МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

И.А. Зикратов, В.В. Косовцев, В.Ю. Петров

Разработка и стандартизация программных средств и информационных технологий.

Учебное пособие

Санкт-Петербург

2010

Зикратов И.А., Косовцев В.В., Петров В.Ю. Разработка и стандартизация программных средств и информационных технологий . Учебное пособие. - СПб: СПбГУ ИТМО, 2010. -91 с.

Учебное пособие преследует цель - практическое усвоение студентами лекционного материала по курсу «Разработка и стандартизация программных средств и информационных технологий» 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 № **15 от 19 октября 2010 г.**

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.

ОГЛАВЛЕНИЕ.

ВВЕДЕНИЕ

Пособие предполагает использование и изучение студентами методов проектирования программного обеспечения и средств стандартизации. При этом, материал продолжает обучение информационным технологиям и технологии визуального офисного программирования. Представленная для пользователей работа базируется на примерах разобранных при изучении таких дисциплин как «Информационные системы» и «Информационные технологии». Большинство вопросов и проблем, связанных с созданием простых программ, пользовательских интерфейсов, приложений для решения пользовательских задач, рассмотрены в пособиях «Информационные системы в экономике» и «Информационные технологии в управлении» [1,2]. Данная работа предполагает изучение более сложных вопросов проектирования программного обеспечения и средств стандартизации.

1. КЛАССЫ в ОФИСНОМ ПРОГРАММИРОВАНИИ

1. СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО КЛАССА.

Офисное программирование предполагает использование встроенного в Office языка Visual Basic for Application (VBA) для создания пользовательских приложений в среде Microsoft Office. VBA является подмножеством Visual Basic-а и представляет собой язык объектно-ориентированного программирования (ООП). Начиная с Office 2000 VBA использует отношения-наследования, которые являются одним из свойств языков ООП. Помимо этого VBA использует встраивание, которое является разновидностью наследования. Связано это с тем, что в Office уже существует огромное количество объектов, которые вложены друг в друга.

Неотъемлемой частью любого языка объектно-ориентированного программирования является понятие – ***класс***. В данной работе рассмотрен пример создания пользовательских классов и использование их в программе.

Как известно, класс должен иметь свойства, методы и над его объектами должны происходить события. В самом общем виде проект может включать множество классов, часть из которых является потомками, а часть родителями. Перед тем как приступать к программированию следует нарисовать схему. В ней, в самом общем виде, отображают классы, их связи, указывающие на потомков и родителей, свойства классов, методы и события.

В качестве примера, рассмотрим вариант создания проекта «Гаражный кооператив». Как вариант модель взаимодействия объектов этого проекта может включать следующие классы объектов:

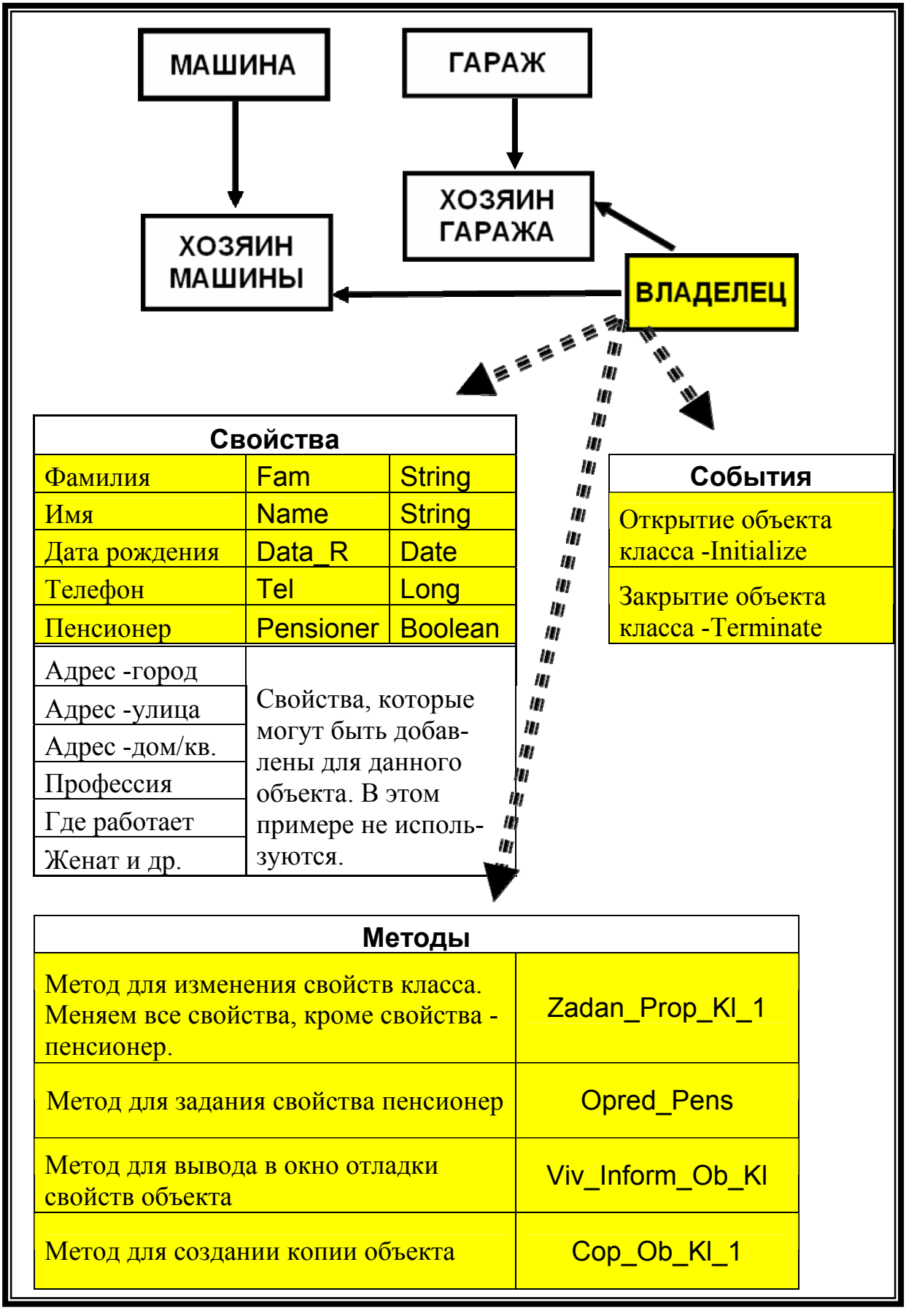
- машина,
- хозяин машины,
- гараж,
- хозяин гаража,
- владелец, человек, который может владеть машиной или гаражом. В принципе, он может не обладать ничем, то есть быть потенциальным владельцем, а может обладать и тем и другим.

Иерархия объектов для данного случая может быть представлена в виде схемы на рисунке 1.1. Каждый класс объектов имеет свои свойства, методы и определенным образом связан с объектами другого класса

На этом рисунке помимо модели объектов отображены свойства, методы и события для объекта Владелец. Все они в дальнейшем могут быть дополнены и изменены. Свойства и методы других классов на схеме не приведены. Их будем добавлять и расширять по мере усложнения модели проекта.

Задание №1

Создать проект и программу, позволяющих использовать один класс, из рассмотренных в схеме семейства классов, а именно – класс Владелец.



Свойства		
Фамилия	Fam	String
Имя	Name	String
Дата рождения	Data_R	Date
Телефон	Tel	Long
Пенсионер	Pensioner	Boolean

Адрес -город
Адрес -улица
Адрес -дом/кв.
Профессия
Где работает
Женат и др.

Свойства, которые могут быть добавлены для данного объекта. В этом примере не используются.

События
Открытие объекта класса -Initialize
Закрытие объекта класса -Terminate

Методы	
Метод для изменения свойств класса. Меняем все свойства, кроме свойства - пенсионер.	Zadan_Prop_Kl_1
Метод для задания свойства пенсионер	Opred_Pens
Метод для вывода в окно отладки свойств объекта	Viv_Inform_Ob_Kl
Метод для создании копии объекта	Cop_Ob_Kl_1

Рисунок 1.1. Схема проекта

Для того чтобы создать *класс* необходимо выполнить команду меню редактора VBA *Insert/Class Module*, после чего в окне свойств данного класса присвоить ему имя и в окне программного кода написать соответствующий текст, так как это сделано на рисунке 1.2.

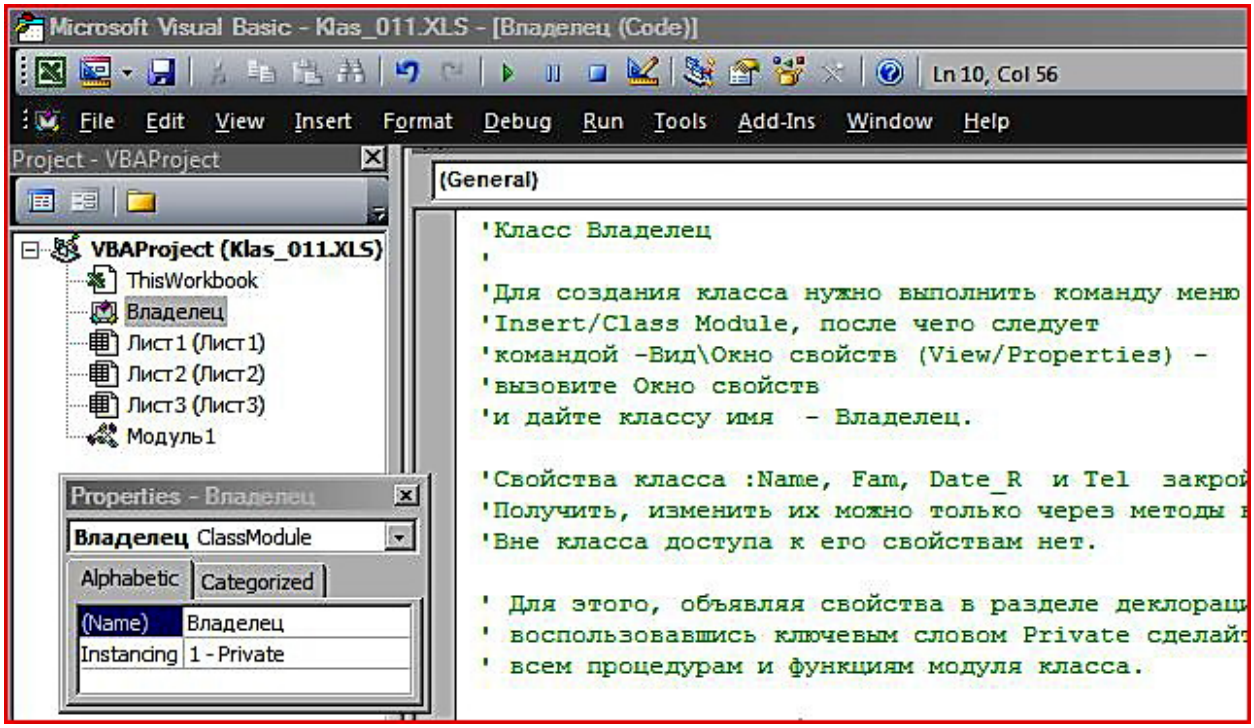


Рисунок 1.2. Создание класса Владелец

Полный текст программного кода модуля класса Владелец с подробными комментариями приведен ниже.

Задание №1 .Наберите текст программ, представленных в листинге для класса «Владелец». Внимательно прочитайте комментарии к инструкциям программы. Добейтесь того, чтобы в вашем программном обеспечении не было ошибок. Тестирование работоспособности данного проекта, будем производить после создания пользовательского приложения (см. п.2).

Задача по созданию текстов, приведенных ниже программ, значительно упрощается тем, что все тексты взяты из работающего проекта. Их можно перенести копированием в соответствующие модули и они должны заработать.

Поэтому основное, что необходимо сделать при изучении этого материала – это внимательно прочитать поясняющий текст и разобраться в нем.

Текст комментариев, набран курсивом для того, чтобы он отличался от текста инструкций программ. Перед копированием его можно сменить на обычный.

'Класс “Владелец”

```
'****************************************
'Для создания класса нужно выполнить команду меню
'Insert/Class Module, после чего следует
'командой -Вид\Окно свойств (View/Properties) -
'вызовите Окно свойств
'и дайте классу имя  - Владелец.

'Свойства класса :    Name,    Fam,    Date_R    и    Tel
'закройте от прямого доступа.
'Получить, изменить их можно только через методы класса.
'Вне класса доступа к его свойствам нет.

' Для этого, объявляя свойства в разделе деклараций модуля класса и
' воспользовавшись ключевым словом Private сделайте их доступными
' всем процедурам и функциям модуля класса.

' Тексты программ набранные ниже расположены в модуле класса.

'  Объявление переменных в разделе деклараций

Private Name As String  ' Это первое свойство класса и т.д.
Private Fam As String
Private Date_R As Date
Private Tel As Long

Private Pensioner As Boolean
      '   Это свойство будем только
      '   инициализировать при создании класса (Class_Initialize),
      '   и передавать при копировании методом   Cop_Ob_Kl_1.
      '   Изменять это свойство  методом Zadan_Prop_Kl_1  при
      '   определении или изменении всех остальных
      '   свойств  объекта не будем.
      '   Для этих целей  создадим свой отдельный метод  -  Opred_Pens
'
'  Открытые методы класса     “Владелец”
'-------------------------------------------------------------------------------

Public Sub Zadan_Prop_Kl_1(ByVal Pro_N As String,  _
    ByVal Prop_F As String,  ByVal Prop_D As Date, ByVal Prop_T As Long)
    '---------------------------------------------------------------------------
    ' 1-й метод класса Владелец (Задание свойств классу 1)
    ' Эта процедура через фактические параметры позволяет присвоить
    ' значения всем свойствам класса
    ' Она практически определяет состояние объекта.
```

```
' Передача параметров в процедуру происходит по значению.
'-------------------------------------------------------------------------------

    Name = Pro_N
    Fam = Prop_F
    Date_R = Prop_D
    Tel = Prop_T
 End Sub

Public Sub Cop_Ob_Kl_1(XXX As Владелец)
'-------------------------------------------------------------------------------
' 2-й метод класса Владелец (Копирование объекта класса 1).
' Этот метод ( процедура )  позволяет копировать значение всех
' полей (свойств) класса Владелец  (см.далее свойства класса).
' Метод создает 2-й объект (близнец первого) !
' Это НЕ ДВЕ ссылки на один объект !
' Обратите внимание, что в качестве формального параметра
' использована переменная объектного типа XXX,
' принадлежащая к  классу  Владелец.

' Для того, чтобы добраться до свойства класса, следует
'  написать его имя, поставить точку и затем имя свойства,
'  введенное при создании процедур для изменения свойств класса
'  (см.далее Свойства объектов класса).

    Name = XXX.Pr_Name
    Fam = XXX.Pr_Fam
    Date_R = XXX.Pr_Date_R
    Tel = XXX.Pr_Tel
End Sub


Public Sub Opred_Pens()
'-------------------------------------------------------------------------------
    ' 3-й метод класса.
    ' Использует вспомогательную закрытую функцию  Is_Pensioner.
    ' Метод присваивает свойству  Pensioner  значение True,
    ' если владелец -пенсионер, и False - если нет.

    Pensioner = Is_Pensioner
End Sub

Public Sub Viv_Inform_Ob_Kl()
'-------------------------------------------------------------------------------
' 4-й метод класса.
```

```
' Вывод информации об объекте класса в окно  отладки
   Dim b_str As String
      If Pensioner Then
            b_str = "пенсионер"
            Else
            b_str = "пенсионером не является"
      End If

      Debug.Print Name, "  ", Fam, "  - ", b_str
      Debug.Print " Его Дата рождения и телефон - ", Date_R, Tel, Chr(13)

End Sub

Private Function Is_Pensioner() As Boolean
'-------------------------------------------------------
' Внутренняя функция.
'=True, если Владелец - пенсионер  ,  False - если не пенсионер
'
'-------------------------------------------------------------------------------
   Dim Pol As Integer, Kol_let As Integer
   Pol = InputBox(""Гражданин  " & Fam & " , если вы инвалид введите" _
         "- 1, если женщина введите -2, мужчина -3", _
          "Информ. для определ-я свойства пенсионер", "2")
   Kol_let = DateDiff("Yyyy", Date_R, Now)
   If Pol = 1 Or (Pol = 2 And Kol_let > 55) Or (Pol = 3 And Kol_let > 60)
         Then
           Is_Pensioner = True
         Else
           Is_Pensioner = False
   End If
End Function

' Методы  обработки  событий  класса
' Встроенные, закрытые для внешнего пользования функции
' по обработке событий начала и конца работы с объектом класса
'=======================================================
```

<u>События объектов (</u><u>сделаем замечания</u>**<u>)</u>**

С каждым объектом класса всегда связаны два события:
- ***Initialize,*** возникающее при его открытии,
- ***Terminate,*** возникающее по окончанию работы с этим объектом класса.

При возникновении любого из них программа автоматически включит обработчик соответствующего события и выполнит, соответственно, один из закрытых методов (их называют конструкторы):

Private Sub Class_Initialize()
Private Sub Class_Terminate()

Для создания программного кода этих методов следует, находясь в *редакторе VBA*, в *модуле класса*, в *окне создания кода*, в *правом раскрывающемся* списке установить *«Class»*, а в *левом раскрывающемся списке* щелкнуть по имени соответствующего события *"Terminate"* так, как это представлено на рисунке 1.3.

Двойной щелчок мыши по имени события приводит к появлению первого и последнего оператора методов, и остается только заполнить тело процедуры необходимыми операторами.

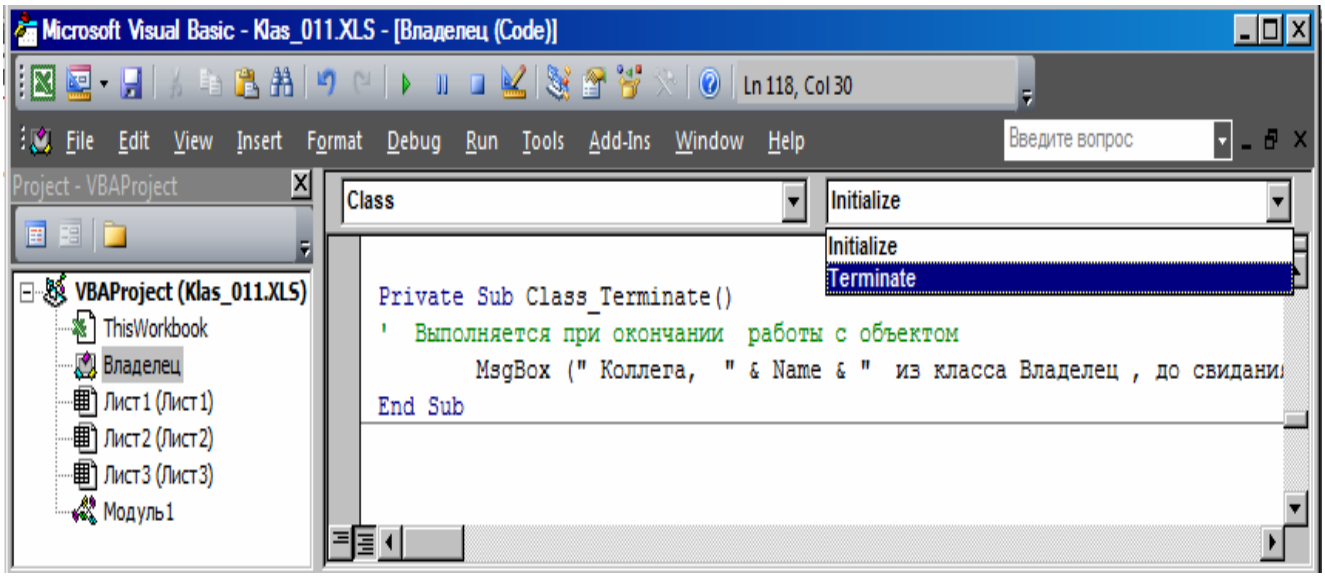


Рисунок 1.3. Создание заготовки для конструкторов класса

Продолжение программного кода модуля класса

```
Private Sub Class_Initialize()
'--------------------------------------------------------------------
' Выполняется при инициализации класса
' При первом обращении к любому объекту класса
    Name = "Илья"
    Fam = "Муромец"
    Date_R = #1/1/1800#
    Tel = 1234567
    Pensioner = False
End Sub
```

```
Private Sub Class_Terminate()
'--------------------------------------------------------------------
' Выполняется при окончании работы с объектом
      MsgBox (" Коллега,  " & Name & _
                  "  из класса Владелец , до свидания")
End Sub

'Свойства объектов класса
'=======================
```

Сделаем замечания.

Свойства объектов класса следует, в общем случае, закрывать. Это предотвращает возможность их *непосредственного* изменения. Для этого использовано ключевое слово *Private*. Но для того, чтобы все-таки иметь возможность добраться до свойств, изменить или получить их значения используют специальные методы (процедуры):

Get – получает значение свойства;

Let - изменяет значение свойства.

Заготовки для каждой такой пары методов строятся автоматически при выполнении команды меню VBA *Insert/Procedure* и установке в появившемся окне диалога *"Add Procedure*" имени методов в текстовом поле "*Name*". После этого следует включить радио кнопку "*Property*" в групповом окне "*Type*" и закрыть методы, используя групповое окно *"Scope"*. Все изменения произведенные Вами должны соответствовать данным, представленным на рисунке 1.3.

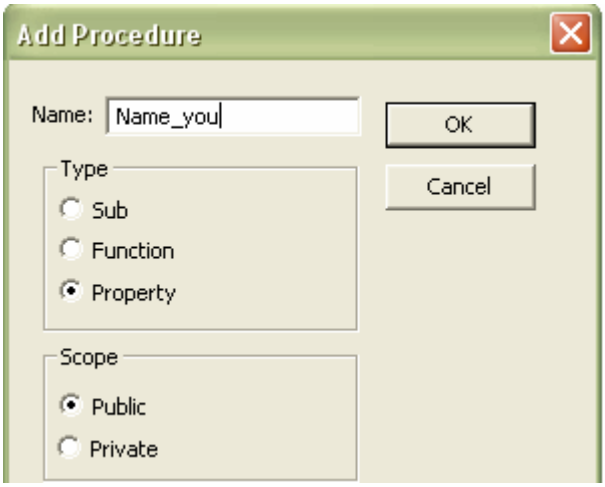


Рисунок 1.3. Окно для заготовки методов, определяющих свойства классов

Продолжение программного кода модуля класса.

```
                'Для изменения и получения значений 5-ти свойств:
                ' - Name,   Fam,  Date_R,  Tel, Pensioner.
                ' используют четыре  пары специальных методов - Get и Let
' Ниже - это процедуры (методы)
' для изменения и получения значений у свойства класса.
```

```
' Заготовки для них вставляются в модуль класса командой меню
' Insert/Procedure...  /_Property
' При этом получаем сразу две заготовки процедур
'  - для изменения и для получения свойств класса

'=========================-
 ' В программах пользователя, расположенных в модулях проекта,
 ' например, в Модуль1 (не модулях классов),
 ' для того, чтобы изменить какое-то свойство у объекта следует
 ' использовать идентификаторы, введенные при создании методов,
 ' для присвоения и получения значений свойств объектов. См.ниже.
 ' Например, для изменения свойства Name у объекта Петров,
 ' следует написать       Петров.Pr_Name

Public Property Get Pr_Name() As String
   Pr_Name = Name
End Property

Public Property Let Pr_Name(ByVal vNewValue As String)
   Name = vNewValue
End Property                    'Pr_Name

Public Property Get Pr_Fam() As String
   Pr_Fam = Fam
End Property

Public Property Let Pr_Fam(ByVal vNewValue As String)
   Fam = vNewValue
End Property                    'Pr_Fam

Public Property Get Pr_Date_R() As Variant
   Pr_Date_R = Date_R
End Property

Public Property Let Pr_Date_R(ByVal vNewValue As Variant)
   Date_R = vNewValue
End Property                    'Pr_Date_R

Public Property Get Pr_Tel() As Variant
   Pr_Tel = Tel
End Property

Public Property Let Pr_Tel(ByVal vNewValue As Variant)
   Tel = vNewValue
End Property                    'Pr_Tel
```

```
' Если не использовать приведенные ниже два метода,
' то объявленная в разделе деклараций переменная Pensioner
' будет все равно видна во всех процедурах модуля класса.
' Она является глобальной переменной.
' Но в этом случае переменная будет иметь одно значение
' для всех объектов класса и свойством объекта являться не будет.
' Это касается и всех остальных свойств.

Public Property Get Pr_Pensioner() As Boolean
   Pr_Pensioner = Pensioner
End Property

Public Property Let Pr_Pensioner(ByVal vNewValue As Boolean)
   Pensioner = rvNewValue
End Property                'Pr_Pensioner
```

2. ИСПОЛЬЗОВАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО КЛАССА

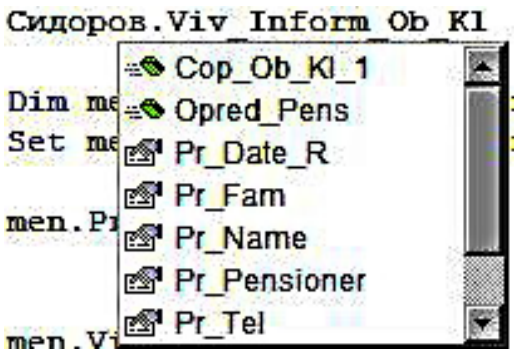
Тест программы, использующей объекты созданного класса "Владелец", представлен ниже. Здесь же показан вид окна *Immediate* после завершения работы этой программы.

```
‘ Листинг Программы
' Набранные тексты программ, используют новые объекты
' класса Владелец  и располагаются в обычных модулях
' НЕ в модулях класса.
' При завершении работы с существующими объектами
' будет выполнен метод   Class_Terminate

Public Sub UserClass()
'*****************************
Dim Петров As New Владелец
Dim Иванова As New Владелец
Dim Сидоров As New Владелец
          ' Первое же обращение к объекту приводит к его инициализации
          ' Вывод свойств объекта Петров  в окно отладки после
          ' инициализации
   Debug.Print "  1  "; Петров.Pr_Name; "  "; Петров.Pr_Fam; "  "; _
```

```
Петров.Pr_Date_R; Петров.Pr_Tel; "  "; Петров.Pr_Pensioner

    ' Изменение свойств у объекта  Петров
    ' и задание свойств объекту Иванова
Петров.Zadan_Prop_Kl_1 Prop_F:="Майоров", Pro_N:="Петр", _
                       Prop_D:=#1/23/1930#, Prop_T:=2345678
Иванова.Zadan_Prop_Kl_1 Prop_F:="Ивановская", Pro_N:="Аня", _
                        Prop_D:=#1/1/1968#, Prop_T:=7654321

    ' Определяем, является ли объект Петров пенсионером
    ' по возрасту или инвалидности и изменяем значение
    ' свойства (переменной) - Pensioner.
Петров.Opred_Pens

    ' Вывод свойств объекта Петров и Иванова
    ' методом   Viv_Inform_Ob_Kl
Петров.Viv_Inform_Ob_Kl
Иванова.Viv_Inform_Ob_Kl

    ' Изменяем значение свойства (Фамилии)  у Ивановой.
Иванова.Pr_Fam = Петров.Pr_Fam & "а"

Debug.Print "  2  ", Петров.Pr_Name
Debug.Print "  3  ", Иванова.Pr_Fam

    ' Копирование свойств объекта Петров в свойства объекта
    ' Сидоров. При этом создается 3-й объект - Сидоров.
Сидоров.Cop_Ob_Kl_1 Петров

    ' Определяем, является ли объект Сидоров пенсионером
    ' по возрасту или инвалидностии изменяем значение свойства
    ' (переменной) - Pensioner.
Сидоров.Opred_Pens

    ' Вывод свойств объекта Сидоров методом   Viv_Inform_Ob_Kl
Сидоров.Viv_Inform_Ob_Kl

Dim men As Object  ' Объявляем объектную переменную
Set men = Сидоров
    ' Теперь переменная тел ссылается на объект Сидоров.
    ' Она может использовать все свойства и методы этого
    ' объекта.
men.Pr_Name = "Гавриил"
    ' Изменяем свойство Name объекта Сидоров,
    ' используя объектную переменную тел
```

```
men.Viv_Inform_Ob_Kl
        ' Выводим значение свойств объекта  Сидоров,  используя
        ' объектную переменную men
Сидоров.Viv_Inform_Ob_Kl
        ' используя объектную переменную men и сам объект
Set men = Nothing
        '  освобождаем ссылку на объект, но объект не пропадает.
Set Сидоров = Nothing    ' Ликвидируем сам объект
End Sub
```

При наборе текста программы, вводя очередную инструкцию, обратите внимание на следующее. При вводе точки следом за именем объекта созданного класса, машина сама предложит продолжить инструкцию именем открытого метода (таких в модуле класса существует всего 4) или ввести в текст программы одно из созданных свойств объекта (таких -5), как это показано на рисунке слева.



