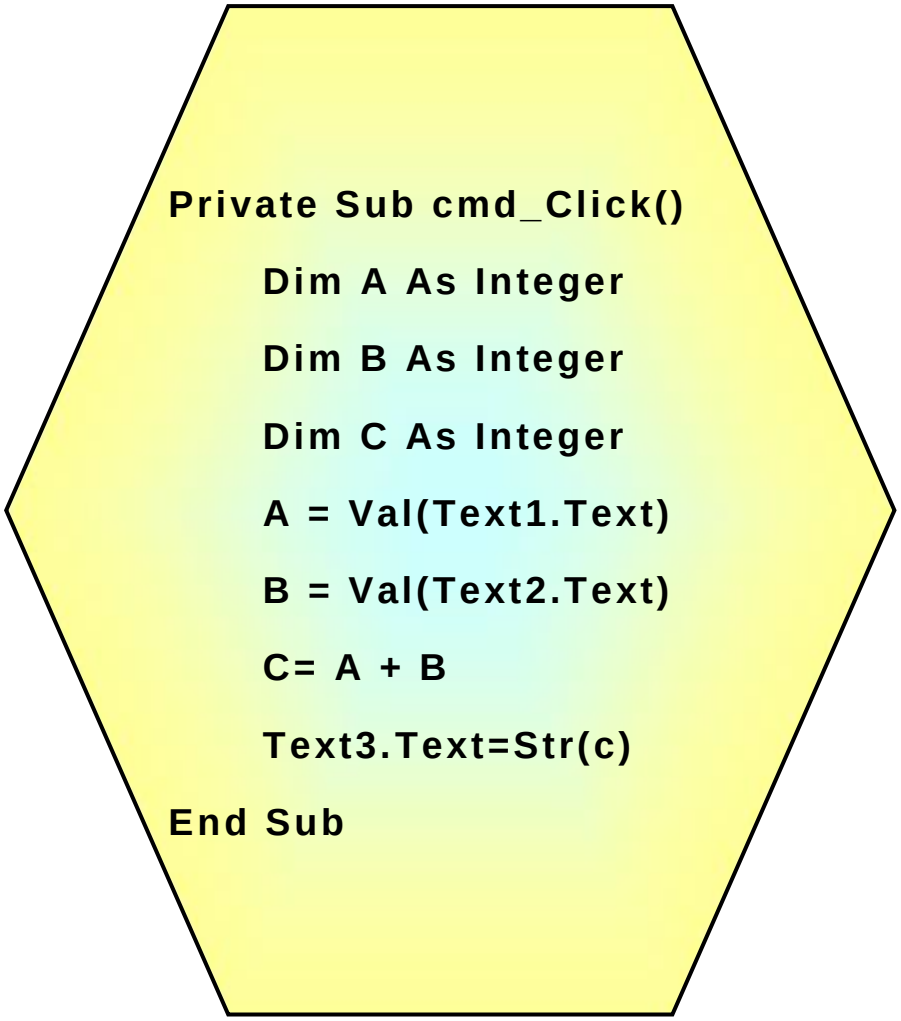
А.СОПУЕВ, А.Ж.КУДУЕВ

ПРАКТИЧЕСКИЕ РАБОТЫ НА VISUAL BASIC



Ош - 2004

ББК 32.974-01
С - 64

Рецензенты: Мансуров К., зав. каф. "Информатики и вычислительной техники" ОшТУ, к.ф.-м.н., доцент

Кыбыраев А.О., зав. каф. "Программирования" ОшГУ, к.ф.-м.н., доцент

Сопуев А., Кудуев А.
С – 64 Практические работы на Visual Basic. – Ош: Изд. центр ОшГУ, 2004. – 55 с.

ISBN 9967-03-181-6

В пособии приведены цели, задания и варианты решения от простых до достаточно сложных типичных примеров практических и лабораторных занятий по языку Visual Basic. Структура изложения материала построена по принципу «Шаг за шагом». Программы, приведенные в книге, снабжены комментариями. Для удобства усвоения приведены рисунки объектов и подготовленных приложений.

Материалы пособия могут быть использованы при проведении занятий по предметам "Язык программирования Visual Basic ", "Языки программирования и методы трансляции" и "Практикум на ЭВМ".

Данное пособие предназначено для студентов вуза. Книга может быть использована и для самостоятельного обучения и как учебное пособие на уроках.

Печатается по решению Ученого Совета ОшГУ

С 2404090000-04 **ББК 32.974-01**

ISBN 9967-03-181-6

Содержание

Введение 4
1. Среда программирования Visual Basic 5
2. Работа с кнопками управления 10
3. Объёмные надписи на форме 11
4. Работа с текстовыми полями 12
5. Создание счетчика времени 13
6. Работа с буфером обмена 14
7. Окно с плавным переходом цвета 15
8. Создание градиентного заголовка 16
9. Изменение контуров формы 18
10. Прокручиваемая картинка 20
11. Стандартное информационное окно 22
12. API-функция GetSystemMetrics 23
13. Бегущая строка 24
14. Графические эффекты 25
15. Работа с переключателями 27
16. Диалоговые окна для обмена сообщениями 28
17. Создание контекстных меню 30
18. Изменение вида курсора в текстовом поле 33
19. Изменение интервала мерцания курсора 35
20. Использование объектов файловой системы 36
21. Разработка простого проигрывателя 38
22. Вызов системных компонентов из VB 40
23. Работа с базами данных 43
24. Проектирование простого отчета 46
25. Работа с элементом управления Rich Textbox 49
26. Автоматизация использования объектов MS Office 52
Литература 55

Введение

Microsoft Visual Basic - это мощная система программирования, позволяющая быстро и эффективно создавать приложения для Microsoft Windows и Microsoft Windows NT.

Язык программирования Visual Basic содержит несколько сотен инструкций, функций и специальных символов. Он предназначен не только для использования в программном продукте Visual Basic и Microsoft Visual Basic for Application, но и включен в состав Microsoft Excel, Microsoft Word, Microsoft Access, Microsoft PowerPoint, Microsoft Project и других приложений для Windows.

Одним из основных преимуществ языка Visual Basic является возможность очень быстрого создания работоспособных приложений, предназначенных для решения конкретных прикладных задач. Поскольку эта среда выполняется на компьютере вместе с операционной средой Windows, то и создавать мы будем программы, называемые Windows-приложения или просто: приложения. Для создания приложения необходимо составить **ПРОЕКТ**. Этим и будем заниматься в среде программирования Visual Basic.

Оригинальный язык программирования Basic был создан Джоном Кемени и Томасом Курцем в 1963 году в Дартмурском колледже. Он быстро завоевал популярность в качестве языка для обучения программированию в университетах и школах и был адаптирован для использования на персональных компьютерах основателем и главой компании Microsoft Биллом Гейтсом в середине 70-х гг. С тех пор для ПК последовательно было выпущено несколько версий Basic, включая Microsoft Quick Basic и MS-DOS Qbasic. Хотя программная оболочка Visual Basic выполнена полностью графической, а сам язык программирования весьма далек от языка, применяемого для ранних версий интерпретаторов Basic, простота и элегантность Basic осталась в большой мере присущей и новым версиям. Широкие возможности Visual Basic и его простота послужили основной причиной для выбора его в качестве языка программирования для создания таких Windows-приложений как Excel.

1. Среда программирования Visual Basic

Цель работы: Изучение способов запуска, сохранение, выход из Visual Basic, настройка среды, работа с инструментами, элементами, свойствами и окнами.

Запуск программы осуществляется одним из обычных способов, характерных для Windows.

- Через систему иерархически организованных меню (<Пуск> - папка *Программы – папка Microsoft Visual Studio 6.0* – пиктограмма *Microsoft Visual Basic*).
- Щелчком мыши по ярлыку VB.

После запуска программы на экран будет выведено диалоговое окно «New Project» (Рис 1.). В нем указан тип программного проекта, установленный по умолчанию (Standard EXE). Окно предоставляет пользователю меню для дальнейших действий.

Меню содержит три пункта:

- **New -** начало создания нового проекта (приложение VB называется проектом);
- **Existing -** выбор приложения из существующий проектов;
- **Recent -** список самых последних проектов;

Рис. 1.

Новый проект откроется в среде программирования IDE (Integrated Development Environment - интегрированная среда разработки) Visual Basic вместе с некоторыми другими окнами и инструментами, как показано на Рис. 2. Главное окно содержит все элементы, которыми обладает любое приложение Windows: строку заголовка с названием проекта, строку меню, строку инструментов.

Строка меню обеспечивает доступ к большинству команд, работающих в соответствии со стандартными соглашениями, общими для всех приложений Windows.

Под строкой меню расположена строка инструментов, представляющая собой набор кнопок, являющихся ярлыками для команд, с помощью которых осуществляется работа в среде Visual Basic.

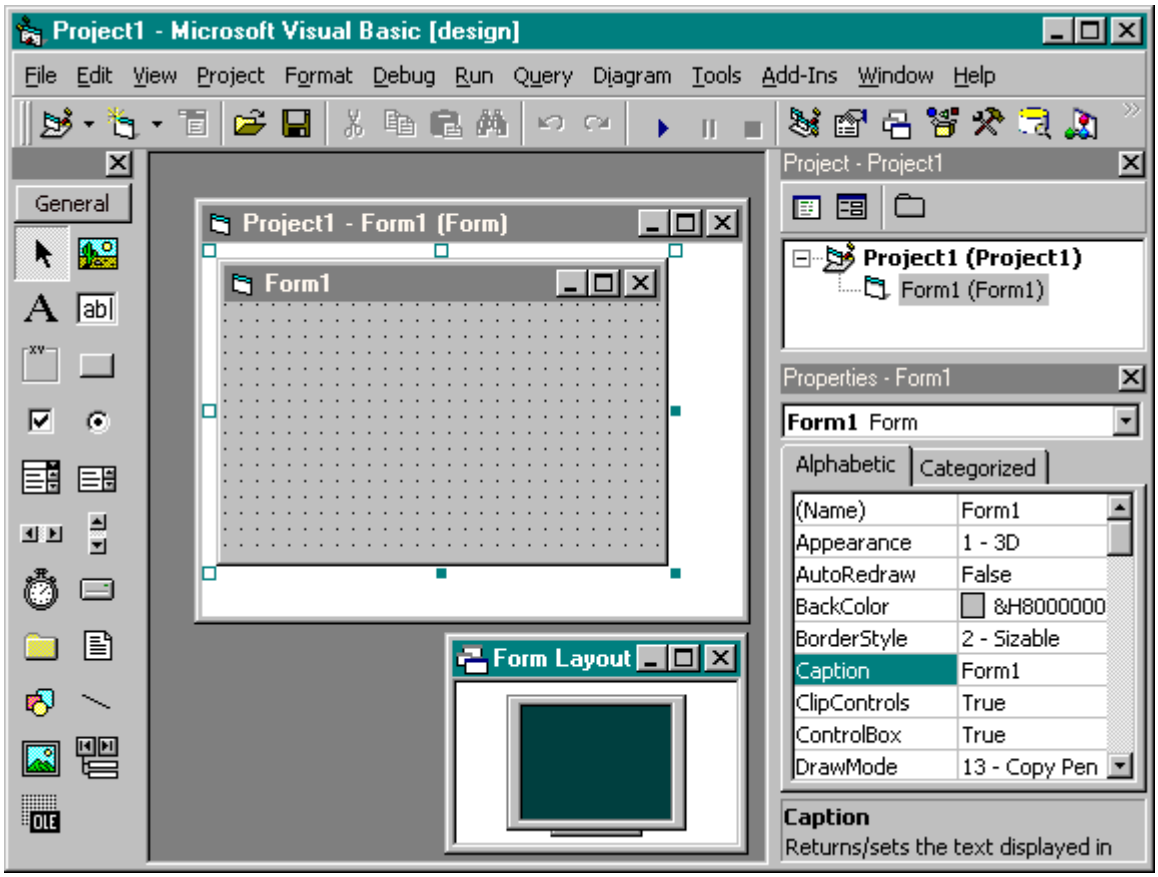


Рис. 2.

Под строкой инструментов (слева направо) располагаются:

- *Окно инструментов* с пустой строкой заголовка («Toolbox»);
- *Окно содержания проекта («Project Container»);*
- *Окно*
- *Окно формы («Form»),* находящееся внутри окна содержания проекта:
- *Окно проекта («Project»).*

Под окном проекта обычно находится окно свойств («Properties»), а под ним – *окно мастер формы («Form Layout»).*

Ниже окна инструментов и окна содержания проекта размещается *окно непосредственного выполнения («Immediate»),* а под ним – панель задач Windows.

Следует иметь в виду, что размер и форма перечисленных окон определяется конкретной конфигурацией системы пользователя. Он может изменить расположение и форму окон, а также сворачивать их, чтобы доступными были на экране необходимые элементы среды программирования.

Настройка среды программирования

В среде программирования Visual Basic имеются семь окон, перечисленных выше, которое используется как инструменты для разработки приложений. Их можно размещать в любом месте рабочего стола и изменять их размеры так, чтобы было удобно работать. Кроме того, имеется возможность прикрепления окон (locking). Прикрепленные окна удобны тем, что они всегда находятся поверх других окон и поэтому всегда видны на экране.

Для прикрепления одного окна к другому их нужно разместить так, чтобы границы окон совпадали. Если требуется увеличить рабочее поле прикрепленного окна, нужно передвинуть одну из его границ.

Рассмотрим назначение каждого из окон среды программирования, их расположение на экран и особенности использования при разработке приложений.

Окна «Form1» и «Project1-Form1(Form»). Любое приложение, разработанное в среде Visual Basic, состоит из одного или более окон. На этапе разработки эти окна называются формами. Форма - это окно в интерфейс пользователя. Она может содержать меню, кнопки, окно списков, полосы прокрутки и другие элементы, существующие в Windows-программах. В начале разработки Visual Basic предлагает одну форму, которая называется Form1, со стандартной сеткой (группа регулярно расположенных точек). Сетки используются для размещения элементов пользовательского интерфейса (Рис. 3).

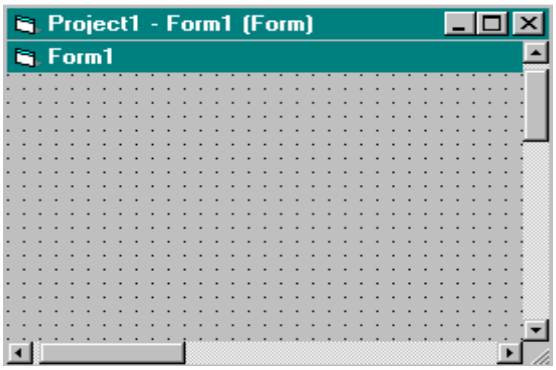


Рис. 3.

Форма окна «Form1» будет поименована, и ее размеры будут изменены при разработке приложения. При необходимости можно добавить новые

формы, щелкнув кнопкой мыши на команде **Add Form (добавить форму)** меню **Project (Проект).** Каждая форма будет окном пользовательского интерфейса на диске.

