции *File_1* в окно отладки. Попробуйте !

!! Объясните, почему и как функция *File_1* возвращает *False*, если файла нет !!

```
Function File_1(File_Name) As Boolean
```

```
File_1 = Dir(File_Name) <> ""
```

```
End Function
```

```
Public Sub Prim()
```

```
Dim buf As Boolean
```

```
buf = File_1("Книга1234")
```

```
Debug.Print buf
```

```
buf = File_1("C:\WINDOWS\WIN.INI")
```

```
Debug.Print buf
```

```
End Sub
```

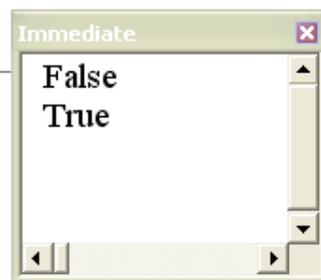


Рисунок 4.1 - Программный код

Замечание. Если Вам кажется, что учтены все случаи в жизни, то Вы глубоко ошибаетесь. Напишите в процедуре *Prim* вместо имени диска *C:* - имя - *P:* и посмотрите что будет !

Задание № 2. Наберите текст процедуры, которая содержала бы следующие операторы:

$$x = 4: y = 0: z = x / y$$

Запустите ее на выполнение и оцените результат.

2. ПЕРЕХВАТ ОШИБОК

Перехватом ошибок называют их обработку.

При обработке ошибок используют инструкции (команды), а также и объекты с их свойствами и методами. К ним относятся:

1. **Элементы, поддерживаемые VBA** для совместимости с предыдущими версиями:

Инструкция Error;

Функция Error.

Эти элементы работоспособны, но лучше ими не пользоваться. Информацию о них можно получить в справке редакторе VBA.

2. **Элементы языка VBA**, используемые для обработки ошибок в приложениях Office:

- **Объект Err** - объект VBA. Этот объект содержит сведения только об одной ошибке. Любая ошибка, возникающая в VBA, записывается как *объект Err*.

- **Объект Error (Errors)** – ошибки, которые использует система (механизм) DAO. DAO – это библиотека объектов доступа к данным, позволяющая работать с библиотеками баз данных в VBA.

- **Метод AssesError** – служит для принудительной генерации ошибок в MS Access. Подобен методу Raise для объектов Err. (о последнем речь пойдет в конце работы)

- **Событие Error** - используется для перехвата ошибок, возникших в форме или отчете MS Access.

В данной работе рассматривается только *Объект Err* и ошибки, генерируемые в приложениях Office-97, кроме тех, которые возникают в MS Access.

3. СИСТЕМА ПЕРЕХВАТА ОШИБОК

Система перехвате ошибок включает:

- **Объект Err.**

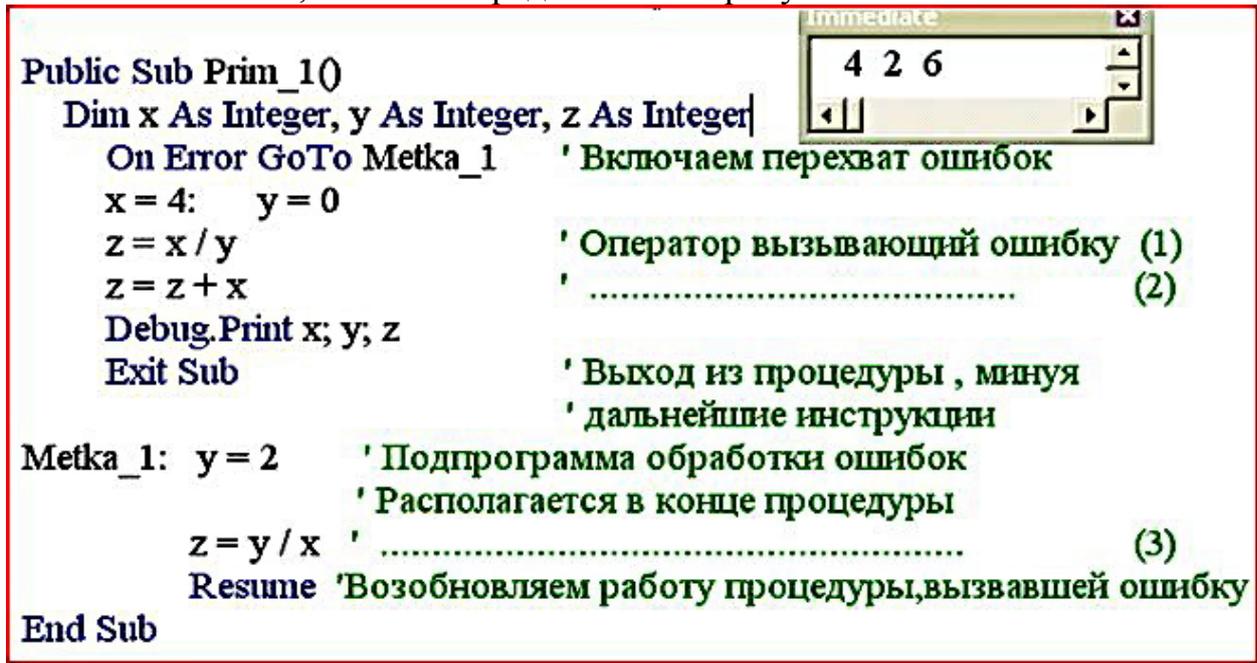
- **Инструкцию On Error**, которая включает перехват ошибок в данной выполняемой процедуре и указывает этой процедуре, что делать, если произошла ошибка.

- **Подпрограмму обработки ошибок**, которая является частью выполняемой процедуры. В ней определяется тип ошибки и производятся действия по ее устранению. Обычно эта подпрограмма располагается в конце процедуры и начинается с метки.

- **Инструкцию Resume**, которая позволяет процедуре после обработки ошибки продолжить выполнение своих операторов.

- *Инструкцию Exit*, позволяющую выйти из процедуры, не выполняя оставшихся операторов. Совершенно очевидно, что она необходима для того, чтобы не выполнять подпрограмму обработки ошибок, находящуюся в теле процедуры. Это необходимо в том случае, если ошибок не возникло.

Задание № 3. Переделайте пример задания № 2, вставив в него перехват ошибок так, чтобы этот пример работал и с ошибками. Один из вариантов, который включает основные компоненты системы перехвата ошибок, может быть таким, каким он представлен на рисунке 4.2.



```

Public Sub Prim_1()
  Dim x As Integer, y As Integer, z As Integer
  On Error GoTo Metka_1 ' Включаем перехват ошибок
  x = 4: y = 0
  z = x / y ' Оператор вызывающий ошибку (1)
  z = z + x ' ..... (2)
  Debug.Print x; y; z
  Exit Sub ' Выход из процедуры , минуя
           ' дальнейшие инструкции
Metka_1: y = 2 ' Подпрограмма обработки ошибок
           ' Располагается в конце процедуры
  z = y / x ' ..... (3)
  Resume ' Возобновляем работу процедуры, вызвавшей ошибку
End Sub

```

Рисунок 4.2 - Процедура с обработкой ошибки

4. КОМПОНЕНТЫ СИСТЕМЫ ПЕРЕХВАТА ОШИБОК.

4.1. Инструкция On Error может применяться многократно в одной и той же процедуре. Способ обработки ошибок при этом может быть каждый раз свой.

Таблица 4.1 - Формы инструкции On Error

<i>Формы инструкции On Error</i>	<i>Особенности использования и что позволяет сделать. Ограничения.</i>
On Error GoTo метка	Управление передается подпрограмме обработки ошибки. Вход в нее определяется меткой
On Error Resume Next	Ошибка игнорируется и происходит передача управления следующей инструкции.
On Error GoTo 0	Отключает обработчик ошибок для данной процедуры.

4.2. Инструкция Resume. Может быть использована только в подпрограмме обработки ошибок! Она является необязательной, а в том случае, ес-

ли после обработки ошибки следует закончить выполнение процедуры, вызвавшей ошибку, *Resume* можно не указывать.

Таблица 4.2 - Формы инструкции Resume

Формы инструкции Resume	Особенности использования и что позволяет сделать. Ограничения.
Resume или Resume O	Передает управление инструкции, в которой произошла ошибка, и происходит снова повторное выполнение этой инструкции.
Resume Next	Передает управление инструкции, которая следует за той, в которой произошла ошибка
Resume метка	Передает управление инструкции, которая помечена соответствующей меткой

4.3. Инструкция Exit

Инструкция имеет 5 форм, для каждого блока процедуры, из которого требуется выйти, не выполняя дальнейших инструкций.

- *Exit Sub*
- *Exit Function*
- *Exit Do* , *Exit For* или *Exit Property*. - встречаются значительно реже.

Задание № 4. Ответьте на вопросы:

1. Что будет, если в примере задания №3 инструкцию *On Error Go To Metka_1* поставить после оператора (2)? Проверьте на примере.
2. Что будет, если убрать инструкцию *Exit Sub*, и почему? Проверьте на примере.
3. Что будет, если убрать инструкцию *Resume*? Проверьте на примере.
4. Сделайте так, чтобы оператор (1) повторно не вызывался на выполнение, а вместо него работал бы оператор из подпрограммы обработки ошибок – (3).
5. Сделайте так, чтобы передача управления из программы обработки ошибок была сделана на оператор *Debug* из основной процедуры.

4.4. Объект Err.

Глобальный объект, содержащий информацию об одной, последней ошибке. Он позволяет выявить тип ошибки, и как всякий объект имеет свойства и методы.

Свойства объекта Err

- Свойство *Number* содержит номер возникшей ошибки. Эти номера и описания ошибок можно уточнить по справочнику VBA в разделе «Перехватываемые ошибки». Рассматриваемое свойство является заданным по умолчанию. Поэтому 2 записи:

Err и *Err.Number* - являются равнозначными.

После обработки ошибок это свойство обнуляется.

- Свойство *Source* –содержит имя проекта, в котором произошла ошибка.
- Свойство *Description* – содержит описание ошибки, но некоторые из них возвращают текст: «Ошибка, определяемая объектом».
- Свойство *HelpFile* – имя файла справки Visual Basic.
- Свойство *HelpContext* – идентификационный номер в справке, соответствующий ошибке.
- Свойство *LastDLLError* –возвращает код системной ошибки с последнего вызова библиотеки DLL.

Задание № 5.

1. Измените в программу задания №3 и получите значения всех 6-ти свойств, возникшей в ней ошибки.

2. С одним типом ошибки, а именно: *(11) - деление на 0* , Вы столкнулись уже в начале работы. При отладке пункта №1 задания №5 Вы, непременно столкнулись еще минимум с двумя типами других ошибок.

Составьте процедуру с этими тремя типами ошибок, взяв за основу переделанный Вами пример из задания №3.

В этом примере, используя блок *Select Case* добейтесь того, чтобы каждый тип ошибок обрабатывался своей подпрограммой. Например, так:

Metka_1:

```

Select Case Err.Number
  Case 11
    ' Первая подпрограмма
  Case .....
    ' Вторая подпрограмма
  Case .....
    ' Третья подпрограмма
End Select

```

Resume

3. Добейтесь того, чтобы пример из задания №1 работал без сбоев.