При выполнении программы последовательно должны появляться следующие окна.

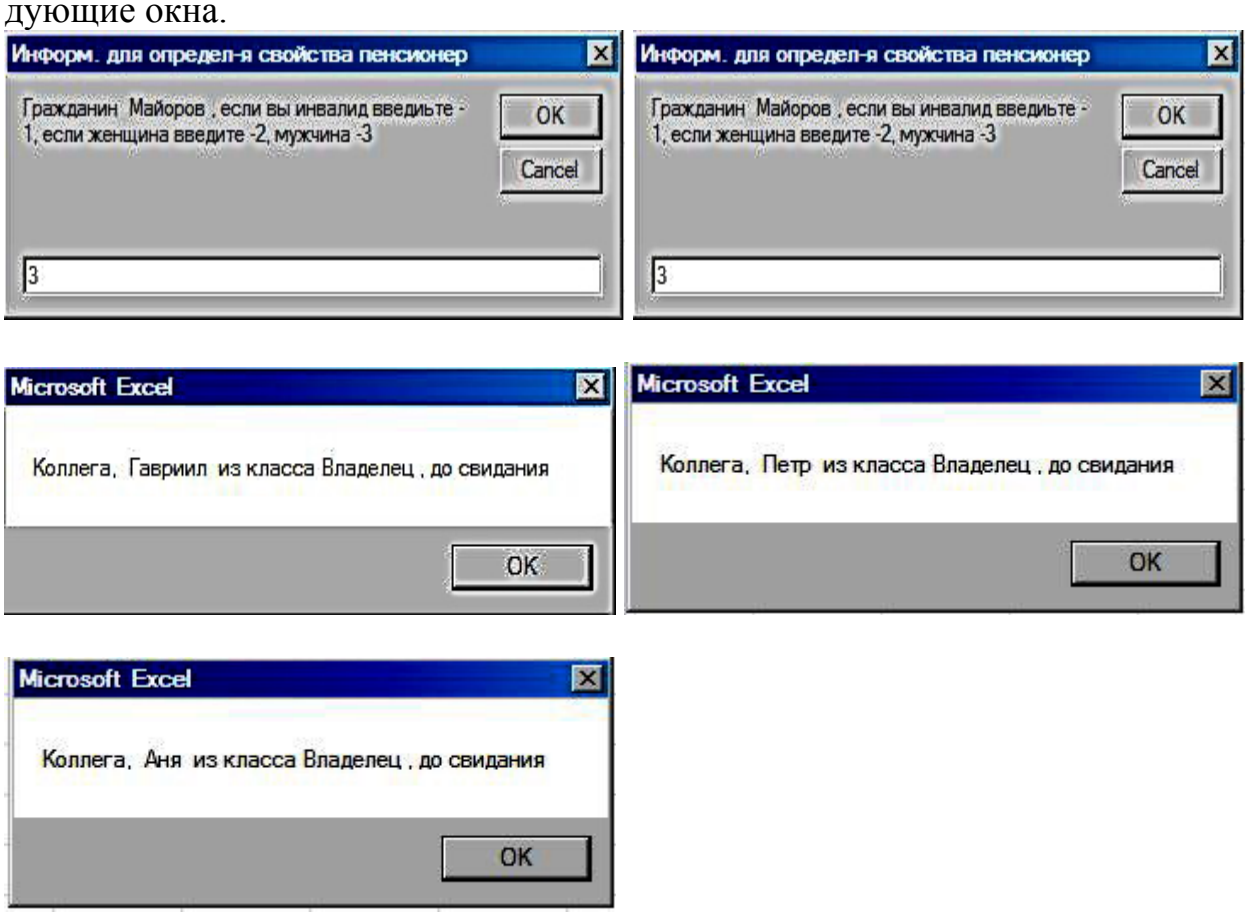


Рисунок 1.6. Рабочие окна, появляющиеся при выполнении пользовательской программы

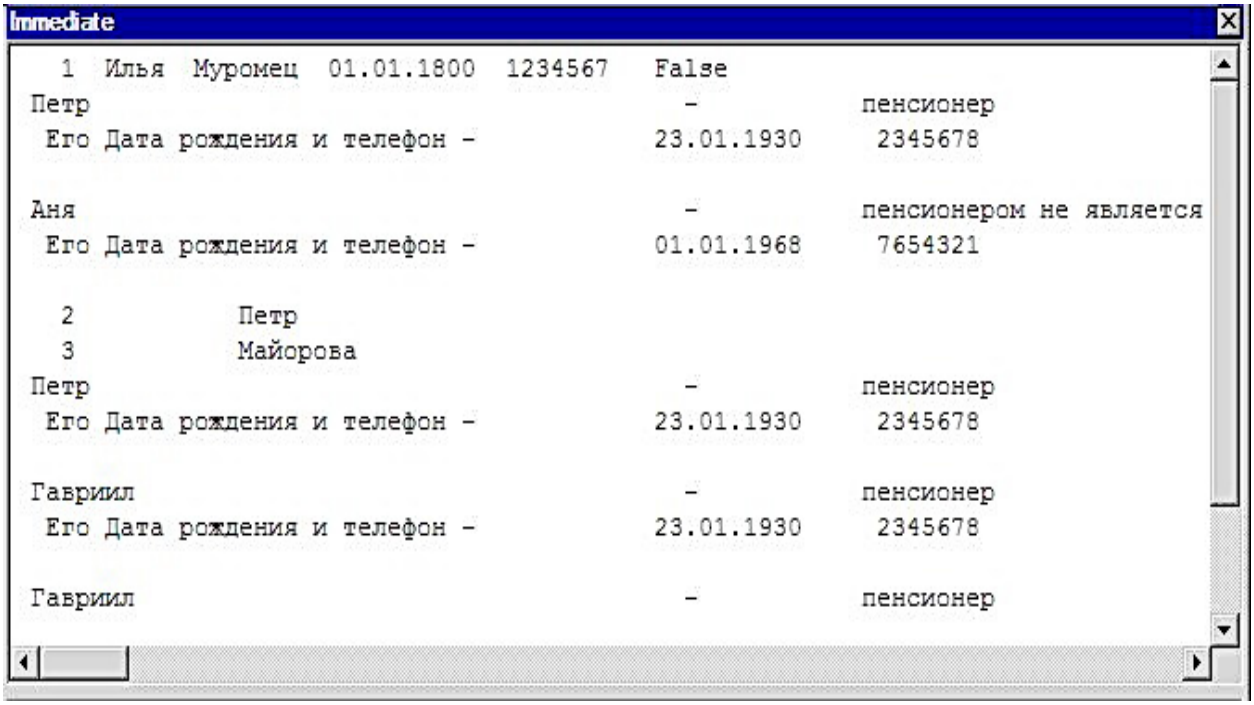


Рисунок 1.7. Вид окна Immediate после выполнения пользовательской программы

Задание №2.

1. Наберите приведенные тексты программ.
2. Составьте алгоритм пользовательской программы.
3. Разберитесь в принципах создания классов, свойств их методов, используя пояснения и соответствующие справки.
4. Убедитесь в работоспособности созданных программных средств.
5. Выясните, что делает каждый оператор основной программы.
6. Сделайте эксперименты. Поменяйте у свойств и (или) методов Public на Private, добавьте свои свойства и методы. Сделайте выводы.

3. СОЗДАНИЕ ПРОЕКТА ДЛЯ СЕМЕЙСТВА КЛАССОВ

Модель семейства классов проекта «Гаражный кооператив» представлена на рисунке 1.1. Создадим проект, который будет включать три класса: Владелец, Гараж и Владелец гаража. Сначала создадим родительский класс – Гараж, а потом класс - Владелец Гаража, который будет потомком двух, созданных ранее родительских классов.

3.1. Проектирование второго класса проекта – Гараж

Создание программного модуля для класса Владелец и его тестирование подробно рассмотрены выше. Аналогичным образом спроектируем класс Гараж и программу для его тестирования.

Вариант листинга программ с комментариями для модуля класса Гараж и пользовательское приложение, использующее этот модуль, приведены ниже. Ввиду того, что многое из вновь созданного, выполнено аналогично рассмотренному при проектировании класса Владелец, комментарии к программному коду приведены только в тех случаях, когда это необходимо.

'Класс “Гараж”

```
'*****************************************

Private G_Name As String        ' Это первое свойство класса (КАС-17)
Private G_Numbe As Integer      ' Номер гаража (613)
Private G_Material As String    ' Бетон, железо, дерево
Private G_Date_P As Date        ' Год постройки -1980
Private G_Comfort As Integer    ' Комфорт :  1-обшит деревом,
                                ' 2- есть деревянный пол, 4-есть яма,
                                '  3- (1+2),  5-(1+4),  6-(2+4), 7- (1+2+4)
Private G_Cena As Long          ' Цена 60 000

'ОТКРЫТЫЕ МЕТОДЫ КЛАССА     Гараж
'-------------------------------------------------------------------------------

Public Sub Zadan_Prop_Garag(ByVal Prop_G_Na As String, ByVal Prop_G_Nu
            As Integer, ByVal Prop_G_M As String, ByVal Prop_G_D As Date, _
           ByVal Prop_G_C As Integer, ByVal Prop_G_Ce As Long)
'-------------------------------------------------------------------------------
' 1-й метод класса Гараж ( Задание свойств классу )

   G_Name = Prop_G_Na
   G_Numbe = Prop_G_Nu
   G_Material = Prop_G_M
   G_Date_P = Prop_G_D
   G_Comfort = Prop_G_C
   G_Cena = Prop_G_Ce
 End Sub

Public Sub Cop_Ob_Kl_1(XXXX As Гараж)
'-------------------------------------------------------------------------------
' 2-й метод класса Гараж ( Копирование объекта класса Гараж ).

   G_Name = XXXX.Pr_G_Name
```

```
    G_Numbe = XXXX.Pr_G_Numbe
    G_Material = XXXX.Pr_G_Material
    G_Date_P = XXXX.Pr_G_Date_P
    G_Comforf = XXXX.Pr_G_Comfort
    G_Cena = XXXX.Pr_G_Cena
End Sub


Public Sub Viv_Inform_Ob_Kl_G()
'  3-й метод класса.
' Вывод информации об объекте класса в окно  отладки
'-------------------------------------------------------------------------------
    Dim b_str As String

        Select Case G_Comfort
        Case 1
            b_str = "Гараж обшит деревом"
        Case 2
            b_str = "Гараж  имеет деревянный пол"
        Case 4
            b_str = "Гараж имеет яму"
        Case 3
            b_str = "Гараж обшит деревом и имеет деревянный пол"
        Case 5
            b_str = "Гараж обшит деревом и имеет яму"
        Case 6
            b_str = "Гараж имет яму и деревянный пол"
        Case 7
            b_str = "Гараж обшит деревом, имеет деревянный пол и яму"
        Case Else
            b_str = "Гараж дрянной"
        End Select

        Debug.Print "Характеристики гаража ", G_Name, "Номер  ", G_Numbe
        Debug.Print "Материал из которого сделан гараж - ", G_Material
        Debug.Print b_str
        Debug.Print "Дата постройки - ", G_Date_P, "Цена - ", G_Cena, Chr(13)
End Sub

' МЕТОДЫ   ОБРАБОТКИ   СОБЫТИЙ   КЛАССА

Private Sub Class_Initialize()
'--------------------------------------------------------------------
    G_Name = "КАС-17"
    G_Numbe = 613
```

```
   G_Material = "Бетон"
   G_Date_P = #1/1/2000#
   G_Comfort = 7
   G_Cena = 100000
End Sub

Private Sub Class_Terminate()
'--------------------------------------------------------------------
' Выполняется при окончании  работы с объектом
      MsgBox (" Гараж с номером - " & G_Numbe & "   из - " & _
                  G_Name & " закрыт")
End Sub

'МЕТОДЫ для получения и задания СВОЙСТВА КЛАССА ГАРАЖ
'--------------------------------------------------------------------

Public Property Get Pr_G_Name() As Variant
   Pr_G_Name = G_Name
End Property

Public Property Let Pr_G_Name(ByVal vNewValue As Variant)
   G_Name = vNewValue
End Property                              'Pr_G_Name

Public Property Get Pr_G_Numbe() As Variant
   Pr_G_Numbe = G_Numbe
End Property

Public Property Let Pr_G_Numbe(ByVal vNewValue As Variant)
   G_Numbe = vNewValue
End Property                              'Pr_G_Numbe

Public Property Get Pr_G_Material() As Variant
   Pr_G_Material = G_Material
End Property

Public Property Let Pr_G_Material(ByVal vNewValue As Variant)
   G_Material = vNewValue
End Property                              'Pr_G_Material

Public Property Get Pr_G_Date_P() As Variant
   Pr_G_Date_P = G_Date_P
End Property

Public Property Let Pr_G_Date_P(ByVal vNewValue As Variant)
```

```
   G_Date_P = vNewValue
End Property                          'Pr_G_Date_P

Public Property Get Pr_G_Comfort() As Variant
   Pr_G_Comfort = G_Comfort
End Property

Public Property Let Pr_G_Comfort(ByVal vNewValue As Variant)
   G_Comfort = vNewValue
End Property                          'Pr_G_Comforf

Public Property Get Pr_G_Cena() As Variant
   Pr_G_Cena = G_Cena
End Property

Public Property Let Pr_G_Cena(ByVal vNewValue As Variant)
   G_Cena = vNewValue
End Property                          'Pr_G_Cena
```

Листинг программы тестирования для данного модуля класса может иметь следующий вид.

```
Public Sub UserClass_02()
‘*****************************
Dim Гараж_01 As New Гараж
Dim Гараж_02 As New Гараж
Dim Гараж_03 As New Гараж

  ' Первое же обращение к объекту приводит к его инициализации
  ' Вывод свойств объекта Гараж_01  в окно отладки после инициализации

Debug.Print " Характеристики объекта после инициализации ", Chr(13); _
   Гараж_01.Pr_G_Name; "  "; Гараж_01.Pr_G_Numbe; "  "; _
   Гараж_01.Pr_G_Material; "  "; Гараж_01.Pr_G_Date_P; "  "; _
   Гараж_01.Pr_G_Comfort; "  "; Гараж_01.Pr_G_Cena; , Chr(13)

  ' Изменение свойств у объекта  Гараж_01
  ' и задание свойств объекту Гараж_02

Гараж_01.Zadan_Prop_Garag Prop_G_Na:="Кооператив 13",  _
     Prop_G_Nu:=1313, Prop_G_M:="Железо", _
      Prop_G_D:=#2/13/2005#, Prop_G_C:=2, Prop_G_Ce:=50000
Гараж_02.Zadan_Prop_Garag Prop_G_Na:="КАС-17",  _
```

```
    Prop_G_Nu:=133, Prop_G_M:="Дерево", _
    Prop_G_D:=#12/13/1999#, Prop_G_C:=0, Prop_G_Ce:=70000

    ' Копирование свойств объекта Петров в свойства объекта Сидоров.
    Гараж_03.Cop_Ob_Kl_1 Гараж_02
    Гараж_03.Pr_G_Numbe = 134

    ' Вывод свойств
    Гараж_01.Viv_Inform_Ob_Kl_G
    Гараж_02.Viv_Inform_Ob_Kl_G
    Гараж_03.Viv_Inform_Ob_Kl_G
End Sub
```

В результате запуска программы тестирования Public Sub UserClass_02() будет получен результат, представленный на рисунке

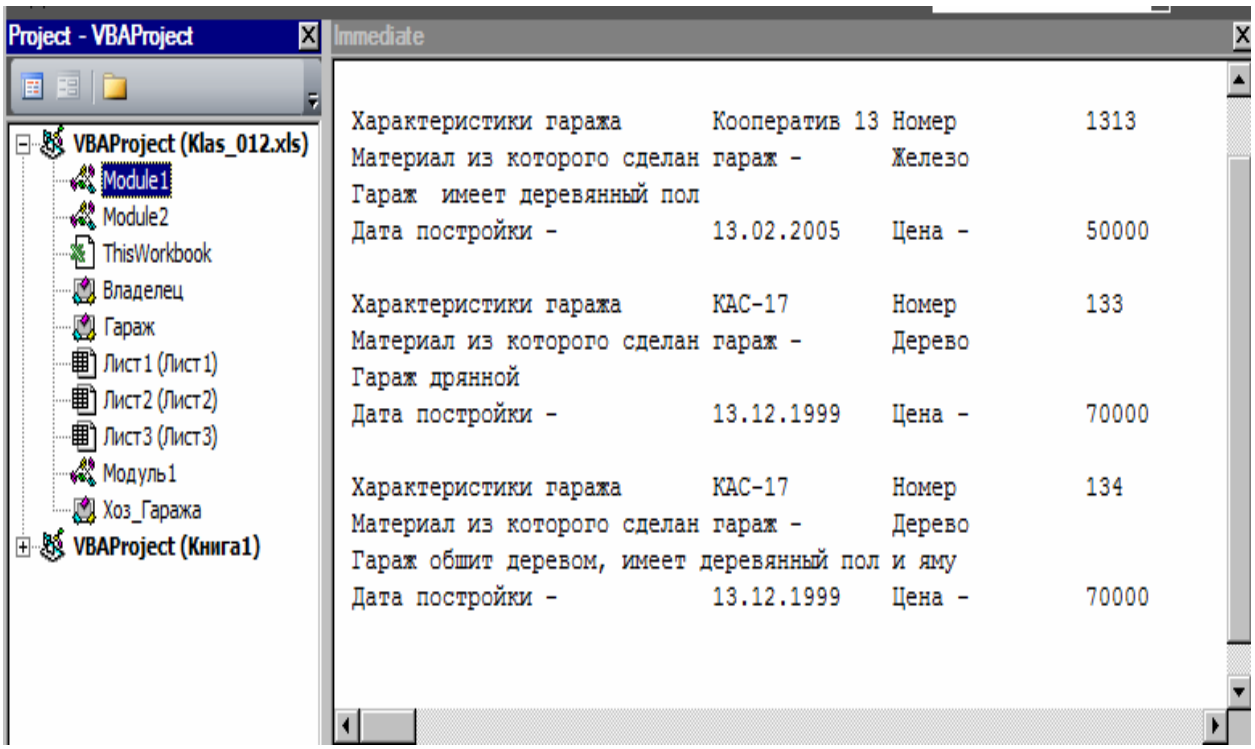


Рисунок 1.4. Результат тестирования

Задание №3.

1. Наберите приведенные тексты программ.
2. Составьте алгоритм пользовательской программы.
3. Убедитесь в работоспособности созданных программных средств.
4. Выясните, что делает каждый оператор основной программы.

3.2. Проектирование проекта, использующего классы с наследованием

Следует признать, что работа с классами, их свойствами и методами достаточно сложна и требует от пользователя проведения экспериментов с программами. Эти исследования позволят выяснить суть происходящего, взаимоотношения классов и их атрибутов, и, в конечном счете, получить верный и предсказуемый результат. Другого пути для того, чтобы научиться программировать, а тем более, работать с классами - нет.

Начиная с Office 2000, в работе с классами появилось понятие - наследование интерфейса. Под интерфейсом всегда понимали средство общения. Такое общение пользователя с программой и ее объектами создается при помощи открытых свойств и методов класса. То есть при наследовании класс-потомок (порождаемый класс) наследует (ему будут доступны) свойства и методы родительского класса, описанные как Public – открытые.

Для того, чтобы порождаемый класс наследовал интерфейс родительского необходимо при декларировании класса записать имя родительского класса с ключевым словом ***Implements***.

'Класс “Хоз_Гаража”

```
'*****************************************
Implements Владелец
Implements Гараж

'СВОЙСТВА КЛАССА     Хоз_Гаража

Public HG_Vladel As Владелец
Public HG_Garag As Гараж


'МЕТОДЫ КЛАССА     Хоз_Гаража

Private Sub Class_Initialize()
'--------------------------------------------------------------------
   Set HG_Vladel = New Владелец
   Set HG_Garag = New Гараж
End Sub

Public Sub Zadan_Prop_Hoz_Garag(ByVal Prop_HG_Na As String,  _
              ByVal Prop_HG_Nu As Integer, ByVal Prop_HG_M As String, _
              ByVal Prop_HG_D As Date, ByVal Prop_HG_C As Integer,  _
              ByVal Prop_HG_Ce As Long)
'-----------------------------------------------------------------------------
```

```
' 1-й метод класса Хоз_Гаража (Задание свойств классу Хоз_Гаража)

'Свойства можно задать, используя приведенные шесть инструкций
        'HG_Garag.Pr_G_Name = Prop_HG_Na
        'HG_Garag.Pr_G_Numbe = Prop_HG_Nu
        'HG_Garag.Pr_G_Material = Prop_HG_M
        'HG_Garag.Pr_G_Date_P = Prop_HG_D
        'HG_Garag.Pr_G_Comfort = Prop_HG_C
        'HG_Garag.Pr_G_Cena = Prop_HG_Ce

 '  Свойства можно также задать, используя метод
'    объекта ГАРАЖ - Zadan_Prop_Garag

 HG_Garag.Zadan_Prop_Garag Prop_HG_Na, Prop_HG_Nu, Prop_HG_M,
Prop_HG_D, Prop_HG_C, Prop_HG_Ce

 End Sub


Public Sub Viv_Inform_Ob_Kl_HG()
'  3-й метод класса.
' Вывод информации об объекте класса Хоз_Гараж в окно  отладки
'-------------------------------------------------------------------------------
  HG_Garag.Viv_Inform_Ob_Kl_G   ' Вывод информации об объекте Гараж
                        ' используем метод  Viv_Inform_Ob_Kl_G объекта Гараж
  HG_Vladel.Viv_Inform_Ob_Kl    ' Вывод информации об объекте Владелец
End Sub

Public Sub Opredel_Cena_Gar()
'  4-й метод класса.
' Расчет стоимости гаража на текущую дату и вывод ее
'-------------------------------------------------------------------------------
  Dim b_Cen As Long, Ist_Cena As Long
  Dim b_dat As Date, b_int As Integer

  b_Cen = CLng(HG_Garag.Pr_G_Cena)
  b_dat = HG_Garag.Pr_G_Date_P
  b_int = DateDiff("YYYY", b_dat, Now)
  Ist_Cena = CLng(b_Cen * (1 - b_int * 0.05))

    Debug.Print "Истинная цена для Гаража номер - " & _
                HG_Garag.Pr_G_Numbe &   Chr(13) &   "В объединении  - " & _
                HG_Garag.Pr_G_Name & " составляет - " & Ist_Cena
 End Sub
```

```
'МЕТОДЫ для получения и задания свойств класа ХОЗЯИН ГАРАЖА

Public Property Get Pr_HG_Name() As Variant
   Pr_HG_Name = HG_Garag.Pr_G_Name
End Property

Public Property Let Pr_HG_Name(ByVal vNewValue As Variant)
   HG_Garag.Pr_G_Name = vNewValue
End Property                          'HG_Garag.Pr_G_Name

Public Property Get Pr_HG_Numbe() As Variant
   Pr_HG_Numbe = HG_Garag.Pr_G_Numbe
End Property

Public Property Let Pr_HG_Numbe(ByVal vNewValue As Variant)
   HG_Garag.Pr_G_Numbe = vNewValue
End Property                          'HG_Garag.Pr_G_Numbe

Public Property Get Pr_HG_Material() As Variant
   Pr_HG_Material = HG_Garag.Pr_G_Material
End Property

Public Property Let Pr_HG_Material(ByVal vNewValue As Variant)
   HG_Garag.Pr_G_Material = vNewValue
End Property                          'HG_Garag.G_Material

Public Property Get Pr_HG_Date_P() As Variant
   Pr_HG_Date_P = HG_Garag.Pr_G_Date_P
End Property

Public Property Let Pr_HG_Date_P(ByVal vNewValue As Variant)
   HG_Garag.Pr_G_Date_P = vNewValue
End Property                          'HG_Garag.G_Date_P

Public Property Get Pr_HG_Comfort() As Variant
   Pr_HG_Comfort = HG_Garag.Pr_G_Comfort
End Property

Public Property Let Pr_HG_Comfort(ByVal vNewValue As Variant)
   HG_Garag.Pr_G_Comfort = vNewValue
End Property                          'HG_Garag.G_Comforf

Public Property Get Pr_HG_Cena() As Variant
```

```
   Pr_HG_Cena = HG_Garag.Pr_G_Cena
End Property

Public Property Let Pr_HG_Cena(ByVal vNewValue As Variant)
   HG_Garag.Pr_G_Cena = vNewValue
End Property                              'HG_Garag.G_Cena
'=============================================================
Public Property Get Pr_XG_Name() As String
   Pr_XG_Name = HG_Vladel.Pr_Name
End Property

Public Property Let Pr_XG_Name(ByVal vNewValue As String)
   HG_Vladel.Pr_Name = vNewValue
End Property                        'Pr_XG_Name

Public Property Get Pr_XG_Fam() As String
   Pr_XG_Fam = HG_Vladel.Pr_Fam
End Property

Public Property Let Pr_XG_Fam(ByVal vNewValue As String)
   HG_Vladel.Pr_Fam = vNewValue
End Property                        'Pr_XG_Fam

Public Property Get Pr_XG_Date_R() As Variant
   Pr_XG_Date_R = HG_Vladel.Pr_Date_R
End Property

Public Property Let Pr_XG_Date_R(ByVal vNewValue As Variant)
   HG_Vladel.Pr_Date_R = vNewValue
End Property                        'Pr_XG_Date_R
Public Property Get Pr_XG_Tel() As Variant
   Pr_XG_Tel = HG_Vladel.Pr_Tel
End Property

Public Property Let Pr_XG_Tel(ByVal vNewValue As Variant)
   HG_Vladel.Pr_Tel = vNewValue
End Property                        'Pr_XG_Tel
```