Окно инструментов. Инструменты и средства управления на панели инструментов служат для того, чтобы создавать новые элементы пользовательского интерфейса. Любое окно (а значит и форма) этого интерфейса содержит различные объекты: командные кнопки, текстовое окна, ярлыки (этикеты) и т. д., все они называются элементами управления. Эти элементы добавляются в формы путем перетаскивания их с панели инструментов

Последняя, как правило, вдоль левой стороны экрана (см. Рис 4). После того как элементы управления внесены в форму, они становятся объектами, или программируемыми элементами пользовательского интерфейса.

Рис. 4.

Панель инструментов содержит также средства управления для создания объектов, выполняющих специальные «за экранные» операции: управления информацией в базах данных, контроль временных интервалов и.т.д. Имеется возможность расширения средств управления, представленных на панели управления. Для этого нужно выбрать в окне пункт *Components* меню **Project** и в появившемся диалоговом окне выбрать нужное средства управления.

Окно «Project» (Проект). Как отмечалось выше, каждая форма - это отдельный файл приложения. Кроме того, существует, по крайней мере, один файл, содержащий код программы для приложения, и если формы используют инструменты не из Visual Basic (такого можно), то они будут содержаться в дополнительных файлах. Все перечисленные файлы вместе образуют проект.

Для удобства работы с отдельными комплектами проекта используется окно «**Project**» В нем перечисляются все файлы проекта, доступ к которым осуществляется при помощи двух кнопок:<View code> (просмотр кода) и <View project> (просмотр объекта).

Файл проекта имеет расширение .vbp (Visual Basic Project) и содержит список всех файлов проекта. В окне проекта отображается структура в виде дерева, которая похожа на структуру папок в окне Explorer Windows.

Если окно проекта закрыто, нужно щелкнуть мышью по кнопке <Project Explorer> (проводник проекта) на панели инструментов. После этого структуру проекта можно просматривать как дерево папок (каталогов).

Окно «Properties» (свойства). Каждый объект (форма или элемент управления) имеет набор свойств. Они определяют внешний вид формы или элемента управления и его поведение. Окно свойств позволяет изме-

нить характеристики (установки) объектов. Оно содержит список всех объектов, использующихся в конкретном пользовательском интерфейсе, и предназначено для установки свойств каждого объекта. Свойства можно просматривать в алфавитном порядке или по категориям. Свойства могут быть числовыми (размеры элементов управления) и др.

В Visual Basic существует определенный формат установки свойства: *Объект.Свойство= значение.*

Окно «Form Layout» (Макет формы). Это окно предназначено для определения начального положения форм на экране при запуске приложения. Достигается это внутри экрана дисплея.

Закрытие Visual Basic

В процессе работы может понадобиться временно выйти из Visual Basic, например, для запуска другого приложения. Сделать это можно, щелкнув мышью по кнопке <свернуть> на главном окне.

Если запущено несколько приложений, то, нажимая несколько раз клавиши [Alt]+[Esc] или [Alt]+ [Tab] (плюс означает одновременное нажатие двух клавиш), нужно пройти открытие приложения, пока Visual Basic не станет активным. Для того чтобы полностью закрыть Visual Basic, нужно выйти в меню **File** и выбрать команду **Exit.** То же самое дает щелчок мышью по кнопке закрытия главного окна. Если были сделаны какие-либо изменения в текущем проекте, Visual Basic спросит, сохранить ли их. Щелчок по кнопке <No> свидетельствует об отказе сохранения изменений. Происходит закрытие программы. Если пользователь желает сохранить сделанные изменения перед закрытием Visual Basic, необходимо щелкнуть мышью по кнопке <Yes>. Если проект сохраняется впервые, появиться диалоговое окно, в котором нужно выбрать папку для сохранения проекта, набрать имя файла и щелкнуть по кнопке <Save>. Visual Basic сначала сохранить форму на диске с расширением .frm. Затем появится диалоговое окно «Save Project As» (сохранение проекта). Нужно поступить так же, как и выше, тогда произойдет сохранение проекта на диске с расширением .vbp, после чего Visual Basic закроется.

В том случае, если пользователь решит не сохранять изменения и предложить работу в Visual Basic, следует щелкнуть по кнопке <Cancel>.

2. Работа с кнопками управления

Цель работы: Создание простейшего Windows-приложения с заданным заголовком окна и цветом формы, содержащие кнопки управления формой.

1. Запустите *Visual Basic* и выберите тип создаваемого проекта, например, *Standard.exe*. Это обычное приложение.

2. Измените заголовок окна формы, установив в окне *Properties* для свойства *Caption* значение «Мое первое приложение».

3. Измените цвет формы со стандартного на «Зеленый» в окне *Properties \ BackColor \ Palette* установив для свойства *BackColor* значение «Зеленый» (См. Рис. 5).

4. Добавьте в форму четыре кнопок управления *CommandButton*.

5. Установите следующие свойства объектов:

Command1	Caption	Развернуть
Command2	Caption	Свернуть
Command3	Caption	Восстановить
Command4	Caption	Закрыть

6. Запишите код для процедуры:

```
Private Sub Command1_Click( )
    WindowState = vbMaximized  ' Развернет форму на весь экран
End Sub
```

7. Самостоятельно запишите код для процедур Private Sub Command2_Click( ) и Private Sub Command3_Click( ), используя следующие положения: vbMinimized, vbNormal.

8. Запишите код процедуры для четвертой кнопки:

```
    Private Sub Command4_Click( )
        Unload Me ' Выгружает форму
    End Sub
```

9. Запустите приложение с помощью меню *Run \ Start* или функциональной клавишей F5 или кнопкой *Start* на панели инструментов.

10. Проверьте работу кнопок и закончите работу приложения.

11. Сохраните сначала файл формы (File \ Save Form1 As…\ FЛаб1), затем файл проекта (File \ Save Project As…\ PЛаб1).

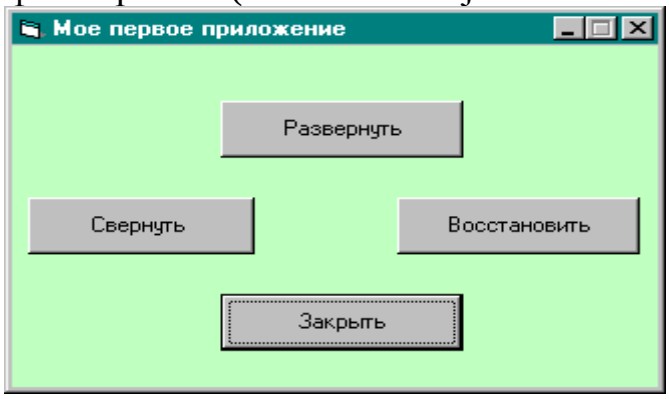


Рис. 5.

3. Объёмные надписи на форме

Цель работы: Подготовка объемных надписей на форме с помощью элемента управления *Label*.

Для этого достаточно написать всего семь строк кода:

1. Выберите место на форме с помощью элементом управления *Label* для ввода надписи и установите следующие свойства:

Свойство / Метод	Значение
Caption	XXI век – век Мультимедиа
Height	2190
Width	6810
Font	Arial Black, Полужирный, 18
Alignment	2 – Center
BackStyle	0 – Transparent
Border Style	Fixed Single

3. Добавьте две кнопки и в свойствах *Caption* введите слова "Изменить цвет" и "Выход" соответственно. (См. Рис. 6.)

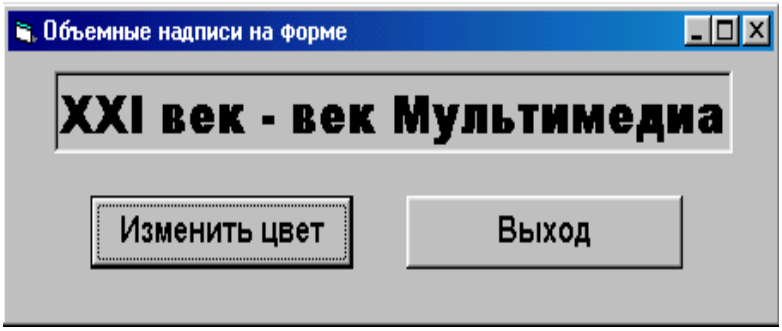


Рис. 6.

4. Двойным щелчком откройте окно кода кнопки "Изменить цвет" и введите следующий фрагмент программы:

```
Private Sub Command1_Click()
Static s As Integer
   s = s + 1
If s > 4 Then s = 1
Select Case s
   Case Is = 1
Label1(1).ForeColor = RGB(255, 0, 0)
   Case Is = 2
Label1(1).ForeColor = RGB(0, 0, 255)
   Case Is = 3
Label1(1).ForeColor = RGB(0, 255, 0)
   Case Is = 4
Label1(1).ForeColor = RGB(255, 0, 255)
   End Select
End Sub
```

Тогда с помощью функции RGB() и методом "Выбора" мы можем изменить цвет надписи при каждом нажатии кнопки "Изменить цвет" последовательно на следующие цвета: красный, голубой, синий, розовый.

5. Запишите код процедуры для кнопки "Выход":

```
Private Sub Command2_Click( )
   End
End Sub
```

6. Сохраните форму и проект под именем FЛаб2.frm и РЛаб2.vbp соответственно.

7. Подготовьте exe файл командой: File \ Make РЛаб2.exe

4. Работа с текстовыми полями

Цель работы: Овладение навыками работы с текстовыми полями *TextBox* и разработка приложения для поиска слов в тексте.

1. Создайте стандартный проект, а на нём три командные кнопки, узкий *TextBox* для искомого слова и большой *TextBox* для текста.

2. Для удобства в свойстве *Name* маленький *TextBox* назовите Tf, а большой *TextBox* назовите T, а в свойстве *MultiLine* выберите режим *True.*

3. В раздел *General Declarations* впишите:

```
Dim Pos As Integer          ' Переменная, хранящая позицию курсора
```

4. Дважды щелкните на форме и впишите в процедуру *Form_Load*:

```
Private Sub Form_Load( )
Pos = 1          ' Позиция курсора по умолчанию станет равной 1
End Sub
```

5. Теперь выберите событие *cmdFind_Click( )* (*cmdFind* имя кнопки поиска текста) и впишите:

```
Private Sub cmdFind_Click( )
' Если в текстовом поле T найдено содержимое окна Tf, то
If InStr(Pos, t.Text, tf.Text) <> 0 Then
T.SetFocus     ' Текстове поле T получает фокус
' Устанавливаем начало выделения - на один символ раньше найденного слова / буквы
T.SelStart = InStr(Pos, T.Text, Tf.Text) – 1     ' Устанавливаем длину выделения длина искомого слова
T.SelLength = Len(tf.Text)          ' Устанавливаем позицию, с которой следует начинать искать следующее идентичное слово
Pos = InStr(Pos, T.Text, Tf.Text) + Len(Tf.Text)
      End If
End Sub
```

6. Теперь если мы меняем искомое слово, то и искать надо с самого начала, соответственно и позиция опять должна быть равна нулю, в событие *Tf_Change( )* впишем:

```
Private Sub
Tf_Change( )
      Pos = 1
End Sub
```

7. Например, созданное вами поисковое приложение может отображаться в следующем виде: (См. Рис. 7).

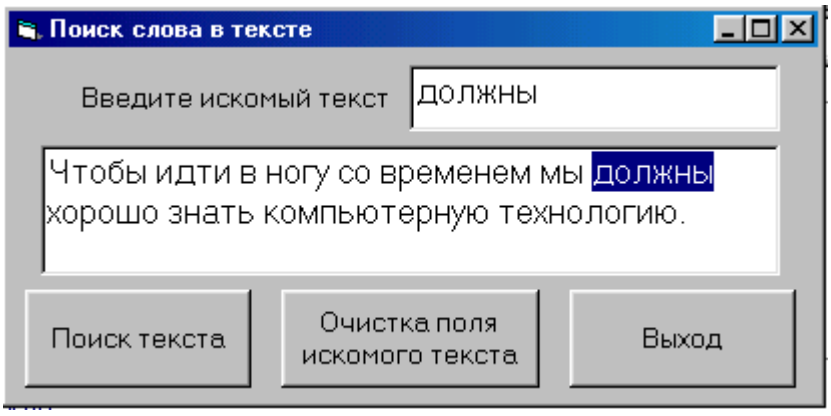


Рис. 7.

5. Создание счетчика времени

Цель работы: Овладение навыками работы с элементом управления *Timer* на примере создания счётчика времени.

1. Создайте стандартный проект, на форму поместите объект *Label* и установите следующие свойства:

Caption	00:00:00:00
Aligment	2 – Center
BackColor	Чёрный
ForeColor	Зелёный
Font	Arial Narrow, Bold, 22
Height	495
Widht	2415

2. Поместите на форму объект *Timer* и установите *Interval = 50. 50* - это одна двадцатая доля секунды (1000 / 20), которая будет расположена на таймере в самом конце.

4. Теперь нам нужны четыре переменные (часы, минуты, секунды и доли секунды), которые будут всегда подсчитываться и отображаться. Дважды щелкнув на форме откройте окно *Code*. Установите курсор в самом верху формы и в разделе *General Declarations* впишите:

```
Dim c As Integer 'объявляем переменную, содержащую количество часов
Dim m As Integer 'объявляем переменную, содержащую количество минут
Dim s As Integer ' объявляем переменную, содержащую количество секунд
Dim ds As Double ' объявляем пер-ную, содер-ю количество долей секунд
```

Двойным щелчком по таймеру откройте окно *Code* и впишите:

```
Private Sub Timer1_Timer( )
    ds = ds +
If ds = 13 Then
    s = s + 1
    ds = 0
End If
  If s = 60 Then
    m = m + 1
    s = 0
  End If
    If m = 60 Then
      c = c + 1
      m = 0
    End If
    If c = 24 Then c = 0
Label1.Caption = Format(c, "0#") & ":" &
Format(m, "0#") & ":" & Format(s, "0#") & ":"
& Format(ds, "0#")
    End Sub
```

Здесь с помощью функции Format (x, "0#") нашей надписи присваиваются отформатированные переменные, содержащие количества часов, минут, секунд и долей секунд.

Вставьте в форму четыре кнопки для управления работы счетчика времени (См. Рис. 8).

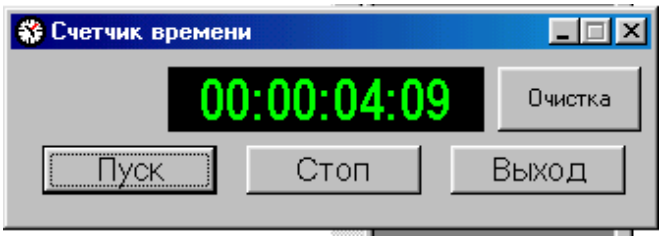


Рис. 8.

6. Работа с буфером обмена

Цель работы: Овладение навыками работы с буфером обмена *Clipboard*.