Задание № 6.

1. Нарисуйте на *Листе1* объект. Это может быть любой рисунок – круг в круге, например. Дайте ему имя, например “*Круги*”.

Доступ к такому объекту может быть осуществлен через метод *Select*:

Sheets(1).DrawingObject(“Круги”).Select

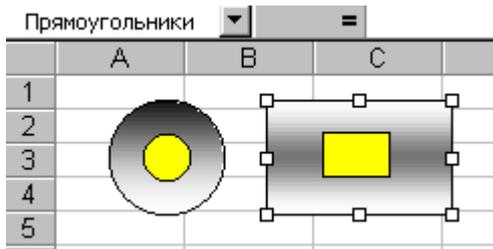


Рисунок 4.3 – Пример объектов для экспериментов

Аналогично поступите еще с одним объектом, например, “*Прямоугольники*”.

Эти рисунки будут являться базой для экспериментов. Их всегда можно скопировать в любое место и уже с копиями производить опыты, не трогая базу и всегда имея под рукой оригинал.

2. Создайте копии этих рисунков на *Листе2*. Именно с копиями рисунков и следует производить эксперименты.

3. Напишите процедуру, которая один за другим удаляла бы эти рисунки с «*Листа2*». Перед каждым новым запуском этой процедуры (перед каждым удалением рисунков) необходимо вручную скопировать эти рисунки на «*Лист2*», если их там нет. Иначе программа выдаст ошибку.

4. Исправьте текст процедуры так, чтобы и в случае отсутствия рисунков, программа работала без сбоев.

Методы объекта Err

а. Метод Raise.

Метод позволяет генерировать ошибку во время выполнения программы.

Этот метод используется для проверки подпрограмм обработки ошибок и для генерирования собственных ошибок. В последнем случае, для того чтобы номер собственной ошибки *Number* гарантированно не совпал с номером стандартной ошибки, следует получить его сложением :

*Number :=*Номер Вашей ошибки + *vbObjectError*

Например, 1

Константа VBA

В результате такого сложения всегда можно определить, внутренняя (встроенная) это ошибка или принадлежит к пользовательскому классу. Номера ошибок пользовательского класса лежат в диапазоне $vbObjectError < Err.Number < vbObjectError + 65536$

Синтаксис метода:

Обязательный параметр

Err.Raise Number, Source, Description, HelpContext, LastDLLError

Объект

Метод

Параметры, имеющие тот же смысл, что и соответствующие свойства объекта Err

Пример использования:

Err.Raise vbObjectError + 1, Description := “ Это моя, родная ошибка !”

б. Метод Clear

Метод сбрасывает (очищает) значения всех свойств объекта *Err*. Используется после обработки ошибки.

Задание № 7.

В соответствии с предложенным ниже алгоритмом, напишите процедуру генерирования встроенной и собственной ошибок и отладьте ее.

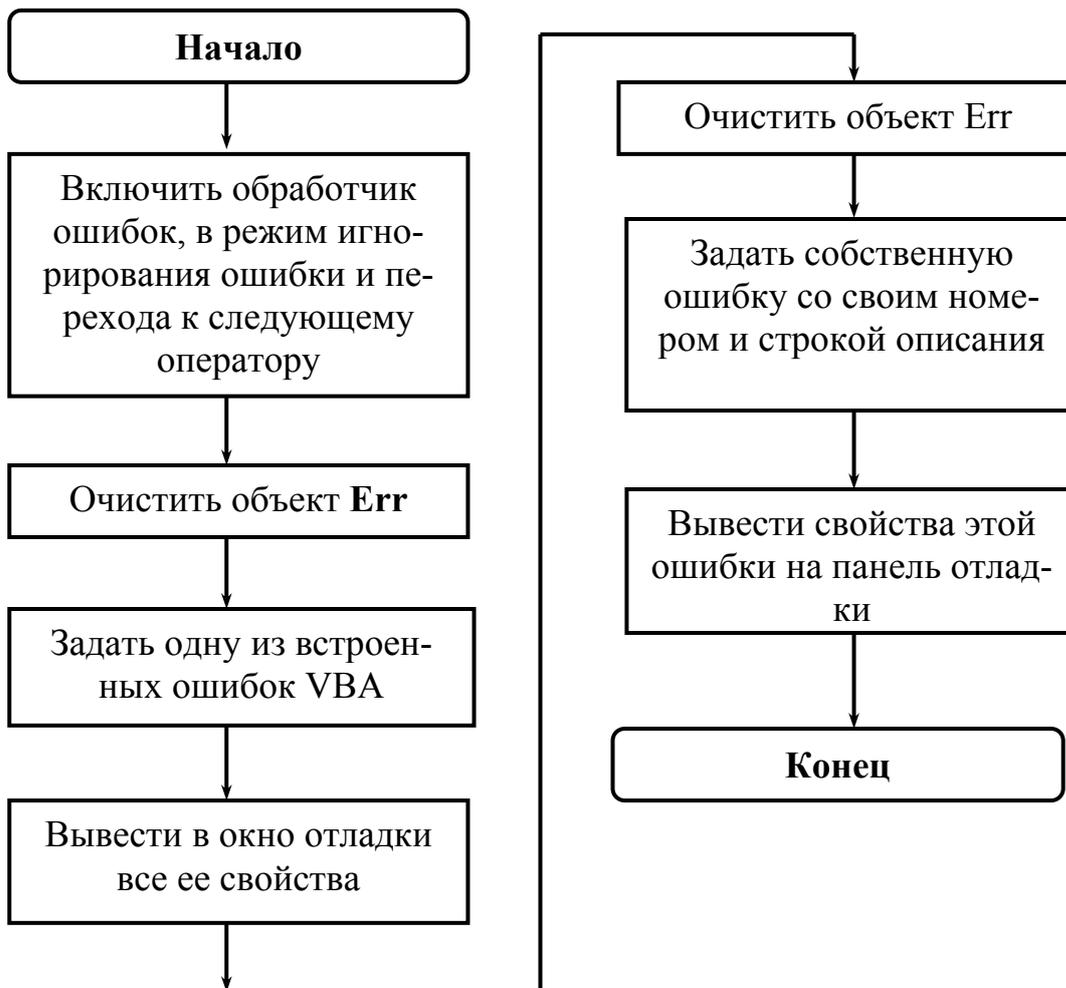


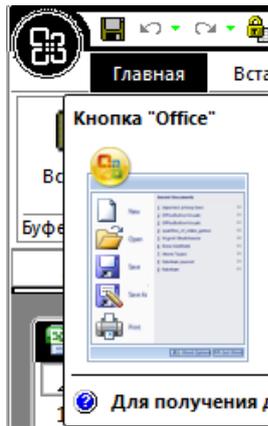
Рисунок 4.4 – Алгоритм для разработки программы

5. СРЕДСТВА СОЗДАНИЯ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА. МЕНЮ И ПАНЕЛИ ИНСТРУМЕНТОВ

1. ОСНОВНЫЕ ПОЛОЖЕНИЯ

Установка на выполнение работы. Ничего не принимайте на веру. Все проверяйте. Примеры и готовые задания запоминайте на отдельных магнитных носителях. Конспектируйте.

В приложениях выпуска 2007 системы Microsoft Office: Word, Excel, PowerPoint, Access - пользовательский интерфейс был значитель-



ительно изменен. *Кнопка Microsoft Office*  заменяет меню *Файл* и расположена в верхнем левом углу окна перечисленных приложений Microsoft Office.

При нажатии на *кнопку Microsoft Office* можно выполнить простые команды по открытию, сохранению и печати файла, как и в прошлых версиях Microsoft Office.

Окно программы Excel имеет новый внешний вид, представленный на рисунке 5.1. На этом рисунке приведены названия элементов пользовательского интерфейса.

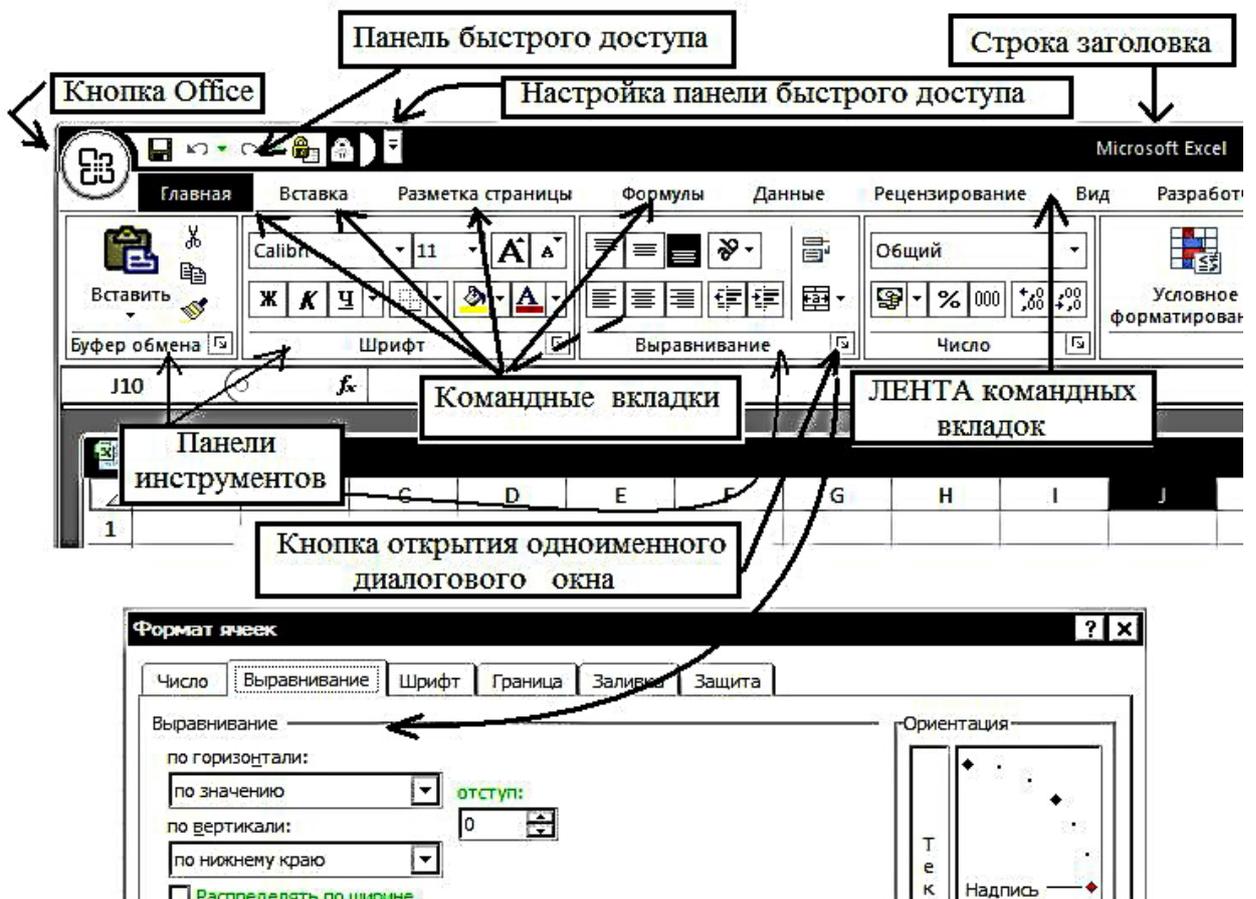


Рисунок 5.1 - Вид окна программы Excel

Верхняя часть окна Excel содержит *Ленту командных вкладок*, каждая из которых содержит ряд *панелей (наборов) инструментов*. Лента динамична и при смене задач, которые решает пользователь, она меняется, вкладки на ней перемещаются, добавляются новые и т.д.