Программа тестирования класса Хоз_Гаража

```
'*****************************************
Public Гараж_01 As New Гараж
Dim Влад_01 As New Владелец
Dim Хоз_Гар_01 As New Хоз_Гаража
```

```
Public Sub UserClass_Вл_Гар()
 '-----------------------------------------------
   ' Первое же обращение к объекту приводит к его инициализации
    Debug.Print "ВЛАДЕЛЕЦ": Влад_01.Viv_Inform_Ob_Kl
    Debug.Print "ГАРАЖ": Гараж_01.Viv_Inform_Ob_Kl_G
    Debug.Print "ХОЗ_ГАРАЖА": Хоз_Гар_01.Viv_Inform_Ob_Kl_HG

    ' Задаем значения свойствам объекта Хоз_Гар_01, из класса Хоз_Гаража
    Хоз_Гар_01.Zadan_Prop_Hoz_Garag Prop_HG_Na:="Кооператив 22", _
                Prop_HG_Nu:=666, Prop_HG_M:="Железо", _
                Prop_HG_D:=#2/9/2009#, Prop_HG_C:=6, Prop_HG_Ce:=80000

    ' Меняем фамилию у объекта Влад_01 из класса Владелец.
    ' Фамилия у объекта Хоз_Гар_01 из класса  Хоз_Гаража осталась прежней
    Влад_01.Pr_Fam = "Петров_01"
    Debug.Print "=  Гараж = ":  Гараж_01.Viv_Inform_Ob_Kl_G
    Debug.Print "=  Владелец  = ": Влад_01.Viv_Inform_Ob_Kl
    Debug.Print "=  Хозяин_Гаража  = ": Хоз_Гар_01.Viv_Inform_Ob_Kl_HG

    ' Меняем фам-ю у хозяина гаража – об. Хоз_Гар_01 из класса Хоз_Гаража.
    ' Фамилия у объекта Влад_01 из класса   Владелец осталась прежней

    Debug.Print "=   КАС - 15  + Иванов_01 = "
    Хоз_Гар_01.Pr_XG_Fam = "Иванов_01"
    Хоз_Гар_01.Pr_HG_Name = "КАС-15"
       Гараж_01.Viv_Inform_Ob_Kl_G
       Хоз_Гар_01.Viv_Inform_Ob_Kl_HG

    ' Определяем цену гаража с учетом амортизации
    Хоз_Гар_01.Opredel_Cena_Gar
    Debug.Print "Владельцем гаража являтся гр." & Хоз_Гар_01.Pr_XG_Fam

    End Sub
```

Задание №4.

1. Наберите приведенные тексты программ и убедитесь в работоспособности созданных программных средств.
2. Сделайте эксперименты. Поменяйте у свойств и (или) методов Public на Private, добавьте свои свойства и методы. Сделайте выводы.

3. Попробуйте создать проект, схема которого представлена на рисунке 1.1, либо какой-то свой, использующий механизм наследования.

2. ДИСКОВЫЕ ФАЙЛЫ (ФАЙЛОВЫЙ ВВОД И ВЫВОД).

1. ОСНОВНЫЕ ПОЛОЖЕНИЯ

Помимо файлов известных форматов, которые определяются использованием того или иного пакета –Word, Excel и т.д., существуют файлы, являющиеся внешними по отношению к используемому приложению. Они хранятся самостоятельно на диске и к ним можно обращаться из любого приложения Office, и, более того, с ними можно работать при программировании на VB, Delphi, C++ и других языках программирования.

Дисковые файлы делят на три группы:

Файлы последовательного доступа. Такие файлы представляют последовательность текстовых строк разной длины. Признаки конца строк являются разделителями элементов файла. Специальных средств поиска для них нет. Для того чтобы получить данные из файла нужно открыть его и прочесть последовательно все данные до нужного места.

Открыть файл последовательного доступа можно в одном из трех режимов. Действия, происходящие с такими файлами при этом, показаны на рисунке 2.1.

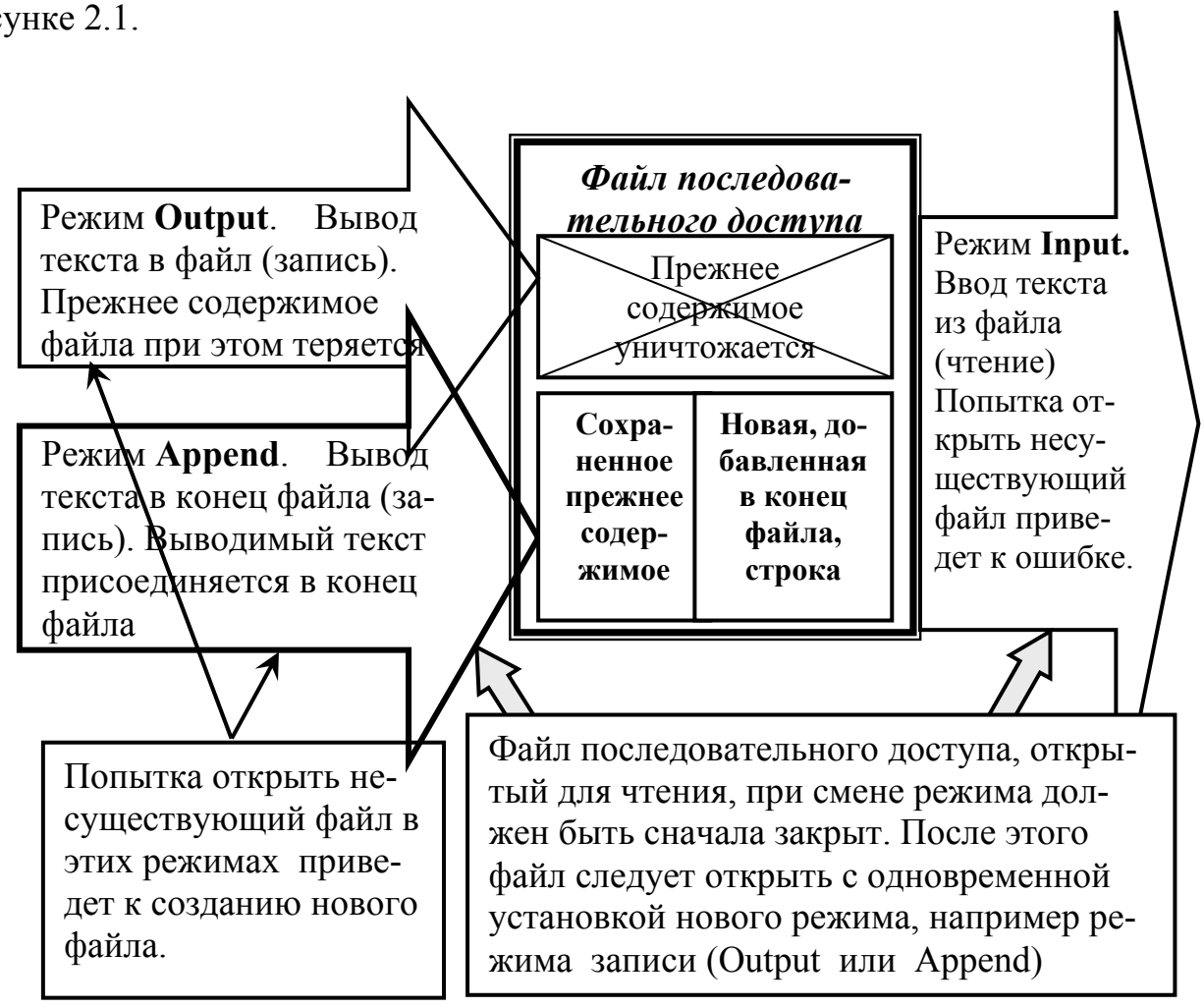


Рисунок 2.1. Режимы работы с фалами последовательного доступа

Файлы произвольного доступа. Такие файлы напоминают базу данных. Они состоят из записей одинакового размера, а каждая запись состоит из полей, в которых и содержатся данные. Обычно для таких файлов используют данные пользовательского типа.

```
Type Gragdanin
        Name as String * 40
        DatRog as Date
End Type
```

В таких файлах поиск данных происходит быстро. Возможен прямой доступ к элементу по его номеру. Серьезным ограничением является требование постоянства длины элементов файла.

Открытие файла произвольного доступа происходит с помощью ключевого слова, определяющего единственно возможный режим - ***Random.*** Причем при этом необходимо указать длину одной записи в байтах (не более 32767).

Файлы бинарного доступа (бинарные файлы). Это файлы с побайтным доступом. В принципе, это такие же как и файлы последовательного доступа , но информация в них не организована в строки. Во время одной операции может быть записано или прочитано любое число байтов.

Открытие бинарного файла возможно в двух режимах:

- ***Binary*** – двоичный доступ (по номеру байта)
- ***Random*** – произвольный доступ. В отличие от файлов произвольного доступа запись (блок байтов) не имеет строго заданного размера.

2. ОТКРЫТИЕ, ЗАКРЫТИЕ И УДАЛЕНИЕ ДИСКОВЫХ ФАЙЛОВ

Процессы закрытия, открытия и удаления для всех дисковых файлов являются общими.

При этом важным моментом является понятие ***Файлового числа*** (идентификатора файла или дескриптора). *Аргумент файловое число* это целочисленное выражение (числа от 1 до 511), которое необходимо присвоить файлу. Обычно первому открытому файлу присваивают 1, второму 2 и т.д. После закрытия файла освободившееся число может использоваться повторно. Для того, чтобы не вести счет открытым файлам допустимое файловое число всегда можно получить, воспользовавшись функцией ***FreeFile***.

Открытие файлов. Синтаксис инструкции для открытия файлов имеет вид:

Open *полное_имя_файла* ***For*** *режим* ***As*** *[#]файловое_число*
*[**Len**=длина_записи_файла]*

Длина указывается для открытия файла с произвольным доступом.
Если длина файла записи не известна, то ее можно получить с помощью функции *Len*

Закрытие файлов. Синтаксис инструкции:

Close [#]файловое_число

Если не указать файловое число, то будут закрыты все файлы.

Удаление файлов. Нижеприведенная инструкция работает с закрытыми файлами. Для открытого файла будет выдано сообщение об ошибке.

Kill полное_имя_файла

3. ОПЕРАТОРЫ ДЛЯ ЗАПИСИ И ЧТЕНИЯ ДАННЫХ ИЗ ФАЙЛОВ

После того как файл открыт, можно манипулировать с информацией помещаемой в файл или извлекаемой из него. Для этого используют операторы, представленные в таблице 2.1.

Таблица 2.1.

Тип доступа	*Запись данных*	*Чтение данных*
Последовательный	Print #файловое_число, аргументы Write #файловое_число, аргументы	Input #файловое_число, аргументы Line Input # файловое_число, имя_переменной
Произвольный	Put [#]файловое_число, [номер_записи], переменная	Get [#]файловое_число, [номер_записи], переменная
Бинарный	Put [#]файловое_число, [номер_записи], переменная	Get [#]файловое_число, [номер_записи], переменная

4. УПРАВЛЕНИЕ ФАЙЛАМИ ПОСЛЕДОВАТЕЛЬНОГО ДОСТУПА

4.1. Запись данных в файл при использовании инструкции *Print#*

Задание №1. Наберите текст следующей процедуры. Убедитесь, что все работает правильно, и объясните результат.

```
Sub Open_File()                          'ошибка т.к. файла нет
   Open "File_Posled_1" For Input As #1
End Sub
```

Задание №2. Наберите текст и запустите на выполнение программу представленную ниже. Убедитесь, что она создает файл. Обратите внимание на то, что файл не закрыт. Как она работает без закрытия файла и закрытием?

```
Sub Open_File_l()
'=============================================
' Создаем пустой файл "File_Posled_1" или обнуляем существующий
Dim FileNum As Integer
Dim PahtFile As String
        FileNum = FreeFile        ' получаем значение файлового числа
        PahtFile = "D:\VU_Work\"      ' определяем путь к файлу
        Open PahtFile & "File_Posled_l.txt" For Output As #FileNum
              ' Close Файл не закрыт !!
End Sub
```

Задание №3

а. Переделайте пример из *задания №2* так, как показано ниже. Результаты выполнения программы контролируйте по содержимому создаваемого файла. Запишите или запротоколируйте это содержимое. Обратите внимание на то, что инструкция ***Print#*** допускает форматирование информации при записи и, в частности, позволяет использовать функцию ***Format().*** Выведите ту же информацию в окно ***Immediate,*** заменив соответствующие операторы. Чем ее вид отличается при записи в файл и выводе в окно ***Immediate***.

```
Sub Open_File_2()
'=============================================
'Заполняем данными файл последовательного доступа
Dim FileNum As Integer
Dim PahtFile As String
Dim b_int As Integer, b_sin As Single ' вспомогательные переменные
   FileNum = FreeFile            ' получаем значение файлового числа
   PahtFile = "D:\VU_Work\"       ' определяем пуль к файлу
   ' создаем  (открываем) файл
   Open PahtFile & "File_Posled_l.txt" For Output As #FileNum

Print #FileNum, "Привет участникам лаб.работ"
   Print #FileNum, "Еще один привет участникам лаб.работ"
      Print #FileNum, 13;   'Что будет, если убрать точку с запятой ?
         b_int = 13
         Print #FileNum, b_int
Print #FileNum, 1; 2; 3; 4; 5
   Print #FileNum, "Оксана"; "Ваня"; "Денис"; "Катя"
      Print #FileNum, 1, 2, 3, 4, 5
         Print #FileNum, "Оксана", "Ваня", "Денис", "Катя"

b_sin = 0.1234            ' Запись с форматированием
Print #FileNum, b_sin, Format(b_sin, "#,##=.00"), Format(b_sin, "0.000")
```

```
Print #FileNum, Tab(13); "Петров"; Tab(20); "Ваня"
Print #FileNum, Spc(13); "13 пробелов слева"
Print #FileNum, "11 строка"
Print #FileNum, "123456789*123456789*123456789*123456789  номер симво-
                ла в строке"
Close FileNum                ' закрываем файл
End Sub
```

После выполнения программы в файле будет храниться информация представленная на рисунке 2.2. Она понадобится при выполнении и анализе следующего задания.

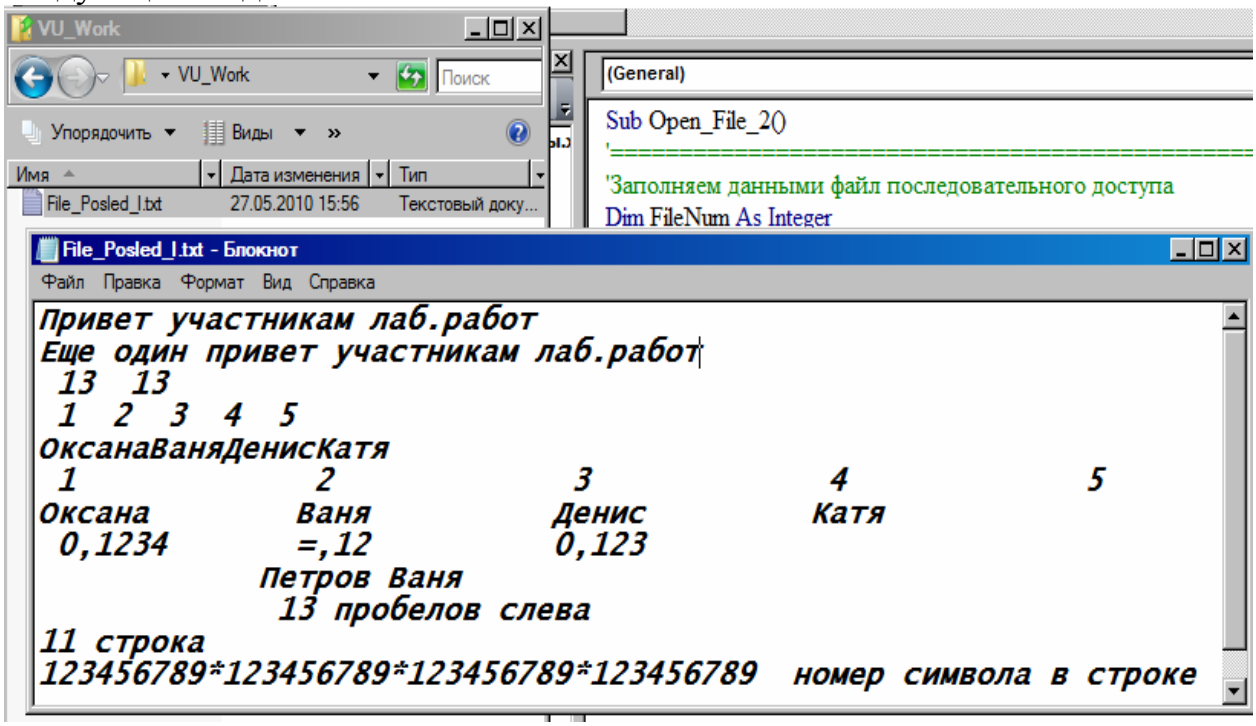


Рисунок 2.2. Содержимое файла File_Posled_l.txt

б. Убедитесь, что при разделении списка аргументов символом запятая (,) элементы будут заполнять файл блоками, заполненными пробелами и разделенными фиксированными табулоотстопами по 14 символов. В том случае, если использовать разделитель точку с запятой (;), то напечатанные в строке символы не будут содержать пробелов (числа в результате преобразования в строку будут иметь по 2 пробела между ними.)

в. Покажите - что конструкция

- *Tab(13); "Петров"; Tab(20); "Ваня"* - позволяет записать в файл аргумент *Петров* начиная с 13 столбца а *Ваня* с 20;
- *Spc(13); "13 пробелов слева"* – позволяет вставить 13 пробелов до первого символа аргумента.

г. Запишите булеву переменную и дату в файл последовательного доступа в виде отдельной строки.

д. Напишите программу, которая бы добавляла данные в конец уже созданного файла

4.2 Запись данных в файл при использовании инструкции (оператора) *Write#*

При использовании оператора ***Print*** данные при записи форматируются в пригодную для чтения форму, подобную текстовому файлу (файлы *read.me* являются такими).

Оператор ***Write*** после каждой записанной строки автоматически вставляет *символы перевода каретки и новой строки (CHR(13) + CHR(10))*. Он хотя и работает подобно оператору ***Print,*** но сохраняет существующие кавычки и разделяет выводимые данные запятыми. Это упрощает определение конца одного элемента данных и начало другого. Поэтому тогда, когда предполагают сохранить данные, а затем прочесть их программой, то пользуются оператором ***Write***.

Задание №4. В процедуре *задания №3* исправьте операторы *Print* и *Write* и заполните файл данными. Запишите содержимое файла так, чтобы была видна структура представления данных. Объясните, чем они отличаются от тех данных, что были созданы оператором *Print.*

4.3. Чтение данных из файла при использовании функции *Input* и инструкции *Input#.*

Функции ***Input*** и ***Input#*** работают в паре с оператором ***Write***. Они имеют следующий синтаксис:

Input(*количество_считываемых_символов*, [#]*файловое_число*)

Функция читает все символы, в том числе запятые, ограничители, символы конца строк и др. Количество считываемых символов следует определят функцией *FileLen*.

Input [#]*файловое_число, список_переменных*.

Тип переменных может быть любой, но согласованный с типом записи в момент ее создания.

! При чтении строковой переменной VB читает ее, начиная с первого непустого символа, и прекращает, если встречает запятую или конец строки. Если первый непустой символ (**“)**, то чтение переменной будет продолжено до следующего символа (**“**) или конца строки. В этих случаях будут прочитаны все символы, включая запятые.

При чтении файла кавычки вокруг строк, разделяющие запятые и пустые строки, игнорируются. При достижении конца файла попытка чтения из него приведет к ошибке. Для проверки на конец файла можно использовать функцию ***EOF***(*файловое_число)*.

Задание №5. Исследуйте работу функций ***Input*** и ***Input [#],*** приняв за основу текст процедуры, представленный ниже. В качестве файла с информацией используйте файл из предыдущего задания.

```
Sub Read_File_5()
    '================================================
    'Читаем данные из файла последовательного доступа
    Dim FileNum As Integer
    Dim PahtFile As String
    Dim i As Integer
    Dim b_int As Integer, b_sin As Single, b_var As Variant
    Dim b_bul As Boolean, b_str As String, b_str_1 As String, b_lon As Long
      FileNum = FreeFile    ' получаем значение файлового числа
      PahtFile = "D:\VU_Work\" ' определяем путь к файлу
      Open PahtFile & "File_Posled_1.txt" For Input As #FileNum

     b_bul = EOF(FileNum)    'Конец файла ?
     b_lon = FileLen(PahtFile & "File_Posled_1.txt")  'Длина файла

    Input #FileNum, b_str, b_str_1, b_int, b_sin 'Читаем из файла 4 строки
    Input #FileNum, b_str  'Читаем из файла следующую пятую строку
    b_var = Input(20, #FileNum)  'Выводим -20 символов, начиная с текущей
                                 позиции указателя.
       For i = 1 To 7        'Читаем   7 строк, начиная с текущей позиции, и
                             записываем их в переменную b_str
          Input #FileNum, b_str
       Next i

    Input #FileNum, b_str, b_int, b_sin 'выводим 3 строки
         b_var = Input(20, #FileNum) 'Выводим -20 символов, начиная с те-
                                     кущей позиции указателя.
         Seek #FileNum, 44 'Позицируем положение указателя на 44 симво-
                           ле, начиная с начала файла
         Input #FileNum, b_str 'Получаем часть строки
       Close FileNum
End Sub
```

Значения переменных на момент начала выполнения цикла FOR, выведены в окне значений локальных переменных и представлены на рисунке 2.3.

Результаты исследования фиксируйте, используя вывод данных на панель *Immediate*.

Объясните все полученные результаты. Как получить ненулевые значения для переменных *b_str, b_int, b_sin?* Как вывести все переменные из файла?

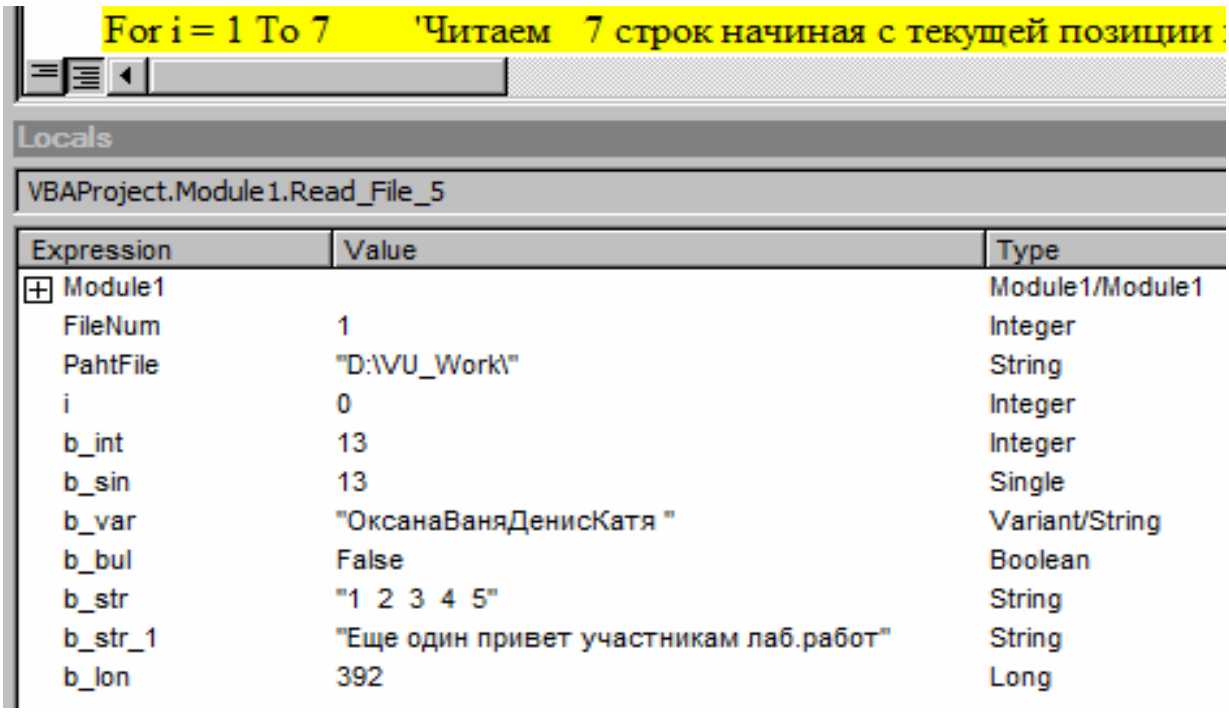


Рисунок 2.3. Окно локальных переменных

4.4. Чтение данных из файла при использовании функции *Line Input*

Оператор ***Line Input*** используется для чтения в программу точной копии содержимого строки файла. Он имеет синтаксис:

Line Input *#файловое_число, имя_переменной.*

Переменная должна быть или строкового типа или типа Variant. Символы возврата каретки и перевода строк не возвращаются в переменную. При чтении данных сохраняются кавычки, запятые, ограничители, в одной строке могут находится данные разного типа. Все это вызвано тем, что они не редактируются при чтении и записи в файл. Оператор ***Line Input*** используют для чтения строк из файлов записанных с помощью инструкции *Print#.* Дальнейшее редактирование на остается совести программиста, который может использовать для этого, как вариант, строковые функции.