1. Поместите тестовое поле *Text* на форме стандартного проекта.
2. В свойствах установите *MultiLine = True*.
3. Создайте три кнопки с именами *cmdCut*, *cmdCopy* и *cmdPaste*, и соответственно надписи к ним: *"Cut"*, *"Copy"*, *"Paste"*.
4. Щелкните два раза на *cmdCut* (кнопка, которая будет вырезать текст) - откроется окно *Code*. Впишите:

```
Private Sub cmdCut_Click( )
Clipboard.SetText Text1.SelText ' выделенный фрагмент текста удаляется
Text1.SelText = ""
End Sub
```

5. Теперь откройте окно выбора (сверху окна *Code*) и выберите событие *cmdCopy* - заполним кнопку *Copy*("Копировать"):

```
Private Sub cmdCopy_Click( )
    'здесь всё тоже, что и выше, но выделенный фрагмент не удаляется
    Clipboard.SetText Text1.SelText
End Sub
```

6. Теперь выберите событие кнопки *Paste* ("Вставить") и впишите:

```
Private Sub cmdPaste_Click( )
Text1.Text = Text1.Text + Clipboard.GetText 'текстовому полю присваевается содержимое буфера обмена
End Sub
```

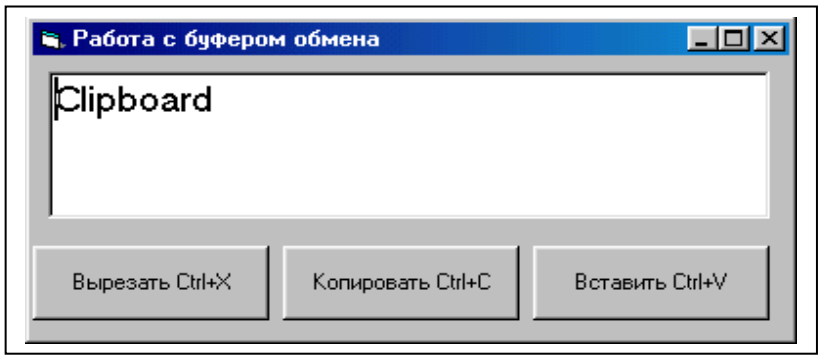


Рис. 9.

Чтобы проводить подобные операции с графикой нужно использовать методы *GetData* и *SetData*.

7. Окно с плавным переходом цвета

Цель работы: Овладение навыками работы с цветами.

Чтобы сделать плавный переход цвета на форме (как в программах *Setup*, но с красным цветом) выполните следующие шаги:

1. Создайте стандартную форму.
2. Свойство *AutoRedraw* надо установить равным *True*.
3. Щелкните два раза на форму и вставьте следующий код:

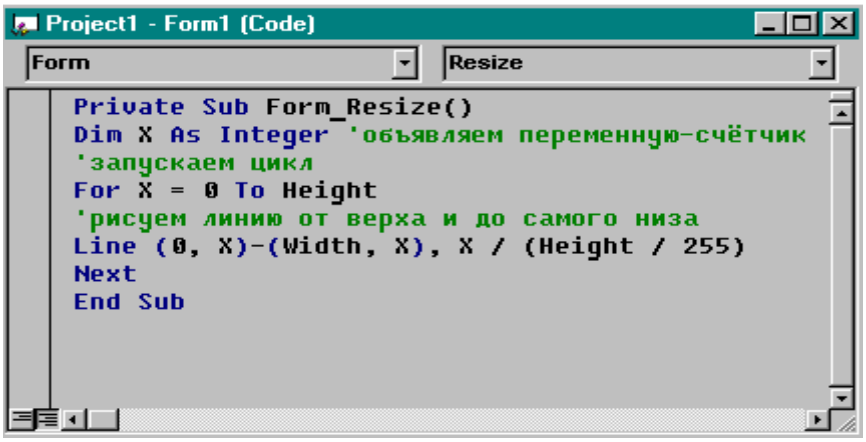


Рис. 10.

Для того, чтобы залить форму слева на право надо использовать следующий код:

```
    For X = 0 To Width
Line (X, 0)-(X, Height), X / (Width / 255)
    Next
```

Есть и другие варианты. Этот способ намного эффективнее, так как заливается быстрее. Пуст цвет будет синий.

```
Private Sub Form_Paint( )
'объявление переменных
Dim lY As Long
Dim lScaleHeight As Long
Dim lScaleWidth As Long
ScaleMode = vbPixels 'единицу измерения устанавливаем равной пикселу
lScaleHeight = ScaleHeight 'получаем кол-во пикселов по высоте
lScaleWidth = ScaleWidth  'получаем кол-во пикселов по ширине
DrawStyle = vbInvisible 'устанавливаем стиль заливки и рисования
FillStyle = vbFSSolid  'устанавливаем стиль заливки и рисования
    For lY = 0 To lScaleHeight 'запускаем цикл закраски
    FillColor = RGB(0, 0, 255 - (lY * 255) \ lScaleHeight) 'закрашиваем
    Line (-1, lY - 1)-(lScaleWidth, lY + 1), , B
Next lY
End Sub
```

8. Создание градиентного заголовка

Цель работы: Создание градиентной заголовки.

1. Создайте стандартный проект.

2. Создайте на нём объект *Image* и назовите его *gradient*. В свойствах установите *Stretch = True*, и на *Visible = False*. Вставьте в него какой-нибудь градиентный рисунок. Его можно быстро сделать в среде Photoshop.

3. Создайте объект *Label* и введите туда какую-нибудь надпись. В свойствах установите *BackStyle* = 0 - *Transparent* и цвет надписи смените на белый. Поместите надпись в верхний левый угол.

4. В свойствах формы установите *ControlBox = False* и из *Caption* сотрите всё. Форма должна получится абсолютно пустой.

5. Сделайте двойной щелчок по форме и откройте окно *Code*. В раздел *General Declarations* впишите:

```
'переменные, которые нам пригодятся при перетаскивании формы
Dim mMove As Boolean
Dim SX As Integer
Dim SY As Integer
```

6. Выберите событие формы *Form_Resize* и впишите:

```
Private Sub Form_Resize()
'рисунок переносим на самый верх и растягиваем
        gradient.Top = 0
        gradient.Left = 0
        gradient.Width = Me.Width
        gradient.Visible = True
End Sub
```

7. А теперь некоторые штрихи для перетаскивания. Впишите:

```
Private Sub Label1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
'в случае нажатия на надпись mMove станет истинной и засекаются значения x и y
        mMove = True
        SX = X
        SY = Y
End Sub
Private Sub Label1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
'реакция на движение мышки по надписи: если мышка нажата, то происходит движение формы за мышкой
        If mMove = True Then Me.Move Left + X - SX, Top + Y – SY
        End If
```

```
End Sub
Private Sub Label1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
        mMove = False     'при отжатии мышки движение отменяется
End Sub
```

8. А теперь тоже самое, но на случай, если пользователь щелкнет на рисунок:

```
Private Sub gradient_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
        mMove = True
        SX = X
        SY = Y
End Sub
Private Sub gradient_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
        If mMove = True Then
        Me.Move Left + X - SX, Top + Y – SY
        End If
End Sub
Private Sub gradient_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
        mMove = False
End Sub
```

9. Теперь введём код, если пользователь сделает двойной щелчок по заголовку:

```
Private Sub gradient_DblClick( )
If Tag <> "2" Then 'если тэг формы не равен двум, то ... состояние формы меняем на растянутое и тэгу присваеваем 2
        Me.WindowState = 2
        Me.Tag = "2"
  Else  'если тэг формы равен двум, то ...
        Me.WindowState = 0          '...состояние формы меняем на свётнутое и тэгу присваеваем 0
        Me.Tag = "0"
        End If
End Sub
```

10. Теперь тоже самое, но для надписи:

```
Private Sub Label1_DblClick( )
        If Tag <> "2" Then Me.WindowState = 2 Me.Tag = "2"
Else
        Me.WindowState = 0 Me.Tag = "0"
        End If
End Sub
```

После выполнение программы готовый градиентный заголовок выглядит следующим образом:

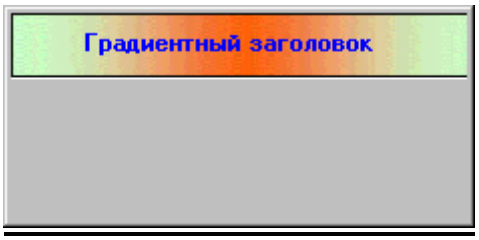


Рис. 11.

Единственный недостаток - это то, что кнопки "свернуть", "увеличить", "закрыть" делать придётся самим. А преимущество в том, что вы можете менять шрифт в заголовке, вставлять графику и прочее, и прочее.

9. Изменение контуров формы

Цель работы: Изучение методов изменения контура формы.

1. Создайте простую форму и измените её размеры на *Height = 4605* и *Widht = 4800*.

2. Двойным щелчком на форме откройте окно *Code*. В разделе формы *General Declarations* добавим объявления двух API-функций:

'эта функция создаёт регион. Требуется идентификатор окна (hWnd), регион, который следует наложить (hRgn) и свойство ReDraw - перерисовка (bReDraw)

Private Declare Function SetWindowRgn Lib "user32" (ByVal hWnd As Long, ByVal hRgn As Long, ByVal bRedraw As Boolean) As Long

'а эта создаёт круглые регионы (эллипсы, круги). Требует X1, X2, Y1, Y2 - точки для создания регионов

Private Declare Function CreateEllipticRgn Lib "gdi32" (ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As Long, ByVal Y2 As Long) As Long

3. Если Вы хотите, чтобы было красиво, то лучше наложить регион ниже заголовка окна, но потом двигать окно Вы уже не сможете, поэтому придётся применить метод перетаскивания формы на любое место. Поэтому, опять-таки в разделе формы *General Declarations* объявим нужные для этого переменные:

Dim IfMove As Boolean 'по этой переменной мы будем определять, нажал ли пользователь кнопку мыши или нет?

Dim X1 As Integer 'этой переменной будет присваиваться значение икса (X) при нажатии на форму

Dim Y1 As Integer 'этой переменной будет присваиваться значение игрека (Y) при нажатии на форму

4. Теперь в *Form_Load* записываем функцию создания региона, а в ней, в передаваемом параметре *hRgn* вписываем функцию создания этого самого региона:

Call SetWindowRgn(Me.hWnd, CreateEllipticRgn (50, 40, 300, 200), True)

'Me.hWnd - это идентификатор нужной формы;

'CreateEllipticRgn(50, 40, 300, 200) - создание овального региона и передача его функции SetWindowRgn; True - режим перерисовки включён

5. Так, осталось добавить код для переноса формы. Выберите событие *Form_MouseDown* и впишите:

Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)

IfMove = True 'обозначаем, что мышка нажата - форма готова к передвижению

X1 = X 'засекаем значение икса на форме в момент нажатия

Y1 = Y 'засекаем значение игрека на форме в момент нажатия

End Sub

6. Теперь там же выбираем событие *Form_Up* и вписываем:

Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)

'этим мы обозначаем, что мышка отпущена и форма не готова к передвижению

IfMove = False

End Sub

7. И последний штрих - выбираем событие *Form_MouseMove* и вписываем:

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)

'проверяем, если кнопка нажата, то начинаем двигать

If IfMove = True Then

'производится движение формы

Move Left + X - X1, Top + Y - Y1

End If

End Sub

8. Всё! Теперь обещанный маленький графический эффект: чтобы передвижение формы выглядело наиболее эффектно, сделайте так:

8.1. В событии Form_Load в строке

CreateEllipticRgn(50, 40, 300, 200) замените 40 на 0.

8.2. И впишите ещё одну строку:

Me.ScaleMode = 3

8.3. Двигайте форму и наслаждайтесь!

Рис. 12.

10. Прокручиваемая картинка

Цели работы: Создание прокручиваемой картинки с помощью элементов *VScrollBar* и *HScrollBar*.

1. Создайте стандартный проект.
2. На форме создайте *PictureBox*, в котором будет бегать картинка. Так как мы не можем заставить гонять картинку по форме.
3. Теперь прямо в *Picture1* создаём объект *Image* и назовём его просто *img*.
4. Справа от объекта *Picture1* создаём вертикальную полосу прокрутки *VScrollBar* и назовём её v.
5. Внизу, под объектом *Picture1*, создаём горизонтальную полосу прокрутки *HScrollBar* и назовём её h.

Тогда у нас получиться следующая картинка. Заметим, что объект *img* невидим. (см. Рис. 13).

Рис. 13.

Теперь осталось вписать код.

6. Откройте окно *Code* двойным щелчком по форме и впишите в событие *Form_Load*:

```
Private Sub Form_Load( )
  'загружаем в img нужный рисунок, желательно очень большой
  img.Picture = LoadPicture("Путь_к_вашему_файлу")
  'выравниваем картинку по левому краю и вверх
  img.Top = 0
  img.Left = 0
'устанавливаем максимальное расстояние прокрутки по вертикали: на всю высоту картинки, а разницу между высотой картинки и высотой полосы прокрутки, чтобы картинка не уползала с окна
  v.Max = img.Height - v.Height
  'здесь устанавливаем скорость прокрутки в режиме "быстрый" - щелкните не по стрелке
  v.LargeChange = 1000
  'а здесь устанавливаем скорость прокрутки в режиме "медленный" - клик по стрелке
  v.SmallChange = 500
  'всё тоже самое только для горизонтальной полосы прокрутки
  h.Max = img.Width - h.Width
  h.LargeChange = 1000
  h.SmallChange = 500
End Sub
```

7. Теперь открываем снова окно с формой и дважды щелкнем по вертикальной полосе прокрутки, тем самым открываем событие *v_Change* и вписываем:

```
Private Sub v_Change( )
  'value это положение бегунка на полосе прокрутки, т.е. если мы жмём вниз, то картинка бежит вверх и наоборот
  img.Top = -v.Value
End Sub
```

8. Теперь открываем сново окно с формой и дважды щелкнем по горизонтальной полосе прокрутки, тем самым открываем событие *h_Change* и вписываем:

```
Private Sub h_Change( )
  'этим мы устанавливаем, что при нажатии вправо картинка бежит влево и наоборот
  img.Left = -h.Value
End Sub
```

Сохранить форму и проект на диске.

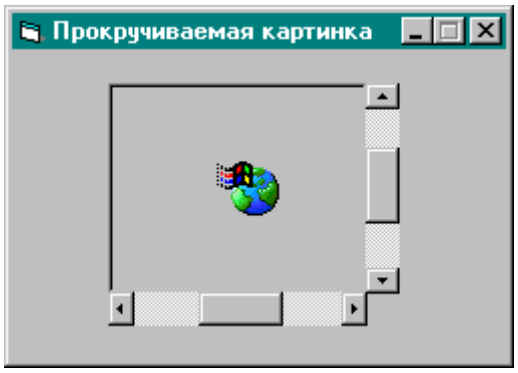


Рис. 14.

11. Стандартное информационное окно

Цель работы: Овладение навыками создания стандартных информационных окон.

Для создания стандартных информационных окон выполните следующие процедуры:

1. Создайте стандартную форму.

2. Измените заголовок окна формы с *Form1* на свойства *Caption* значение “Стандартное информационное окно”.

3. Создайте на нём объект *Command* и назовите его *cmdAbout.*

4. Установите свойство *Caption* объекта *Command1* в значение "О программе"

5. Двойным щелчком на форме откройте окно *Code*. На верху формы декларируем API-функцию *ShellAbout*:

```
Private Declare Function ShellAbout Lib "shell32.dll" Alias "ShellAboutA" (ByVal hwnd As Long, ByVal szApp As String, ByVal szOtherStuff As String, ByVal hIcon As Long) As Long
```

Теперь выберите событие кнопки *cmdAbout* ("О программе") и впишите:

```
Private Sub cmdAbout_Click( )
  Call ShellAbout(Form1.hwnd, App.Title, "Ваше Ф.И.О.", Me.Icon)
End Sub
```

Заметим, что *ShellAbout* функции нужны передаваемые значения. Такими значениями могут быть:

hwnd - уникальный индикатор формы, о которой "пойдёт речь"

szApp - это может быть название нашей программы или какая-нибудь другая строка, в заголовке которой будет стоять "Информация о программе", но вместо точек содержимое этого параметра.

szOtherStuff - здесь можно вставить имя разработчика

hIcon - и иконка!