Кроме ленты, для доступа к наиболее часто используемым командам, появилась панель инструментов – *Панель быстрого доступа*. Она доступна для добавления новых команд и для их удаления.

При этом изменились и способы модификации пользовательского интерфейса.

Для изменения или создания пользовательского интерфейса в приложениях Office предусмотрено:

а). Изменения панели быстрого доступа встроенными средствами меню Excel. Они называются *изменениями на этапе разработки*.

б). Использование *программ на VBA*. В том случае, если программный код изменяет систему меню в период выполнения программ, изменения меню называют *изменениями на этапе выполнения*.

2. ИЗМЕНЕНИЯ ИНТЕРФЕЙСА НА ЭТАПЕ РАЗРАБОТКИ

Если в ранних версиях Office на этапе разработки легко модифицировались практически все панели инструментов и меню, то теперь *Панели инструментов пользователя* по сути дела представлены одной *Панелью быстрого доступа*. Только в нее могут быть добавлены (удалены) команды, из числа предложенных разработчиками.

Для добавления кнопок (элементов встроенных панелей инструментов), вызывающих те или иные команды приложения, следует щелкнуть правой кнопкой мыши по этой команде (кнопке-значку), расположенной на соответствующей вкладке Ленты, и в контекстном меню выбрать инструкцию «*Добавить на панель быстрого доступа*».

Если есть необходимость поместить на *Панель быстрого доступа* всей панели инструментов, например «*Шрифт*» со вкладки «*Главная*», то щелкнуть нужно по названию панели инструментов, а далее аналогично.

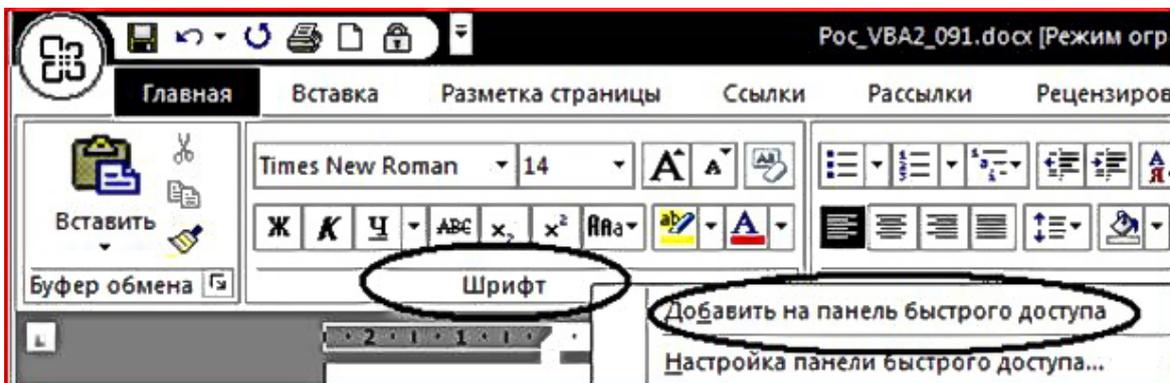


Рисунок 5.2 – Формирование *Панели быстрого доступа*

В том случае, если требуется отыскать и поместить на панель быстрого доступа какую-нибудь команду, которой нет на ленте, следует из контекстного меню (см. рисунок 5.2) выполнить инструкцию *Настройка панели быстрого доступа*. Появится окно «*Параметры Excel*». Используя правую часть этого окна, при выборе категории «*Настройка*», можно сформировать «*Панель быстрого доступа*».

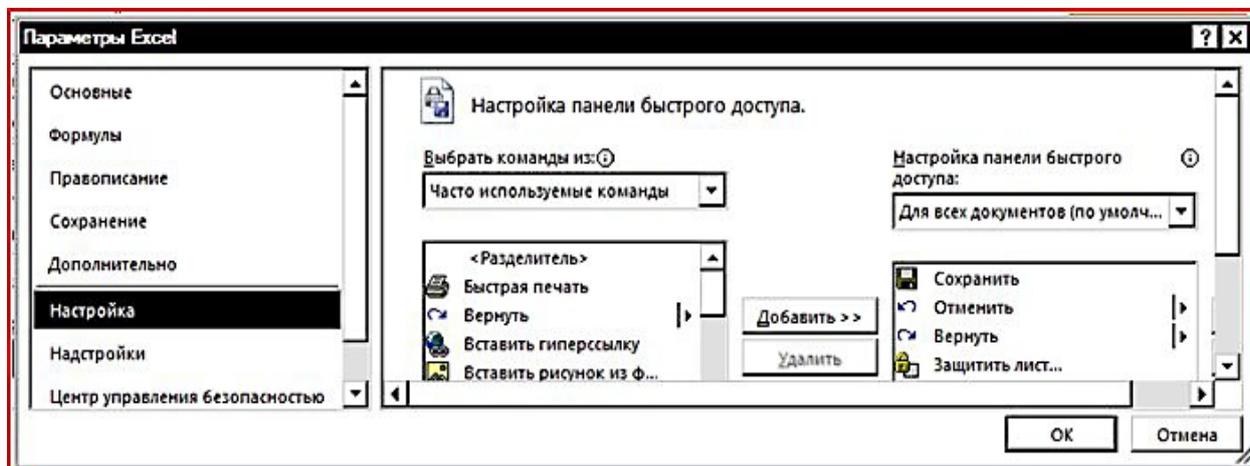


Рисунок 5.3 – Окно Параметры Excel

Окно «*Параметры Excel*» можно вызвать кнопкой «*Настройка панели быстрого доступа*» (см. рисунок 5.1) /*Другие команды*.

При работе с макросами и элементами управления весьма полезной оказывается вкладка «*Разработчик*», которая появится на ленте в том случае, если флажок «*Показывать вкладку «Разработчик» на ленте*» поднят. Для того чтобы добраться до флажка нужно щелкнуть по кнопке *Office*, далее по кнопке «*параметры Excel*» и в одноименном окне выбрать категорию «*Основные*».

Заметим, что в более ранних версиях (до Office 2007) процесс создания пользовательского меню имел значительно больше возможностей и степеней свободы. На этапе разработки можно было создавать меню с кнопками, полями ввода, раскрывающимися меню и т.д. Теперь процесс ограничен вставкой кнопок в панель быстрого доступа.

3. ИЗМЕНЕНИЯ ИНТЕРФЕЙСА НА ЭТАПЕ ВЫПОЛНЕНИЯ

В ранних версиях Office различали понятия: строка меню и панель инструментов. При этом они являлись объектами одной коллекции и поэтому работа с ними, особенно программно, была во многом схожа.

В Office 2007, объекты для создания панелей с инструментами не претерпели особых изменений ни в названии, ни в методах. Но размещаются все пользовательские панели с инструментами на вкладке «*Настройки*» в группе «*Настраиваемые панели инструментов*». В данном случае речь может идти

и о создании меню в том понимании, которое присутствовало в ранних версиях Office. Один из вариантов пользовательского меню представлен на рисунке 5.4. Здесь же приведены основные термины, которые используются при создании таких меню.

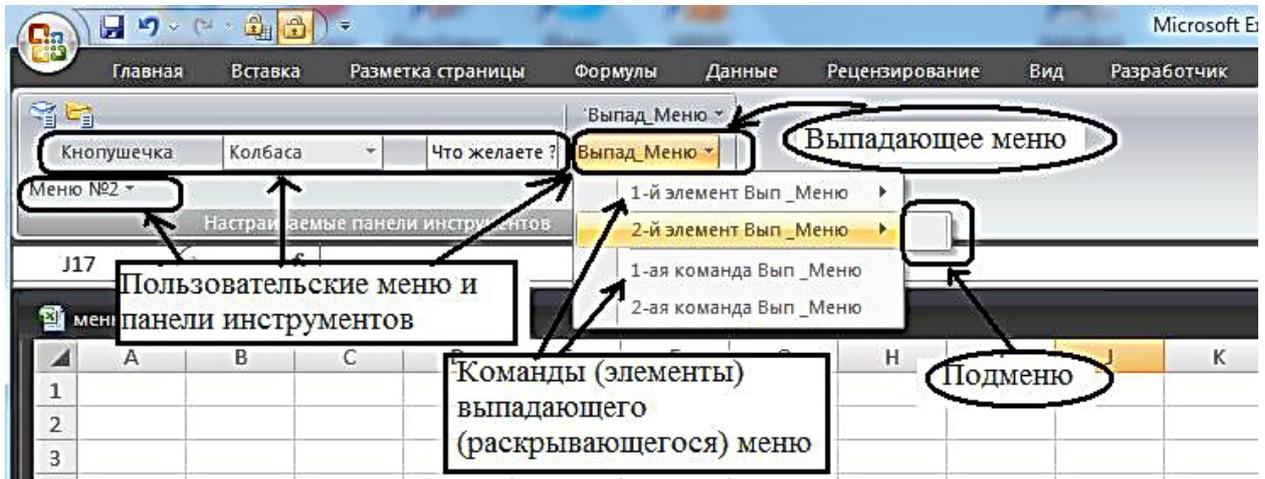


Рисунок 5.4 – Основные элементы пользовательского меню

На рисунке четыре строки меню. Первая строка содержит три элемента управления: кнопку, раскрывающийся список, поле ввода. Вторая - одну команду. Две последующие строки вызывают выпадающие меню, причем, первые две команды у них вызывают свои подменю, которые в свою очередь могут содержать любые элементы управления или инструкции.

3.1 Основные объекты

Модификация меню происходит через объект верхнего уровня в модели объектов Office - *CommanBars*. Каждый объект этой коллекции - *CommandBar* содержит, в свою очередь, коллекцию объектов *CommandBarControls*, составляющих отдельные элементы меню и панелей инструментов. Для того, чтобы получить эти элементы используют свойство *Controls* объекта *CommandBar*.

Все элементы *CommandBarControls* объекта *CommandBar* принадлежат к одному из трех классов:

CommandBarButton – определяет кнопку или команду меню,

CommandBarPopur – определяет подменю,

CommandBarComboBox – сложно-организованные кнопки, заданные элементами управления.

3.2 Создание строк меню (панелей инструментов)

Объект *CommandBar*

Для создания строк меню служит объект *CommandBar*. Процедура создания выглядит следующим образом:

Public Sub qq_1()

Dim CBar As CommandBar ‘Определяем объектную переменную – CBar как объект класса CommandBar - панель инструментов.

**Set CBar = CommandBars.Add(Name:="PetrovBar", _
Position:=msoBarTop, MenuBar:=True, _
Temporary:=True)**

‘Создаем новый объект семейства CommandBars, а, именно, панель инструментов с именем PetrovBar, используя метод Add. Объектная переменная CBar ссылается на этот вновь созданный объект.