Задание №6. Напишите программу для вывода всех данных из файла последовательного доступа, используя оператор ***Line Input.*** Отладьте ее, результаты выводите на панель *Immediate*. Сохраните результат исследования.

5. УПРАВЛЕНИЕ ФАЙЛАМИ ПРОИЗВОЛЬНОГО И БИНАРНОГО ДОСТУПА

Если для последовательных файлов запись производится в конец, а чтение с начала файла, то для файлов произвольного доступа (и бинарных) чтение и запись может осуществляться в любое место и с любого места.

Установить нужный номер записи или чтения можно оператором ***Seek****:*

Seek *[#]файловое_число, номер_записи*

Номер записи это число от 1 до 2147483647, куда будут помещены данные следующим оператором ***Put*** или откуда они будут прочитаны оператором ***Get***. Для бинарных файлов –это номер байта. Если не указывать *номер_записи*, то по умолчанию принимается текущая позиция указателя записи.

Определение текущей позиции возможно при использовании конструкции:

Seek [#]*файловое_число*

5.1 Операторы записи - *Put* и чтения – *Get*

Операторы имеют следующий синтаксис:

Put [#]*файловое_число, [номер_записи], переменная*
Get [#]*файловое_число, [номер_записи], переменная*

Задание №7. Создайте файл произвольного доступа, запишите в него информацию и прочитайте ее, воспользовавшись следующей программой:

```
Sub Sozd_WR_File_7()
'=============================================================
   'Создать файл произвольного доступа ( Файловое число =1 ), _
   записать данные в него и прочитать их. _
   Файл в Word-е не читается ! _
   При повторном обращении файл не обнуляется.
Dim PahtFile As String, FileNum As Integer
Dim i As Integer, b_int As Integer, b_int1 As Integer
   'Открываем файл произвольного доступа File_Proizv_l.dat
   'Длину файла определим в 2 байта
PahtFile = "D:\VU_Work\"      ' определяем путь к файлу
 FileNum = FreeFile
 Open PahtFile & "File_Proizv_l.dat" For Random As #FileNum Len = 2
   'Запишем и считаем последовательно в файл цифры 1. 2, 3, 4, 5, _
   'проверяя как изменяется при этом длина записи файла
For i = 1 To 5
   Put FileNum, i, i
   Get FileNum, i, b_int
   b_intl = LOF(FileNum) 'Получаем длину файла в байтах (вариант №1)
   b_intl = Len(i)
   Debug.Print "Запись №:"; i, "Содержимое="; b_int, "Длина="; b_intl
Next i
Close #1
End Sub
```

Поясните работу всех операторов и их особенности.

Как увидеть содержимое файла?

Задание №8. Создайте файл произвольного доступа, запишите в него информацию, воспользовавшись набором процедур, представленных ниже.

Объясните назначение и работу каждой процедуры, и каждого оператора в них. Информацию, полученную в процессе работы программы, сохраните либо в отдельном файле, либо запишите.

Сделайте так, чтобы ввод информации можно было осуществлять из экранной формы.

Создайте программу, выполняющую те же действия, но использующую динамический массив для хранения данных в переменной пользовательского типа.

Листинг программы.

```
'В разделе деклараций создаем переменную пользовательского типа
'По общепринятой терминологии она представляет собой
'запись с тремя (в данном случае) полями

   Type Gragdanin
      Name As String * 30
      DatRog As Date
      Doplnf As String * 25
   End Type
'--------------------------------------------------------------------------

Public Sub Gen_Zap(Zapis As Gragdanin, _
                   Имя As String, Д_Рож As Date, Прим As String)
   '=============================================
   ' Процедура заполнения полей переменной пользовательского типа
                              ' Гененерирует запись, например
   Zapis.Name = Имя           '"Петров Иван Денисович"
   Zapis.DatRog = Д_Рож       '#11/10/1992#
   Zapis.Doplnf = Прим        ' Допол. информация - "Внучек"
End Sub

Public Sub Pechat_Zap(Zapis As Gragdanin)
  '=============================================
' Печать записи в окно отладки
Debug.Print "Имя:"; Zapis.Name, _
          "Дата рождения:"; Zapis.DatRog, _
          "Примечания:"; Zapis.Doplnf
End Sub
```

```
Sub Sozd_WR_File_8()
 '==========================================
' Создаем файл с 4-мя записями
Dim Zap_Gr As Gragdanin ' переменная типа Gragdanin
                          (пользовательского типа)
Dim PahtFile As String
Dim i As Integer, b_int As Integer, b_intl As Integer
   PahtFile = "D:\VU_Work\"
      'Открываем файл произвольного доступа с длиной записи, равной
      'длине переменной пользовательского типа Gragdanin
   Open PahtFile & "File_BD_l.dat" For Random As #1 Len = Len(Zap_Gr)

'Создадим 4 записи и выводим их в окно отладки
   ' Генерируем одну (первую) запись (значения переменной польз. типа)
   Gen_Zap Zap_Gr, "Петров Иван Денисович", #10/11/1992#, "Внук"
   ' Размещаем запись в дисковом файле как первую
   Put #1, 1, Zap_Gr '
   ' Выводим (запись)значения переменной пользю типа в окно отладки
   Pechat_Zap Zap_Gr

   ' Создаем и размещаем запись в дисковом файле как вторую
   Gen_Zap Zapis:=Zap_Gr, Имя:="Виктория Олеговна ", _
            Д_Рож:=#2/18/1973#, Прим:="Бухгалтер"
   Put #1, 2, Zap_Gr
   Pechat_Zap Zapis:=Zap_Gr

   Call Gen_Zap(Zap_Gr, "Мороховец Катя", #6/24/1999#, "Ученица")
   Put #1, 3, Zap_Gr
   Call Pechat_Zap(Zap_Gr)

      Call Gen_Zap(Zap_Gr, "Денис Вадимыч", #2/1/1973#, "Автомобилист")
      Put #1, 4, Zap_Gr
      Call Pechat_Zap(Zap_Gr)

   ' Обращения к процедурам может быть любым. В примере рассмотрено 3
варианта.  На наш взгляд, удобней последней вариант. В нем оператор Call
указывает на вызов процедуры, и ее название не виснет непонятным словом
в тексте программы. Фактические параметры помещены в скобки и видны.
   ' Наиболее путанным обращением к процедуре следует признать первое.

b_int = LOF(1)  '- Функция для определения числа записей
Debug.Print "Длина файла="; b_int
Close #1
End Sub
```

5.2. Создание запроса на получение доступа к произвольной записи из файлов произвольного доступа.

Задание №9. Приведенная ниже процедура позволяет определить количество записей и получить любую из них. Она обращается к процедурам и пользовательской переменной, описанными в предыдущем *задании №8,* и к созданному ранее файлу произвольного доступа. Наберите ее и поэкспериментируйте.

```
Sub Poluch_File_9()
'==========================================

Dim Zap_Gr As Gragdanin
Dim PahtFile As String, b_str As String
Dim i As Integer, b_int As Integer, b_intl As Integer
   PahtFile = "D:\VU_Work\"
   Open PahtFile & "File_BD_l.dat" For Random As #1 Len = Len(Zap_Gr)
   b_int = LOF(1) \ Len(Zap_Gr)  ' определяем число записей
      Debug.Print "Число записей файла="; b_int
   For i = 1 To b_int
      Seek 1, i              ' устанавливаем ПОЗИЦИЮ записи
      Get #1, , Zap_Gr       ' получаем данные из файла
      Call Pechat_Zap(Zap_Gr)
   Next i
   Close #1
End Sub
```

Результат выполнения процедуры приведен на рисунке 2.4.

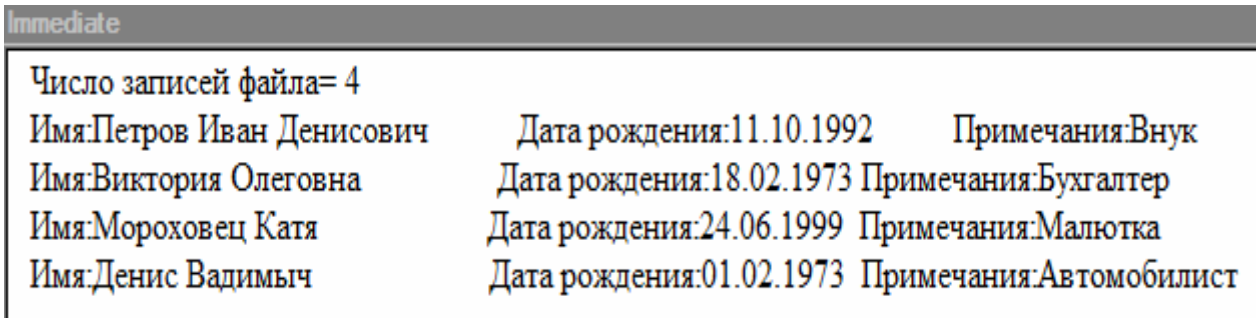


Рисунок 2.4.

Создайте экранную форму, из которой можно будет изменить любую запись в файле.

5.3 Удаление записей и строк из файлов.

Задание №10. Самостоятельно создайте программу, в которой можно удалить:

а. выбранную пользователем запись в файле произвольного доступа,

б. выбранную пользователем строку в файле последовательного доступа.

Замечание: рассмотрите варианты:

- с обнуление данных в выбранных элементах записи и строк,
- с удалением данных из файла, при котором все записи и строки файла смещаются на одну вверх, а последняя из них стирается .

Задание №11. Организуйте вывод информации на принтер, используя файловые инструкции:

Open "LPT1" For Output As #4
Print #4, "Имя:"; Zap_Gr.Name
Close #4

3. ИСПОЛЬЗОВАНИЕ ФУНКЦИЙ WIN32 API В VBA

1. ОСНОВНЫЕ ПОЛОЖЕНИЯ

Функции ***Win32 API*** используются для расширения возможностей разрабатываемого приложения VBA. Средства ***Win32 API*** применяются в разрабатываемых на разных языках (VB, C и др.) программах, в приложениях Windows и в самой операционной системе Windows различных версий. Эти средства очень сильно отличаются друг от друга. Одни подают звуковой сигнал, другие изменяют поведение операционной системы. При неверном обращении с функциями ***Win32 API*** будут возникать сбои в работе приложений VBA и операционной системы.

API – Application Programming Interface переводят как «***Интерфейс прикладного программирования»*** или ***«Интерфейс программирования приложений»***. ***Win32 API*** используется в 32 разрядных операционных системах. Причем при разработке информационных технологий и операционных систем фирма Microsoft использовала модель ***WOSA –Windows Open Service Architecture*** – «***Открытую архитектуру служб Windows».***

Преимущества открытой архитектуры состоят в следующем:

- ✓ способствует стандартизации программ,
- ✓ программное обеспечение становится переносимым на компьютеры с различными процессорами,
- ✓ улучшается взаимодействие программного и аппаратного обеспечения, разработанного различными фирмами.

Интерфейс API был создан в результате формализации процесса неоднократного использования программного кода. Он оформлен в виде библиотек динамической компоновки *(DLL)* с файлами, имеющими расширение *DLL* или *EXE*. *Динамическая компоновка* обозначает, что связь с процедурой устанавливается динамически, во время выполнения программы, и только тогда, когда процедуру требуется вызвать. *При статической компоновке* – библиотеки присоединяются к программе на этапе компиляции или редактирования связей.

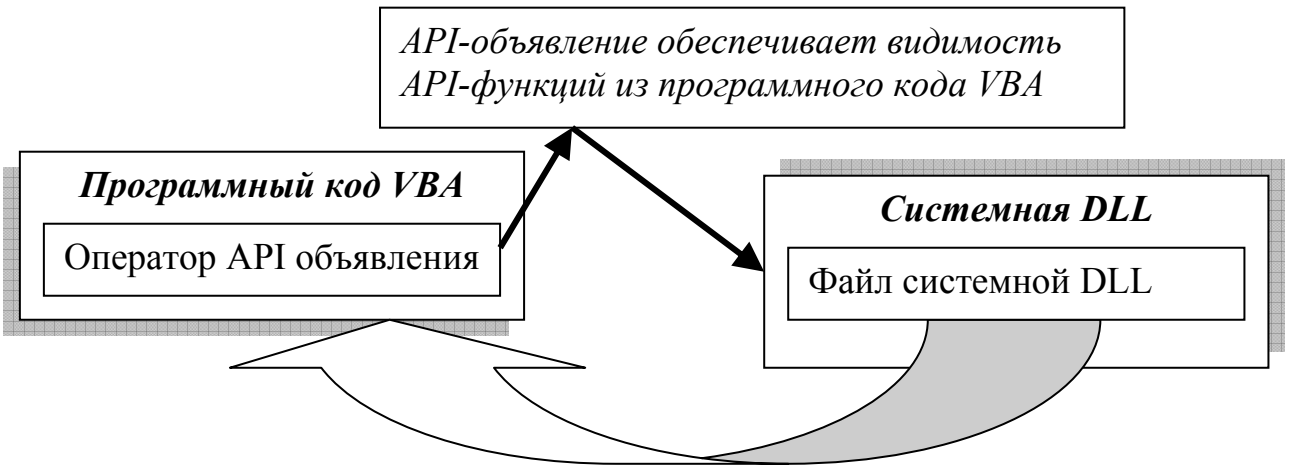


Рисунок 3.1 Работа с функциями API

Библиотеки DLL могут обновляться независимо от использующих их программ и не занимают место в памяти до тех пор, пока не понадобятся. При создании нового API или собственных DLL библиотек происходит лишь расширение старого набора библиотек.

Существует более 1000 функций API. *Классификация функций API* может быть представлена в следующем виде:

- ✓ управление Windows (нажатие клавиш, действия мыши …),
- ✓ элементы управления (диалоговые окна, кнопки ..),
- ✓ настройка приложений,
- ✓ графические средства (рисование точек, цвет..),
- ✓ системные средства (управление памятью, системным временем..),
- ✓ языковая поддержка,
- ✓ сетевые средства.

Основные библиотеки, составляющие большую часть Win32 API это:

- ✓ ***User32.dll.*** В этой библиотеке расположены функции управления окнами, меню и таймерами;
- ✓ ***COMDLG32.dll.*** Библиотека функций, связанных с использованием диалоговых окон общего назначения.
- ✓ ***Kernel32.dll.*** Функции этой библиотеки управляют памятью и другими системными ресурсами;
- ✓ ***GDI32.dll.*** Библиотека функций, которые управляют выводом на экран дисплея и на принтер.

Следует отметить, что использование VBA для программирования в Office позволяет создавать полноценные приложения, но использование при этом *функций Win32 API* значительно расширяет возможности VBA, а, следовательно, и разработчика программ и документов.

2. ОПИСАНИЕ ФУНКЦИЙ *Win32 API* И ИХ ИСПОЛЬЗОВАНИЕ.

Перед использованием *функций Win32 API* их **обязательно** необходимо объявить (описать), используя для этого специальную *инструкцию (описатель) Declare*. Это объявление производят в разделе деклараций любого модуля. Оно должно быть сделано для того, чтобы указать программе пользователя где расположена искомая функция API, какие аргументы и как передаются функции, что эта функция возвращает.

При объявлении функции необходимо указать в описателе следующую информацию:

1. имя функции,
2. используемую DLL,
3. передаваемые в функцию параметры,
4. тип возвращаемых функцией данных.

Пример описателя для одной из функций API представлен ниже, на рисунке 3.2.

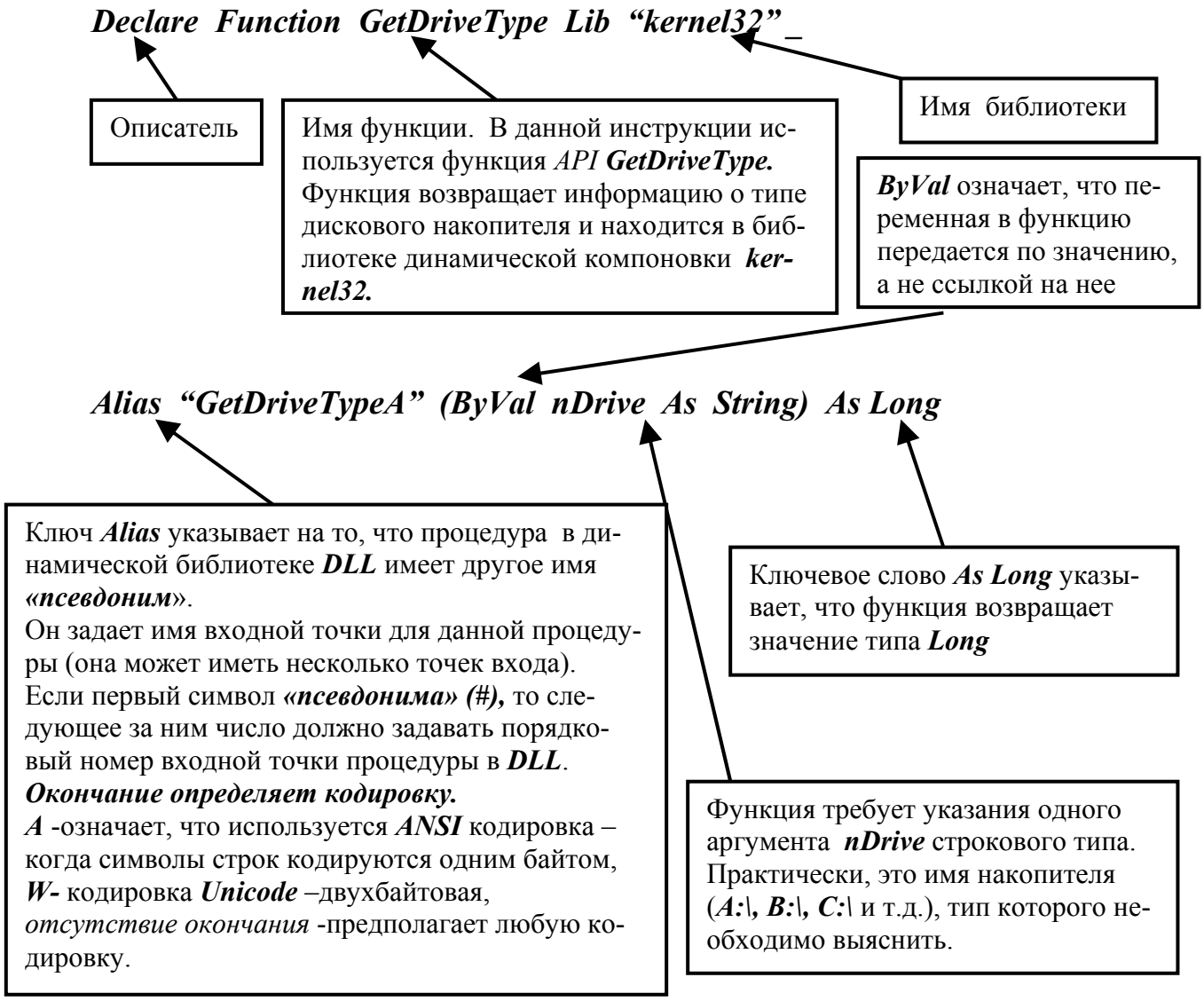


Рисунок 3.2. Пример описателя

Для хранения возвращаемой информации необходимо использовать переменную, тип которой соответствует типу значения, возвращаемого функцией. Для данного примера в рабочих процедурах необходимо использовать операторы:

```
Dim var_Drive as Long              'объявляем тип переменной
var_Drive= GetDriveType("C:\")     'обращение к функции
```

Заметим следующее!

В поставках Office-а вместе с встроенным в него VBA найти функции *Win32 API* невозможно. Поэтому прототипы объявлений *функций WinAPI,* а также используемые ими константы и типы данных, можно найти в файле ***WIN32API.TXT***, поставляемом со многими приложениями, включая Visual Basic или СИ. Этот файл инсталлируется в подкаталог ***Common\Tools\Winapi*** программного каталога ***Visual Basic.*** Чтобы воспользоваться в своей программе какой-то функцией *API*, достаточно скопировать из этого файла не-

обходимую информацию и вставить ее в свой модуль. Комментарии, включенные в этот файл, помогут выяснить назначение функций.

В поставку *Visual Basic* входит специальное приложение ***API Text Viewer***, которое упорядочивает все функции *API* по алфавиту и помогает быстро добавить объявление нужной функции. Причем возможно копирование сразу нескольких объявлений *API* функций. ***API Text***.

Для запуска этого приложения можно щелкнуть по пиктограмме *API Text Viewer***,** расположенной в меню "*Пуск"* системы *Windows* в группе программ *Visual Basic*. После этого в меню "*Файл"* следует выбрать команду *"Load Text File*", которая загрузит *WIN32API.TXT.* Как только *API Text Viewer* загрузит указанный файл, в разделе *Available Items* появится список доступных объявлений *API.*

Документация к *Windows API* никогда не входила в состав Visual Basic или VBA. Для эффективного и профессионального использования *Windows API* следует обратиться на сервер ***Microsoft Developers Network Online***, который находится в ***Internet*** по адресу: ***http://www/microsoft.com/msdn/.***

Для обращения к *WIN32API.TXT* могут быть использованы любые поисковики, например Yandex. В принципе, при использовании *функций API* достаточно ознакомиться со справочной информацией, которая с избытком содержится в литературе по программированию.

Задание №1.

Рассмотрите работу функции ***GetDriveType***, описанной выше на рисунке 3.2. Наберите текст программы представленной ниже. Убедитесь в том, что она работает. Модуль с программой может быть расположен в любом приложении Office.

‘ Описатель и константы разместите в разделе деклараций модуля

```
Declare Function GetDriveType Lib "kernel32" _
        Alias "GetDriveTypeA" (ByVal nDrive As String) As Long
            'Значения, возвращаемые функцией GetDriveType и соответствующие различным типам накопителей, поставим в соответствие с константами, смысл которых, в общем-то ясен.
Public Const DRIVE_UNKNOWN = 0
Public Const DRIVE_NO_ROOT_DIR = 1
Public Const DRIVE_REMOVABLE = 2
Public Const DRIVE_FIXED = 3
Public Const DRIVE_REMOTE = 4
Public Const DRIVE_CDROM = 5
Public Const DRIVE_RAMDISK = 6
```

```
Sub devGetDriveType()
'================================
  Dim var_lng_Drive As Long
  Dim var_str_Drive As String
var_str_Drive = InputBox("Введите имя диска:", _
                                "Исследование функций Win32API", "C:\")
If var_str_Drive <> "" Then  'Если ввели пустую строку то анализ не _
                              производим   и переходим к  оператору End If
   var_lng_Drive = GetDriveType(var_str_Drive)  ' Определяем тип накопителя
   Select Case var_lng_Drive
      Case DRIVE_UNKNOWN
            MsgBox "Тип накопителя неизвестен", _
                        vbInformation, "Исследование функций Win32API"
      Case DRIVE_NO_ROOT_DIR
            MsgBox "Корневой каталог отсутствует", _
                        vbInformation, "Исследование функций Win32API"
      Case DRIVE_REMOVABLE
            MsgBox "Накопитель является съемным диском.", _
                        vbInformation, "Исследование функций Win32API"
      Case DRIVE_FIXED
            MsgBox "Накопитель является жестким диском.", _
                        vbInformation, "Исследование функций Win32API"
      Case DRIVE_REMOTE
            MsgBox "Накопитель является сетевым диском.", _
                        vbInformation, "Исследование функций Win32API"
      Case DRIVE_CDROM
            MsgBox "Накопитель является приводом CD-дисков.", _
                        vbInformation, "Исследование функций Win32API"
      Case DRIVE_RAMDISK
            MsgBox "Накопитель является электронным диском.", _
                        vbInformation, "Исследование функций Win32API"
   End Select
End If
End Sub
```

<u>Задание №2</u>.

Наберите текст нижеследующей процедуры. Она служит для определения интервала времени между событиями и достаточно полно прокомментирована. Создайте свой вариант программы с использованием рассмотренной функции *WIN32 API* – ***GetTickCount.***

```
Declare Function GetTickCount Lib "kernel32" () As Long 'Декларируем
                                                          функцию
```

```
Sub Delta_Time()
' Определение временного интервала
' ==========================
Dim Time_Int  As Long
Dim b_str As String
      Time_Int = GetTickCount()     'Получение начала отсчета  времени
      MsgBox "Нажмите кнопку ОК!", vbInformation, _
                "Фиксация конца интервала"
      Time_Int = GetTickCount() - Time_Int
                                    'Подсчет времени до нажатия кнопки ОК.
      b_str = "Прошло " & Time_Int
      b_str = b_str & " миллисекунд от запуска программы _
                           до нажатия кнопки ОК."
      MsgBox b_str, vbInformation, "Исследование функций Win32API"
End Sub
```

3. СТРОКОВЫЕ ФУНКЦИИ *Win32API*

При возвращении строк функциями *WIN32API* необходимо заранее указать размер возвращаемой строки для того, чтобы под нее была выделена память. Длина строк должна быть на единицу больше возвращаемой строки, т.к. любая строка заканчивается служебным символом, ASCII-код которого равен 0.