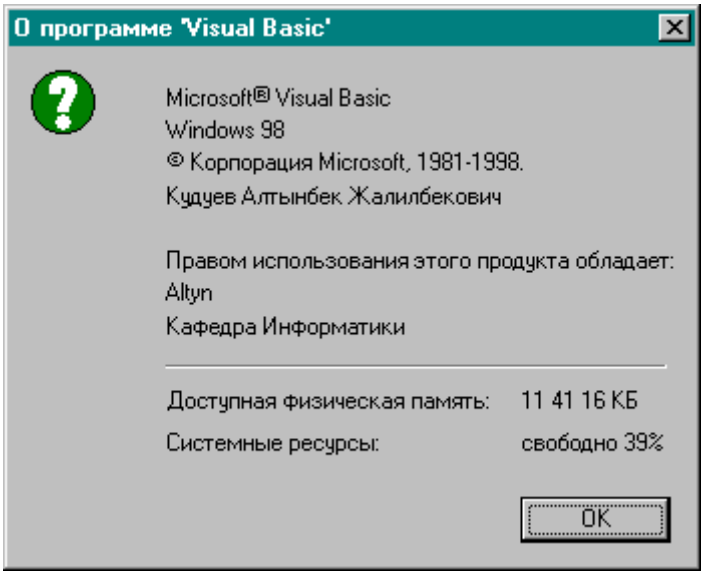


Рис. 15.

12. API-функция GetSystemMetrics

Цель работы: Изучение свойств API-функции *GetSystemMetrics*, возвращающей некоторые системные значения, установленные на компьютере пользователя.

Для API-функции *GetSystemMetrics* существует целый ряд констант:

Константа	Значение	Описание
SM_CXSCREEN	0	Резолюция по горизонтали.
SM_CYSCREEN	1	Резолюция по вертикали.
SM_CMOUSEBUTTONS	43	Количество кнопок мыши.

Эти константы продекламировать можно как в модуле, так и в форме (только впереди нужно поставить Private)

Private Declare Function GetSystemMetrics Lib "user32" Alias "GetSystemMetrics" (ByVal nIndex As Long) As Long

Например, для определения разрешения экрана пользователя (неважно где, в модуле или на форме) можно использовать следующий прием:

Запишем код для диалогового окна MsgBox "Ваше разрешение экрана: " & GetSystemMetrics (0) & "x" & GetSystemMetrics (1). Тогда на экран выводится следующая информация (если у вас действительно 800x600, если нет, то там будет ваша резолюция):

"Ваше разрешение экрана: 800x600"

Заметим, что пользователи часто могут использовать эту функцию. Многие из нас используют экранную резолюцию 800x600, так как она наиболее удобно и оптимальна. Даже тогда, когда мы пишем приложения, мы даже и не задумываемся, что есть люди, использующие 640x480 или 1024x768. А окна в наших программах зачастую оптимизированы для своей резолюции. Если мы выберем другие параметры экрана, мы можем допустить ошибку, так как в этом случае форма может оказаться чрезвычайно маленькой, или не влезеть в экран. Поэтому либо написать универсальные окна, либо осуществить проверку резолюции и подгонят окно под неё.

Разрешение экрана можно также определить следующим образом:

По горизонтали: **Screen.Width / Screen.TwipsPerPixelX**

По вертикали: **Screen.Height / Screen.TwipsPerPixelY**

Пример:

```
MsgBox "Ваше разрешение экрана: " & Screen.Width / _
Screen.TwipsPerPixelX & "x" & Screen.Height / Screen.TwipsPerPixelY
```

13. Бегущая строка

Цель работы: Создание бегущей строки, которая будет "бегать" от начала формы к концу и по форме туда и обратно.

Для создания такой строки, предлагается два варианта:

***Вариант 1*:** 1.Разместим в форму таймер. *(Interval: чем больше, тем медленнее и наоборот)*

2. На форму *Label1*, который бегать должен и переименовать его в скажем, *lblRunStr*. Впишем туда, например текст "Язык программирования Visual Basic".

3. В событие "таймер":

```
Private Sub Timer1_Timer( )    'Если левая точка lblRunStr больше ширины формы тогда пусть эта точка будет на столько меньше нуля сколько lblRunStr есть в длину.
  If lblRunStr.Left > Me.Width Then lblRunStr.Left = -lblRunStr.Width
'Движение в лево на 40 пикселов если надо быстрее, то надо ставить больше 40 и наоборот
  lblRunStr.Move lblRunStr.Left + 40, lblRunStr.Top
End Sub
```

***Вариант 2*:** Первые два пункта такие, как и в первом варианте. Только в окне свойств нужно вписать у *lblRunStr* в тэге *(Tag)* "->".

1. В событие "таймер":

```
Private Sub Timer1_Timer( )
If lblRunStr.Left + lblRunStr.Width > Me.Width Then 'Если правая точка достигла конца формы, то пусть тэг у lblRunStr будет "<-"
```

```
lblRunStr.Tag = "<-"
End If
If lblRunStr.Left < 0 Then
'Если левая точка достигла начала формы, то пусть тэг будет "->"
lblRunStr.Tag = "->"
End If
Select Case lblRunStr.Tag 'Выбор тэга
Case "->"       'Если тэг равен "->"то пусть строка движется вправо
  lblRunStr.Move lblRunStr.Left + 40, lblRunStr.Top
Case "<-" 'Если тэг равен "< - " то пусть строка движется влево
  lblRunStr.Move lblRunStr.Left - 40, lblRunStr.Top
End Select      End Sub
```

2. Сохраните проект и выполните его.

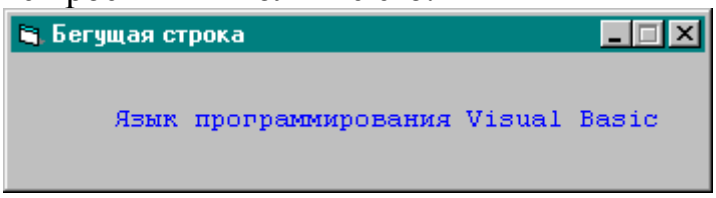


Рис. 16.

14. Графические эффекты

Цель работы: Овладение навыками работы с графическими эффектами и методами изменения графических изображений форм.

1. Для начала создайте стандартный проект, объект *Image1* с нужным рисунком.

2. Создайте простую форму и измените её размеры на *Height = 3780* и *Widht = 4905*. Измените цвет формы, со стандартного на другой, т.е. в окне *Properties* установите для свойства *BackColor* значение &H8000000E&.

3. Полностью заполните форму (Рис.17) рисунком. А это можно сделать следующим кодом:

```
Private Sub Form_Paint( )
  'объявление переменных-счётчиков
  Dim X As Integer
  Dim Y As Integer
'запускаем цикл, заполняющий форму по горизонтали, т.е. картинка "нашлёпывается" через расстояние своей ширины
For X = 0 To Me.Width Step Image1.Width
'запускаем цикл, заполняющий форму по вертикали, т.е. картинка "нашлёпывается" через расстояние своей высоты
For Y = 0 To Me.Height Step Image1.Height
```

```
            Me.PaintPicture Image1, X, Y
            Next Y
  Next X
End Sub
```

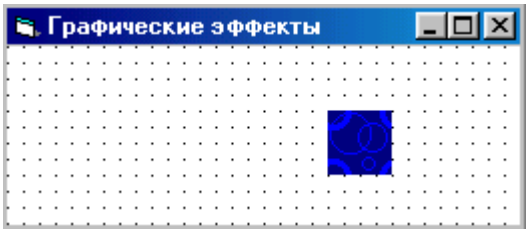


Рис.17.

4. Ниже следующим кодом можно заполнить форму картинкой в шахматном порядке (Рис.18):

```
Private Sub Form_Paint( )
  'объявление переменных-счётчиков
  Dim A As Integer
  Dim B As Integer
  Dim X As Integer
  Dim Y As Integer
  'запускаем цикл, заполняющий форму по горизонтали через одну, т.е. картинка "нашлёпывается" через двойное расстояние своей ширины
  For X = 0 To Me.Width Step 2 * Image1.Width
  'запускаем цикл, заполняющий форму по вертикали через одну, т.е. картинка "нашлёпывается" через двойное расстояние своей высоты
  For Y = 0 To Me.Height Step 2 * Image1.Height
            Me.PaintPicture Image1, X, Y
            Next Y
  Next X
  'запускаем цикл, заполняющий форму по горизонтали через одну, т.е. со сдвигом вправо на расстояние своей ширины, картинка также "нашлёпывается" через двойное расстояние своей ширины
  For A = Image1.Width To Me.Width Step 2 * Image1.Width
  'запускаем цикл, заполняющий форму по вертикали через одну, т.е. со сдвигом вниз на расстояние своей высоты, картинка также "нашлёпывается" через двойное расстояние своей высоты
  For B = Image1.Height To Me.Height Step 2 * Image1.Height
            Me.PaintPicture Image1, A, B
            Next B
  Next A
End Sub
```

5. После выполнения нашего приложения эффект возможно выглядит таким образом: *2 Заполнение формы в шахматном порядке.*

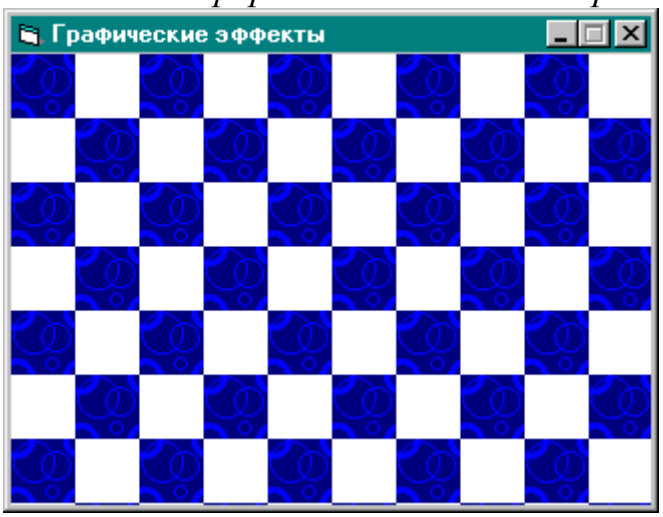


Рис. 18.

15. Работа переключателей

Цель работы: Создание простейшего Windows-приложения с различными переключателями, которые влияют на отображаемый рисунок:

Вид геометрической фигуры изменяется с помощью переключателей. Фигура – это элемент управления *Shape.*

1.Создайте стандартный проект.

2. Измените заголовок окна формы с Form1 на «Обработка событий» в окне Properties установите для свойства Caption значение «Обработка событий».

3. Создайте объект *Command* и назовите его *cmdExit*.

4. Поместите на форму элемент *Shape1*.

5. Цвет фигуры задается свойствами *BackColor* и *BackStyle*.

6. Добавьте в форму переключатель *Option*.

7. Для создания второго переключателя поместите на форму рамку, щелкните на панели *ToolBox* элемент *Option* и нарисуйте его, которая изменяет вид геометрический фигуры.

Вид фигуры определяется свойством *Shape*, возможные значения свойства *Shape* можно просмотреть в окне свойств. Создание процедуры можно начинать от объекта: выбираем объект (позицию переключателя) двойным щелчком и получаем объявление процедуры обработки события, которое является событием по умолчанию для данного объекта. Другой способ – работа со списками редактора кода.

8. Теперь выберите событие переключатель Option и впишите:

```
Private Sub Option1_Click( )
  Shape1.Shape = 0
End Sub
```

9. Теперь выберем событие *Option2* впишем:

```
Private Sub Option1_Click( )
  Shape1.Shape = 1
End Sub
```

10. Далее выберем cmdExit и впишем:

```
Private Sub cmdExit_Click( )
If MsgBox("Уверены?", vbYesNo, "Выход из окне") = vbYes Then
Unload Me
  Else
        Cancel = 1
  End If
End Sub
```

11. Самостоятельно записать код для процедур:

Private Sub Option2_Click( ), Private Sub Option3_Click( ), Private Sub Option4_Click( ) и Private Sub Option5_Click( ).

12. Сохраните проект и дайте команду на выполнение.

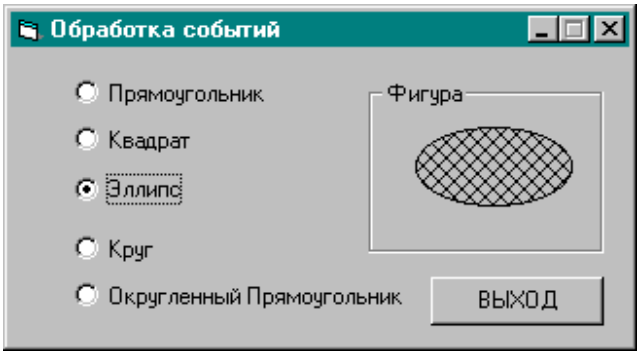


Рис. 19.

16. Диалоговые окна для обмена сообщениями

Цель работы: Использование возможностей применения диалового окна, создаваемая функцией *MsgBox( ).*

Существует несколько типов диалоговых окон, которые необходимы для поддержания в программе интерактивного режима работы конечного пользователя (ввод сообщений пользователю, прием и интерпретация указаний, введенных пользователем, и др.).

Для создания диалоговых окон, мы предлагаем два варианта:

Вариант 1: Простое окно-сообщение.

MsgBox(“строка_сообщения”)

Если в сообщении должно присутствовать значение переменной или элементы массива переменной, элемент структуры пользовательского типа данных и т. п., следует преобразовать значение в строковый тип и обеспечить конкатенацию строк.

1. Создайте стандартный проект.

2. Создайте объект *Command* и и назовите его *cmdRun*.

3. Теперь выберите событие *cmdRun_Click( )* и впишите:

```
Private Sub cmdRun_Click( )      'Объявление процедуры
Dim A As Single     'Объявление переменной
Randomize     'Запуск генератора случайных чисел
  A = Rnd       'Присвоение переменной значения случайного числа
  MsgBox "Значение случайного числа" & Str(A)  ‘Вывод сообщения
End Sub                  'Конец процедуры
```

4. В результате выполнения процедуры будет выведено окно сообщение (Рис. 20).

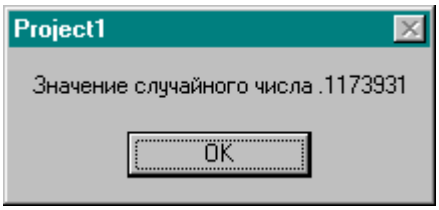


Рис. 20.