CBar.Visible = True ‘Используя свойство Visible объекта CommandBar делаем его (панель инструментов (меню) видимым.

End Sub

Если потребуется запустить эту процедуру повторно, то сначала уничтожьте созданное меню. Это можно сделать используя следующий макрос:

Public Sub qq_1_Del()

CommandBars(Name:="PetrovBar").Delete

End Sub

Другой путь удаления меню - вызвать контекстное меню (рисунок 5.5) для данного элемента управления. После этого необходимо выполнить команду «Удалить настраиваемую панель инструментов»

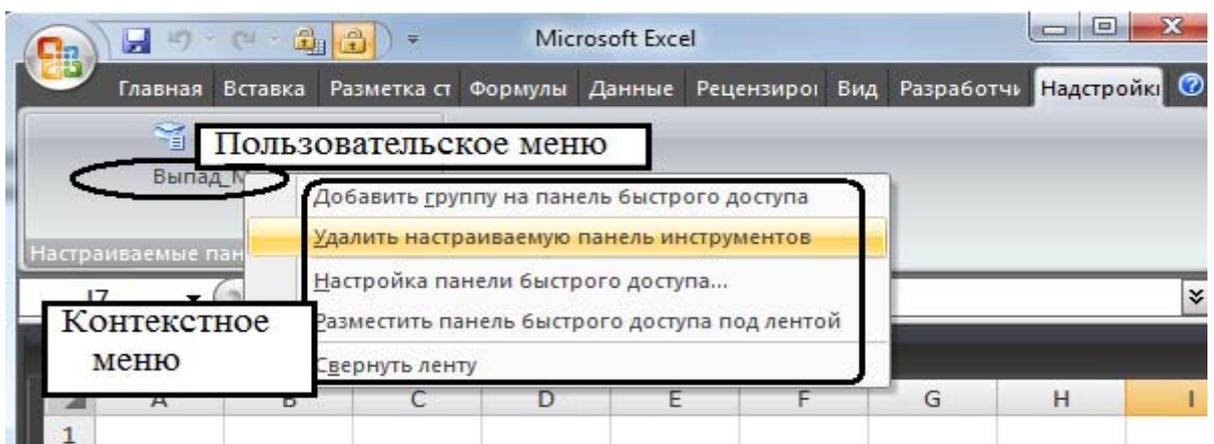


Рисунок 5.5 – Удаление пользовательского меню

В противном случае (если запустите на выполнение макрос qq_1 дважды), машина выдаст ошибку. Дважды создать одно меню в одном месте нельзя. Тоже самое Вы получите, если дважды запустите макрос qq_1_Del, так как удалить объект можно только один раз.

✓ Представленная процедура создаст новое **головное меню**. Это такое, которое расположено сверху на панели «*Настраиваемые панели инструментов*» на вкладке «*Настройки*», и которое заменит существующее. Этот факт определяет значение параметра *MenuBar:=True*. Если установить *MenuBar:=False*, то будет создано **обычная строка меню**, положение которой определит параметр *Position*.

✓ Параметр *Position* может принимать следующие значения (рядом с именем константы проставлено ее значение):

msoBarLeft (0)– привязка строки меню к левой границе окна основного приложения,

msoBarRight (2)– к правой границе,

msoBarTop(1) – к верхней границе,

msoBarBottom(3) – к нижней границе,

msoBarFloating(4) – задает свободное размещение строки меню,

msoBarPopup(5)- указывает на то, что меню является контекстным меню.

✓ В случае создания обычной строки меню (*MenuBar:=False*), ее появление на экране (видимость) определяется программным путем. За это отвечает свойство *Visible*.

Значение свойства *Visible = True* определяет то, что меню будет видно на экране.

По умолчанию свойство *Visible =False*. Если это значение не изменить, то для представленной процедуры головное меню на экране дисплея не появится.

✓ Значение параметра *Temporary:=True* определяет то, что строка меню удаляется после закрытия основного приложения. При значении *Temporary:=False* строка меню будет восстановлена при очередном открытии этого приложения.

Примечание.

Свойство *Visible* определяет видимость элементов меню и самого меню, в целом. Но пользовательское меню будет видно на экране только после того, как вы программно разместите в нем элементы меню.

После выполнения процедуры *qq_1* внешний вид экрана не измениться, так как элементов в нашем меню нет.

3.3 Свойства объекта *CommandBar*, представляющего отдельную панель (строку)

Задание № 1. Наберите и запустите в пошаговом выполнении процедуру, которая выводит имена и др. свойства всех панелей.

Public Sub qq_2()

Dim CBar As CommandBar

For Each CBar In CommandBars

```

Debug.Print "Name="; CBar.Name; "NameLocal="; CBar.NameLocal
Debug.Print "Enabled="; CBar.Enabled; " Visible="; CBar.Visible; ""
Debug.Print " ="; CBar.BuiltIn; " Position ="; CBar.Position
Debug.Print " ="; CBar.Protection; " ="; CBar.Index

```

Next**End Sub**

Помимо известных свойств объекта *CommandBar* в программе присутствуют следующие:

- **Name u NameLocal** –родное имя панели на английском языке и локализованное – на русском.
- **Enabled** –если свойство равно True, то имя панели (строки меню) появляется в списке доступных меню,
- **BuiltIn** – если равно True, то это встроенная панель, если False – то определенная пользователем.
- **Protection** – задает защиту панели от изменений пользователем и представляет сумму констант. Если =0, то панель доступна во всем.
- **Index** –порядковый номер элемента в коллекции. В данном случае, в коллекции панелей и строк меню.
- **Controls** – центральное свойство, возвращающее коллекцию элементов, располагаемых на панели.

3.4 Методы объекта CommandBar

- **Add** – добавляет элемент,
- **Delete** – удаляет элемент,
- **Reset** – восстанавливает установки встроенных панелей.

3.5 Программирование элементов строк меню (панелей) CommandBar

Доступ к отдельным элементам меню осуществляется через свойство *Controls* коллекции *CommandBarControls*. В свойстве *Controls* хранятся ссылки на все элементы одного меню.

а. *Обращение к существующим элементам меню (панели инструментов)*

Задание № 2. Наберите текст процедуры. Запустите ее. Не встречавшееся ранее, свойство *Caption* содержит имя элемента меню. Укажите на объекты, коллекции, свойства. Расшифруйте работу операторов. Что делает процедура?

```

Public Sub qq_4()
    Debug.Print CommandBars("Standard").Name;
Dim My_Element As CommandBarControl
    For Each My_Element In CommandBars("Standard").Controls
        Debug.Print My_Element.Index; " =="; My_Element.Caption
    Next
End Sub

```

б. Добавление новых элементов в меню (панель инструментов)

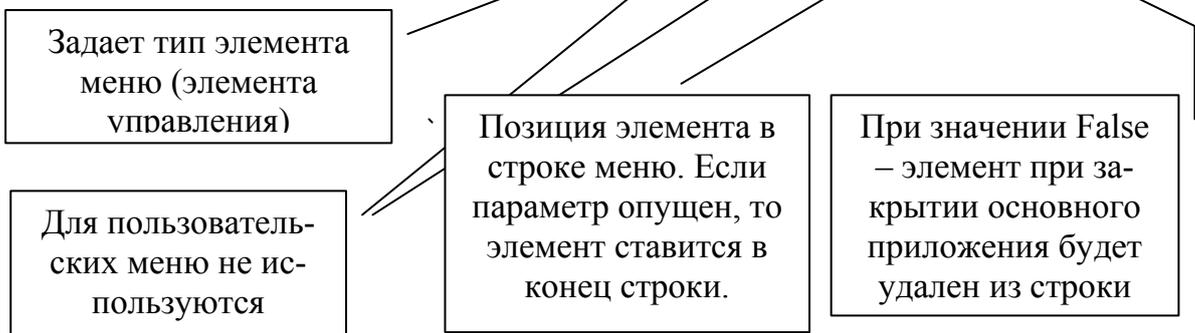
Создайте строку меню с именем «*PetrovBar*» с помощью процедуры *qq_1()*.

Добавление строк меню (панелей) производит метод *Add* семейства *CommandBarControls*. При этом можно размещать на создаваемой панели объекты *CommandBarControls* всех трех, указанных выше классов. Вызов метода производится оператором:

```

CommandBars("PetrovBar ").Controls.Add(Type, Id, Parameter, Before, Temporary).

```



Задание № 3. Создайте панель инструментов с одной кнопкой, раскрывающимся списком и полем ввода. Ее вид представлен на рисунке 5.6, а текст макроса *qq_5()*, позволяющий это сделать, приведен ниже.

Кнопке присвойте макрос, удаляющий созданную пользовательскую панель.

Для списка самостоятельно напишите макрос, который переписет имя выбранного элемента списка в поле ввода.

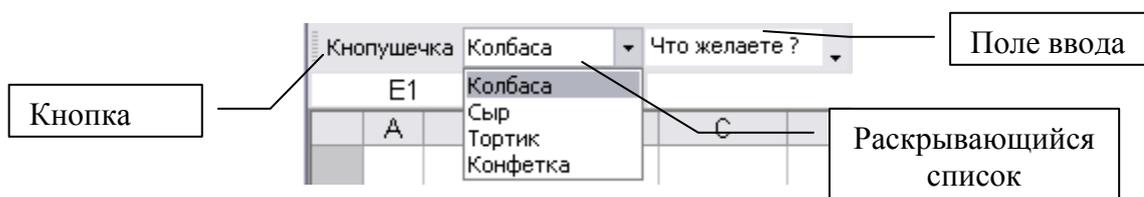


Рисунок 5.6 - Вариант панели инструментов пользователя

Public Sub qq_5()

```

'=====

Dim My_Button As CommandBarButton           ' Кнопка
Dim My_Edit As CommandBarControl           ' Поле ввода
Dim My_DropDown As CommandBarComboBox     ' Раскрывающийся
                                           ' список

With CommandBars("PetrovBar")
'-----
    Set My_Button = .Controls.Add(Type:=msoControlButton)
                                           'тип элемента меню

    With My_Button
        .Style = msoButtonCaption           '(стиль кнопки –в данном случае это надпись на кнопке, ' при msoButtonAutomatic –чистая кнопка, и тд.)

        .Caption = "Кнопушечка"
        .Enabled = True
        .OnAction = "qq_Del"               '(связываем с макросом)
        .TooltipText = "Кнопка удаления этой строки меню" ' текст, появляющийся рядом с 'кнопкой при установке на кнопку мыши

    End With

'-----
    Set My_DropDown = .Controls.Add(Type:=msoControlDropdown)
    With My_DropDown
        .Caption = "Список_Ш" ' определяем имя списку
        .AddItem "Колбаса", 1 ' добавляем первый эл-т списка и определяем его номер.
        .AddItem "Сыр", 2
        .AddItem "Тортик", 3
        .AddItem "Конфетка", 4
        .ListIndex = 1 ' задает номер выбранного элемента списка .
    End With

'-----
    Set My_Edit = .Controls.Add(Type:=msoControlEdit)
    With My_Edit
        .Caption = "Поле_Ввода" ' это имя поля.
        .Text = "Что желаете ?" ' текст в поле
    End With

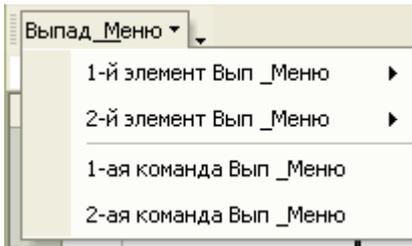
'-----
End With
End Sub

```

в. Создание выпадающих меню и подменю.