При работе с функциями используют ***дескрипторы и указатели***.

Дескриптор –специальное число, используемое Windows для идентификации объектов (окна и др.). Почти все объекты имеют дескрипторы, а большинство функций API получает и возвращает дескрипторы.

Указатели -адрес объекта, переменной или структуры. VBA их не использует.

Библиотеки динамической компоновки различаются на следующие виды:

- ✓ ***DLL*** – могут использоваться любыми приложениями,
- ✓ ***XLL*** - могут использоваться только Excel,
- ✓ ***WLL*** - могут использоваться только Word.

Задание №3. Рассмотрите вариант использования строковых функций. Для примера возьмите функцию ***mciSendString,*** которая позволяет управлять устройствами мультимедиа при помощи строковых команд, являясь одним из вариантов использования специального интерфейса управления устройствами мультимедиа *(MCI)*.

На диске D находится звуковой файл:

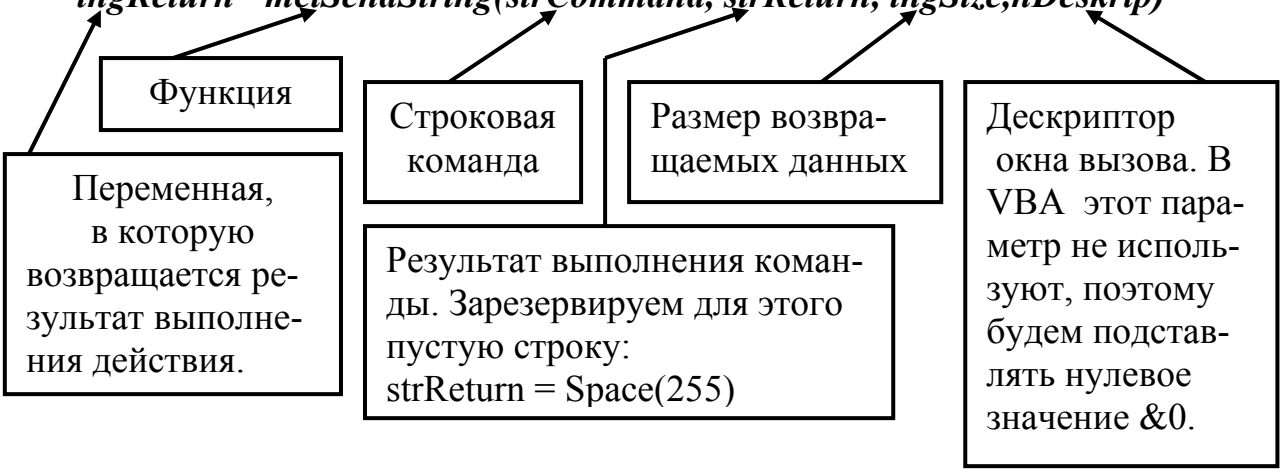
D:\VU_Bas\VBASIC\Mulnimedia\VU_S\Debusi.rmi.

Для того, чтобы воспроизвести его, используют три команды:

- ✓ ***open*** – открывает устройство (файл),
- ✓ ***close*** – закрывает устройство (файл),
- ✓ ***play*** – запускает воспроизведение данных.

Синтаксис функции ***mciSendString*** *имеет следующий вид:*

lngReturn= mciSendString(strCommand, strReturn, lngSize,hDeskrip)



Запишите текст процедуры, представленный ниже, в модуль документа и запустите ее.

```
Declare Function mciSendString Lib "winmm.dll" _
Alias "mciSendStringA"
(ByVal lpstrCommand As String, ByVal lpstrReturn As String, _
ByVal lngSize As Long, ByVal hwndCall As Long) As Long
                    ' описание функции в разделе деклараций
'-------------------------------------------------------------------------------


Sub Muzon()
'=================
'Проигрывает файл, расположенный на диске D:
Dim File_Name As String
        File_Name = "D:\VU\ЛИТМО\ПОСОБИЯ\VBA_03\Debusi.rmi"
        Player File_Name
End Sub

Sub Player(File_Name)
'=================
Dim lng_Return As Long
Dim str_Command As String
Dim str_Return As String
Dim lng_Size As Long

' Открываем файл
        str_Command = "open " & File_Name & " alias music"
        str_Return = Space(255)
        lng_Return = mciSendString(str_Command, str_Return, lng_Size, 0&)
```

```
' Запускаем воспроизведение файла
        str_Command = "play music"
        str_Return = Space(255)
        lng_Return = mciSendString(str_Command, str_Return, lng_Size, 0&)

' Закрываем файл
        str_Command = "close " & File_Name
        str_Return = Space(255)
        lng_Return = mciSendString(str_Command, str_Return, lng_Size, 0&)

End Sub
```

Попробуйте связать какое-нибудь событие с запуском звукового файла.

Задание №4. Самостоятельно разберитесь в синтаксисе и работе одной из существующих функций WIN32 API. Приведите пример ее использования.

4. РАСШИРЕННЫЕ СРЕДСТВА СОЗДАНИЯ ПРИЛОЖЕНИЙ Mc.OFFICE. ТЕХНОЛОГИЯ ACTIVEX

1. ОСНОВНЫЕ ПОЛОЖЕНИЯ.

Все приложения Office позволяют так или иначе обрабатывать данные: Excel производит расчеты и их анализ, Word позволяет работать с текстовыми документами, Access управляет базами данных и т.д. Программирование на VBA позволяет объединять отдельные приложения, использовать объекты и возможности разнообразных приложений не покидая того, в котором работает пользователь, т.е интегрировать приложения. К таким расширенным средствам создания приложений относят**:**

- ✓ использование WinAPI,
- ✓ технологию ActiveX и элементы управления ActiveX

Знакомству с последним средством, а именно, с технологией ActiveX посвящена данная работа.

В соответствии с терминологией, используемой фирмой Microsoft, термин *“ActiveX”* во многих случаях заменяет термин “*OLE” (Object Linking and Embedding* – связывание и внедрение объектов).

В связи с этим то, что раньше называлось "*объект OLE*", теперь имеет название "*компонент ActiveX*" или "*объект ActiveX*", а в некоторых случаях просто *"объект".*

Претерпел изменение и термин "*OLE Automation* ". Теперь его название "*ActiveX Automation*" или просто " *Automation*".

Отметим, что фирмой Microsoft при переходе к новыми технологиям, понятие "*объекта"* расширено до понятия *"программируемого объекта*", чье поведение можно изменять, используя *Automation.* Именно это позволяет конструировать приложения, управляющие другими объектами-приложениями.

По большому счету, технология *ActiveX* собирательное понятие. Это сумма технологий таких как:

- ✓ технология создания элементов управления *ActiveX,*
- ✓ технология создания документов, работающих в браузерах Internet,
- ✓ *Remote Automation* – технология удаленного доступа,
- ✓ технология *Active DeskTop* – средств интеграции HTML-документов и элементов непосредственно на экран пользователя,
- ✓ *Active Server Pages*- технология создания и редактирования сценариев на Web-серверах.

ActiveX – это открытый стандарт, применяемый для совместного использования данных различными приложениями. Многие производители программного обеспечения, такие как *IBM/Lotus, Corel/WordPerfect* и др. создают приложения, поддерживающие технологию *ActiveX.*

Базовый элемент технологии *ActiveX* это – составляющие технологии *COM (Component Object Model* – компонентная модель объектов или модель

составных объектов), которая является одной из ключевых технологий компании Microsoft.

Технология ActiveX обеспечивает два мощных средства для разработки приложений:

- ✓ редактирование на месте, позволяющее внедрять объект из одного приложения в документ, созданный другим приложением,
- ✓ *Automation*, с помощью которого можно программным путем устанавливать свойства, вызывать методы и обрабатывать события внедренных объектов.

Первое средство тривиально. Им пользуются постоянно, не задумываясь о технологиях и других сложных вопросах. Например, находясь в текстовом редакторе Word, внедряем в документ рабочий лист Excel. Для этого используем команду *меню Вставка/Объект/Вкладка Создание/Тип объекта - Лист Microsoft Excel* или обычным копированием с использованием специальной вставки. После того, как объект (лист Excel) внедрен в документ, все связанные с ним данные хранятся в этом документе. Двойной щелчок по объекту приводит к его активизации и после этого с ним можно работать точно так же, как и в табличном процессоре.

Второе средство – *Automation* более сложное в технологии *ActiveX.* Ранее использовавшаяся средство (технология) называлось *DDE (Dynamic Data Exchange* - динамический обмен данными). Работало оно значительнее медленнее.

Приложения могут поддерживать Automation двумя способами:

- ✓ *Приложение – объект Automation ("сервер Automation")* – приложение, представляющее свои объекты для получения команд от управляющего приложения;
- ✓ *Управляющее приложение ("клиент Automation")* – приложение, которое воздействует на объекты *Automation* и посылает команды серверу *Automation*.

Отметим, что в качестве клиентов не могут выступать такие приложения Office как Office Binder и Outlook.

2. СОЗДАНИЕ И УПРАВЛЕНИЕ ОБЪЕКТАМИ *AUTOMATION* ИЗ ПРОГРАММ

В принципе, ***Automation*** можно рассматривать как применение двух функций:

- ✓ ***CreateObject***. Функцию можно использовать в приложении-клиенте для доступа к объектам Automation приложения-сервера без установки ссылки на него. Ее используют для создания объекта. Эта функция имеет один строковый аргумент - класс, представляющий имя соответствующего объекта Automation. Функция возвращает ссылку на объект Automation.

Например: *CreateObject("Excel.Application")* возвращает приложение Excel.

✓ ***GetObject.*** Функция используется для доступа к существующим документам (объектам), хранящимся в файлах. Она имеет два аргумента: путь к файлу и класс.

Например: GetObject("C:\Книга1.xls", "Excel.Sheet") возвращает ссылку на рабочий лист Excel.

2.1. Использование функции *CreateObject*

Типы объектов Automation, их классы, используемые функцией, и библиотеки представлены ниже в таблице 4.1.

Таблица 4.1

Библиотека	**Класс**	**Тип объекта**
Microsoft Excel 12.0 Object Library	Excel.Application	Приложение
	Excel.Sheet	Лист (Раб.книга с 1-м листом)
	Excel.AddIn	Раб.книга
	Excel.Chart	Диаграмма (Раб.книга с 2-мя листами)
Microsoft Access 12.0 Object Library	Access.Application	Приложение
Microsoft PowerPoint 12.0 Object Library	PowerPoint.Application	Приложение
Microsoft Word 12.0 Object Library	Word.Application	Приложение
Microsoft Binder 12.0 Object Library	OfficeBinder.Binder	Приложение
Microsoft OutLook 12.0 Object Model	OutLook.Application	Приложение

При работе с различными объектами используют два способа, позволяющие обеспечить к ним доступ: ***раннее связывание и позднее связывание.***

Раннее связывание предполагает создание ссылки на существующий объект или конкретную библиотеку объектов.

При создании объекта из приложения, в котором находится данная процедура, используют следующую инструкцию:

```
----------
Dim theRange as Range
```

В данном случае объектная переменная theRange ссылается на объект класса Range.

Если, работая в приложении Excel, создают объект в другом приложении Office, то после создании ссылки на библиотеку этих объектов, используют инструкцию (для Microsoft Word 10.0 Object Library):

```
----------
Dim ObjWord as New Word.Applicatoin
```

Раннее связывание позволяет применять в программном коде константы библиотек. Оно считается более предпочтительным и позволяет работать с более высокой скоростью.

Позднее связывание более универсально.

В этом случае сначала объявляют объектную переменную без привязки к конкретному классу. Это переменная типа *Object*. Класс объекта может быть любой и память под объект не выделяется.

После этого специальным оператором *Set* переменную связывают с объектом того или иного класса.

```
----------------------
Dim theRange  As Object
Set theRange =  ActiveSheet.Range(“A1”)
```

2.1.1.Позднее связывание.

Задание №1. Наберите текст процедуры, представленной ниже, в модуле VBA. Попробуйте запустить процедуру находясь в Excel и Word. Поэкспериментируйте.

```
Sub Prim_E_01()
'=======================================
'Доступ к объекту Excel Application из Excel ( Word и др ")
'Позднее связывание

   Dim Obj_Excel As Object   ' Сначала объявим объектную переменную.
         'После этого объектную переменную свяжем с объектом _
         - приложением Excel. При этом приложение будет открыто.
   Set Obj_Excel = CreateObject("Excel.Application") '
      With Obj_Excel
         .Visible = True      'Сделаем приложение видимым.
         .Workbooks.Add  'Добавим в приложение книгу Excel.
       End With
Obj_Excel.Worksheets("Лист1").Range("A1").Value=“Проба Excel”
                    'Запишем информацию в ячейку A1 первого листа
Obj_Excel.Quit                'Закроем объект
Set Obj_Excel = Nothing     'Очистим переменную _
                               (освободим ее от ссылки на объект)
End Sub
```

Класс *Excel.Application* используют тогда, когда требуется представить данные в Excel, причем оставить пользователя в нем. Для того чтобы выйти из приложения, вызвавшего Excel, не закрывая сам Excel, следует свойство этого класса *Visible* установить равным *True*, а метод *Quit* "заремить”.

Что будет:

- если положить свойство Visible=False или вообще убрать?
- если не использовать метод Add ?
- куда попадет информация "Проба Excel" если клиентское приложение находится в Excel?
- объясните как реализовано позднее связывание и что это такое,
- добавьте программно еще несколько книг в приложение.

Задание №2. Наберите текст процедуры, используя Word, а потом Excel в качестве клиентского приложения, и поэкспериментируйте, добавляя инструкции в программу, убирая их. Прокомментируйте действие всех инструкций и операторов.

```
Sub Prim_E_02()
'=====================================
'Доступ к объекту Excel Application из Word и др.
Dim Obj_Sheet As Object
   Set Obj_Sheet = CreateObject("Excel.Sheet")
   With Obj_Sheet
      .Application.Visible = True
   End With

  ' Если этот макрос расположен в Excel, то будут выполнены все три
  ' инструкции, расположенные ниже.  Если он расположен в Word-е, то
  ' две последние из трех дадут сбой, т.к. не указан объект - книга,
  ' созданная в Excel -Obj_Sheet.
   Obj_Sheet.Worksheets("Лист1").Range("A1").Value = "Проба EEEExcel"
   Worksheets("Лист1").Range("A1").Value = "Проба Excel"
   Range("A1").Value = 222222

      'Obj_Sheet.Range("A1").Value = "Ошибка!!"
      ' Эта инструкция ошибочна, т.к. объект Obj_Sheet не лист и
      ' свойства Range не имеет

   Set Obj_Sheet = Nothing       'Очистим переменную (освободим ее
                                 'от ссылки на объект).
   Application.Quit              ' Закрываем приложение, в котором
                                 ' находится этот макрос.
End Sub
```

Обратите внимание на то, что при использование класса: объекта Excel - *Excel.Sheet (Excel.Chart)* в действительности возвращается книга, а не лист. Это можно легко проверить выполнив команду меню «*Сохранить файл*», остановив выполнение процедуры перед инструкцией *Set Obj_Sheet = Nothing*. Кроме того, созданные объекты существуют максимум до того момента, пока

существует клиентское приложение. Можно удалить их и раньше. Для этого нужно использовать оператор очистки переменной объекта *Set Obj_Sheet = Nothing*.

Этот класс объектов рекомендуется использовать тогда, когда требуется создать промежуточные вычисления. При работе со скрытым объектом скорость обработки данных значительно возрастает.

Ответьте, подтвердив ответы программно: какое имя получит созданная книга, какое имя у рабочего листа, как переименовать рабочий лист, как добавить рабочий лист, как перемножить содержимое двух ячеек на созданном рабочем листе и получить результат

Задание №3. Создайте две программы для работы с объектами класса *Excel.Chart* и *Excel.AddIn* (см. таблицу №1) Какие листы и сколько содержит образованное приложение Excel в этом случае. Разместите на листах какую-либо информацию.

Задание №4 . Наберите текст процедуры, используя Excel в качестве клиентского приложения, и поэкспериментируйте. Объясните действие каждого оператора.

```
Sub Prim_W_01()
'=========================================
'Доступ к объекту Word Application из Excel
'Dim ObjWord As New Word Application              = 1 =
Const wdAlignParagraphCenter = 1                 ' = 2 =
Set ObjWord = CreateObject("Word.Application")    ' =3=
   With ObjWord
      .Visible = True
      .Documents.Add
         With .Selection.Font
            .Bold = True
            .Size = 20
         End With
            With .Selection
               .TypeText "Внедяемся в Word из др.приложения"
               .ParagraphFormat.Alignment = 1 'wdAlignParagraphCenter
         End With
   End With
ObjWord.Quit

End Sub
```

Важным моментом в этом примере является то, что из-за позднего связывания программа не имеет доступа к библиотеке объектов Word, а поэтому необходимо определить в процедуре константу *wdAlignParagraphCenter*.

2.1.2. Раннее связывание.

Раннее связывание происходит на этапе компиляции, если тип объекта-переменной известен заранее. Перед написанием кода, использующего *Automation* необходимо сослаться *на библиотеку серверов Automation*. Для этого, находясь в редакторе VBA, следует выполнить команду меню *Tools/References* и в окне *Available References* (доступные ссылки) из списка библиотек выбрать те, которые необходимы. В нашем случае - это *Microsoft Word 12.0 Object Library* (см. рисунок 4.1).

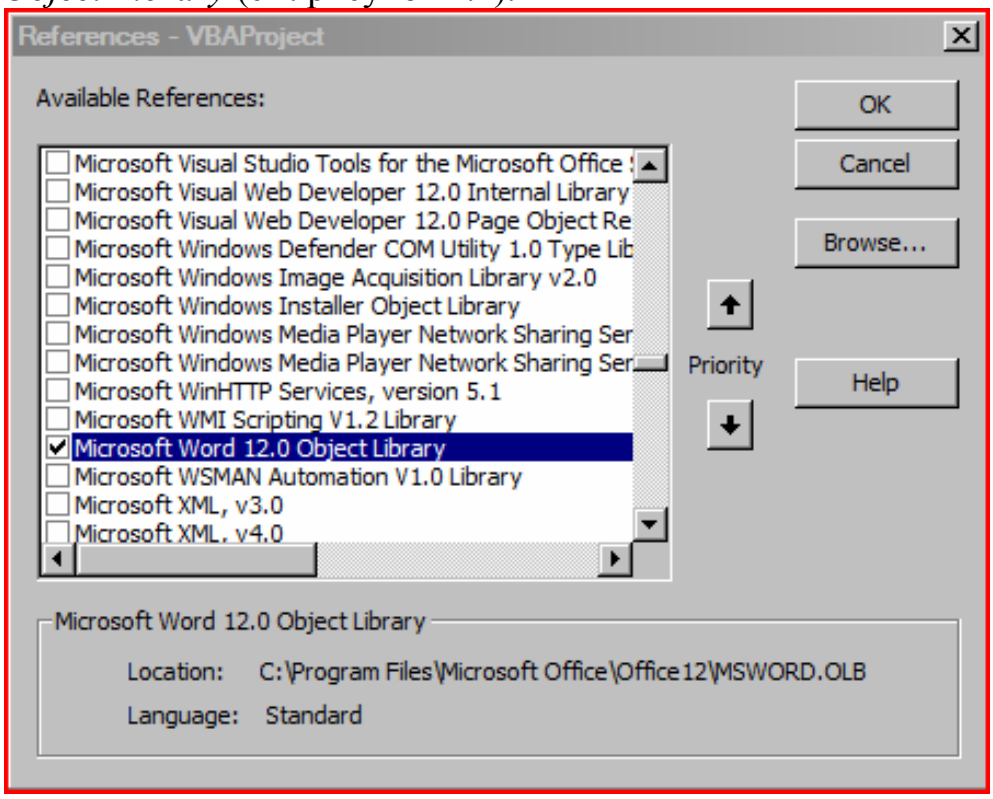


Рисунок 4.1. Окно установления ссылок

После установки ссылки нужно создать переменную, ссылающуюся на нужный объект из объектной модели данного приложения.

Для редактора Word такую ссылку задает оператор №1 в *Задании №4 (Prim_W_01)*. Ключевое слово ***New*** говорит о том, что нужно запустить новую копию приложения и поместить указатель на это приложение в объектную переменную. При этом второй и третий операторы не нужны.

Задание №5. Исправьте текст процедуры Prim_W_01 так, чтобы осуществить раннее связывание и убедитесь в ее работоспособности.

Задание №6. Создайте объекты из оставшихся классов из таблицы 1, присвойте соответствующим переменным ссылки на созданные объекты в клиентских приложениях, попробуйте осуществить какие-нибудь действия с этими объектами программным путем.

2.2. Использование функции *GetObject*

Функция используется для доступа к существующим объектам и имеет следующий синтаксис:

- ✓ *Для позднего связывания*

  Dim ObjName as Object
  Set ObjName=GetObject([PathName] [, Class])

PathName - путь к файлу. Если необходим доступ к объекту Application любого уже запущенного приложения – первый аргумент можно опустить. Например GetObject(, 'Excel.Application)
Для открытия и отображения документа имя файла, но не весь путь, следует поместить в квадратные скобки.

Class - имя_приложения (класс_объекта)

- ✓ *Для раннего связывания*

  Dim ObjName as Class
  Set ObjName=GetObject([PathName] [, Class])

Задание №7.. Создайте в корневом каталоге диска *C:* книгу с именем *"Книга1"* и в ячейку *"A1"* поместите любой текст. Наберите текст процедуры, используя Excel в качестве клиентского приложения, и поэкспериментируйте.

```
Sub Prim_W_02()
'================================
' Доступ к объекту Excel Sheet in Excel и др.
Dim Obj_Sheet As Object
Set Obj_Sheet = GetObject("C:\Книга1.XLS", "Excel.Sheet").ActiveSheet
MsgBox Obj_XLSheet.Range("Al").Value
Set Obj_Sheet = Nothing
End Sub
```

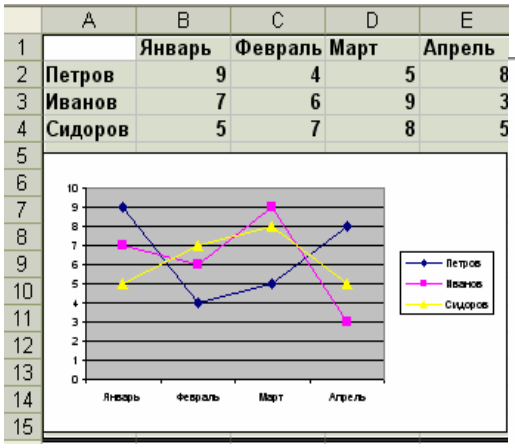
Сделайте так, чтобы открываемая книга была видна.
Добейтесь работоспособности программы без использования метода *ActiveSheet*
Получите объект *Application* уже запущенного экземпляра *Excel*.

3. УПРАВЛЕНИЕ СВЯЗАННЫМИ И ВНЕДРЕННЫМИ ОБЪЕКТАМИ.

Разговор о механизме (технологии) OLE был начат в начале этой главы. Отметим некоторые моменты этой технологии. Особенность ее состоит в том, что вместе со вставляемыми данными (например, таблицей Excel – это внедренный объект) в приложение-приемник (например, Word) , будет передана и сохранена в нем информация о приложении, создавшем внедренный объект.

Документ приложения-приемника называют контейнером. *При внедрении* все данные об объекте копируются в документ-контейнер. При редактировании объекта, расположенного в документе-контейнере в исходном файле изменений не происходит. Кроме того, изменения в исходном файле не влияют на внедренный объект. *При связывании* в документ-контейнер записывается информация о документе, из которого скопирован объект. При этом изменение документа-источника изменяет связанный объект.

Задание №8. Скопируйте объект Excel (часть листа, показанного на рисунке 4.2 в Word). После копирования объекта в буфер, вставляя его в документ Word, используйте "*Специальную вставку",* и не забудьте поднять флажок "*Связать".* Напишите процедуру, представленную рядом с рисунком. Проверьте, как она работает.




```
Sub Prim_E_W_01()
'=========================================
' Изменение данных Excel из Word
 Dim xlObj As Object
 Set xlObj = GetObject(, "Excel.Application")
        xlObj.Range("A2") = "Proba"
        xlObj.Range("B2") = 99
        xlObj.Range("B3") = 77
End Sub
```

Рисунок 4.2. Исследование связанных и внедренных объектов

Как будет работать программа если не "связывать"копируемый объект?
Что будет если скопировать объект как рисунок?

3.1. Использование семейства объектов *OLEObjects.*

В приложении-клиенте внедренный объект можно активизировать с помощью семейства ***OLEObjects*** , а затем отредактировать его на месте, используя *Automation*.

Задание №9. Создайте диаграмму в одной книге Excel и скопируйте ее в другую книгу. Во второй книге напишите процедуру, представленную ниже и убедитесь в том, что она работает.

```
Sub Prim_E_E_2()
'================================
   Dim objOLE As ChartObject
   Set objOLE = ActiveSheet.ChartObjects(1)
         objOLE.Activate
         MsgBox objOLE.Name
         MsgBox objOLE.Parent.Name
End Sub
```

Задание №10. Внедрите последний заголовок и абзац текущего документа в *Excel* на *Лист2*. Для этого сначала выделите фрагмент текста в Word-е., после чего вставьте его в Excel, используя команду *Правка/Специальная вставка/Вставить как Объект Документ Microsoft Word*. При этом должен получиться результат, представленный на рисунке 4.3. Причем, информация о внедренном объекте будет расположена в поле имени *«Объект 1»* и строке формул *(=ВНЕДРИТЬ "Word Document.12","")*.