Вариант 2:

II. Окно сообщение с командами кнопками

1. Общий формат оператора:

MsgBox(“строка_сообщение”[, < кнопки >] [, ”заголовок_окна”] [, <файл - подсказки>, <контекст>])

где <строка_сообщение> - максимальная длина строки – 1024 символа;

<кнопки> - число, являющееся суммой кодов выбранных типов кнопок и пиктограммы, или имена кнопок;

<файл - подсказки> - имя файла – подсказки для контекстно-зависимой помощи при работе в окне, строка символов;

<контекст> - число, которое назначено подсказке для данного окна.

2. В таблице 1 приведены коды задания командных кнопок и пиктограмм в функции MsgBox ( ). Таблица 1.

Код	Константа	Описание
0	vbOKOnly	OK
1	vbOKCancel	OK, Отмена
2	vbAbortRetryIgnore	Прекратить, Повторить, Игнорировать

3	vbYesNoCancel	Да, Нет, Отмена
4	vbYesNo	Да, Нет
5	vbRetryCancel	Повторить, Отмена

3. Создайте объект *Command* и и назовите его *cmdShow*.

4. Далее выберем *cmdShow* и впишем:

```
Private Sub Command2_Click( )
  Dim response As Integer
  Dim Msg As String
  Dim Title As String
  Dim Help As String
  Dim Style As Integer
  Dim Ctxt As Integer
        Msg = "Вы хотите продолжить?"
  Style = 35
        Title = "Пример окна-сообщения"
  Help = "Demo.hlp"
  Ctxt = 0
        Responce = MsgBox(Msg, Style, Title, Help, Ctxt)
End Sub
```

5. В результате выполнения программы будет сформировано следующее окно сообщения с кнопками (Рис. 21).

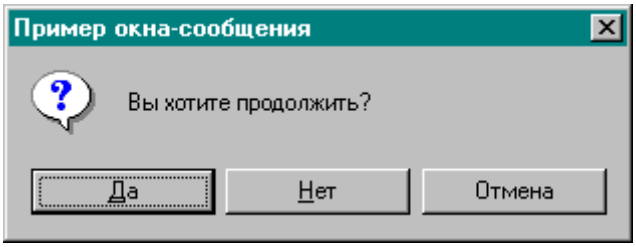


Рис. 21.

17. Создание контекстных меню

Цель работы: Изучение методов создания контекстных меню, выводимых при щелчке правой кнопки мыши.

1. Сначала создайте простые две формы.

2. Щелкните на кнопочке Menu Editor вверху слева. Там меню нарисовано  .

3. Щёлкните на нём.

4. Теперь нам нужно создать меню, которое и будет контекстным. В поле *Caption* вписываем какое-нибудь имя, неважно какое, оно чисто для вас, т.к. отображаться эта надпись нигде не будет. Введём, например,

My_Popup. В поле *Name* вводим, к примеру *mnuPopup* и убираем галочку с *Visible* - делаем меню невидимым. Всё, теперь окно *Menu Editor* должно выглядеть как на следующем рисунке:

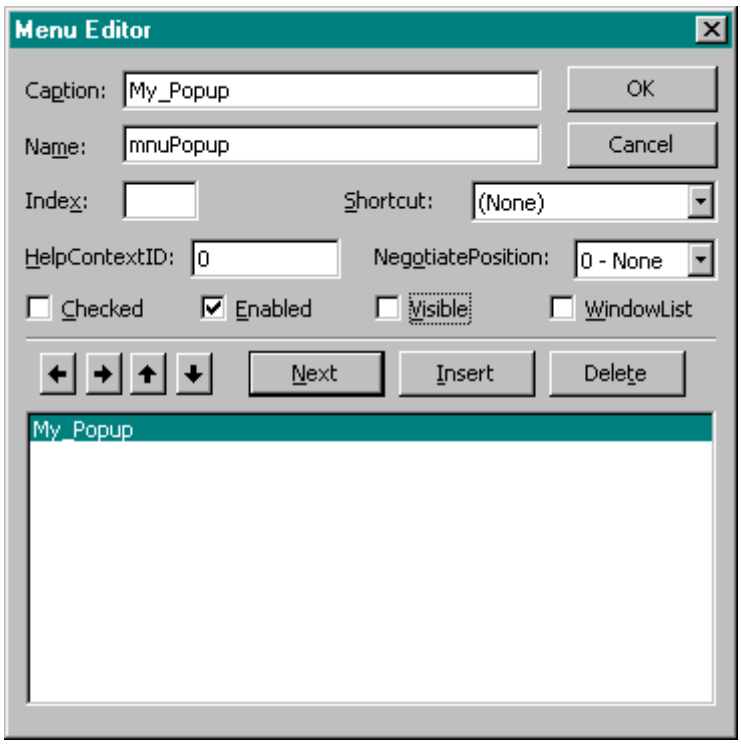


Рис. 22.

Теперь надо ввести нужные пункты меню. Щёлкните на кнопке *Next*, а потом на стрелке вправо. Появятся четыре точки. Это значит, что следующий пункт находится в подуровне меню *Popup*, так сказать является его составляющим компонентом. В пустое поле *Caption* введите «О программе» (показать сообщение) и в поле *Name* введите *mnuAboutMe*. Щелкните ещё раз на *Next*, но стрелку больше нажимать не надо - подуровень стоит теперь по умолчанию. В *Caption* введите «Введите имя», в *Name* вписываем *mnuYN*, потом снова на *Next*. Теперь сделаем маленький визуальный эффект - введём разделитель между командами меню и командой выхода. Для примера, запустите любую программу под *Windows*, щёлкните на меню *File* и посмотрите вниз, команда *Exit* всегда отделена сепаратором от всех остальных. Так сделаем такой же разделитель. В поле *Caption* вводим "-", только без кавычек (минус). В поле *Name* введите, что хотите это не нужно Вам вообще, например *Sep*. Теперь снова *Next*, и в поле *Caption* введите «Выход», а в поле *Name* введите *mnuExit*. Всё, наше меню готово. Осталось теперь только "нафаршировать" его кодом и перекрепить как контекстное. Нажмите кнопку **OK**.

5. Сделайте двойной щелчок на форме и откройте окно *Code*. В окошке, где стоит *Load* (событие формы стоящие по умолчанию) и выберите событие *MouseUp*. Т.е. событие происходит при отпускании кнопки мыши.

Почему так объясним чуть позже. Передаваемые значения для подпрограммы *MouseUp* это *Button, Shift, X, Y* но нам нужен *Button*. Этот аргумент возвращает значение, показывающее, какая кнопка была нажата. Значение 1 это левая кнопка, 2 - это правая. Так же есть бейсиковские константы *vbLeftButton* и *vbRightButton*, их значения непосредственно 1 и 2. Поэтому в событие *MouseUp* вписываем:

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
'проверяем: если нажата была правая кнопка, то цепляем к форме My_Popup-меню, как PopUp
If Button = vbRightButton Then
        Me.PopupMenu mnuPopup, , X, Y, mnuExit
        End If
End Sub
```

Теперь объясним строку Me.PopupMenu mnuPopup, , X, Y, mnuExit отдельно. Во-первых, вот синтаксис:

object.**PopupMenu** menuname, flags, x, y, boldcommand

object объект, к которому цепляем меню.

menuname имя меню, которое хотим прицепить

flags константа, описывающая место появление меню

x расположение меню по иксу

y расположение меню по игрику

boldcommand пункт меню, который должен быть выделен жирным (только один пункт!)

Во-вторых, вот константы для *flags*:

Константы	Значение	Описание
VbPopupMenuLeftAlign	0	(по умолчанию) меню появляется слева под курсором.
VbPopupMenuCenterAlign	4	меню появляется по центру под курсором.
VbPopupMenuRightAlign	8	меню появляется справа под курсором.

Т. е. мы можем подставлять как константы, так и значения. А если пропустим, то меню появится там, где оно стоит по умолчанию.

6. Теперь выберем событие *MouseDown* и впишем:

```
Private Sub Form_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
  'при каждом нажатии контекстное меню спрятывается
  Me.mnuPopup.Visible = False
End Sub
```

Теперь объясняем, почему так. Здесь мы с имитируем точное появление контекстного меню как в *Windows Explorer*. Обратите внимание на то,

что меню появляется только при нажатии правой клавиши, а при повторном нажатии исчезает и появляется на новом месте.

Теперь там, где стоит форм, выберем *mnuExit* - появится событие на щелчок по пункту меню "Выход". Впишем внутрь только три буквы:

```
Private Sub mnuExit_Click( )
  End 'закрытие программы
End Sub
```

Теперь щелкнем там же и выберем событие *mnuYN*, затем введём:

```
Private Sub mnuYN_Click( )
  YourName = InputBox("Введите имя!") 'запрос имени
End Sub
```

Далее выберем *mnuAboutMe* и впишем:

```
Private Sub mnuAboutMe_Click( )
  Form2.Show
End Sub
```

Всё, теперь осталось запустить проект и проверить работоспособность контекстного меню.

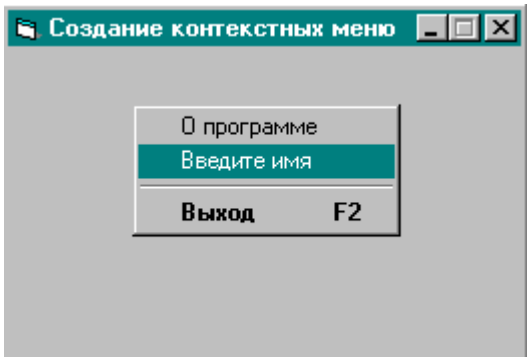


Рис. 23.

18. Изменение вида курсора в текстовом поле

Цель работы: Изучение методов изменения курсора в текстовом поле.

Часто требуется сделать в текстовом поле жирный курсор, вместо маленького обычного, или сделать необычный курсор как в DOS, только сверху. Иногда задается вопрос: а можно ли заменить курсор вообще маленькой картинкой? Ниже дается ответы на все эти вопросы.

1. Создайте стандартный проект и текстовое поле на нём.

2. Щелкните два раза на форме и впишите в раздел формы *General Declarations* объявление двух, нужных нам API-функций:

```
'это сама функция создания курсора в текстовом поле
```

Private Declare Function CreateCaret Lib "user32" (ByVal hwnd As Long, ByVal hBitmap As Long, ByVal nWidth As Long, ByVal nHeight As Long) As Long

'эта функция показывает созданный курсор

Private Declare Function ShowCaret Lib "user32" (ByVal hwnd As Long) As Long

Теперь об этих функциях. Значит, первая создаёт курсор из следующих заданных параметров:

hwnd - идентификатор текстового поля

hBitmap - картинка (название объекта или, если не надо, 0)

nWidth - ширина курсора

nHeight - высота курсора

Вторая функция отображает курсор:

hwnd - идентификатор поля с курсором

Прежде чем вызывать функцию создания курсора, нужно показать форму с окном.

3. В событие формы *Form_Load* впишите:

```
Private Sub Form_Load( )
  Show
        Call CreateCaret(Text1.hwnd, 0, 10, 13)
        Call ShowCaret(Text1.hwnd)
End Sub
```

4. В событие текста *Text_MouseDown* пишите следующий код:

```
Private Sub Text1_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
Select Case Index
Case 0
        Call CreateCaret(Text1(Index).hwnd, 0, 6, 13)
        Call ShowCaret(Text1(Index).hwnd)
        Text1(0).SelStart = Len(Text1(Index).Text)
Case 1
        Call CreateCaret(Text1(Index).hwnd, 0, 8, 1)
        Call ShowCaret(Text1(Index).hwnd)
        Text1(0).SelStart = Len(Text1(Index).Text)
Case 2
        Call CreateCaret(Text1(Index).hwnd, 0, 2, 30)
        Call ShowCaret(Text1(Index).hwnd)
        Text1(0).SelStart = Len(Text1(Index).Text)
Case 3
        Call CreateCaret(Text1(Index).hwnd, 0, 1, 8)
        Call ShowCaret(Text1(Index).hwnd)
        Text1(0).SelStart = Len(Text1(Index).Text)
  End Select
```

End Sub

Теперь запустите проект, в текстовом окне в самом начале должен стоять жирный курсор. Нужен свой, оригинальный курсор, какого нет ни у кого? Нарисуйте же его сами, а потом создайте проект на форме и вставьте.

Для того чтобы поместить эту картинку вместо курсора используйте примерно следующую строку:

Call CreateCaret(Text1.hwnd, Image1.Picture, 0, 0)

и вместо курсора действительно будет картинка!

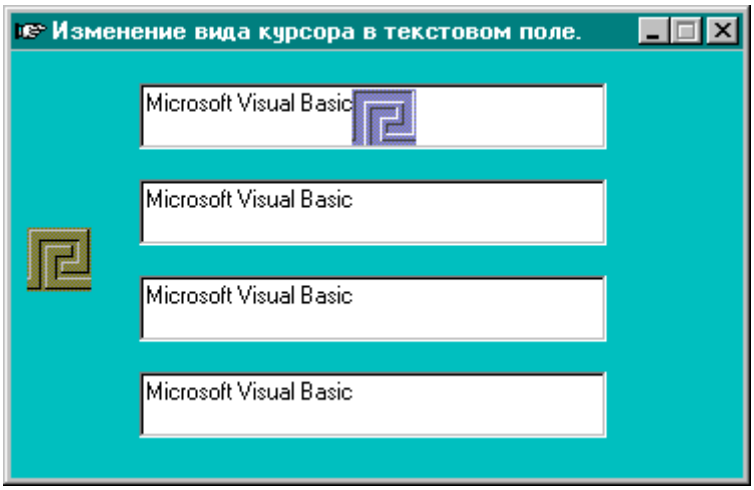


Рис. 24.

19. Изменение интервала мерцания курсора

Цель работы: Изучение методов, с помощью которого изменяется интервал мерцания курсора.

Мы будем использовать две функции:

1. Создайте стандартный проект с длинным текстовым полем (*Text1*), коротеньким(*Text2*) и кнопкой(*Command1*) и впишите в раздел формы *General Declarations*:

'функция, возвращающая текущую скорость мерцания курсора в милисекундах (не требует параметров)

Private Declare Function GetCaretBlinkTime Lib "user32" ( ) As Long

'а эта функция устанавливает скорость равную переданному ей параметру

Private Declare Function SetCaretBlinkTime Lib "user32" (ByVal wMSeconds As Long) As Long

2. Уберите стандартный текст из маленького текстового поля.

3. Сделайте двойной щелчок по форме и в раздел *Form_Load* впишите:

```
Private Sub Form_Load( )
'в текстовом поле Text2 отображаем текущую скорость мерцания курсора
  Text2.Text = GetCaretBlinkTime
End Sub
```

4. Теперь в событие *Command1_Click* впишите:

```
Private Sub Command1_Click( )
'при щелчке вызываем функцию установки скорости мерцания и передаём ей значение текстового поля в надежде, что оно было изменено:
  Call SetCaretBlinkTime(Text2.Text)
'потом устанавливаем фокус на большом текстовом поле и посадим курсор конец текста
  Text1.SetFocus
  Text1.SelStart = Len(Text1.Text)
End Sub
```

Этот пример особенно хорошо отображает сходство API и простых функций.

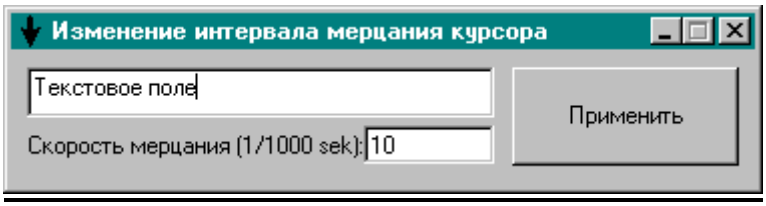


Рис. 25.

20. Использование объектов файловой системы

Цель работы: Использование объектов файловой системы.

Visual Basic предлагает три очень полезных объекта для предоставления доступа к файловой системе. Это списки дисковых накопителей, которые позволяют вам пролистывать доступные диски в системе; списки директорий, которые позволяют вам ориентироваться в папках на выбранном диске, и списки файлов, позволяющие вам выбрать нужный файл в папке. В следующем упражнении мы используем три объекта файловой системы для построения программы, называемой Browser (Просмотр), которая обнаруживает и показывает файлы, содержащие картинки.

1. В меню *File* (Файл) щелкните на строке *New Project* (Новый проект), затем щелкните на OK для создания нового стандартного выполняемого файла.

2. Увеличьте размер вашей формы так, чтобы в нем поместились средства управления файловой системой и окно для отображения картинок. Вокруг формы появится полосы прокрутки.

3. Щелкните на кнопке *DriveListBox* (Список дисководов) в панели инструментов.

4. Еще добавьте в форму кнопку *DirListBox* (Список директорий) в панели инструментов и затем добавьте окно списка директорий в форму под списком дисководов. Оставьте место примерно для пяти – шести папок в окне списка.

5. Еще поместите в форму элемент *FileListBox* (Список файлов) в панели инструментов и добавьте окно списка файлов в форму под списком директорий. Отведите место для отображения шести – семи имен файлов. Объект «Список файлов» позволяет пользователю выбирать файл в файловой системе.

6. Добавьте в форму кнопки *Image (Рисунок)*. Экран будет выглядеть так, как показано на рисунке 26.

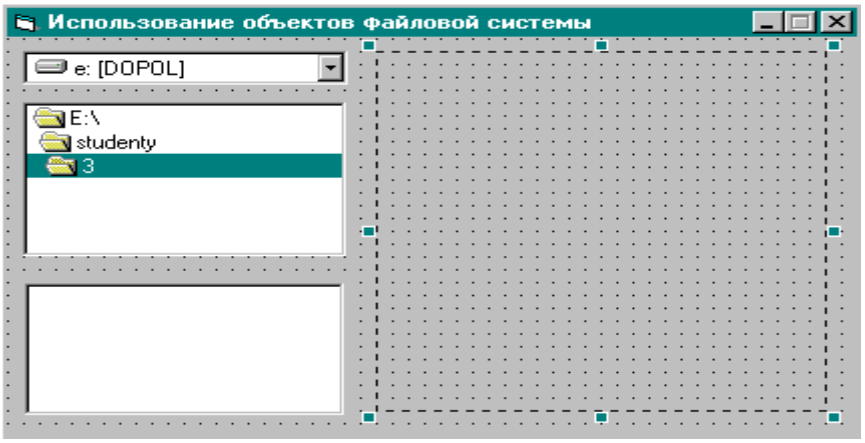


Рис. 26.

7. Теперь в окне *Properties* установите следующие свойства.

Объект	Свойства	Установка
File1	Pattern	*.bmp; *.wmf; *.ico
Image1	Stretch	True
Image2	BorderStyle	1-Fixed Single

8. Дважды щелкните на объекте «Список дисководов» в форме, а затем в событие *Drive1_Change* наберите между строками *Private Sub* и End Sub следующее программное утверждение:

```
Private Sub Drive1_Change( )
  Dir1.Path = Drive1.Drive
End Sub
```

9. Закройте окно программного кода, затем дважды щелкните на списке директорий в форме и добавьте следующее программное утверждение к процедуре события *Dir_Change*.

```
Private Sub Dir1_Change( )
  File1.Path = Dir1.Path
End Sub
```

10. Дважды щелкните на списке файлов на форме и добавьте следующий код к процедуре событие File1_Click:

```
Private Sub File1_Click( )
  SelectedFile = File1.Path & "\" & File1.FileName
  Image1.Picture = LoadPicture(SelectedFile)
End Sub
```

11. Запустите и при успешной работе сохраните программу. Например, экран будет выглядеть так, как показано на рисунке 27.

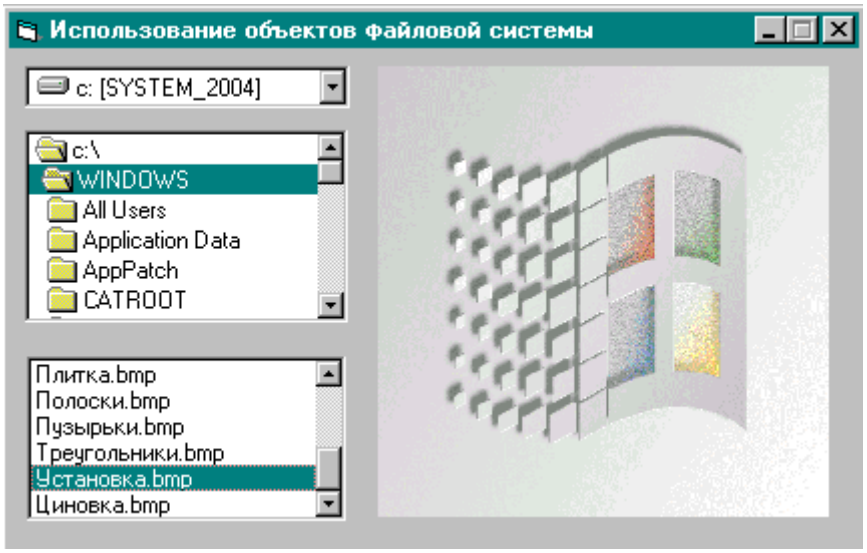


Рис. 27.

21. Разработка простого проигрывателя

Цель работы: Использование возможностей элемента управления MCI.

Для управления устройствами мультимедиа в Visual Basic применяется специальный интерфейс *MCI (Multimedia Control Interface)*. Основным компонентом этого интерфейса является элемент управления *MMControl* с большим набором свойств и команд, позволяющих управлять мультимедиа и полностью их контролировать. Для изучения работы элемента управле-

ния *MMControl* создадим приложения для проигрывателя звуковых файлов в формате *WAV*. Чтобы создать такой проигрыватель, выполните следующие действия:

1. Создайте новый стандартный проект.

2. Присвойте проекту имя *MyMultiMedia*. Для этого откройте окно свойств проекта, выбрав команду *Project1 Properties* (Свойства *Project1*) меню *Project* (Проект).

3. Присвойте форме проекта имя *FormPlayer*. В свойства Caption формы введите заголовок **«Мультимедиа плейер»**.

4. Присоедините к проекту библиотеку компонентов *Microsoft Multi-Media Control 6.0*, воспользовавшись диалоговым окном *Components* (Компоненты), которое открывается в меню *Project* (Проект) командой *Components* или *Ctrl+T*.

5. Нам потребуется диалоговое окно поиска проигрываемых файлов, поэтому установите также в окне *Components* флажок для библиотеки компонентов *Microsoft CommonDialog Control 6.0*.

6. Добавьте в форму *FormPlayer* элемент управления *MMControl*, и присвойте ему имя *MMControlCDPlayer*.

7. Еще добавьте в форму элемент управления *CommonDialog*, и присвойте ему имя *cdPlayer*.

8. Поместите объект *CommandButton* в окно формы *Form1*. Назовите эту кнопку *cbFindFile* и введите в свойства *Caption* значение, **Найти файл**. Созданный проект имеет следующий вид:

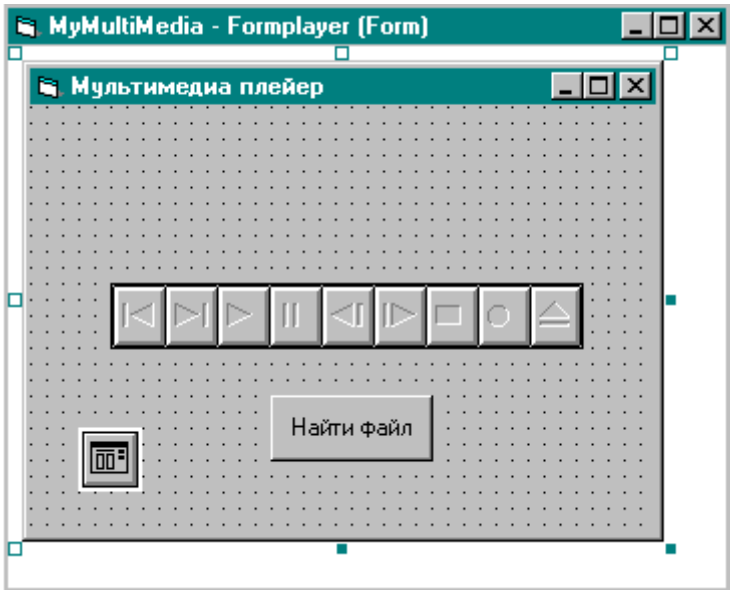


Рис. 28.

10. Сделайте двойной щелчок по форме и в раздел *Form_Load* впишите:

```
Private Sub Form_Load( )
  MMControlCDPlayer.Notify = False
```

```
  MMControlCDPlayer.Wait = True
  MMControlCDPlayer.Shareable = False
  MMControlCDPlayer.DeviceType = "WaveAudio"
End Sub
```

11. Теперь в событие cbFindFile_Click впишите:

```
Private Sub cbFindFile_Click( )
  cdPlayer.ShowOpen
  MMControlCDPlayer.FileName = cdPlayer.FileName
  MMControlCDPlayer.Command = "Open"
End Sub
```

12. Щелкните два раза на форму и вставьте следующий код:

```
Private Sub Form_Unload(Cancel As Integer)
  MMControlCDPlayer.Command = "Close"
End Sub
```

13. Запустите приложение. При нажатии кнопки **«Найти файл»** открывается диалоговое окно Открытие файла, позволяющее осуществить поиск, а затем открыть файл для воспроизведения.

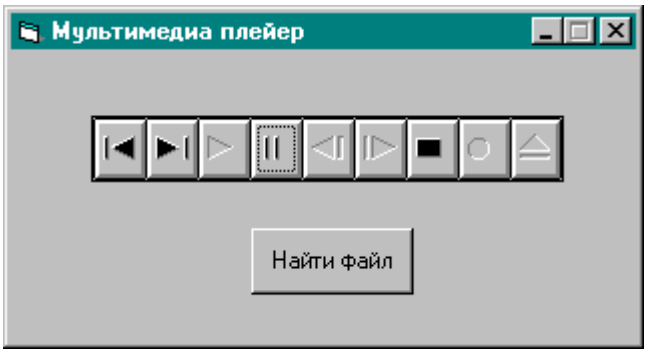


Рис. 29.

22. Вызов системных компонентов из VB

Цель работы: Изучение методов вызова системных компонентов из VB.

Для большинства системных установок *Windows 95/98/NT* предлагает свои компоненты. Например, чтобы поменять расширение экрана, мы щелкнем правой кнопкой на рабочем столе --> "Свойства" --> "Установки". Также существует ещё несколько десятков таких окон, предлагающих что-либо изменить в системе. Так вот каждое из таких окон можно вызвать просто из VB, не используя ни одной API-функции. Каждый набор таких окон находится в библиотеках *Windows* с расширением "*.cpl"*. Например, в библиотеке **«Desk.cpl»** находятся четыре стандартных окна (точнее ***одно***

окно с четырьмя закладками): "Фон", "Заставка", "Настройка", "Оформление". Для того чтобы вызвать не просто нужное окно, а ещё и нужную закладку, то в параметрах вызова надо установить индекс нужной закладки. Индексы некоторых из них Вы найдёте в этом же примере.

Вызов происходит при помощи обычной функции *Shell*, но немного необычным способом:

```
Call Shell("rundll32.exe shell32.dll,Control_RunDLL desk.cpl,,0" )
```

Откроет окно свойств экрана с закладкой "Фон".

А теперь некоторые параметры:

Свойства экрана / фоновый рисунок = "desk.cpl,,0"
Свойства экрана / заставка = "desk.cpl,,1"
Свойства экрана / оформление = "desk.cpl,,2"
Свойства экрана / настройка = "desk.cpl,,3"
Спец. возможности / клавиатура = "access.cpl,,1"
Спец. возможности / звуки = "access.cpl,,2"
Спец. возможности / экран = "access.cpl,,3"
Спец. возможности / мышь = "access.cpl,,4"
Спец. возможности / общие = "access.cpl,,5"
Главная для
@0 мышь = "main.cpl @0"
Дата и время/
дата и время = "timedate.cpl, ,0"
часовой пояс = "timedate.cpl, ,1"
Модем = "modem.cpl"
Сеть/конфигурация = "netcpl.cpl"
Пароль = "password.cpl"
Система = "sysdm.cpl, ,0"
' ,,0 система / общие
' ,,1 система / устройства
' ,,2 система / оборудование
' ,,3 система / быстродействия
ODBC = "odbccp32.cpl"
Установка и удаление программ = "appwiz.cpl,,0"
' ,,0/1 инсталляция / деинсталляция
' ,,2 установка Windows
' ,,3 загрузочная дискета
Язык и стандарты = "intl.cpl,,0"
' ,,0 региональные стандарты
' ,,1 числа
' ,,2 денежная единица
' ,,3 время
' ,,4 дата
Джойстик = "joy.cpl,,1"

```
        ' ,,0 общие
        ' ,,1 прочие
Мультимедия = "mmsys.cpl,,0"
        ' ,,0 аудио
        ' ,,1 видео
        ' ,,2 миди
        ' ,,3 компакт – диск
        ' ,,4 устройства
Интернет = "inetcpl.cpl,,0"
        ' ,,0 общие
        ' ,,1 безопасность
        ' ,,2 содержание
        ' ,,3 соединение
        ' ,,4 программы
        ' ,,5 дополнительно
DeskTop-темы (Plus! - пакет) = "Themes.cpl "
```

Теперь для укрепления полученных знаний; напишем процедуру следующим образом:

1. Создайте стандартный проект.

2. Поместите объект Frame1 в окно формы Form1, и введите в свойства Caption значение, **Выбрать.**

3. Теперь скопируйте её в буфер обмена и вставьте. Появится фрейм (“хотите ли Вы создать массив из этих элементов - скажите, что хотите”). Второй объект OptionButton1(1) появится на форме. Щелкните на Вставить ещё четыре раза. Появятся OptionButton1(2), OptionButton1(3), OptionButton1(4), OptionButton1(5). Сделайте форму "пошире" и разместите все элементы так, чтобы все они были на виду.

4. Щелкните два раза на Option1_Click (которая будет выбирать включатели) - откроется окно Code. Впишите:

```
Private Sub Option1_Click(Index As Integer)
  W = Index
End Sub
```

5. Добавьте в форму CommandButton, и присвойте ему имя cmdShow, и введите в свойства Caption значение **Выставка**. Записать код для обработки события Click на объекте CommandButton1:

```
Private Sub cmdShow_Click( )
  Dim W As Integer ' объявление переменных
  Select Case W
  Case 0 ' свойства экрана/фоновый рисунок
         Tx$ = "desk.cpl,,0"
  Case 1 ' мультимедиа/
         ' ,,0   аудио
         ' ,,1   видео
```