Задание № 4 Удалите все пользовательские меню.

Используя макрос, приведенный ниже, создайте выпадающее меню с 4-мя элементами, вид которого представлен на рисунке 5.7.



Это выпадающее меню содержит 2 группы, отделенные объемной линией. Первая содержит два элемента, которые вызывают подменю. Вторая группа включает две простых команды, причем, первая связана с макросом.

Рисунок 5.7 - Меню
Public Sub qq_6()

```

Dim CBar As CommandBar                'стока меню

Dim Element_Vip_menu As CommandBarControl 'выпадающее меню
Dim Element_Pod_menu_1 As CommandBarControl 'подменю в вып.меню
Dim Element_Pod_menu_2 As CommandBarControl 'подменю в вып.меню
Dim Element_Komanda_1 As CommandBarButton '1-ая команда в вып.меню
Dim Element_Komanda_2 As CommandBarButton '2-ая команда .меню

'Создание строки меню. В том случае, если оно у Вас уже создано,
'сначала удалите его

Set CBar =CommandBars.Add(Name:="PetpovBar", _
Temporary:=True, Position:=msoBarTop, MenuBar:=False)
CBar.Visible = True
'Добавление элемента - выпадающее меню В СТРОКУ МЕНЮ
Set Element_Vip_menu = CBar.Controls.Add(Type:=msoControlPopup)
Element_Vip_menu.Caption = "Выпад_ &Меню"
'Добавление 1-го элемента выпод.меню - подменю.
Set Element_Pod_menu_1 = CBar.Controls("Выпад_ &Меню").Controls _
.Add(Type:=msoControlPopup)
Element_Pod_menu_1.Caption = "1-й элемент Вып_ Меню"
'Добавление 2-го элемента выпод.меню - подменю.
Set Element_Pod_menu_2 = CBar.Controls("Выпад_ &Меню").Controls _
.Add(Type:=msoControlPopup)
Element_Pod_menu_2.Caption = "2-й элемент Вып_ Меню"
'Добавление 3-го элемента выпод.меню - команды №1.
Set Element_Pod_menu_2 = CBar.Controls("Выпад_ &Меню").Controls _
.Add(Type:=msoControlButton)

With Element_Pod_menu_2
.Caption = "1-ая команда Вып_ Меню"
.BeginGroup = True 'с этого эл-та начнется новая группа
.OnAction = "qq_Del" 'связываем элемент с макросом

```

End With

'Добавление 4-го элемента выпод.меню - команды №2.

```
Set Element_Pod_menu_2 = CBar.Controls("Выпад_&Меню").Controls _
    .Add(Type:=msoControlButton)
```

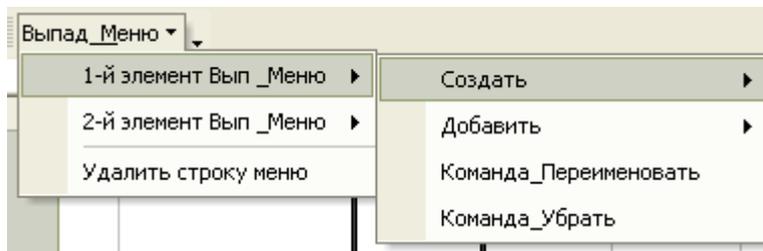
```
Element_Pod_menu_2.Caption = "2-ая команда Вып_Меню"
```

End Sub

Задание №5. Напишите две процедуры.

Первая процедура должна:

- преобразовывать полученное в задании 4 выпадающее меню в другое, вид которого представлен на рисунке 5.8;
- удалять вторую команду из меню (рисунок 5.7);
- менять название у первой команды меню (рисунок 5.7).



Вторая процедура должна создавать подменю с элементами, представленными на рисунке 5.8.

Рисунок 5.8 - Выпадающее меню

4. ПРОГРАММИРОВАНИЕ КОНТЕКСТНЫХ МЕНЮ

Контекстные меню появляются обычно в результате щелчка правой кнопкой мышью по какому-то элементу на экране дисплея. Но, в принципе, можно заставить приложение активизировать контекстное меню в ответ на любое событие. Для примера, организуем перехват события: *активизацию объекта «Лист1»* из текущей книги. В случае его наступления выведем контекстное меню.

При создании контекстного меню следует учесть 2 момента:

- Свойство *Position* должно быть равно константе *msoBarPopup* (5) (см. п.п. 3.2).
- Имя объекта - контекстного меню - сделайте доступным везде, т.е. *Public*, иначе программа Ваше замечательное меню не увидит.

Так как к текущему моменту у Вас накопилось уже очень много исходного материала, воспользуетесь для создания конкретного вида контекстного меню процедурой *qq_50*.

Итак, для начала напишите процедуру создающую меню.

Она должна запускаться при запуске приложения!!

Option Explicit

Public Kont_Cbar As CommandBar

```
-----
Public Sub Soz_Kon_Men() ' Создание контекстного меню
Set Kont_Cbar = CommandBars.Add(Name:="PetrovBar", _
    Position:=msoBarPopup, MenuBar:=False,
    Temporary:=True)
qq_5
End Sub
```

Отображение контекстного меню происходит благодаря использованию метода *ShowPopUp* объекта *CommandBarPopup*. Поэтому событию активизации листа должен быть поставлен в соответствие макрос вызывающий это меню:

```
Private Sub Worksheet_Activate()
Kont_Cbar.ShowPopUp
End Sub
```

Это все. Запустите первую процедуру. Активизируйте Лист1 и получите контекстное меню.

Задание №6. Создайте свое контекстное меню. Свяжите его с каким-нибудь событием.

6. СОЗДАНИЕ ПРИЛОЖЕНИЯ EXCEL ДЛЯ АВТОМАТИЗАЦИИ РАСЧЕТОВ

В результате ультразвукового обследования щитовидной железы получают линейные размеры (длину, ширину, толщину) ее двух долей - левой и правой. После этого определяют объем каждой доли и общий объем железы.

Объем долей определяют по формуле:

*Объем доли = длина * ширина * толщина * коэффициент,*

Общий объем = Объем левой доли + Объем правой доли,

а коэффициент выбирают исходя из условия:

коэффициент 0.526, если возраст человека > 16 лет,

0.474, если возраст человека ≤ 16 лет.

После сравнения рассчитанных объемов с их допустимыми нормами, представленными в таблице 6.1, делают вывод о том: находятся истинные размеры желез в норме, меньше или больше нормы. Результат обследования оформляют в виде бланка, приведенного на рисунке 6.1.

Задание. Автоматизируйте расчеты анализа обследования щитовидной железы. Для этого выполните действия представленные ниже.

1. На одном из листов Excel создайте бланк для размещения исходных данных и результатов анализов (см. рисунке 6.1).
2. На другом листе наберите таблицу 6.1.
3. Создайте экранную форму (окно пользовательского интерфейса) для заполнения данных по расчету и анализу щитовидной железы, например такую, как представленная на рисунке 6.2.
4. Предусмотрите автозапуск экранной формы при открытии рабочей книги.
5. В *пользовательское меню* поместите команду для запуска экранной формы, так как это показано на рисунке 6.2.

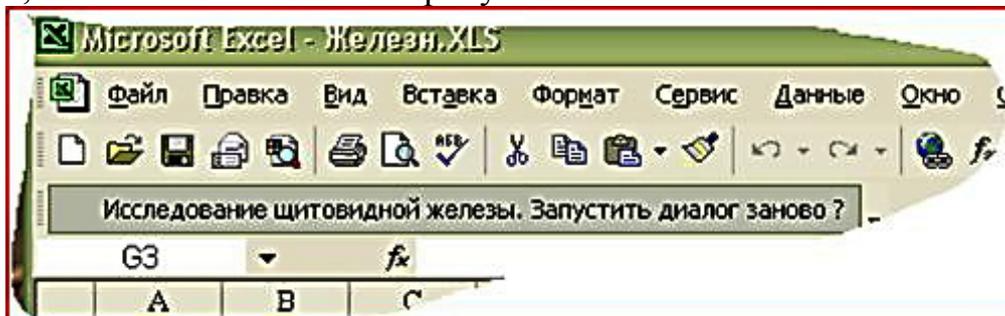


Рисунок 6.1 – Пользовательское меню

Предусмотрите удаление пользовательского меню, в том случае, если наступит событие "Закреть книгу".

По требованию преподавателя:

6. Предусмотрите печать бланка из экранной формы.

7. Предусмотрите возможные ошибки пользователя при вводе чисел в текстовые поля экранной формы.

Например, когда пользователь вместо запятой, как разделителя дробной и целой части, ввел точку. Для обработки ошибок используйте встроенные средства Excel (объект Error).

8. Создайте дополнительную таблицу (базу данных) для хранения полученной в результате исследования информации.

9. Создайте дополнительную экранную форму или добавьте необходимые элементы управления в существующую, чтобы появилась возможность просматривать созданную базу данных, производить поиск информации (например, по фамилии), стирать и корректировать записи.