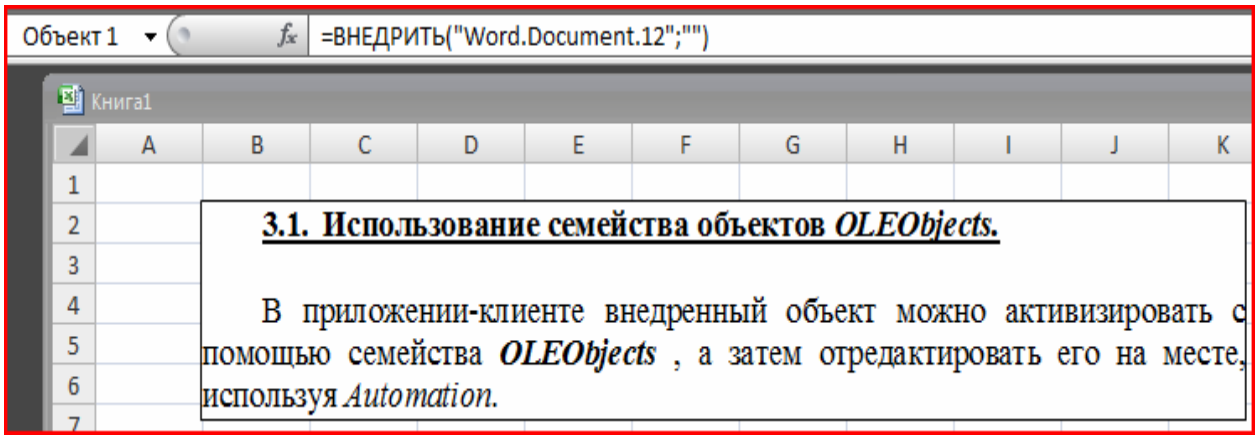


Рисунок 4.3. Внедренный в Excel объект из Word

Используя программу, представленную ниже, измените шрифт заголовка и выровняйте его по центру.

Результат работы программы изображен на рисунке 4.4

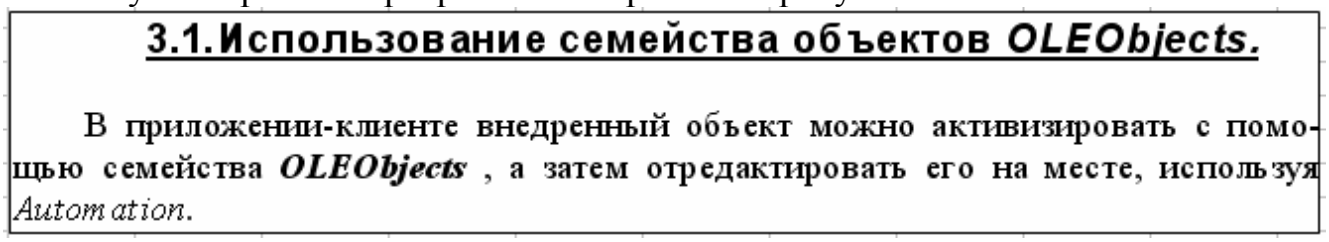


Рисунок 4.4. Изменение внедренного в Excel объекта

```
Sub Prim_W_04()
' Управление внедренным объектом
   Dim WordApp As Object, i As Integer
   Const wdAlignParagraphCenter = 1
   Const wdLine = 5
   Const wdExend = 1

   With Worksheets(2)
         .Select
        .OLEObjects(1).Activate
         Set WordApp = GetObject(, "Word.Application")
         With WordApp.Selection
              .HomeKey
             .MoveDown wdLine, 1, wdExend
              With .Font
                       .Bold = True
                       .Size = 16
                       .Name = "Arial"
              End With
             .ParagraphFormat.Alignment = wdAlignParagraphCenter
        End With
    Set WordApp = Nothing
    .Range("A1").Select
   End With
End Sub
```

Ответьте: зачем объявлены постоянные вначале программы? Что будет если это не делать? Что будет, если не использовать метод *Activate* для внедренного объекта? Что будет, если Word закрыт, а Вы запустили свою программу?

Объект *Selection* имеет методы *HomeKey* и *MoveDown,* которые использованы в примере. Оба имеют параметры. Поясните и покажите, что делают и как работают методы, а также, что определяют параметры этих методов.

Создайте еще один внедренный объект. Выделите у него вторую строчку и измените в ней шрифт.

3.2. Работа с функцией *GetOject* без запуска приложения-сервера.

В рассмотренном выше примере была использована функция ***GetOject***, причем для безошибочной работы программы было необходимо запустить *приложение-сервер*. Последнего можно избежать, используя следующую, приведенную ниже процедуру.

Задание №11. Запустите процедуру на выполнение и убедитесь, что метод Activate служит для открытия документа и редактирования его по месту. Для экспериментов используйте предыдущий пример, где Excel является клиентским приложением с двумя внедренными объектами.

```
Sub Prim_W_05()
    '=============================================
      'Управление внедренным в  Excel  из  Word-а объектом
      'программу выполняем из  Excel
      Dim objOLE As OLEObject
      Set objOLE = ActiveSheet.OLEObjects("Object 2")
      objOLE.Activate          'редактирование на месте
      'objOLE.Verb (xlVerbOpen)   'редактирование в отд.окне
End Sub
```

Контроллер *Automation* при работе с внедренными объектами поддерживает и метод *Activate*, в чем Вы только что убедились, и метод *Verb* (аналогичные ему *DoVerb* –для Word и PowerPoint). Метод для Excel имеет две константы *xlVerbPrimary* (в этом случае он работает аналогично методу Activate) и *xlVerbOpen* (в этом случае открывается отдельное окно для редактирования объекта). Измените код программы и проверьте это.

Задание №12. По аналогии с примером *задания №11* можно управлять нарисованным в *Paint*, а затем помещенным в *Word* рисунком. Такой рисунок приведен рядом с текстом программы.

```
Sub Prim_W_P_1()
'=====================================
   Dim objOLE As Shape, aa As String
   'Set objOLE = ActiveDocument.Shapes(1)
   'aa = objOLE.Name

   Set objOLE = ActiveDocument.Shapes("Object 4")
   objOLE.Activate
   'objOLE.OLEFormat.DoVerb (wdOLEVerbOpen)
End Sub
```

Наберите текст и запустите процедуру на выполнение. Объясните работу каждого оператора.

4. ТЕХНОЛОГИЯ СОЗДАНИЯ ЭЛЕМЕНТОВ УПРАВЛЕНИЯ *ACTIVEX.*

Кроме стандартных элементов управления (текстовых полей, меток и т.д.) в проектах можно использовать дополнительные *элементы управления ActiveX*. Обычно они расположены в файлах с расширениями *.ocx, .dll, .exe*.

Для того чтобы в проекте на экранной форме появились дополнительные элементы управления нужно выполнить следующее. Находясь в редакторе VBA, после вызова пользовательской формы выполнить команду меню *Tools/Additional Controls* или после щелчка правой кнопкой мыши по панели инструментов команду *Additional Controls*. После этого в рабочем окне *Additional Control*s поднять флажки у соответствующих управляющих элементов.

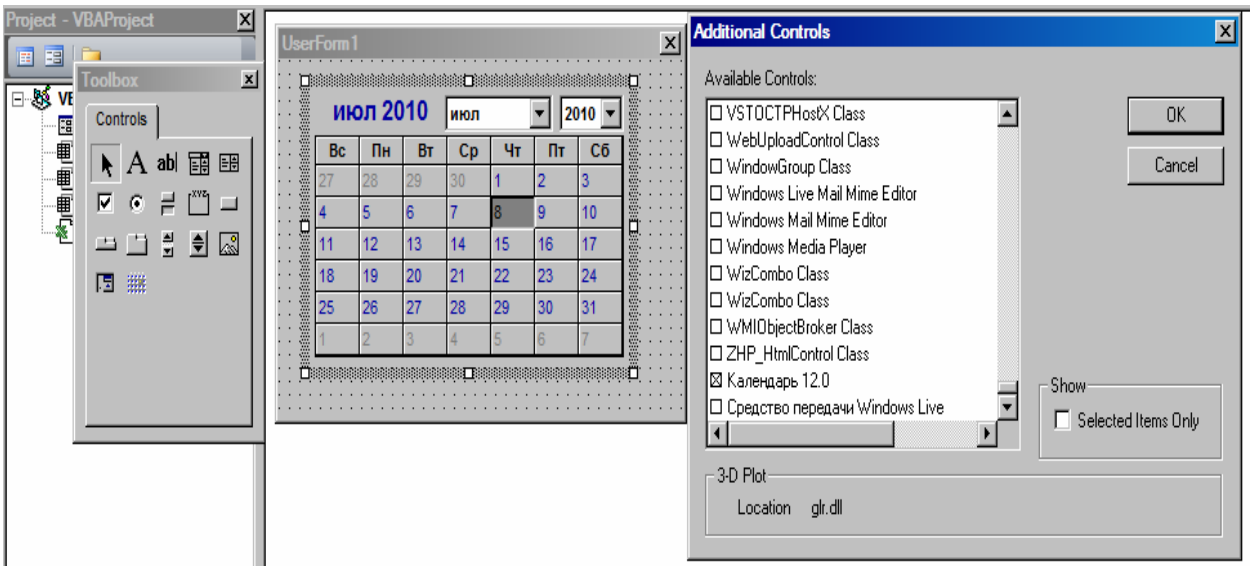


Рисунок 4.5. Вставка дополнительного элемента управления

Кроме этого, на рабочие листы Excel командой меню Вставка\Объект можно вставить листы документа Word, документы Mc.Visio и т.д.

Задание №13. Разместите указанные элементы в рабочем документе Word. Напишите программный код, который позволял бы обращаться к этим элементам.

5. СПРАВОЧНАЯ СИСТЕМА В OFFICE

1. ОСНОВНЫЕ ПОЛОЖЕНИЯ.

Никогда еще не было такого, чтобы сложная информационная система работала без ошибок и у пользователя не возникало вопросов к ней и разработчикам. Одним из возможных вариантов решения проблем служит использование справочной системы или справки.

Такие пакеты как *Office 2007 Developer Edition* и *Visual Basic 6* включают инструментальные средства для создания собственных справочных систем по образцу справки, которая используется в Office. В поставку этого программного обеспечения входит утилита *HTML Help Workshop*, позволяющая создать справочную систему в виде скомпилированного файла. Файлы, составляющие разделы этой системы, содержат текст, графику и другие элементы, которые могут появляться в разделах справки.

В комплект стандартной поставки VBA и других версий Office указанный компонент не входит, но его можно установить, используя Интернет, по адресу: *http://www.microsoft.com/workshop/autor/htmlhelp/default.asp.*

При установке *Microsoft HTML Help* должны появиться следующие инструментальные средства:

- ✓ *HTML Help Workshop (hhw.exe)* – основное средство, позволяющее создавать отдельные файлы справки и объединять их, создавая один файл,
- ✓ *Microsoft HTML Help Image Editor (Flash.*exe) - средство, позволяющее работать с графическим материалом,
- ✓ *HTML Help Compiler (hhc.exe)*– компилятор файлов.

Основу справочной системы составляют темы, оформленные как поясняющий текст в виде *html-файлов*. Эти файлы могут быть созданы в любом редакторе, позволяющем работать с таким форматом, в *Word, FrontPage, BestAddress HTML Editor 2004 Professional* и т.д.. При этом разработчики могут использовать весь дизайн Web- страниц.

Создаваемая справочная система должна иметь хотя бы *оглавление* (его могут называть – *таблица содержания*). Одно это позволяет пользоваться ей как справочником. На этапе его создания определяется иерархическая структура оглавления, а также все то, что должно быть включено в систему. Первый, черновой вариант оглавления следует создать на бумаге, обсудив его с разработчиками информационной системы.

Формально, непосредственное создание оглавления происходит в системе *HTML Help Workshop* только после того, как создан его черновой вариант и его разделы -*html-файлы*. Недостающие разделы можно добавить потом, а часть из них удалить или изменить при необходимости.

Таблицы содержания используют следующие элементы: *книги и страницы*. В *книги*, вместе со страницами, могут быть вложены другие книги, по-

этому каждая книга может иметь структуру дерева, а само оглавление выглядит как лес.

Создавая справочную систему, следует иметь в виду:

- ✓ для создания сложных разделов следует использовать такие редакторы как *FrontPage, BestAddress HTML Editor* или какой либо другой, *а не Word*. В данном случае использовался *Adobe Dreamweaver.*

Это вызвано тем, что не все файлы, созданные в Word, будут без ошибок компилироваться, и просматриваться в *HTML Help Viewer*. Могут быть не видны рисунки.

- ✓ Имена файлов следует писать на латинице. Это связано с тем, что в ряде случаев *Viewer* может не понимать русского языка, если использовать имена файлов в гиперссылках.
- ✓ При создании файлов справочной системы следует установить кодировку «Кирилица (Windows). Использование других кодировок приведет к тому, что вместо русского текста получаться нечитаемые символы.
- ✓ Расширение *html* следует набирать маленькими буквами.
- ✓ Создаваемые файлы лучше всего хранить в одном каталоге.

Справочная система может быть достаточно сложной. По мере необходимости ее можно усложнять и совершенствовать, используя создание индексов, ссылок, мастер ответов –*Answer Wizard* и т.д. Создадим сначала самый простой вариант такой системы. Для ее создания необходимо выполнить ряд шагов–этапов, основными из которых являются:

1. Создание исходных файлов справочной системы:
    а. html-файлов, описывающих отдельные темы;
    б. файла проекта справочной системы;
    в. таблицы содержания (оглавления), задающую иерархическую структуру справочной системы;
    г. индексного файла для быстрого поиска информации (в принципе, на первом этапе это можно опустить);
2. Компиляция файла проекта, с созданием одного файла справочной системы.
3. Отладка системы.
4. Создание системы запуска созданного приложения.

Процедура создания справки в основном формализирована. При этом не решается никаких научных задач – есть только заданная последовательность действий, которая приводит к определенному результату. Отметим, что эта последовательность не является жестко заданной. Что-то можно сделать сначала, что-то потом, а в некоторых случаях наоборот, но, тем не менее, общая канва остается постоянной.

Создадим простую справочную систему, используя как основу данное пособие, и выполнив указанные выше этапы.

2. СОЗДАНИЕ *HTML-ФАЙЛОВ*, ОПИСЫВАЮЩИХ ОТДЕЛЬНЫЕ ТЕМЫ (подготовительная работа)

Прежде всего, создайте исходный вариант оглавления справочной системы. Для учебных целей была взять часть пособия «Информационные технологии в управлении» [2]. Создавая *таблицу содержания* желательно воспроизвести его иерархическую структуру. На этом этапе отметьте, какие файлы будут книгами (K), какие разделами (P) и подберите для них соответствующие имена. Например:

Таблица 5.1 – Таблица содержания

	Раздел/ Книга	***Имя файла***
Справочная система пользователя	***K***	SSP_k.html
Титульный лист	*P*	Tit_r.html
ОГЛАВЛЕНИЕ.	***K***	Ogl_k.html
Введение	*P*	Vved_r.html
7. Графические элементы в Office	***K***	LR7_k.html
1. Основные положения.	*P*	Raz_71.html
2. Объект Shapes.	*P*	Raz_72.html
2.1. Методы объекта Shapes.	*P*	Raz_721.html
2.2. Свойства объекта Shapes.	*P*	Raz_722.html
3. Объект Shape	*P*	Raz_73.html
3.1. Методы объекта Shape	*P*	Raz_731.html
3.2 Свойства объекта Shape	*P*	Raz_732.html
8. Построение диаграмм.	***K***	LR8_k.html
9. Создание приложения для автоматизации заполнения бланка «Расчета структуры цены»	***K***	LR9_k.html
Литература.	*P*	Lit_r.html
Разное.	*P*	Razn_r.html
1. Рисунки.	*P*	Ris_01_r.html
2. Ссылки. (создайте пока пустой файл !)	*P*	Ssilk_r.html

Для создания разделов справки используйте редактор HTML-файлов, всю свою фантазию и умение. Не забывайте вводить имена заголовков ***<title>.*** Старайтесь давать им емкие и короткие названия, состоящие из одного, двух, максимум трех слов. В дальнейшем это пригодиться при организации системы поиска информации и создании индексного файла. Создавая HTML-файлы, переносите в них готовые части пособия, для которого создается справка.

Код файла Ogl_k.html представлен ниже на рисунке 5.1

Создайте файлы с именами, представленными в таблице 5.1

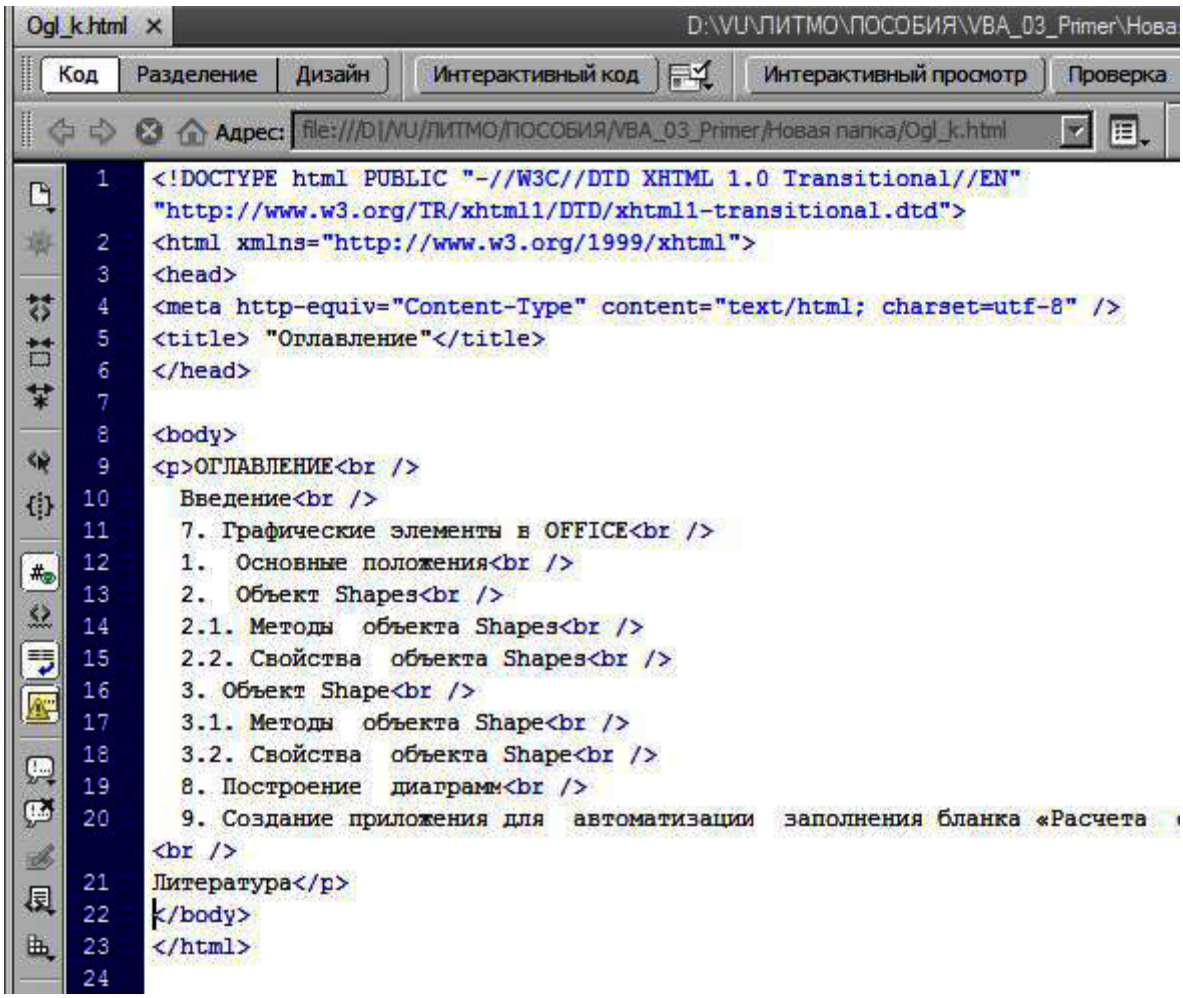


Рисунок 5.1. Код файла Ogl_k.html

3. СОЗДАНИЕ ПРОСТОЙ СПРАВОЧНОЙ СИСТЕМЫ

3.1 Создание файла проекта.

На деталях процесса создания справки останавливаться не будем. Это связано с тем, что интерфейс каждого этапа достаточно понятен и соответствует здравому смыслу и интуитивным представлениям.

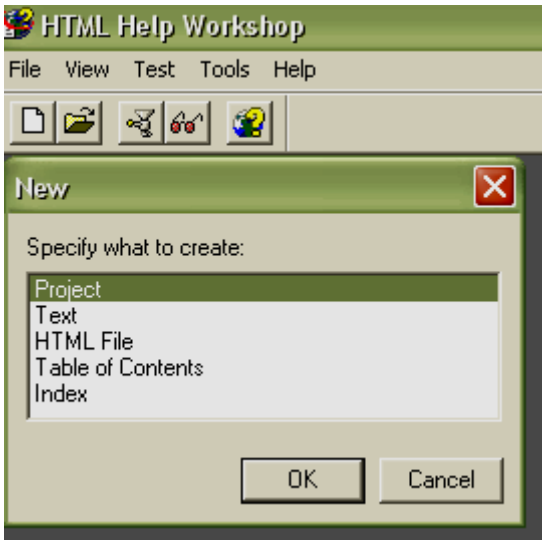


Рисунок 5.2. Окно диалога

Запустите программу *HTML Help Workshop (hhw.exe).*

✓ Выполните команду меню *Файл\New.*

✓ В Окне диалога *New* (см. рисунок 5.2) выберите вариант *Project* и нажмите кнопку *OK.* Должно появиться окно *New Project.*

В принципе, в окне *New* есть команды, которые вызывают блокнот для создания текстового файла, простой редактор HTML-файлов и т.д., но они пока не нужны.

✓ В окне *New Project* щелкните по кнопке *Далее*, отказавшись от конвертации файла справки.

✓ В новом окне *New Project- Destination* введите имя файла новой справочной системы, указывая полный путь: *E:\VU_HELP\ALL\Spravka_00.hhp.* При этом удобно использовать кнопку *Browse.* Щелкните по кнопке *Далее.*

✓ В открывшемся окне *New Project- Existing Files* выберите форматы заготовок файлов, которые предполагается включить в состав справочной системы. Так как темы справки хранятся в виде *html- файлов,* поднимите флажок *HTML files* как на рисунке 5.3.

Рисунок 5.3. Рабочее окно New Project- Existing Files

✓ Следующим должно появиться окно *New Project- HTML Files* (рис.20.3.). В нем, используя кнопки *Add* и *Remove,* необходимо в текстовое поле ввести имена ранее созданных *HTML-файлов,* которые связаны с темами справочной системы.

После щелчка по кнопке *Add* появиться окно диалога *Открыть.* Выделение в нем необходимых файлов и щелчок по кнопке *Открыть* переносит их имена в текстовое поле окна *New Project- HTML Files.*

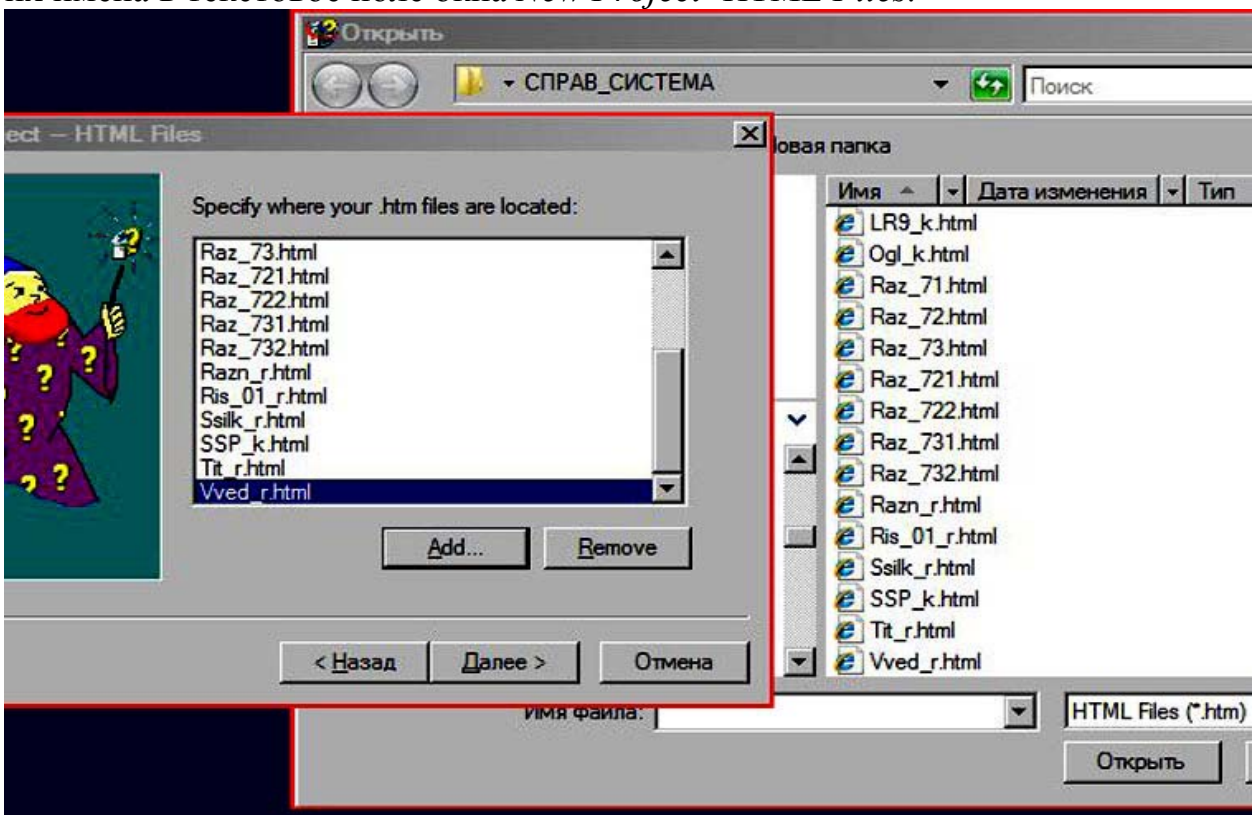


Рисунок 5.4. Наполнения справочной системы файлами

Нажав кнопку *Далее*, а затем в следующем окне *New Project-Finish* кнопку *Готово* –получаем новый созданный файл проекта. Диалоговое окно *HTML Help Workshop* примет вид, представленный на рисунке 5.5.