```
        ' ,,2   миди
        ' ,,3   компакт – диск
        ' ,,4   устройства
        Tx$ = "mmsys.cpl,,4"
    Case 2
        ' ,,0  дата и время
        ' ,,1  часовой пояс
        Tx$ = "timedate.cpl, ,0"
    Case 3 ' сеть/конфигурация
    Tx$ = "netcpl.cpl"
    Case 4 ' спец. возможности/клавиатура
        Tx$ = "access.cpl,,1"
    Case Else
        ' DeskTop-темы (Plus! - пакет)
        Tx$ = "Themes.cpl "
    End Select
     X = Shell("rundll32.exe shell32.dll,Control_RunDLL " + Tx$)
End Sub
```

Что делает эта процедура? Она всего лишь соединяет имя библиотеки с индексом, переведённым в строку, и запускает всё это дело. В итоге проект, возможно, выглядит следующим образом:

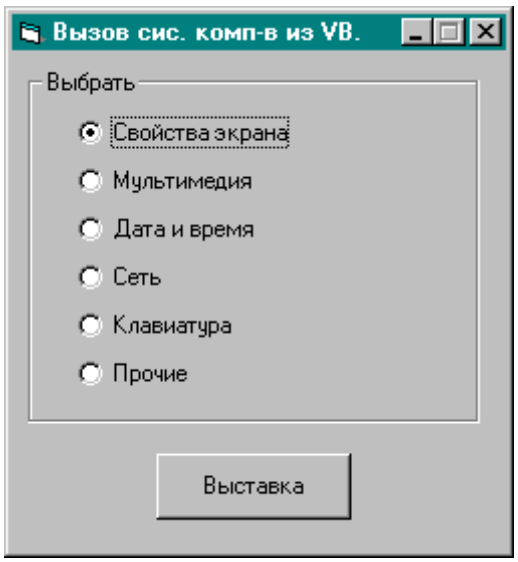


Рис. 30.

23. Работа с базами данных

Цель работы: Умение обращаться с файлами данных посредством объекта Data.

Для создания базы данных:

1. Создайте стандартный проект, щелкните на *ToolBox,* выберите кнопку *Data* и натяните на форму такую полоску, чтобы она была не очень широкой и в самом низу формы.

3. Теперь создайте на форме четыре кнопки и назовите их (по порядку создания): *cmdAdd (Добавить), cmdDelete (Удалить), cmdUpdate (Сохранить), cmdExit (Выход).*

4. Создайте на форме три текстовых поля одинаковой длины.

5. Теперь свяжите объект *Data* с какой-нибудь базой данных. Например, с базой данных Biblio.mdb. Активизируйте объект *Data* и в свойствах, в *DatabaseName* выберите этот файл. Теперь там же, но в поле *RecordSource* выберите раздел *"Authors"*. Этим мы выберем только нужную часть базы данных.

6. Выделите первое текстовое поле и в свойствах, в *DataSource* выберите единственную в списке, созданную нами базу данных *Data1*. Тем самым мы связываем это текстовое поле с объектом *Data1*. А в поле свойств *DataField* выберите *"Au_ID"*. Теперь это поле отображает идентификационный номер каждого автора. То же самое проделайте со следующими двумя другими полями, но в *DataField* вместо *"Au_ID"* выберите *"Author"* и *"Year Born"*.

Таким образом, мы написали простейшую программу для отображения содержания базы данных. Теперь научимся редактировать её.

7. Щёлкните два раза на кнопку **«Добавить»** и введите:

```
Private Sub cmdAdd_Click( )
Data1.Recordset.AddNew
'Все поля, которые могут быть отредактированы будут очищены и подготовлены, поле Year Born будет установлено на 0, т. к. оно должно иметь какое-то значение поле Au_ID не будет очищено вообще, а изменено на самую последнюю позицию.
End Sub
```

8. Следующая кнопка сохраняет внесённые изменения в базе данных. Сделайте двойной щелчок по кнопке **«Сохранить»** и введите:

```
Private Sub cmdSave_Click( )
Data1.UpdateRecord        'Сохраняем
'Без следующей строки после сохранения в полях автоматически появились бы самые первые записи. Поэтому закладке присваиваем идентификатор последней изменённой записи и в полях останутся наши записи
Data1.Recordset.Bookmark = Data1.Recordset.LastModified
End Sub
```

9. Для удаления записей сделайте двойной щелчок по кнопке **«Удалить»** и введите:

```
Private Sub cmdDelete_Click( )
```

'Чтобы выполнить последовательность инструкций над одиночным объектом Data1, не перечисляя его каждый раз, используем инструкцию With

```
With Data1.Recordset
  .Delete
  .MoveNext
  If .EOF Then .MoveLast
End With
End Sub
```

10. Теперь для кнопки **«Выход»** введите:

```
Private Sub cmdExit_Click( )
  End
End Sub
```

Таким образом, мы создали почти настоящую программу с базой данных.

10. Запускайте проект. Если нажать на правую кнопку >, то содержание полей изменяется. Теперь нажмем на правую кнопку >|. В полях появляется самые последние записи. То же самое будет происходить при нажатии на левые кнопки |< и <. Нетрудно заметить, что поле, содержащие год рождения, всегда пустое.

11. Теперь нажмите кнопку **«Добавить».**

12. Первое поле не исправляйте, во второе введите, например, свое имя, а в третье - год рождения.

13. Теперь сохраните эту запись, нажав кнопку **«Сохранить».**

14. Таким образом, наша запись сохранена на самом последнем месте. Можем снова "прогуляться" по записям и вернуться, наконец, с помощью правой кнопки >|. Наша запись на месте.

15. Теперь нажмите на кнопку **«Удалить».** Тогда запись исчезает. Но, в этом случае мы не можем удалять записи, сделанные не нами.

16. Поиск записей в базе данных. Если нам нужно найти конкретное имя автора из тысяч других, то для этого нужно воспользоваться одним из способов нахождения:

FindFirst, FindLast, FindNext или *FindPrevious*. В следующей таблице представлены их значения:

Метод	Пояснение
FindFirst	Ищет первую запись в БД
FindLast	Ищет последнюю запись в БД
FindNext	Ищет каждую следующую запись в БД
FindPrevious	Ищет предыдущую запись в БД

17. Создайте кнопку *cmdFind* с надписью ***"Поиск"*** и впишите:

```
Private Sub cmdFind_Click( )
```

'У пользователя запрашивается имя автора, а потом происходит поиск его в БД

```
Data1.Recordset.FindFirst "Author = '" _
        & Trim(InputBox("Введите имя автора")) & "'"
'Если запись не найдена, то появляется следующая надпись:
If Data1.Recordset.NoMatch Then MsgBox "Имя не найдено"
End Sub
```

18. Теперь запустите проект. Найдите какое-нибудь имя из БД и нажмите кнопку **Поиск**.

19. Введите имя и нажмите ОК.

После выполнения этого проекта БД будет выглядит так:

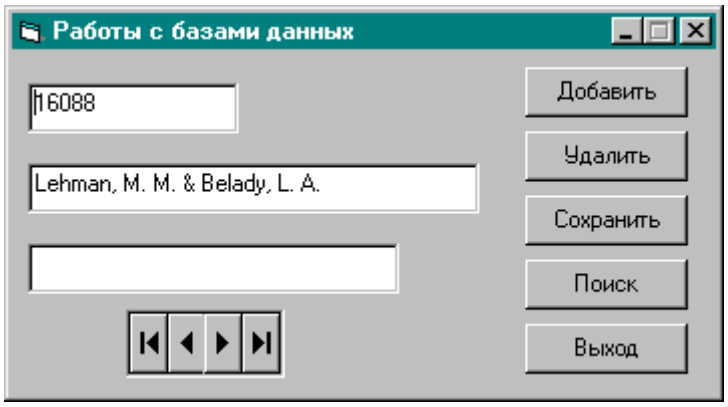


Рис. 31.

24. Проектирование простого отчета

Цель работы: Изучение методов создания простого отчета.

Для проектирования и управления отчетами в Visual Basic используется объектно-ориентированный подход: в расположении пользователя имеется специальный объект *DataReport* и инструментальное средство *Data Report Designer* (Конструктор Отчетов). Прежде чем приступить к созданию отчета следует проверить подключено ли к проекту приложения конструкторов *Data Environment* (Окружение данных) и *DataReport* (Отчет данных). Для этого выберите команду *Components* (Компоненты) меню *Project* (Проект), в открывшемся диалоговом окне *Components* перейдите на вкладку *Designers* (Конструкторы) и проверьте, установлены ли флажки *Data Environment* и *Data Report*. Обычно они устанавливается по умолчанию.

Прежде чем начать проектирование отчета мы рассмотрим простой пример для этой цели:

Для создания отчета необходимо добавить в проект приложения объект *DataReport*. Выполните это следующим образом:

1. В меню *Project* (Проект) выберите команду *Add Data Report* (Добавить отчет). После выполнения этой команды появляется окно объекта *DataReport*. При помощи окна свойство Caption, присвойте ему заголовок **Отчет**.

2. В меню *Project* (Проект) выберите команду *More ActiveX Designers* (другие конструкторы *ActiveX*), а затем значение *Data Environment* (Окружение данных). Открывается окно *Data Environment,* содержащее соединение *Connection1*.

3. Установите курсор на соединение *Connection1*, нажмите правую кнопку мыши и выберите команду *Properties* (Свойства) контекстного меню. Открывается диалоговое окно *Data Link Properties* (Свойства связи с данными). На вкладке *Provider* (Поставщик услуг) выберите поставщика услуг.

4. Перейдите на вкладку *Connection* (Подключение) и укажите имя базы данных, для которой будет создаваться отчет.

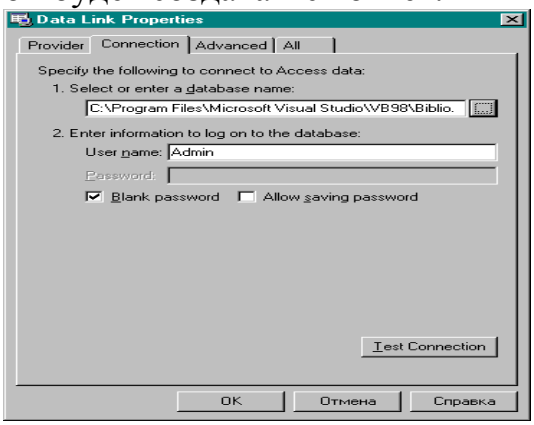


Рис. 32.

5. Чтобы проверить правильность соединения, нажмите кнопку Test Connection (Проверить подключение). Для потверждения выполненных действий закройте окно *Data Link Properties*, нажав кнопку *Ok.*

6. Для этого в окне *Data Environment1* выберите соединение *Connection1* и нажмите кнопку *Add Command* (Добавить команду) на панели инструментов окна или нажмите правую кнопку мыши и выберите команду *Add Command* контекстного меню.

7. После того как источник данных (объект *Command1*) появится в окне окружения, откройте окно свойств данного объекта, выбрав команду *Properties* (Свойства). Присвойте источнику данных имя MyReportCommand.

8. Установите значения полей этого окна как рис. 33, то есть укажите имя таблицы, для которой создается отчет.

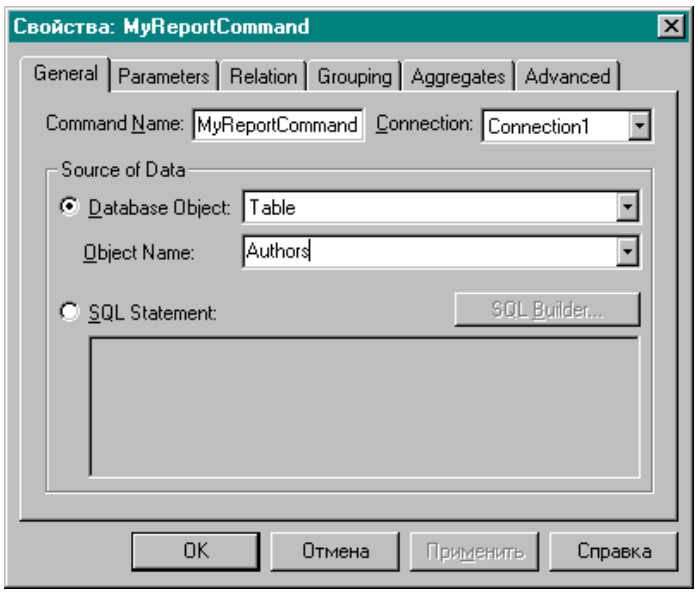


Рис. 33.

9. После создания источника можно установить связь отчета с источником данных с помощью свойств отчета *DataSource* и *DataMember*. В свойтсве *DataSource* выберите значение *DataEnviroment1*, а в свойстве *DataMember* - значение *MyReportCommand.*

10. Теперь можно приступить к проектированию формы отчета. Начнем с заголовка отчета, то есть будем работать с разделом *Report Header*. В отчет будут выведены следующие поля таблицы *Authors*: код книги, автор книги и год выпуска. В заголовке отчета необходимо расположить заголовок самого отчета и заголовки столбцов отчета.

11. Чтобы создать заголовок отчета, выполните следующие действия:

- ✓ щелкните мышью кнопку RptLabel на панели элементов управления, установите курсор в первую строку раздела Report Header и нарисуйте рамку большого размера;
- ✓ при помощи окно свойств установите имя объекта rptRepTitle;
- ✓ в свойство Caption введите текст заголовка **Список литератур**;
- ✓ для свойства Alignment установите значение начертание шрифта Полужирный и размер 12.

12. Добавьте в следующую строку этого раздела еще три объектов *RptLabel* для заголовок столбцов отчета. Назовите их в соответствии с полями таблицы, добавляя префикс *rpt* и окончание *Header*: *rptAu_IDHeader, rptAuthorHeader, rptYearBornHeader.* В свойство *Caption* введите соответствующие тексты заголовков: **Код книги, Автор книги, Год выпуска.** Шрифт установите полужирный, размера 10.

13. Вставьте в первую строку раздела *Detail* объект RptTextBox. В поле установлено значение *Unbound*, означающее, что объект пока не связан с источником данных. Это будет поле для вывода кода книги. Назовите этот объект *rptAu_ID*. В свойстве *DataMember* выберите источник данных *MyReportCommand*, затем в свойстве DataField выберите поле *Au_ID* базы

данных, соответствующее этому поля отчета. Аналогично добавьте текстовые поля для остальных полей базы данных.

14. Теперь создайте на форме одну кнопку и назовите его *cbStartReport*, а в свойстве *Caption* кнопки введите текст **«Отчет»**.

15. Добавьте в событие *Click* кнопки *cbStartReport* код, указанный ниже:

```
Private Sub cbStartReport_Click( )
  DataReport1.Show
End Sub
```

16. Запустите приложение на выполнение и нажмите кнопку **«Отчет»**. Отчет будет выведен в окно просмотра рис. 34.

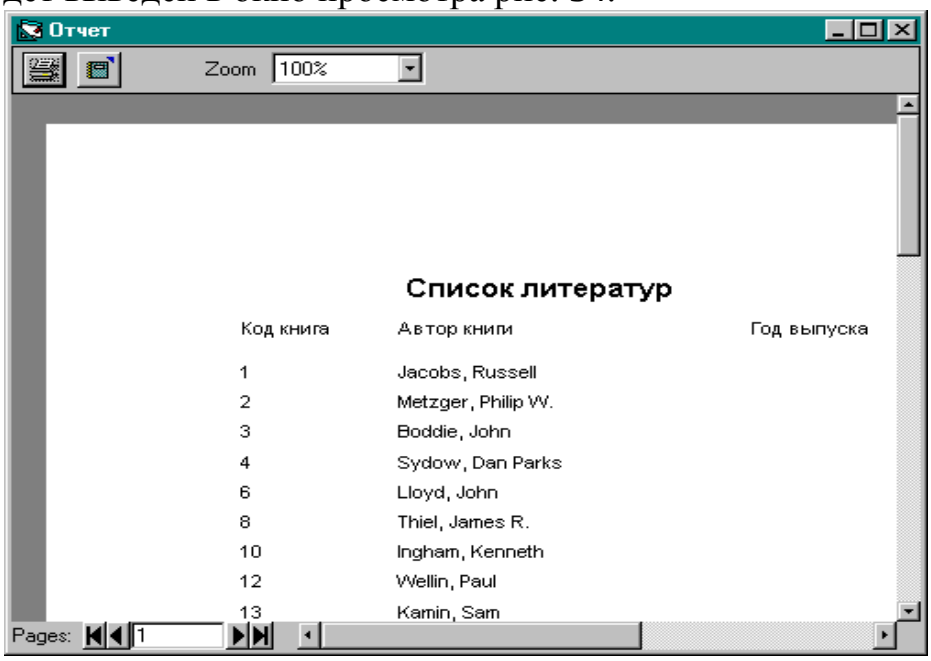


Рис. 34.

25. Работа с элементом управления Rich Textbox

Цель работы: Использование возможностей элемента управления Rich Textbox при создании текстовых полей.

Средство Rich Textbox дает возможности добавлять в приложение расширенные функции текстового процессора. В отличие от более простого средства Toolbox (Текстовый блок), позволяющего выполнять лишь ограниченный набор стандартных функций, средство Rich Textbox (Форматированный текстовый блок) дает возможность осуществлять форматирование текста в популярном стандарте RTF.

Вариант 1: Добавим в панель инструментов средство управления RichTextbox.

Прежде чем приступить к использованию средства Rich Textbox, мы должны добавить его в панель инструментов.

1. В меню Project (Проект) щелкните на команде Components (Компоненты), а затем щелкните на закладке Controls (Средства управления).

2. Измените имя формы с Form1 на «RTFEdit». В окне Properties установите для свойства Name значение «RTFEdit».

3. Осуществите прокрутки списка средств управления, пока не найдете строку Microsoft Rich Textbox Control 6.0.

4. Щелкните на поле флажка рядом с названием этого средства управления, затем щелкните на кнопке OK. Visual Basic добавить средство управления Rich Textbox в панел инструментов.

Средство управления Rich Textbox

Рис. 35.

5. Создание текстового блока с помощью средства управления Rich Textbox аналогично созданию текстового блока с помощью стандартного средства управления Textbox.

6. Для этого нужно щелкнуть на значке Rich Textbox в панели инструментов, а затем перетащить его в форму, в результате чего будет создан текстовый блок, необходимый для нашего приложения размера. Однако различие между этими двумя средствами управления станет очевидным, если мы начнем манипулировать свойствами и функциями средства Rich Textbox.

7. Теперь надо связать объект Rich Textbox. Для этого активируйте объект Rich Textbox и в свойствах, в FileName выберите файл с расширением *.rtf.

8. Теперь щелкните на кнопке Start (Пуск) на панели инструментов для запуска программы.

Программа RTFEdit появится на экране, как представлено на следующем рисунке.

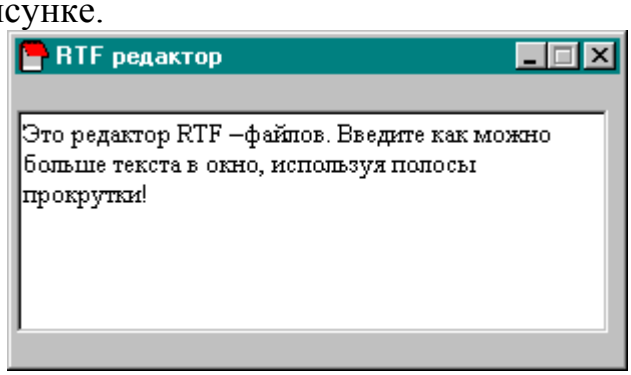


Рис. 36.

Вариант 2. Воспользуемся командами из меню File (Файл). Теперь попробуем выполнить команды RTF для работы с файлами.

1. Теперь нужно щелкнуть на кнопочке Menu Editor вверху слева. Там нарисовано меню. 

2. Щёлкните на нём. Теперь нам нужно создать меню.

3. Меню состоит из следующих подменю: Открыть (Open), Закрыть (Close), Печать (Print) и Выход (Exit) в меню Файл (File).

4. Нам потребуется диалоговое окно открытия файла с расширением RTF, поэтому установите также в окне Components флажок для библиотеки компонентов Microsoft CommonDialog Control 6.0.

5. Щелкните два раза на подменю Открыть (Open) - откроется окно Code. Впишите:

```
Private Sub mnuFileOpen_Click( )
CommonDialog1.CancelError = True
On Error GoTo Errhandler:
CommonDialog1.Flags = cdlOFNFileMustExist
CommonDialog1.ShowOpen
RichTextBox1.LoadFile CommonDialog1.FileName, rtfRTF
Errhandler:
'Если нажата кнопка Cancel - выйти из процедуры
End Sub
```

6. С помощью команды Закрыть (Close) из меню Файл (File) вы можете закрыть открытый RTF-файл путем отчистки содержимого текстового окна, используя функцию Text с аргументом в виде пустой строки (“ ”). Чтобы реализовать эту возможность, нужно создать глобальную переменную с именем UnsaveChanges с типом Boolean (Булева) в разделе объявлений Declarations.

```
Dim UnsaveChanges As Boolean
```

Переменная имеет тип Boolean (Булева), поскольку будет принимать значения True (Истина) или False (Ложь), отражающие текущее состояние (сохраненные или не сохраненные) любых изменений в документе. Переменная UnsaveChanges будет принимать значение True (Истина), если в документе имеются не сохраненные изменения, значение False (Ложь), если их нет. Программа управляет статусом изменений с помощью события Change (Изменение), относящегося к средству управления Rich Textbox.

Текст процедуры обработки события Rich Textbox1_Change выглядит следующими образом:

```
Private Sub RichTextBox1_Change()
'Устанавливать каждый раз значение True в глобальной переменой
UnsavedChanges когда текст в окне изменяется
UnsavedChanges = True
End Sub
```

7. Теперь сделайте двойной щелчок на подменю Печать (Print) и введите следующие коды:

```
Private Sub mnuFilePrint_Click( )
'печатать текущий документ использую
'установки принтера по умолчанию
RichTextBox1.SelPrint (Printer.hDC)
```

```
End Sub
```

8. Теперь подменю Выход (Exit) просто введите:

```
Private Sub mnuFileExit_Click()
Dim prompt As String
Dim reply As Integer
CommonDialog1.CancelError = True
On Error GoTo Errhandler:
If UnsavedChanges = True Then
prompt = "Сохранить Ваши изменения?"
reply = MsgBox(prompt, vbYesNo)
If reply = vbYes Then
    CommonDialog1.ShowSave
    RichTextBox1.SaveFile CommonDialog1.FileName, rtfRTF
    UnsavedChanges = False
End If
'после сохранения файла выходим из программы
Errhandler:
'если нажата кнопка cancel, то возвращаемся в программу
End If
End Sub
```

9. Запустить приложение - меню Run, Run или F5 или кнопка на панели инструментов. В итоге получается следующее окно.

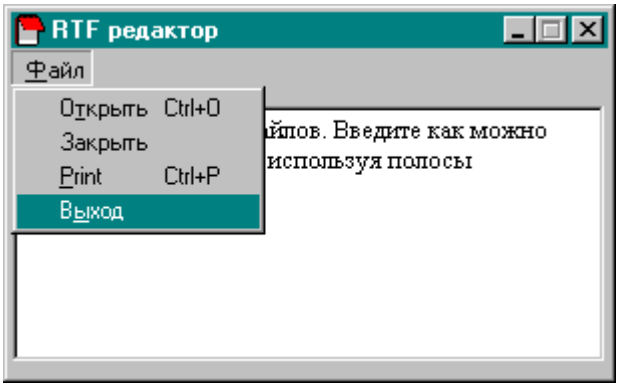


Рис. 37.

26. Автоматизация использования объектов MS Office

Цель работы: Приобретение навыков внедрения объектов других приложений в процедуры VB посредством технологии автоматизации.

При работе со средствами автоматизации надо хорошо понимать роль и назначение каждого участвующего приложения: управляющее приложение, приложение-клиент, приложение-сервер. Приложение-клиент

управляет приложением-сервером. Приложение-сервер открывает свои объекты, которые может использовать другое приложение.

Многие приложения, которые поддерживают технологию автоматизации, имеют библиотеки объектов. Библиотека объектов содержит информацию, необходимую приложению-клиенту для управления объектами приложения-сервера. Для получения доступа к этой библиотеке надо создать на нее ссылку из управляющего приложения, такого как Excel. Для того чтобы Excel управлял приложением Microsoft Word, т.е. Word был приложением-сервером, надо создать ссылку на библиотеку его объектов.

1. Для этого, находясь в редакторе Visual Basic, выполните команду Tools → References (Сервис → Ссылки). Откроется диалоговое окно References. Установите флажок Microsoft Office 10.0. Object library и щелкните на ОК. В текущий проект будет добавлена ссылка на эту библиотеку.

2. Измените заголовок окна формы с Form1 на «Использование объектов Microsoft Office». В окне Properties установите для свойства Caption значение «Использование объектов Microsoft Office».

3. Создайте пять кнопок с именами cmdWord, cmdExcel, cmdPPoint и cmdExit, и соответственно надписи к ним: "Открыть документ", "Открыть таблицы", "Просмотр презентации", “ Выход”.

4. Щелкните два раза на cmdWord (кнопка, которая будет открывать файл) - откроется окно Code. Впишите:

```
Private Sub cmdWord_Click( )
Dim doc As Object      'определяем объекты-переменные
Dim reply, prompt      'определяем переменные для окон сообщений
prompt = "Для просмотра файла нажмите клавишу пробел или кнопку
<Да>?"
reply = MsgBox(prompt, vbYesNo, "Microsoft Word")
If reply = vbYes Then
Set doc = CreateObject("Word.Application.10")
doc.Visible = True      'открываем и запускаем файл
doc.Documents.Open "c:\Мои документы\Письмо.doc"
Set doc = Nothing       'очищаем объекты - переменные
End If
End Sub
```

5. Теперь откройте окно выбора (сверху окна Code) и выберите событие cmdExcel - заполним кнопку «Открыть таблицу»:

```
Private Sub cmdExcel_Click()
Dim xls As Object      'определяем объекты-переменные
Dim reply, prompt      'определяем переменные для окон сообщений
prompt = "Для просмотра таблицу нажмите клавишу пробел или кнопку
<Да>?"
reply = MsgBox(prompt, vbYesNo, "Microsoft Excel")
If reply = vbYes Then
```

```
Set xls = CreateObject("Excel.Application.10")
xls.Visible = True        'открываем и запускаем таблицу
xls.Workbooks.Open "c:\Мои документы\Файл_1.xls"
Set xls = Nothing         'очищаем объекты - переменные
End If
End Sub
```

6. Теперь выберите событие кнопки cmdPPoint и впишите:

```
Private Sub cmdPP_Click(  )
Dim ppt As Object       'определяем объекты-переменные
Dim reply, prompt       'определяем переменные для окон сообщений
prompt = "Нажать клавишу Пробел для перехода от слайду" & _
"в презентации." & vbCrLf & "Стартовать?"
reply = MsgBox(prompt, vbYesNo, "Изумительные факты PowerPoint")
If reply = vbYes Then
Set ppt = CreateObject("PowerPoint.Application.10")
ppt.Visible = True        'открываем и запускаем презентацию
ppt.Presentations.Open "c:\Мои документы\Slide_1.ppt"
ppt.activePresentation.slideshowsettings.Run
Set ppt = Nothing         'очищаем объекты - переменные
End If
End Sub
```

7. Добавьте в событие Click кнопки cmdExit код, указанный ниже:

```
Private Sub cmdExit_Click( )
End          'Закрытие программы
End Sub
```

8. В меню Run (Выполнить) щелкните на команде Start (Старт).

В итоге ваш проект, возможно, выглядит следующим видом:

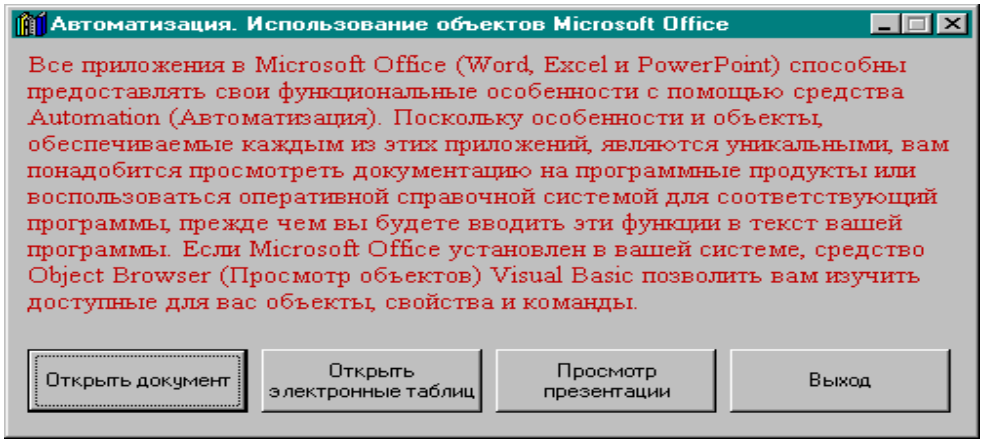


Рис. 38.

Литература

1. Visual Basic 6.0. – СПб.: БХВ - Санкт-Петербург, 1999. – 992 с.
2. Истомина Т.Л. Программирование на Visual Basic в школе. – М.: Дограф, 2000. – 96 с.
3. Ананьев А., Федоров А. Самоучитель Visual Basic 6.0. - БХВ-Петербург, 2001. – 624 с.
4. Макарова Н.В., Матвеев Л.А., Бройдо В.Л. и др. Информатика. Учебник – М.: Финансы и статистика, 1999. – 768 с.
5. Сайлер Б., Спаттс Д. Использование Visual Basic 6. Специальное издание. – М.; СПб.; К.: Издательский дом «Вильямс», 2001. – 832 с.
6. Мак-Манус, Джеффри, П. Обработка баз данных на Visual Basic 6. – М.; СПб.; К.: Издательский дом «Вильямс», 2001. – 672 с.
7. Браун С. Visual Basic 6.0. – СПб.: Питер, 2001. – 576 с.
8. Титаренко Г. Visual Basic 6.0. Практика использования. – К.: BHV, 2001. - 416 с.