Таблица 6.1- Нормы объема железа

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1															
2		Нормы объема щитовидной железы													
3															
4		Женщины						Мужчины							
5	Возраст	Правая		Левая		Общий (Пр+Лев)		Правая		Левая		Общий (Пр+Лев)			
6		min	max	min	max	min	max	min	max	min	max	min	max		
7	4	0,51	1,25	0,44	1,18	0,97	2,41	0,54	1,24	0,5	1,18	1,06	2,4		
8	5	0,61	1,19	0,54	1,14	1,17	2,31	0,54	1,66	0,52	1,38	1,06	3,01		
9	6	0,7	1,76	0,68	1,62	1,4	3,34	0,86	1,78	0,78	1,64	1,66	3,38		
10	7	0,91	1,97	0,8	1,82	1,75	3,75	0,96	2,32	0,79	2,17	1,79	4,45		
11	8	1,24	2,48	1,09	2,33	2,38	4,76	1,19	2,37	1,11	2,21	2,32	4,56		
12	9	1,3	3,38	1,19	3,09	2,53	6,43	1,28	2,66	1,21	2,51	2,53	5,13		
13	10	1,65	3,03	1,44	2,42	3,13	5,91	1,49	2,83	1,28	2,66	2,83	5,43		
14	11	1,72	3,43	1,62	3,18	3,38	6,58	1,76	3,2	1,57	3,09	3,38	6,54		
15	12	3,33	4,06	2,39	4,59	5,18	9,18	1,85	3,39	1,67	3,81	3,57	7,67		
16	13	2,54	5,74	2,37	5,05	5,03	10,69	2,13	5,31	1,96	4,6	4,19	9,81		
17	14	2,99	5,79	2,42	5,32	5,57	10,95	2,68	5,58	2,19	5,09	4,96	10,58		
18	15	3,2	7,36	2,65	6,79	5,98	14,04	3,02	6,7	2,46	5,84	5,58	12,14		
19	16	3,38	6,58	3,03	5,75	6,61	12,15	3,7	7,86	3,5	6,76	7,43	14,39		
20	17	4,43	7,95	4,13	7,19	8,7	14,98	4,22	8,74	3,72	8,18	8,06	16,8		
21	18	3,94	7,46	3,49	7,59	7,45	15,05	3,68	5,5	3,38	4,58	7,46	9,68		
22	19	2,85	7,51	2,82	7,06	5,7	14,56	3,68	5,5	3,38	4,57	7,46	9,68		
23	20	3,74	6,89	3,47	6,75	7,26	13,62	7,57	7,57	7,57	7,57	15,13	15,13		
24	21-25	3,94	8,64	3,81	7,51	7,88	16	4,74	9,14	4,22	8,14	9,16	17,28		
25	26-30	4,16	8,78	3,98	7,86	8,28	16,5	5,13	9,37	3,94	9,18	9,31	18,33		
26	31-35	3,4	10,14	2,11	9,61	5,84	19,44	4,65	10,97	4,17	10,37	8,98	21,58		
27	36-40	3,83	8,57	3,66	7,32	7,78	15,64	4,6	9,26	4,15	8,35	8,43	17,43		
28	41-45	4,39	10,11	2,92	9,66	7,59	19,47	5,72	8,72	4,92	7,68	11,32	15,72		
29	46-50	3,65	7,21	3,19	6,55	7,06	13,56	4,31	10,41	3,8	10,3	8,17	20,65		
30	51-55	2,65	9,84	2,37	8,15	5,64	17,42	4,5	9,48	4,12	9,3	8,68	18,72		
31	56-60	3,55	7,57	3,35	7,69	6,9	15,26	3,42	9,7	3,72	9,02	7,78	18,6		
32	61-65	2,38	7,96	2,45	6,49	5,15	14,09	3,19	8,57	2,3	10,04	5,55	18,55		
33	66-70	1,77	5,23	1,5	4,62	3,32	9,8	1,89	8,47	1,85	8,47	3,77	17,05		
34	71-75	1,36	4,14	1,29	4,09	2,65	8,21	2,7	5,12	2,59	4,81	5,39	9,89		
35	76-80	1,2	3,44	1,2	3,18	2,45	6,57	2,47	8,19	1,87	8,07	4,88	16,2		
36	81-95	1,02	2,92	1,01	2,92	2,06	5,86	1,02	2,92	1,01	2,92	2,06	5,86		

	A	B	C	D	E	F	G	H
1								
2								
3		Результат ультразвукового исследования щитовидной железы						
4								
5		Фамилия И.О.						
6		Возраст (лет)		17				
7		Пол (муж/жен)		Мужской				
8								
9		Размеры щитовидной железы						
10		Правая доля	(см)		Объем правой доли		0,486024	
11		длина	1,2		Норма		4,22-8,74	
12		ширина	0,7		Отклонение		-3,733976	
13		толщина	1,1		Заключение		Меньше нормы	
14		Левая доля						
15		длина	3,7		Объем левой доли		4,631956	
16		ширина	1,7		Норма		3,72-8,18	
17		толщина	1,4		Отклонение		Норма	
18					Заключение		В норме	
19								
20					Общий объем		5,11798	
21					Норма		8,06-16,8	
22					Отклонение		-2,94202	
23					Заключение		Меньше нормы	
24								
25								

Рисунок 6.2 – Отчет о результате обследования

Сравнение результатов исследования с нормативными значениями

Фамилия И.О. Коэффициент 0,526

Возраст (лет) Пол Мужской Женский

Размеры щитовидной железы (см)

Левая доля		Правая доля	
Длина	<input type="text" value="10,7"/>	Длина	<input type="text" value="2,2"/>
Ширина	<input type="text" value="5,7"/>	Ширина	<input type="text" value="2,7"/>
Толщина	<input type="text" value="7,4"/>	Толщина	<input type="text" value="2,1"/>

Результаты расчета		Нормы объема		Отклонения объема	
Объем левой доли	<input type="text" value="237,3975"/>	Объем левой доли	<input type="text" value="422-8,14"/>	Объем левой доли	<input type="text" value="-184,6"/>
Объем правой доли	<input type="text" value="6,561324"/>	Объем правой доли	<input type="text" value="4,74-9,14"/>	Объем правой доли	<input type="text" value="Норма"/>
Общий объем	<input type="text" value="243,9588"/>	Общий объем	<input type="text" value="9,16-17,28"/>	Общий объем	<input type="text" value="226,67"/>

Общее заключение

Левая доля	Правая доля	Общий объем
<input type="text" value="Меньше нормы"/>	<input type="text" value="В норме"/>	<input type="text" value="Больше нормы"/>

Рисунок 6.3 - Пользовательский интерфейс

7. ГРАФИЧЕСКИЕ ЭЛЕМЕНТЫ В OFFICE.

1. ОСНОВНЫЕ ПОЛОЖЕНИЯ.

Графические элементы – линии, фигуры, рамки, рисунки могут быть помещены в документы Office и откорректированы с помощью программирования на VBA. Для этого используется коллекция *Shapes*, которая содержит все графические элементы.

Для создания графических элементов используются различные методы, которыми обладает объект *Shapes*. С именами этих методов Вы можете познакомиться в редакторе VBA, вызвав в *Object Browser* и выбрав класс *Shapes* в библиотеке *Word*. Аналогично можно поступить и с библиотекой других приложений *Office*.

Помимо коллекции *Shapes* используют одиночный объект *Shape*. Его применяют для изменения или форматирования одного объекта. Существует также объект *ShapeRange*, представляющий подмножество набора *Shapes*.

2. ОБЪЕКТ SHAPES

2.1 Методы объекта Shapes

✓ *Метод AddShape.* Центральный метод, возвращающий (добавляющий) графический элемент заданного типа. Синтаксис для этого метода:

AddShape (TypeObject, Left, Top, Width, Height)

где: *Left, Top* – координаты верхнего, левого угла,
Width, Height – ширина и высота объекта,
TypeObject – тип объекта: линия, куб и т.д., которые определяются константами,

Например:

<i>msoShape5pointStar</i>	—		<i>msoShapeSmileyFace</i>	—	
<i>msoShapeOval</i>	—		<i>msoShape32pointStar</i>	—	
<i>msoShapeRectangle</i>	—		<i>msoShapeSun</i>	—	
			<i>msoShapeFlowchartSequent</i>	—	

Все константы, которые определяют вид и поведение объектов, можно увидеть вызвав справку. Для этого, находясь в окне модуля редактора VBA, поставьте курсор мыши на слово *AddShape*, и вызовите справку, нажав кнопку *F1* на клавиатуре.

Константы *msoAutoShapeType* будут также доступны при выполнении команды меню редактора VBA View/Object Browser. Результат ее выполнения представлен на рисунке 7.1.

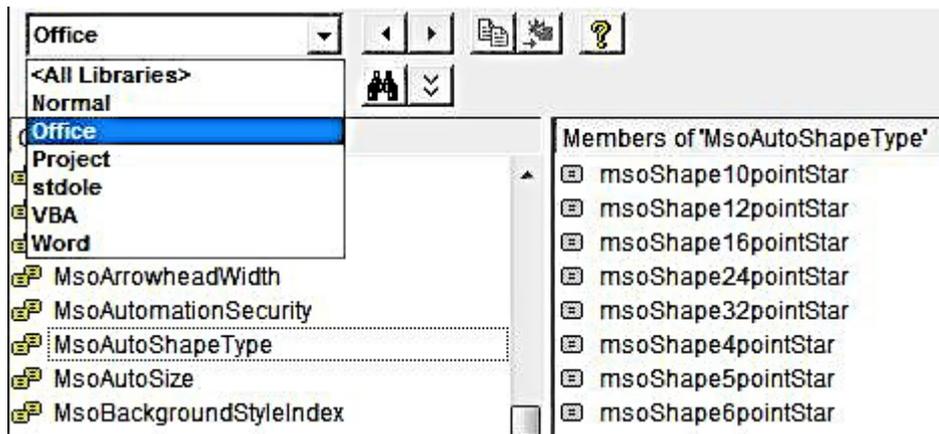


Рисунок 7.1 - Окно `Browser-a`

Задание №1. Познакомитесь со справочными данными и напишите процедуру, добавляющую несколько рисованных объектов в документ. Пример такой процедуры приведен ниже. В данном случае неважно работаете вы в Word-е или Excel-е.

```
Private Sub Add_Shape()
    Call ActiveDocument.Shapes.AddShape(msoShapeCube, 50, 50, 50, 50)
    ActiveDocument.Shapes.AddShape msoShapeCube, 100, 50, 50, 50
End Sub
```

Объясните, что обозначает каждый оператор или инструкция. Какие объекты еще добавлены в документ?

✓ **Методы *AddLine* и *AddCurve*.** Это специальные методы, возвращающие линии. Например, *прямую линию* заданной ширины можно нарисовать с помощью макроса, приведенного ниже.

```
Private Sub Add_Shape_01()
    ActiveDocument.Shapes.AddLine 100, 50, 300, 100
    ActiveDocument.Shapes(1).Select
    Selection.ShapeRange.Line.Weight = 50
End Sub
```

Задание №2. Самостоятельно нарисуйте кривую линию *Curve*, воспользовавшись частью текста предыдущего примера и справкой.

✓ **Метод *AddTextBox*.** Создает текстовое поле.

Задание №3. Используя метод *AddTextBox(Orientation, Left, Top, Width, Height)*, поставьте в заданное место текстовое поле с введенным в него символом -

X

, и поле с любой надписью, расположенной по вертикали.

Задание №4. С помощью программирования на VBA вставьте сначала в документ различные графические элементы.

Используя программу, приведенную ниже, замените символы "?????????" на необходимые для этого операторы так, чтобы все отдельные

2.2 Свойства объекта Shapes

Свойств два. Первое – *Count*, возвращающее число графических объектов, второе – *Range*, которое возвращает объект *ShareRange*.

Задание №7. Напишите процедуру, которая бы выводила количество рисованных объектов и выделяла второй из них.

3. ОБЪЕКТ SHAPE

3.1 Методы объекта Shape

✓ Метод *PickUp, Apply*. Первый метод копирует формат объекта, а второй – применяет этот формат к указанному объекту. Например:

```
Set myDoc=WorkSheets(" Лист1")
With myDoc: .Shapes(1).PickUp: .Shapes(2).Apply: End With
```

Задание №8. Напишите процедуру, в которой эти методы работали бы.

- ✓ Метод *Copy* – копирует объект Shape в буфер обмена.
 - ✓ Метод *CopyPicture* – копирует выбранный объект в буфер обмена как изображение.
 - ✓ Метод *Cut* – вырезает объект и помещает его в буфер (только для внедренных объектов).
 - ✓ Метод *Delete* – удаляет указанные объекты.
- Метод *Duplicate* - создает дубль объекта и возвращает на него ссылку.

Задание №9. Напишите процедуру, в которой использовались бы приведенные методы.

3.2 Свойства объекта Shape

Работа со свойствами объекта *Shape* имеет некоторые особенности.