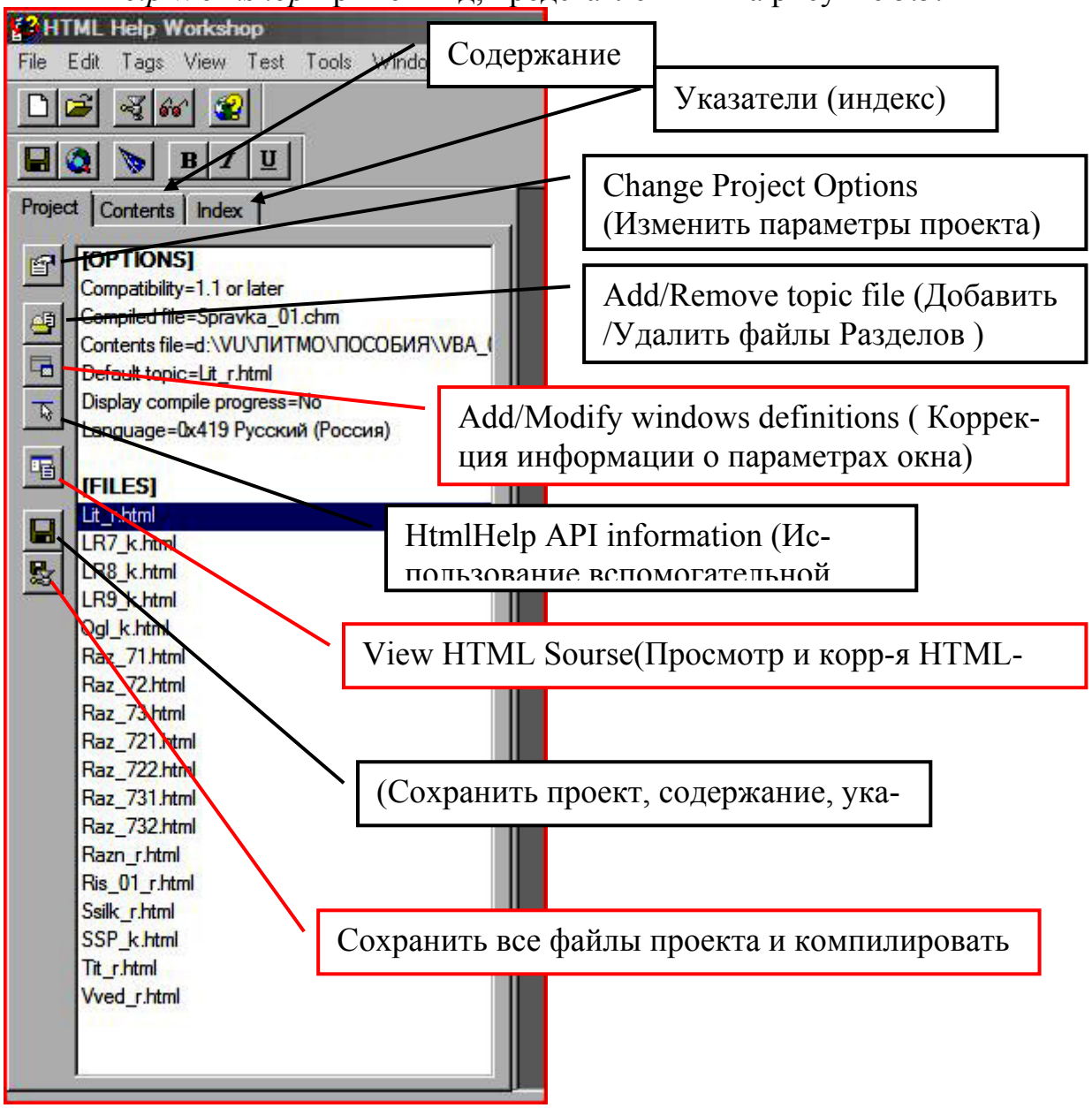
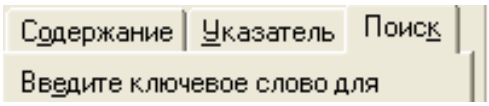


Рисунок 5.5 – Окончательный вид окна HTML Help Workshop

3.2 Создание содержания справочной системы.



Окно справочной системы обычно содержит три вкладки: *Содержание, Указатель и Поиск*.

Процесс создания содержания сводится к следующему:

1. Перейдите на вкладку *Contents*. Файл содержания еще не создан, поэтому появится окно диалога с предложением создать новый файл (рисунок 5.6):

Рисунок 5.6

Щелкнув по кнопке *ОК*, задав в окне *Сохранить как,* путь и имя файла, согласившись с предлагаемым расширением файла *HHC*, запомните его. Пока этот файл содержания пустой и область раздела *Contents* окна *HTML Help Workshop* не содержит никакой информации.

2. Остаемся на вкладке *Contents*.

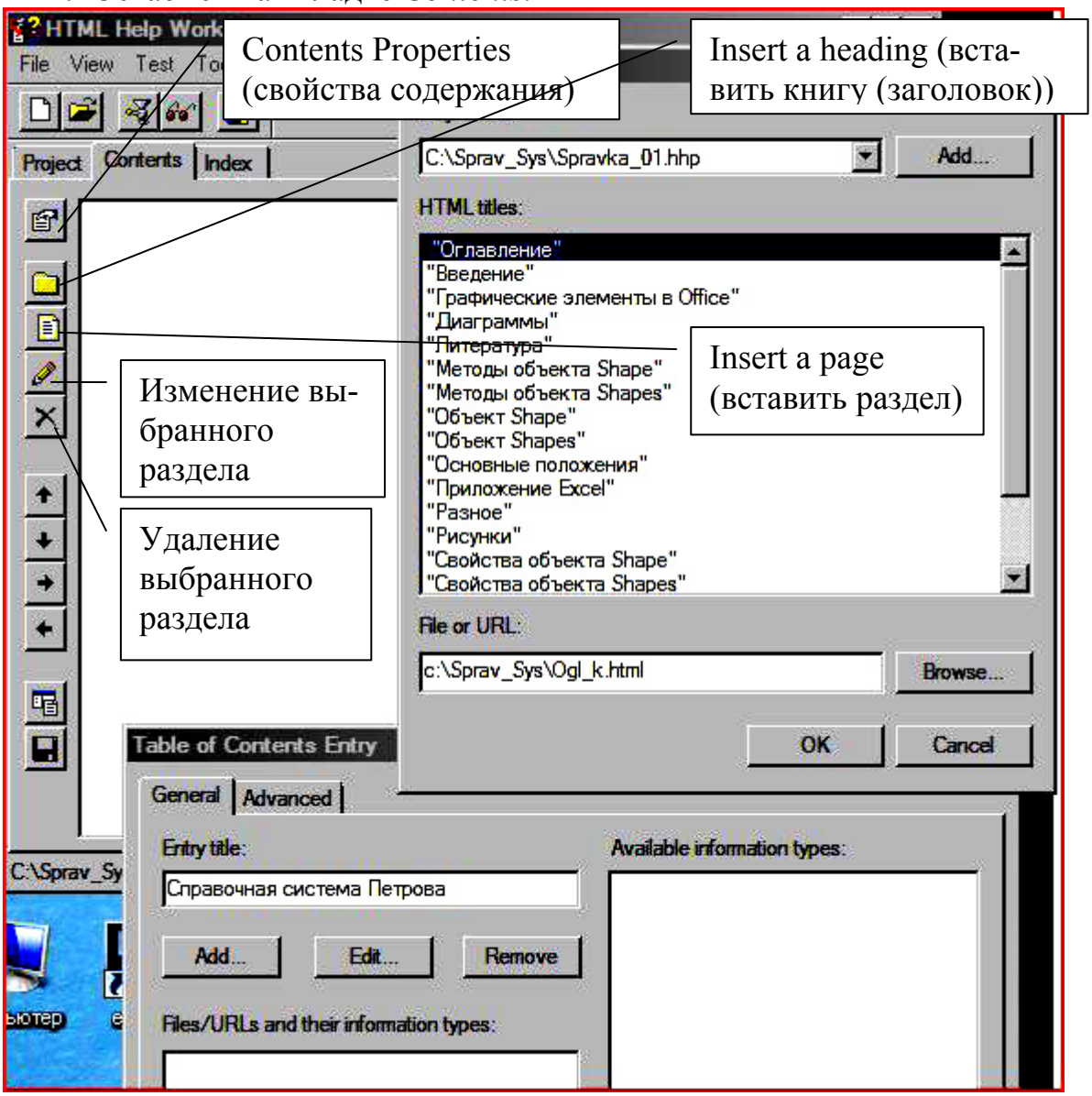
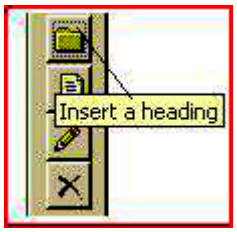


Рисунок 5.7. Окна для формирования содержания справочной системы

На рисунке 5.7 изображены все окна диалога и кнопки, определяющие режимы создания *содержания* справки.

В соответствии с данными таблицы 5.1 и информацией представленной на рисунке 5.7 создайте книги и разделы.



3. Создайте книгу.

Щелкните по кнопке *Insert a heading*. Это вызовет появление окна *Table of contents entry* (см рис.5.7).

3.1. В этом окне *Table of contents entry* присутствуют три кнопки:

Add - добавить раздел или книгу;

Edit - редактировать данные (можно заменить один раздел на другой и т.д.);

Remove- удалить книгу или раздел.

3.2. Щелкните по кнопке *Add.* Появится диалоговое окно *Path or URL.*

3.3. Щелкнув по кнопке *Add*, но уже в окне *Path or URL,* выберите имя проекта *c:\Sprav_Sys\Spravka_01.hhp*(см. рис.5.7.).

3.4. В поле ввода *HTML titles* окна *Path or URL* (рис.20.6.) выделите первую *Книгу (Раздел*). Ее название – *"Справочная система".*

Тут же в поле *File or URL* появится информация о адресе файла, содержащего данную тему.

На рисунке 5.7 выделена книга «Оглавление» и адрес файла - *c:\Sprav_Sys\Ogl_k.html.*

3.5. Нажмите на кнопку *OK* и вернитесь в окно *Table of contents entry.*

Здесь необходимо заполнить текстовое поле *Entry tytle.* В это поле можно ввести любую информацию. Очевидно, информация должна отражать суть содержания раздела или книги. Поэтому следует либо повторить имя заголовка файла, которое было отражено в поле *HTML titles* окна *Path or URL*, либо несколько его модифицировать. Вместо текста - *Справочная система* напишите, *Справочная система Петрова,* например.

3.6. Щелкните по кнопке *OK* и получите первую запись в оглавлении. Запись появится в левом поле окна *HTML Help Workshop* (рис.5.7*).*

4. На следующем этапе создайте в содержании раздел с именем *"Титульный*". Он содержится в файле *Tit_r.html.* Для создания раздела выполните действия п.п. 3.1-3.6 с тем отличием, что изначально следует щелкнуть по кнопке *Insert a page* в окне *HTML Help Workshop* (рис.5.7).

5. Повторяя рассмотренные действия многократно, получите все оглавление.

6. Используя кнопки со стрелками, добейтесь нужной структуры оглавления и получите вариант таблицы содержания, представленный на рисунке 5.8.

7. Сохраните измененный вариант проекта. После этого должен появиться новый файл *Table of Contents.hhc*, содержащий оглавление.

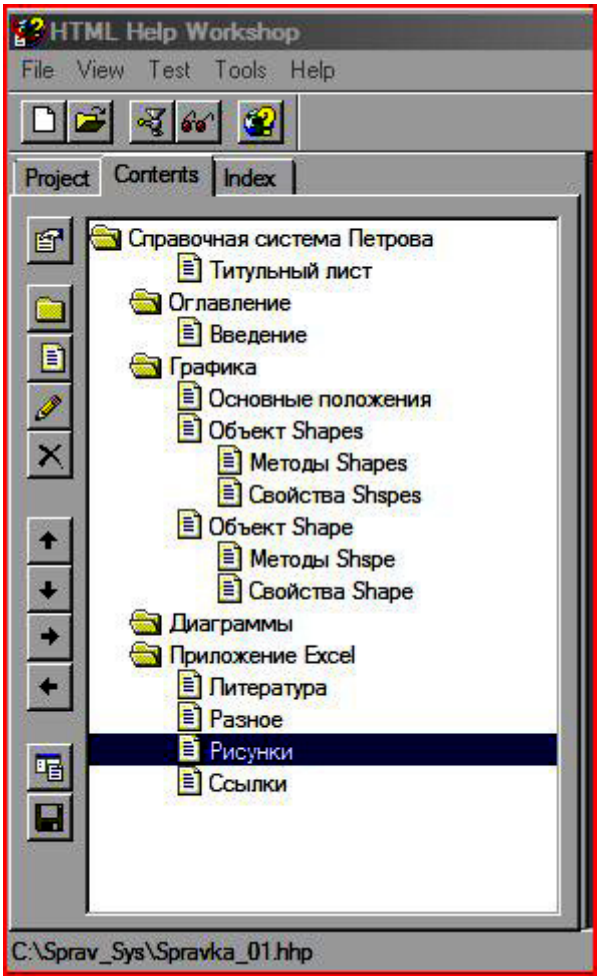
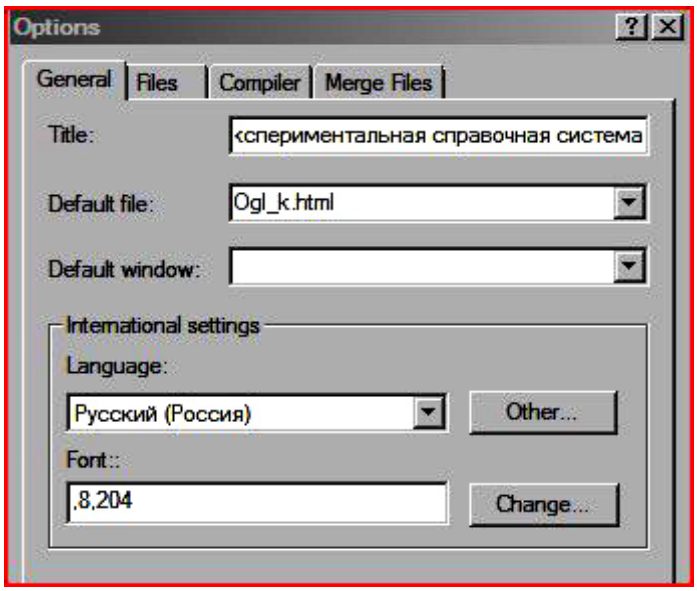


Рисунок 5.8. Окончательный вариант оглавления

3.3 Настройка параметров справочной системы



Щелкните по кнопке *Change project options*, вызовите окно *Option*, представленное на рисунке слева, и введите или укажите требуемые параметры проекта.

На вкладке *General (Общие)* введите заголовок окна справочной системы *(Title)*.

Укажите файл темы и окно, которые будут выбраны при ее открытии. (*Default file (window))*

На вкладке Files указано расположение файлов справочной системы-*Compiled file*, файлов с указателями –*Index file* , и содержанием –*Contents file.* Некоторые из этих файлов появятся позже.

Параметры компиляции можно установить на вкладке Compiler.

3.4 Компиляция и тестирование справочной системы

Для того, чтобы компилировать файл справки, следует выполнить команду меню *File\Compile* окна *HTML Help Workshop* или щелкнуть по кнопке *Save all project files and compile*.

Для просмотра созданного файла справочной системы выполните команду *View\Compiled file*…. Результат выполнения такой команды представлен на рисунке 5.9.

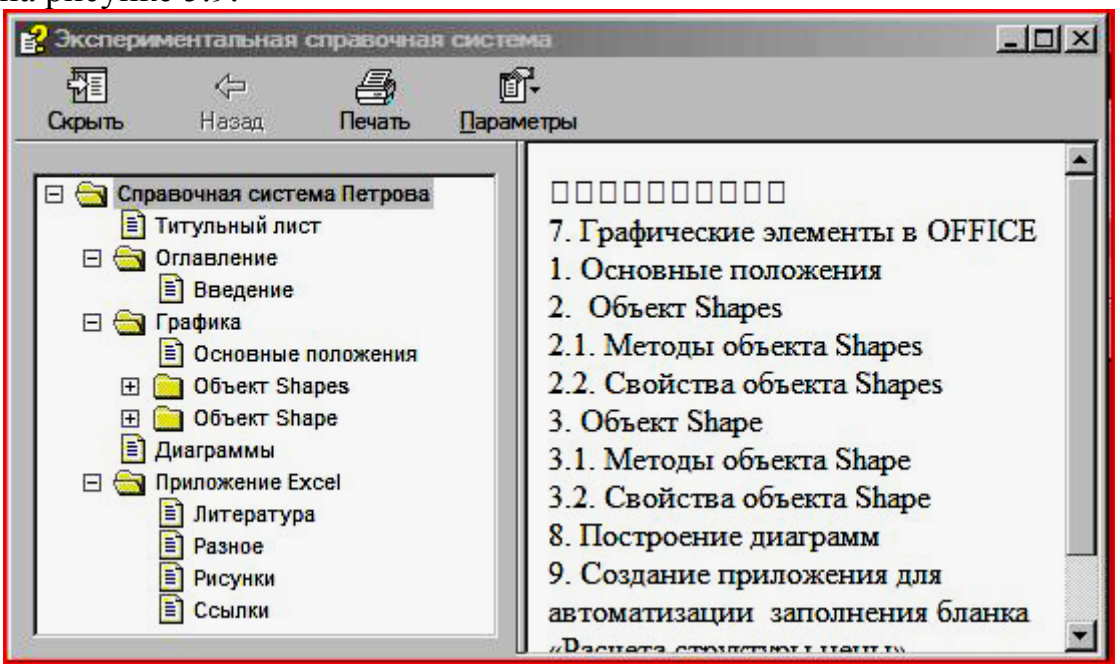


Рисунок 5.9. Исходный вариант справочной системы.

4. МОДИФИКАЦИЯ И УСЛОЖНЕНИЕ СПРАВОЧНОЙ СИСТЕМЫ

Созданная система обеспечивает поиск нужной информации по содержанию. Усложните созданную систему. Для этого рассмотрим организацию других способов поиска информации.

4.1 Создание ключей (индексов) для поиска информации.

В этом случае, с каждым разделом связывается некоторое множество индексов (ключевых слов или фраз). Задавая индекс или выбирая его, можно получить доступ ко всем темам, связанным с ним.

1. Сначала создайте пустой индексный файл. Для этого перейдите на вкладку *Index* и в окне диалога *Index Not Specified* согласитесь с предложенным по умолчанию вариантом создать новый индексный файл –*Create a new index file*. Указав каталог и имя (оставьте *Index.hhk*), запомните его.

2. Создайте ключи (указатели). Для этого, щелкнув по кнопке *Insert a keyword*, откройте окно диалога *Index Entry*. Его вид подобен окну *Table of contents entry* (см рисунок 5.6), которое использовалось при соз

дании оглавления. Отличие состоит только в том, что вместо поля *Entry title*

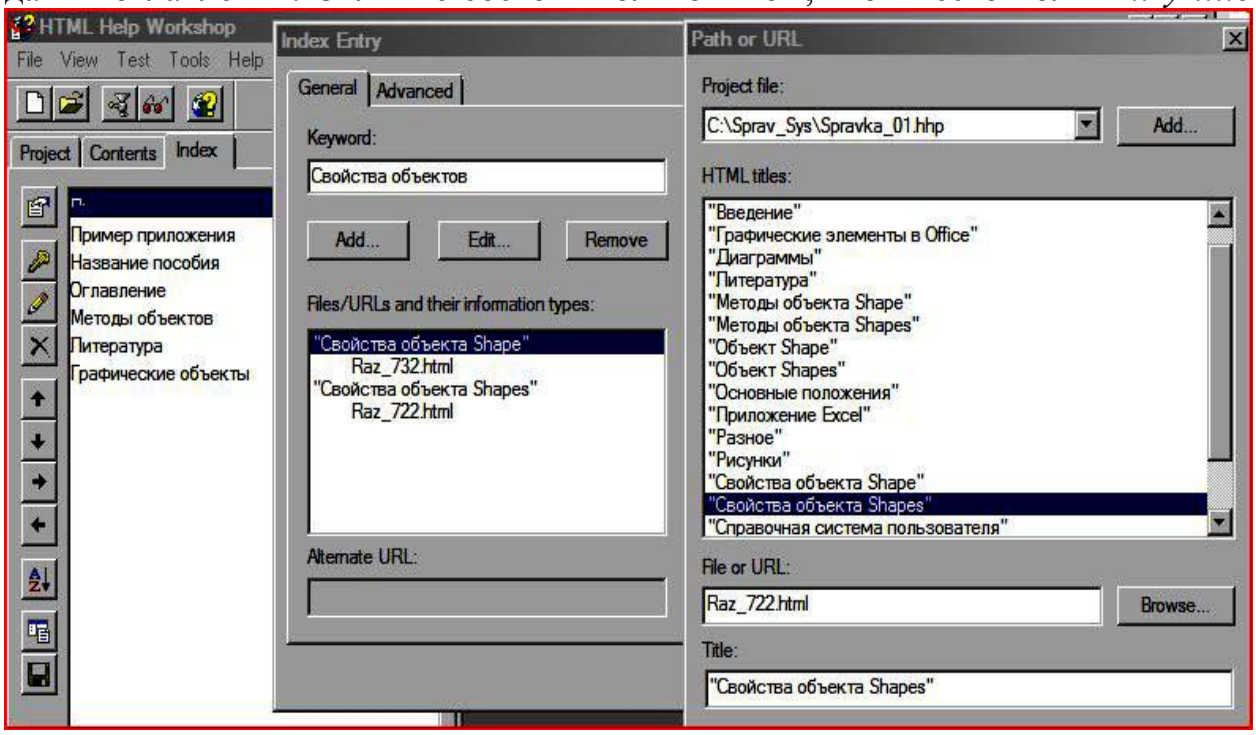


Рисунок 5.10. Вид рабочих окон при создании индексов

в новом окне использовано поле *Keyword*. В это поле следует вводить *ключевые фразы (ключи*), после чего, нажав кнопку *ADD*, выбрать имена разделов, которые будут связаны с ключом. С одним ключом можно связать несколько разделов.

Технология подобна той, которая использовалась при создании таблицы содержания. Если в процессе работы требуется что-то изменить в индексном файле, то необходимо подсветить ключевое слово, щелкнуть по кнопке с карандашном, и откорректировать данные. Ключевые слова и файлы, связанные с ними, представлены в таблице 5.2., а вид окна с ключами на рисунке 5.10.

Таблица 5.2. Таблица для формирования индексов

Авторы	Tit_r
Графические объекты	LR7_k, Raz_71, Raz_72, Raz_73
Литература	Lit_r
Методы объектов	Raz_721, Raz_731
Название пособия	Tit_r
Оглавление	Ogl_k
Пример приложения	LR9_k
Рисунки	Ris_01
Свойства объектов	Raz_722, Raz_732

3. Сохраните файлы и проект. Перекомпилируйте проект. Запустите файл на просмотр. Новый вариант справочной системы представлен на рисунке 5.11.

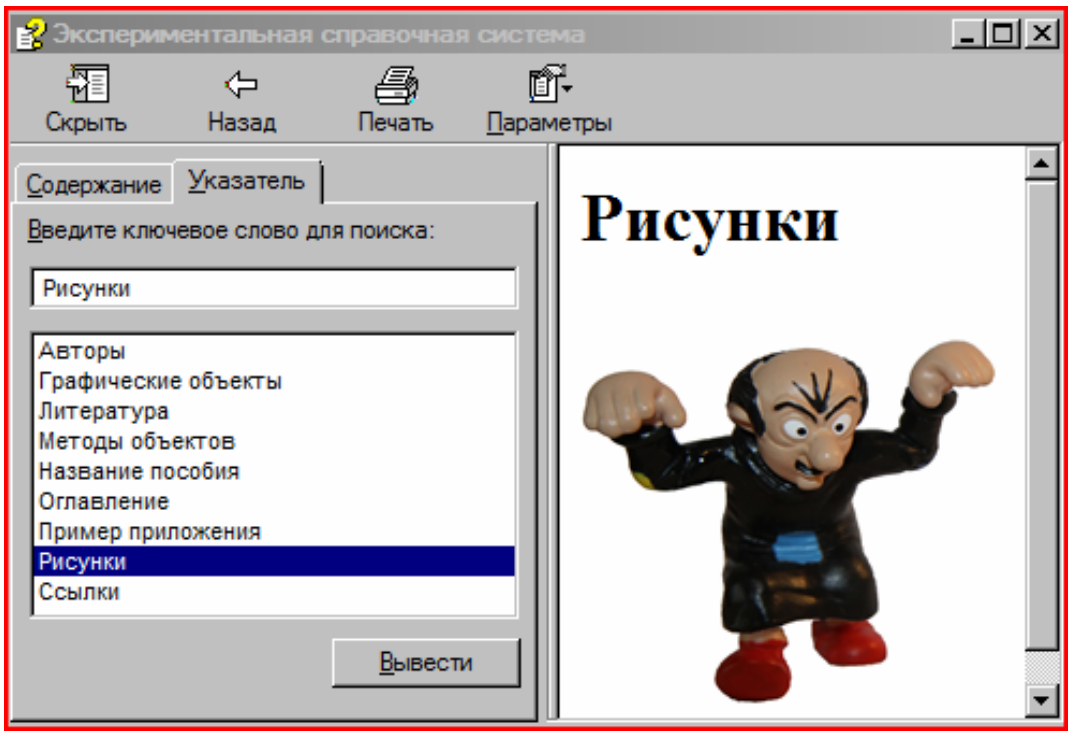


Рисунок 5.11. Справочная система с поиском

Теперь эта система позволяет искать темы по ключевым фразам. Попробуйте.

4.2 Создание справочной системы с полнотекстовым поиском.