Объект имеет достаточно большое количество свойств: *Name, Line, Visible* и т.д. Большинство из них предназначено, чтобы вернуть конкретный графический объект для дальнейшей работы с ним, так как в большинстве случаев форматирование объектов *Shape* невозможно при помощи свойств, применяемых непосредственно к объекту *Shape* или *ShapeRange*.

Для этого атрибуты *Shape* сгруппированы под вторичными объектами, которые содержат те или иные свойства. Чтобы установить свойство для объекта *Shape*, необходимо сначала вернуть вторичный объект, представляющий набор связанных атрибутов объекта *Shape*, а потом изменить или установить необходимое свойство.

Задание №10. Наберите текст программы представленный ниже. Выполните ее. Программа должна вернуть два объекта и отформатировать их, используя свойства объекта *Shape*. Используя свойство *Fill*, программа возвращает объект *FillFormat*, а затем обратившись к свойству *ForeColor* объекта *FillFormat* изменяет цвет фона.

Свойство *Line* возвращает объект *LineFormat*, который содержит свойства форматирования линии.

```
Private Sub Add_111()  
    Set MyDoc = ActiveDocument  
    With MyDoc.Shapes.AddShape(msoShapeRectangle, 50, 50, 50, 50).Fill  
        .ForeColor.RGB = RGB(0, 60, 0)  
        .BackColor.RGB = RGB(13, 13, 13)  
    End With  
    With MyDoc.Shapes.AddShape(msoShapeCross, 150, 50, 50, 50).Line  
        .ForeColor.RGB = RGB(0, 0, 60)  
        .Weight = 13  
    End With  
  
End Sub
```

Задание №11. Используя справку и литературу, выясните, какими свойствами обладает объект *Shape*. Составьте программу, использующую одно из них.

8. ПОСТРОЕНИЕ ДИАГРАММ

Чаще всего диаграммы строят вручную с помощью мастера диаграмм. Но в ряде случаев это требуется сделать программным путем. В последнем варианте используют объект *Chart* и все вложенные в него объекты, их свойства и методы. Программный способ построения диаграмм не является интерактивным и имеет меньше возможностей.

Одним из возможных вариантов создания диаграммы является использование метода *SetSourceData* объекта *Chart*. Примерный текст программы, использующей этот метод, представлен ниже на рисунке 8.2, а результат ее работы на рисунке 8.1.

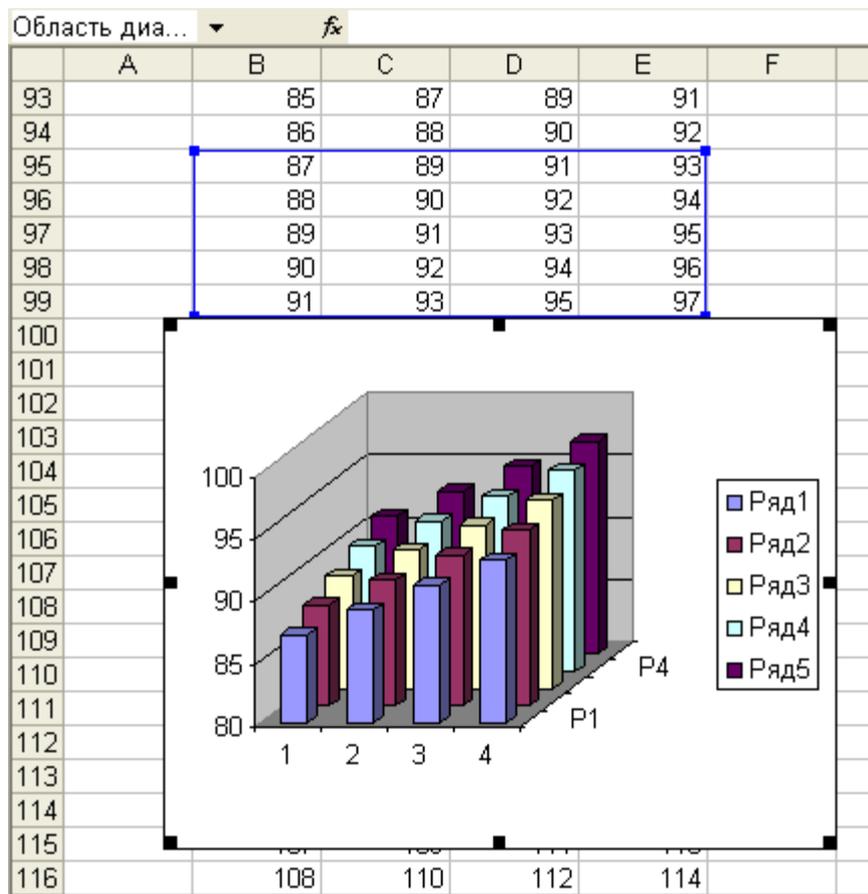


Рисунок 8.1 - Результат работы программы

Задание №1.

1. Наберите текст программы и убедитесь в ее работоспособности.
2. Программным путем:
 - 2.1. Измените положение диаграммы на листе (сместить ее).
 - 2.2. Измените размер диаграммы на листе.
 - 2.3. Свяжите с другой область данных.
 - 2.4. Поменяйте значения свойств объектов-участников объекта *Chart*.
(измените названия осей, заголовков и т.д.)
 - 2.5. Используя метод *Location*, поместите диаграмму на другой лист.

2.6. Создайте диаграмму на отдельном листе диаграмм (не на рабочем листе).

2.7. Раскрасьте объекты в другие цвета.

Задание №2. Создайте самостоятельно диаграмму, используя метод *ChartWizard* вместо *SetSourceData*.

```

(General) Prim0
Sub Prim0()
'=====
'Пример построения диаграммы на 1-ом листе активной книги

Dim myRange As Range, myChart As ChartObject
| With Worksheets(1)
  Set myRange = .Range("B95:E99")
  myRange.Activate
  'Данные для построения возьмем из области ячеек "B95:E99"
  'Активизируем область построения диаграмм

  Set myChart = .ChartObjects.Add(myRange.Left - 10, _
                                  myRange.Top + myRange.Height, _
                                  250, 200)

  'Создадим объект myChart (диаграмму)
  'Свяжем его размеры с размерами области данных для диаграммы.
  'Объект пока пустой. Имеет размеры, но без данных.
End With

With myChart.Chart
  ' Свойство Chart возвращает объект Chart
  .ChartType = xl3DColumn
  .SetSourceData Source:=myRange, PlotBy:=xlRows
  ' Метод SetSourceData указывает на источник данных
  .Location where:=xlLocationAsObject, Name:="Лист1"
  ' Метод Location передвигает диаграмму в новое место
End With

ActiveChart.RightAngleAxes = True
  ' Свойство повышающее наглядность диаграмм
  ' Изменяет угол зрения на диаграмму
End Sub

```

Рисунок 8.2 – Вид окна Модуля с программой

Подобъекты объекта Chart и их названия приведены на рисунке 8.3.

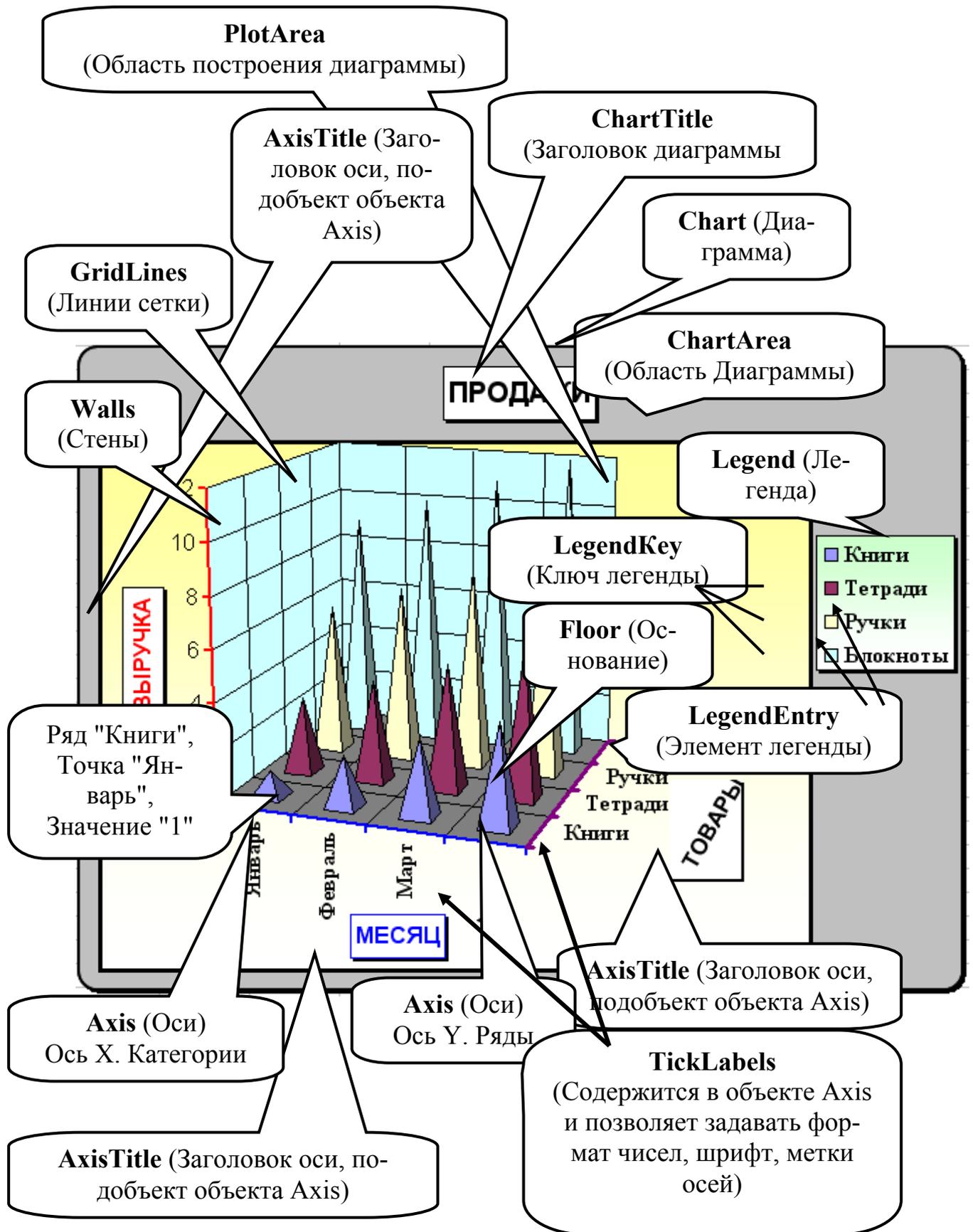


Рисунок 8.3 - Подъобъекты объекта Char

9. СОЗДАНИЕ ПРИЛОЖЕНИЯ ДЛЯ АВТОМАТИЗАЦИИ ЗАПОЛНЕНИЯ БЛАНКА «РАСЧЕТА СТРУКТУРЫ ЦЕНЫ»

Во втором разделе пособия [1] "Создание шаблонов документов (Word). Создание автоматизированного приложения (Excel)" были сформулированы условия и исходные данные для решения прямой и обратной задачи по расчету структуры цены. Решение самих задач возлагалось на студентов.