По отношению к поиску по указателям полнотекстовый поиск является более мощным инструментом, позволяющим найти любой раздел, включающий искомый текст. Но такой поиск требует больших затрат памяти и времени. В данном варианте поиск ведется на точное соответствие слов и фраз.

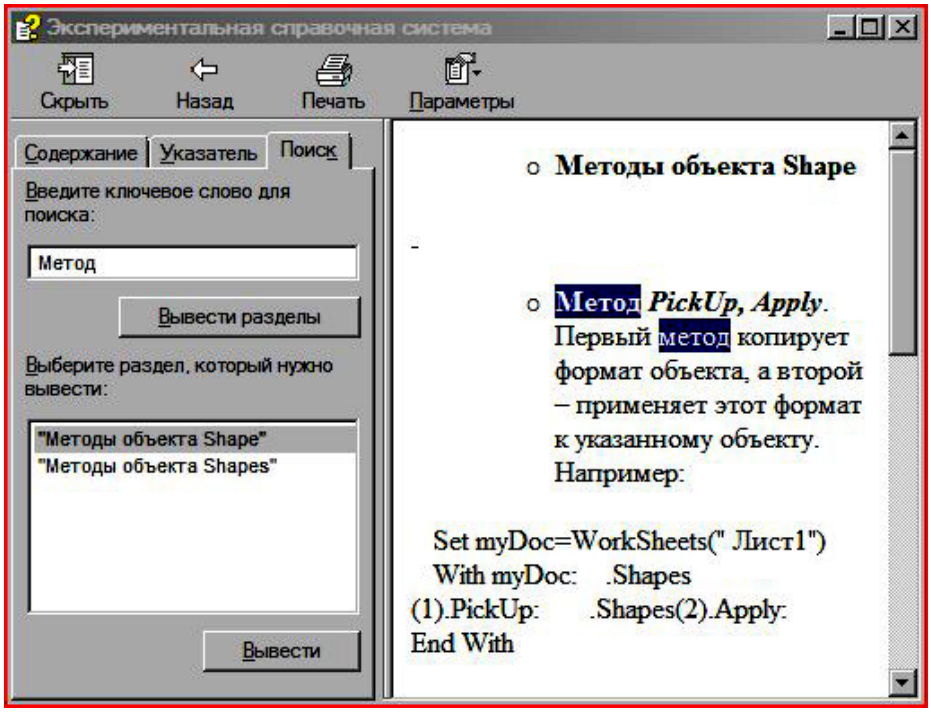


Рисунок 5.12. Окончательный вид справочной системы

Для изменения справочной системы, добавление в нее полнотекстового поиска в окне свойств проекта, на вкладке *Compiler* поднимите флажок *Compile full-text search information.* После компиляции в справке появится еще одна вкладка *Поиск*. Вариант справочной системы с использованием такого поиска показан на рисунке 5.12.

Создайте справочную систему с полнотекстовым поиском и покажите как он работает.

4.3. Создание расширенного варианта полнотекстового поиска.

Поиск, который может производиться по образцу, причем, когда выражение задающее образец может включать шаблоны - заменители (**?** и *****), называют расширенным поиском.

Для включения этого вида поиска в состав справочной системы следует:

- выбрать вкладку *Project*,
- щелкнуть по кнопке *Add/Modify window definitions,*
- задать имя окна –любое имя на латинице и нажать кнопку *OK,*
- в диалоговом окне *Window Types* перейти на вкладку *Navigation Pane,*
- включить флажок *Search* и связанный с ним флажок *Advanced ,*
- если вы поднимите флажок *Favorites,* задающий папку *Избранное,* то в окно справки будет добавлена еще одна вкладка – *Избранное*, и пользователь, не покидая справочную систему сможет работать с этой папкой.

5. ЗАПУСК СПРАВОЧНОЙ СИСТЕМЫ

После того, как проектирование справочной системы закончено, выполнена компиляция и создан выходной файл (в данном случае - *Spravka_01.chm*), необходимо написать макрос и связать его с какой-нибудь кнопкой или событием.

В состав *Microsoft HTML Help* входит программа *HTML Help executable program (hh.exe)*, предназначенная для запуска и выполнения справочного руководства, которое имеет вид скомпилированного html-файла.

Для запуска справочной системы следует написать макрос . Текст макроса представлен ниже.

Наберите этот текст и запустите справку.

```
Public Sub Spravka()
'Запуск справочной системы
    Dim Name_F As String, b_doub As Double
    Name_F = "C:\Sprav_Sys\Spravka_01.chm"
    b_doub = Shell("hh.exe" & Name_F, vbNormalFocus)
    'Application.Help "C:\Sprav_Sys\Spravka_01.chm"
End Sub
```

В принципе в процедуре можно использовать и метод Help, но в ряде случаев он работает неустойчиво и приводит к ошибке. Положительного результата в этом случае можно добиться, расположив рабочий файл и файл справки в одной папке.

6. ЗАДАНИЕ

Создайте свой вариант справочной системы.

В систему, помимо всего прочего, обязательно включите файл с рисунками разного формата и файл "*Ссылки"*.

В последнем файле используйте ссылки на сайты в Интернете и свои HTML-файлы, созданные для справочной системы.

Запустите созданную справку в Word, Excel, Access.

6. ПРИМЕРЫ И ПРАКТИЧЕСКИЕ ЗАДАНИЯ

Задание № 0 Откройте одну книгу из другой

В рабочей папке *"WORK_01"* создайте 2 файла с именами *"Книга_А"* и *"Книга_В"*, причем во втором (*"Книга_В"*) дайте рабочему листу имя *"Особый"*.

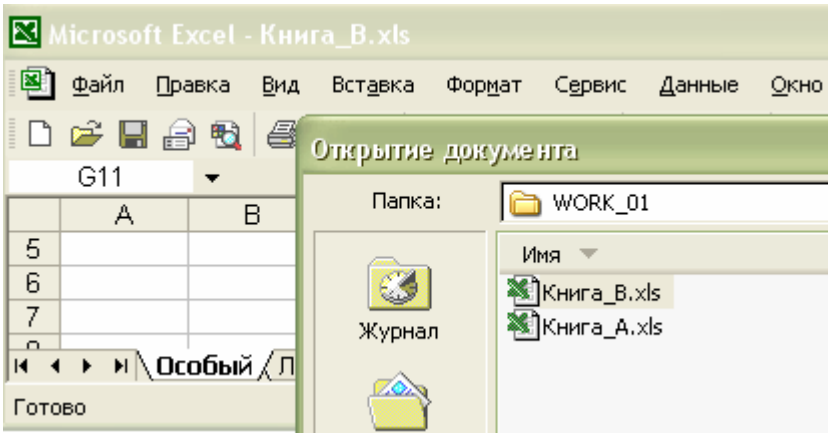


Рисунок 6.1.

“Книга_А” должна содержать команду пользовательского меню, открывающую *“Книгу_В”*

“Книга_В” должна включать автомакрос *"Auto_Open",* который должен запускаться при ее открытии (содержание макроса пусть будет любое).

В том случае, если *“Книга_В”* отсутствует в текущем каталоге, программа сгенерирует ошибку. Вы должны обработать ее, и с помощью метода *Application.GetOpenFilename* обеспечить возможность поиска и открытие файла "Книгу_В" в других папках.

Если *"Книга_В"* найдена, проверьте (при ее открытии), есть ли у нее рабочий лист с именем "Особый". Если такого листа нет, следует выдать сообщение о том, что открытый файл не является искомым, его следует закрыть и продолжить поиск или закончить работу.

Задание №1 Создайте автомакросы:

Auto_Open и Auto_Close.

Книгу с этими автомакросами назовите *"Книга_С"* . Запомните и закройте ее .

Добейтесь того, чтобы при открытии рабочей книги *"Книга_С*" из какой-нибудь другой рабочей книги, выполнялись:

- ✓ автомакрос *Auto_Open* для случая, если объект *WorkBook* ("*Книга_С*") открывается методом *Open*;
- ✓ автомакрос *Auto_Close* для случая, если объект *WorkBook* закрывается методом *Close* и при этом создан метод обработки события *BeforeClose;*

Задание №2 Рассчитайте количество лет, через которое окупится покупка. Обеспечьте возможность изменения всех исходных данных.

Задачу решить двумя способами:

- ✓ с помощью такой команды Excel, как "*Поиск решения*"
- ✓ программированием на VBA.

Покупка №1. Гараж куплен за 3000 долларов. Каждый год владелец вносит плату за эксплуатацию и аренду земли в размере 5000 рублей.

Сравните с вариантом, если владелец будет держать машину на платной стоянке с оплатой за сутки 120 рублей.

Покупка №2. Квартира куплена за 100 000 долларов. За нее каждый месяц вносится квартплата в размере 4000 рублей.

Сравните с вариантом, если хозяин будет снимать квартиру, выплачивая 400 долларов в месяц..

Задание № 3 Используя программирование на VBA, скопируйте содержимое выделенной ячейки во все листы рабочей книги в ячейки с тем же адресом, что и выделенная ячейка.

Алгоритм решения задачи:

- o выделяем ячейку на рабочем листе, например "B2";
- o запускаем макрос на выполнение, который должен:
  - ✓ скопировать содержимое ячейки в буфер;
  - ✓ определить адрес ячейки;
  - ✓ определить количество листов рабочей книги;
  - ✓ в цикле For –Next перебрать листы рабочей книги, вставляя в ячейку с искомым адресом содержимое буфера.

Задачу решить:

- ❖ используя цикл *For-Next*;
- ❖ используя цикл *For-Each -Next*;

Задание № 4 Используя средства VBA, создайте процедуру автозаполнения столбца ячеек числами (номерами) от 3 до 20. В программу добавить средства, позволяющие вводить адрес первой ячейки для ввода информации.

Проверить работоспособность созданного программного обеспечения.

Задание № 5 Создайте процедуру, с динамическим массивом, используя минимальное количество операторов.

Алгоритм решения задачи:

- ✓ Определите (опишите) динамический массив
- ✓ Заполните первые 2 элемента массива.
- ✓ Переопределите его размерность, не стирая старых данных.
- ✓ Заполните первые дополнительно добавленные 2 элемента массива.
- ✓ Используя переменную типа *Variant* и функцию *Array*, создайте второй массив с той же размерностью, как и первый.
- ✓ Присвойте всем элементам первого массива значения, хранящиеся в элементах второго.

Задание № 6 Приведите пример (создайте процедуру), на использование в программе цикла *For Each-Next* для случая, если *группа объектов* является *массивом*.

Проверить работоспособность созданного программного обеспечения.

Задание № 7 Приведите пример (создайте процедуру), на использование в программе цикла *For Each-Next* для случая, если *группа объектов* является *семейством объектов*.

Проверить работоспособность созданного программного обеспечения.

Задание №8 Создайте процедуру, использующую цикл *For Each-Next*, которая закрашивает ячейки со значениями менее нуля (<0) в заданный преподавателем цвет в диапазоне *CurrentRegion*.

Проверьте работоспособность созданного программного обеспечения.

Задание №9 Создайте процедуру, использующую цикл *For Each-Next*, которая закрашивает ячейки со значениями менее нуля (<0) в заданный преподавателем цвет в диапазоне *UsedRange.*

Проверьте работоспособность созданного программного обеспечения.

Задание №10 Создайте процедуру, использующую цикл *For Each-Next*, которая закрашивает ячейки со значениями менее нуля (<0) в заданный преподавателем цвет в диапазоне *CurrentArray*

Проверьте работоспособность созданного программного обеспечения.

Задание №11 Создайте процедуру, использующую цикл *Do Loop* для случаев:

- ✓ с проверкой условия выполнения цикла в его начале,
- ✓ с проверкой условия выполнения цикла в его конце.

Проверьте работоспособность созданного программного обеспечения.

Задание №12 Создайте процедуру, использующую *пользовательский тип данных.*

Поля у этих переменных должны быть разного типа, в том числе:

- массивом,
- записью (переменных пользовательского типа).

Проверьте работоспособность созданного программного обеспечения.

Задание №13 Создайте процедуру, использующую *объектные переменные и раннее связывание*.

Проверьте работоспособность созданного программного обеспечения.

Задание №14 Создайте процедуру, использующую *объектные переменные и позднее связывание*.

Проверьте работоспособность созданного программного обеспечения.

Задание №15 Создайте процедуру, использующую функцию *MsgBox(),* создающую окно сообщений.

При формировании атрибутов использовать два варианта их задания:

- ✓ суммой кодов
- ✓ константами.

В процедуре произведите анализ кода нажатой кнопки и запуск какого-либо метода (печать листа, форматирование и т.д.) в зависимости от того, по какой кнопке окна произведен щелчок мышью.

Проверьте работоспособность созданного программного обеспечения.

При выводе окна на экран, предусмотрите один из представленных ниже вариантов заполнения (по заданию преподавателя):

	Вариант №					
	A	**B**	**C**	**D**	**E**	**F**
Размещаемые в окне кнопки	OK	OK + Cancel	Abort + Retry + Ignore	Yes + No + Cancel	Yes + No	Retry + Cancel
Номер активной кнопки		2	3	3	2	1
Пиктограмма	⊕	?	!	⊕	¡	!

Задание №16 Создайте процедуру, использующую функцию *InPutBox(),* создающую окно ввода.

Преобразуйте полученную от функции строку в число (или дату).

В том случае, если нажата кнопка *Cancel*, организуйте вывод окна снова. Для этого использовать цикл *Do Loop*.

Проверьте работоспособность созданного программного обеспечения

Задание №17 Создайте и запустите процедуру обработки одного из событий для объекта WorkBook:

- ✓ Activate
- ✓ BeforeClose
- ✓ BeforeSave
- ✓ Open
- ✓ NewSheet
- ✓ SheetBeforeRightClick

Задание №18 Создайте и запустите процедуру обработки одного из событий для объекта WorkSheet:

- ✓ Activate
- ✓ BeforeRightClick
- ✓ Calculate
- ✓ Change

Задание № 19 Добейтесь того, чтобы при нажатии определенной комбинации клавиш на клавиатуре выполнялась ваша процедура.

Задание № 20 Добейтесь того, чтобы ваша процедура запускалась на выполнение в заданное время.

Задание №21 Приведите пример использования метода *OnKey.*

Задание №22 Приведите пример использования метода *OnTime*

Задание №23 Используя программные средства:

1. измените ширину столбца;
2. скройте формулу в ячейке;
3. скройте строку;
4. произведите горизонтальное и вертикальное выравнивание текста в ячейке;
5. расположите текст в ячейке под заданным углом к горизонту;
6. установите режим автоматического изменения размеров шрифта при заполнении ячейки содержимым по всей ширине;
7. установите режим переноса текста на другую строку внутри ячейки, расширяя ее ширину;
8. установите режим запрещающий ввод информации в ячейку;
9. распределите слишком длинный текст в ячейки диапазона, находящиеся под ним;
10. Объедините и разъедините ячейки.

Задание № 24 Создайте объектную переменную указанного типа.

Используя эту переменную, заполните значениями указанные ячейки, залейте их любым цветом

```
Dim myRange As Range
Set myRange = Range("C2:C5")
```

Задание №25 Создайте пересечение и залейте его любым цветом.

Задание №26 Создайте объединение и залейте его любым цветом.

Задание №27 Приведите пример на использование методов *Cells* и *Offset.* Покажите их различие.

Задание №28 Объявите две объектные переменные указанного типа.

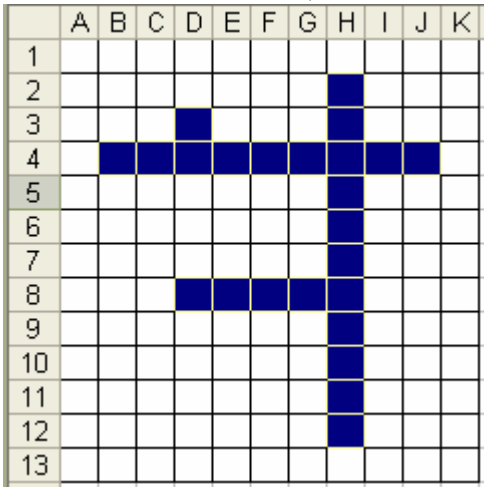
Dim myRange As Range, myRange_n As Range

Свяжите первую переменную *myRange* с указанным диапазоном: *("C2:C5")*

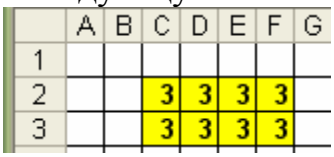
Используя метод *Offset*, создайте новый объект, на который будет ссылаться переменная *myRange_n*, смещенный относительно *myRange* на 3 строки вниз и 5 столбцов влево.

Заполните значениями ячейки нового диапазона, используя объектную переменную *myRange_n*, залейте его любым цветом

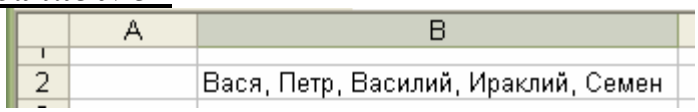
Задание №29 Используя пересечения и объединения, выделите сложную область и залейте ее цветом.



Задание №30 Используя метод Range с двумя аргументами, заполните числами следующую область и залейте ее цветом



Задание №31



Напишите программу на VBA, которая в тексте, помещенном в ячейку *B2,* поменяет все запятые на точки.

Количество слов, букв в словах, запятых в этом тексте может быть любым.

В принципе, простое решение получается при использовании цикла While-Wend.

Задание №32 Задана таблица №1
Необходимо:

1) Создать таблицу №2 и поместить в соответствующие ячейки необходимые формулы для подсчета количества указанных отметок.

2) В какую-нибудь ячейку поместить формулу, позволяющую на основании данных таблицы №1 выполнить действия в соответствии с алгоритмом 1 и определить, в конечном итоге, склонность учащихся к математике или литературе.

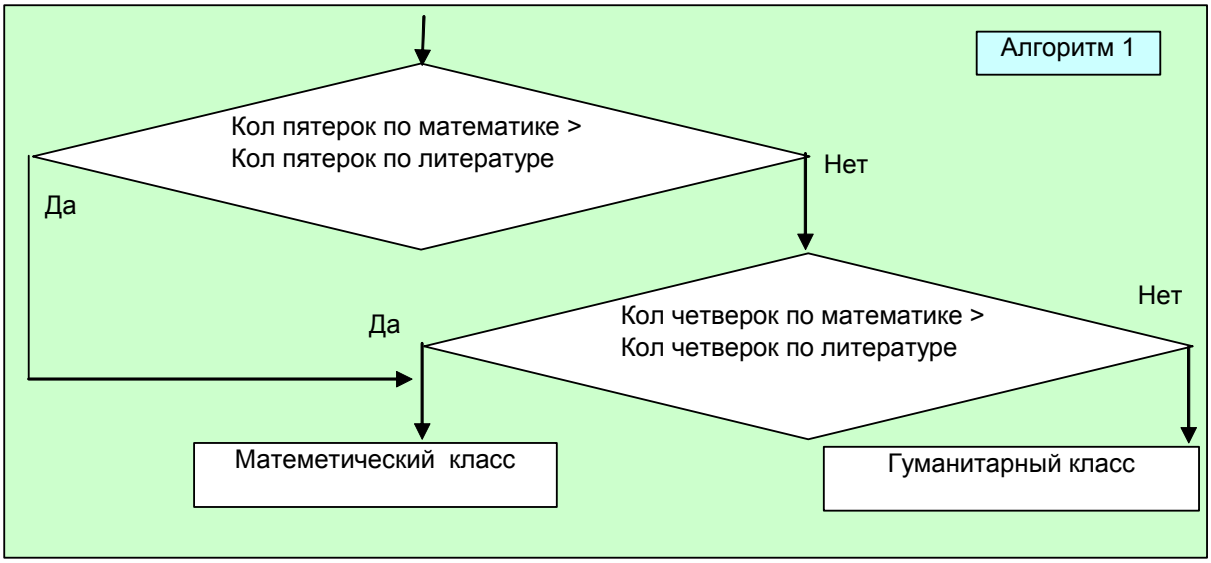
3) В какую-нибудь ячейку поместить формулу, позволяющую на основании данных таблицы №1 выполнить действия в соответствии с алгоритмом 2 и определить, в конечном итоге, склонность учащихся к математике или литературе**.**

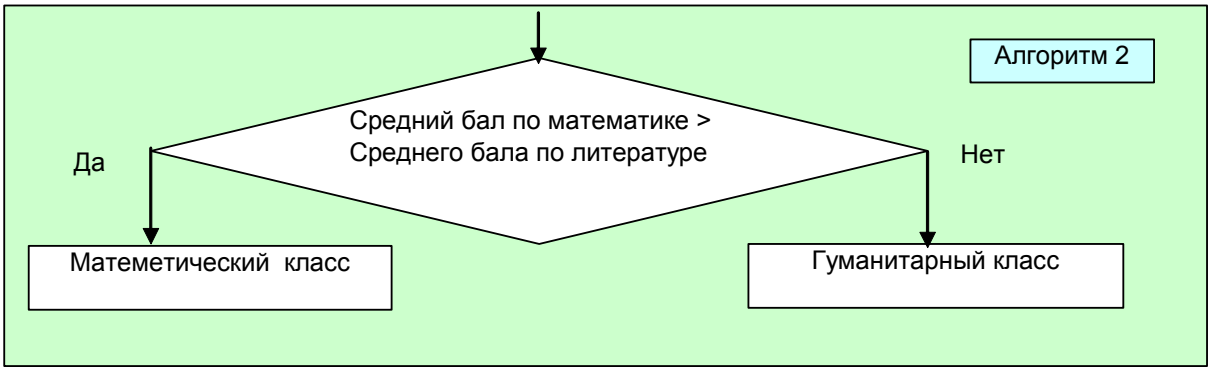
Таблица №1

Студенты	Дисциплины	
	Математика	Физика
Иванов	5	4
Петров	2	3
Сидоров	4	5
Курамшин	3	3
Кацман	5	5
Ерицян	4	5
Басаев	2	2

Таблица №2

Количество указанных оценок по предметам			
	мат	физ	Всего
Пятерок	2	3	5
Четверок	2	1	3
Троек	1	2	3
Двоек	2	1	3





Задание №33 Поместите в ячейку дату *25 января 1990*

Напишите функцию которая бы помещала число в одну ячейку, месяц в другую, а год в третью.

Используйте две функции *InStr и InStrRev*

Задание №34 Напишите функцию, которая бы меняла бы в тексте, расположенном в ячейке, один символ на другой. Например, все символы «н» на «с».

Покажите, что функция работоспособна.

Используйте оператор *Replace*.

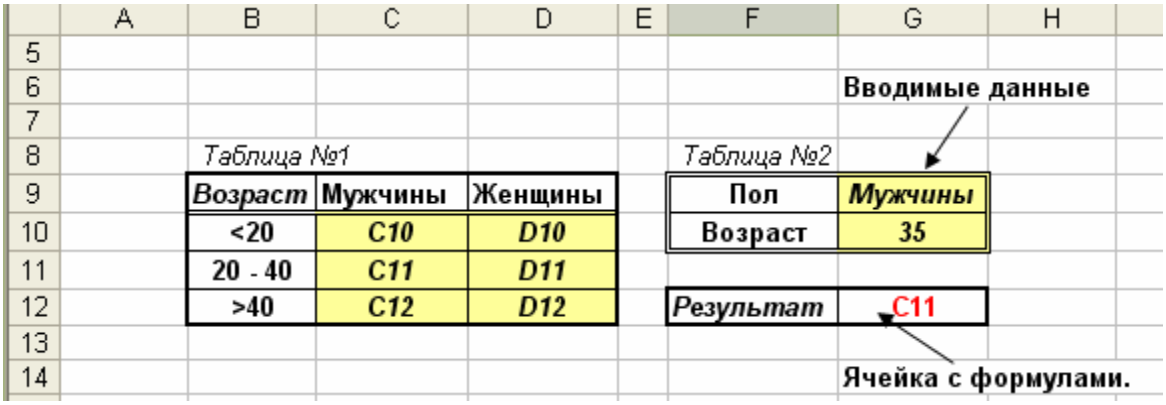
Задание №35 Напишите пользовательскую функцию, которая бы в ячейку *"A2"* помещала информацию о том, к какому типу принадлежат данные, вводимые в соседнюю ячейку *"A3"* . (текст, число, дата).

Покажите, что функция работоспособна.

Покажите, как то же самое можно сделать с помощью встроенных функций Excel.

Задание №36 Используя мастер функций вставьте в ячейку "*G12"* функцию, которая бы в соответствии с введенными данными таблицы №2, выбирала соответствующую информацию из таблицы №1.

Используйте такие функции как "ИНДЕКС" , "ВПР", "ГПР", "ВЫБОР".



Задание №37 Определите, сколько дней Вы прожили со дня рождения.

Определите, сколько часов Вы прожили со дня рождения.

Задание №38 Заданы числа

А=4567б,617283 и Б=0,00003546

Задайте такой формат, чтобы эти числа были напечатаны так:

А 4567б,6 4567б,61 4567б,617 4567б,6172 45 67б
Б ,0 0,000 0,00003546 0,00003546000

.

Задание №39 Выполните один из нижеперечисленных примеров:

1. Приведите примеры и поясните, как использовать формат “General Number” и “Fixit.”
2. Приведите примеры и поясните, как использовать формат “Standard” и ”Currency”.
3. Приведите примеры и поясните, как использовать формат “Long Date” и ”Shot date”.
4. Приведите примеры и поясните, как использовать функцию Len, InStr, Mid.
5. Приведите примеры и поясните, как использовать функцию Lset, Rset.
6. Приведите примеры и поясните, как использовать функцию Trim, Asc, Val.
7. Приведите примеры и поясните, как использовать функцию DateSerial.
8. Приведите примеры и поясните, как использовать функцию DateValue

ЛИТЕРАТУРА

1. Бураков П.В., Петров В.Ю. Информационные системы в экономике. Учебное пособие . СПб:, СПбГУИТМО, 2010, 59с
2. Зикратов И.А., Петров В.Ю. Информационные технологии в управлении. Учебное пособие . СПб:, СПбГУИТМО, 2010, 64с
3. Вильям Дж.Орвис, Visual Basic for Application на примерах. - М.: Бином, 1995. -512 с.
4. 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
Владимир Владимирович Косовцев
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