Прямая задача проста и не вызывает трудностей. Обратная задача требует составления простых уравнений и их решения. Один из возможных вариантов ее решения и реализации приведен далее. Все ссылки на рисунки в тексте сделаны на соответствующие иллюстрации пособия [1].

1. Создание приложения следует выполнять в соответствии с установками «Пояснительной записки». Исходными данными для расчета являются: Стоимость работ (**Ст.Раб**); Расходы на служебные командировки (**Сл.Ком**); Прямые прочие расходы (**Проч.Расх.**); Затраты на работы, выполненные сторонними организациями (**Зат.СтО**); Процент денежного довольствия военнослужащих в собственных расходах по теме (**%Воен**).

2. Создайте на одном из рабочих листов Excel бланк «Структура цены», по аналогии с приведенным на рис.2.2 в работе [1]. Книгу назовите «Srt_Ceni», а лист с бланком - «2этана».

3. Необходимо сделать так, чтобы бланк «Структура цены» заполнялся автоматически в соответствии с исходными данными, располагаемыми на отдельном листе, а к листу с бланком пользователи обращались бы только для вывода его на принтер.

4. Для ввода этих исходных данных и расчета промежуточных величин отведите один из листов рабочей книги, и назовите его «Исх_данные». Примерный вид такого листа представлен на рис.2.3 в работе [1].

Ячейки **C1, C2, D1** содержат сведения общего характера.

Ячейки (**D5:E9**), залитые более темным цветом, предназначены для ввода информации (с клавиатуры, например).

Ячейки (**B14:B23**) содержат постоянные величины, взятые из пояснительной записки.

5. Сначала следует выполнить подготовительную работу и вывести формулы для основных зависимостей, определить трудоемкость темы на каждом из этапов, исходя из стоимости работ, а потом уже рассчитать значения всех статей расходов. Прделаем это.

$$\text{Цена} = \text{Ст.Раб.}/1,2 \dots\dots\dots (1);$$

$$\text{НДС} = \text{Ст.Раб.} - \text{Цена} \dots\dots\dots (2);$$

$$\text{Прибыль} = \text{Цена} * K_{0,091}; \dots\dots\dots (3);$$

$$\text{Цена} = \text{Итого} + \text{Прибыль} + \text{Зат.СтО};$$

$$\text{Итого} = \text{Цена} * (1 - K_{0,091}) - \text{Зат.СтО} \dots\dots\dots (4);$$

Выводим выражения для Тпв и Тпг.

$$\text{Расходы Военных} = (\text{Итого} - \text{Сл.Ком.} - \text{Проч.Расх.}) * (\% \text{Воен.}) \quad (5);$$

$$\text{Расходы Гражданских} = \text{Итого} - \text{Расходы Военных} \quad (6);$$

С другой стороны:

$$\text{Расходы Военных} = \text{ДДД} + \text{ДДФ} + \text{ДДФ} * K_{1,592}; \quad (7);$$

$$\text{Расходы Военных} = K_{dd} * T_{nv} * (1 + K_{0,776}) + K_{dd} * T_{nv} * K_{0,455} + \\ + K_{dd} * T_{nv} * (1 + K_{0,776}) * K_{1,592};$$

$$\text{Обозначив : } Coef1 = K_{dd} + K_{dd} * K_{0,455} + \\ + K_{dd} * (1 + K_{0,776}) * K_{1,592} = \\ = K_{dd} * ((1 + K_{0,776}) * (1 + K_{1,592}) + K_{0,455}) \quad (8);$$

и приравняв: (5) и (7), получим:

$$T_{nv} = \text{Расходы Военных} / Coef1 \quad (9);$$

Поступая аналогично для гражданского населения будем иметь:

$$\text{Расходы Гражданских} = \text{РОТ} + \text{ДОТ} + \text{РОТ} * K_{1,592} \\ Coef2 = K_{zp} * T_{nz} * ((1 + K_{0,686}) * (1 + K_{0,385} + K_{1,592}) + \\ + K_{0,153} * (1 + K_{0,385})) \quad (10);$$

$$T_{nz} = \text{Расходы Гражданских} / Coef2 \quad (11);$$

6. На листе «Исх_данные» отведите ячейки для расчета и хранения $Coef1$ и $Coef2$, а также введите в них соответствующие формулы.

	А	В	С	Д	Е
24					
25	Coef1 (Военн)	57,463333	Считает машина		
26	Coef2 (Граж)	59,425602			

7. Отдельным ячейкам на листе «Исх_данных» удобно дать имена.

Адрес	Имя ячейки
D5	StRab_1
E5	StRab_2
D10	Тпв_1
E10	Тпв_2
D11	Тпг_1
E11	Тпг_2
B14:B18	Имена
B20:B23	соответствуют
B25:B26	обозначению

В ячейки B25 и B26 вставьте формулы (8) и (10), соответственно.

В ячейку D10 запишите формулу для расчета Тпв_1 (см. (9))

$$= ('2mana' D17 - D6 - D7) * (D9 / 100) / Coef_1,$$

а в ячейку D11 запишите, соответственно

$$= ('2mana' D17 - D6 - D7) * ((100 - D9) / 100) / Coef_2$$

Размножьте эти формулы в соседние ячейки E10 и E11.

7. Заполните формулами лист с именем «2этан»

Адрес ячейки	Формула
D14	=Исх_данные!D6
D15	=Исх_данные!D7
D19	=Исх_данные!D8
D22	=StRab_1
D20	=D22/1,2
D21	=D22-D20
D18	=ОКРУГЛ(D20*K_0.091;0)
D17	=D20-D18-D19
D9	=ОКРУГЛ(Kdd* T _{пв_1} *(1+K_0.776);0)
D10	=ОКРУГЛ(Kdd*K_0.455* T _{пв_1} ;0)
D11	=Kzp*(1+K_0.686)* T _{пг_1}
D12	=Kzp* T _{пг_1} *K_0.153
D13	=(D11+D12)*K_0.385
D16	=D17-СУММ(D9:D15)
A5	=СЦЕПИТЬ("по теме: ";Исх_данные!E1)
F31	=Исх_данные!C2
F33	=Исх_данные!C1

В столбце **C** вставьте формулы суммирования по строкам.

Соседний столбец **E** заполните аналогично, используя размножение.

8. Защитите листы.

9. Запомните созданную книгу.

10. Проверьте работоспособность Вашего программного продукта.

11. Оформите ввод данных с использованием экранных форм.

ЛИТЕРАТУРА

1. Бураков П.В., Петров В.Ю. Информационные системы в экономике. Учебное пособие. СПб.: СПбГУИТМО, 2010, 59с.
2. Вильям Дж.Орвис, Visual Basic for Application на примерах. - М.: Бином, 1995. -512 с.
2. К.Соломон. Microsoft Office 97: разработка приложений - СПб.: - СПб,1998. -560 с
3. Камминг Стив. VBA для чайников , 3-издание.:Пер. с англ. - М.: Издательский дом "Вильямс", 2001. - 448 с.
4. Биллиг В.А.. Средства разработки VBA- программиста. Офисное программирование. Т.1. -М.: Издательско-торговый дом "Русская редакция", 2001. - 480 с.
5. Биллиг В.А.. VBA в OFFICE 2000.Офисное программирование. - М.: Издательско-торговый дом "Русская редакция", 1999. - 480 с.
6. Васильев А., Андреев А. VBA в OFFICE 2000. -Спб.: "Питер", 2001. - 432 с.



В 2009 году Университет стал победителем многоэтапного конкурса, в результате которого определены 12 ведущих университетов России, которым присвоена категория «Национальный исследовательский университет». Министерством образования и науки Российской Федерации была утверждена Программа развития государственного образовательного учреждения высшего профессионального образования «Санкт-Петербургский государственный университет информационных технологий, механики и оптики» на 2009–2018 годы.

КАФЕДРА ПРИКЛАДНОЙ ЭКОНОМИКИ И МАРКЕТИНГА

Кафедра прикладной экономики и маркетинга была основана 25 мая 1995 года в связи с началом подготовки в СПбГУ ИТМО бакалавров по направлению 521600 «Экономика». В 1997 году кафедра стала готовить сначала бакалавров, а затем и специалистов по специальности 071900 «Информационные системы в экономике». Со дня основания и по настоящее время кафедрой руководит Почётный работник высшего профессионального образования Российской Федерации, доктор экономических наук, профессор, действительный член Российской академии естествознания Олег Валентинович Васюхин.

В настоящее время кафедра прикладной экономики и маркетинга обучает студентов по специальности 080801 «Прикладная информатика в экономике», а также готовит бакалавров и магистров по направлению 080100 «Экономика»

Кадровый состав кафедры представлен специалистами высшей квалификации – три доктора экономических наук, профессора; 9 кандидатов наук, доцентов и 6 старших преподавателей и ассистентов. Около 30% преподавателей – это молодые специалисты, обучающиеся в аспирантуре или недавно закончившие её.

В соответствии с утверждёнными учебными планами, преподаватели кафедры читают более 40 экономико-управленческих и информационных дисциплин как для студентов своих специальностей и направлений, так и для студентов всего университета. С целью обеспечения более эффективного учебного процесса преподавателями кафедры разработаны более 25

учебно-методических пособий, в том числе, часть из них в виде электронных учебников.

Кафедра обладает современной материально-технической базой. Большая часть учебного процесса реализуется в компьютерных классах Гуманитарного факультета, подключённых к сети Интернет. Все виды занятий, текущий контроль знаний, а также разнообразные виды самоподготовки студентов осуществляются на основе балльно-рейтинговой системы организации учебного процесса.

Ежегодно кафедра выпускает около 50 специалистов, бакалавров и магистров, которые успешно работают на предприятиях различных форм собственности и направлений деятельности. Часть выпускников каждый год продолжают обучение в аспирантуре СПбГУИТМО. Практически все студенты кафедры, начиная с 3-4 курса, в свободное время работают на предприятиях Санкт-Петербурга, что в большинстве случаев является основой для прохождения различного рода практик и подготовки выпускной квалификационной работы.

Преподаватели и аспиранты кафедры ведут активную научно-исследовательскую деятельность, участвуя в хоздоговорных исследованиях для предприятий и организаций, а также в крупных госбюджетных НИР. В учебном процессе кафедры принимают участие представители промышленности и науки Санкт-Петербурга.

В настоящее время кафедра прикладной экономики и маркетинга является одной из ведущих выпускающих кафедр Гуманитарного факультета СПбГУ ИТМО.

Игорь Алексеевич Зикратов
Вадим Юрьевич Петров

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В УПРАВЛЕНИИ

Учебное пособие

В авторской редакции

Дизайн

В.Ю.Петров

Верстка

В.Ю.Петров

Редакционно-издательский отдел Санкт-Петербургского государственного
университета информационных технологий, механики и оптики

Зав. РИО

Н.Ф. Гусарова

Лицензия ИД №

Подписано к печати _____

Заказ № _____

Тираж 50

Отпечатано на ризографе

Редакционно-издательский отдел
Санкт-Петербургского государственного
университета информационных технологий,
механики и оптики
197101, Санкт-Петербург, Кронверкский пр., 49

