МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРЫГЫЗСКОЙ РЕСПУБЛИКИ

КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. И. РАЗЗАКОВА

КАРАБАЛТИНСКИЙ ТЕХНИЧЕСКИЙ ИНСТИТУТ

ПРОГРАММИРОВАНИЕ НА «DELPHI»

Бишкек – 2011

Рекомендовано к печати УМК КТИ КГТУ им. И.Раззакова

Протокол № 8 от 3 февраля 2011 г.

Составитель: БАРАКОВА Ж. Т.

Программирование на «Delphi». Методическое пособие к лабораторным работам по дисциплине «Программирование и основы алгоритмизации». / КТИ КГТУ им. И.Раззакова. – Б.: ИЦ «Текник», 2011. – 56 с.

Пособие включает методические указания для изучения программирования в среде Delphi. Подробно описаны основные действия для выполнения лабораторных заданий, способы использования свойств и событий компонентов, приемы разработки приложений.

Преднозначено студентам специальности автоматического управления производством для самостоятельной подготовки и выполнения отчета для защиты лабораторных работ рекомендуется следующая литература:

Рецензент к.ф.-м.н., доц. Имаш кызы Мээрим

Программирование на «Delphi». Методическое пособие к лабораторным работам по дисциплине «Программирование и основы алгоритмизации»

Составитель ***Баракова Жанна Токтобековна***

Тех. редактор ***Субанбердиева Н.Е.***

Подписано к печати 03.03.2011 г. Формат бумаги 60x84 $^{1}/_{16}$.
Бумага офс. Печать офс. Объем 3,5 п.л. Тираж 75 экз. Заказ 98. Цена 59 сом.

Бишкек, ул. Сухомлинова, 20. ИЦ “Текник” КГТУ им. И.Раззакова, т.: 54-29-43
e-mail: beknur@mail.ru

ОГЛАВЛЕНИЕ

Лабораторная работа № 1

Создание и сохранение файла. Создание файла проекта

Цель работы:
Создать простое приложение Delphi с двумя взаимодействующими формами.

Решаемые в работе задачи:
- Создать каталог и подкаталог, где будет храниться проект.
- Сохранить файл проекта.
- Протестировать проект.

Задание:
1) Создать две формы, текст которых пересылается из одной формы в другую.
2) Установить связи между формами.

Порядок выполнения работы:
1. Если **Delphi** еще не запущена, запустите ее. Создайте новый проект, используя команду File→ New; выберите пиктограмму Application в диалоговом окне New Items (новые элементы) и щелкните на кнопке OK.
2. Щелкните на пиктограмме Form или File→ New Form, чтобы добавить в проект вторую форму.
3. Используя Инспектор Объектов, измените свойство Caption (заголовок) для формы 1 на “Входная форма”, для формы 2 на “Выходная форма”.
4. Свойства форм Name (имя формы) измените на “InputForm” и “OutputForm” соответственно.
5. Сохраните файлы модуля unit1.pas и unit2.pas следующим образом: File→Save As **C:\program files\borland\delphi7.0\mydir\forminfo\input** и **C:\program files\borland\delphi7.0\mydir\forminfo\output.**

Сохранение файла проекта

1. File →Save Project AS.
2. В диалоговом окне Save Project AS выберите каталог и имя проекта для его сохранения, изменив его с **project1.dpr на**
C:\program files\borland\delphi7.0\mydir\forminfo\ forminfo.dpr

Пример построения формы.
1. Добавьте на каждую форму по окну редактирования Edit.
2. На входную форму установите две кнопки Button1 и Button2.
3. Очистите свойство **text** в обоих окнах Edit.
4. Присвойте имя InText кнопке Edit на входной форме.
5. Дайте имя OutText кнопке на выходной форме.
6. Присвойте заголовку (Caption) кнопки Button входной формы значение “Переслать”, а ее свойству **name-** значение “SendText”.

6. Чтобы гарантировать визуальное отображение форм на экране, измените значение Visible (видимая) на **true**.
7. Для кнопки Button2 присвойте заголовку (Caption) “Показать выходную форму” а ее свойство Name- значение “formshow”.
8. Добавьте строку кода для кнопки Button1 (“Переслать”). Дважды щелкните мышью на кнопке SendText и добавьте следующее сообщение: outputForm.outtext.text:=inputform.intext.text;
9. Добавьте кода для кнопки (Button2) formshow следующее сообщение: outputform.show;
10. Нажмите клавишу [F9] или щелкните на пиктограмме Run. Появится сообщение о добавлении списка (рис.1.1)

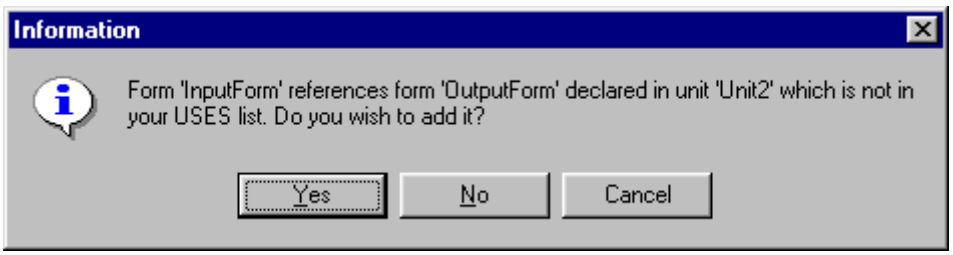


Рис. 1.1

11. Щелкните на кнопку [Yes].
12. File→ Save All.
13. Запустите программу снова на выполнение (Run или F9). Появится ваша форма (рис.1.2).

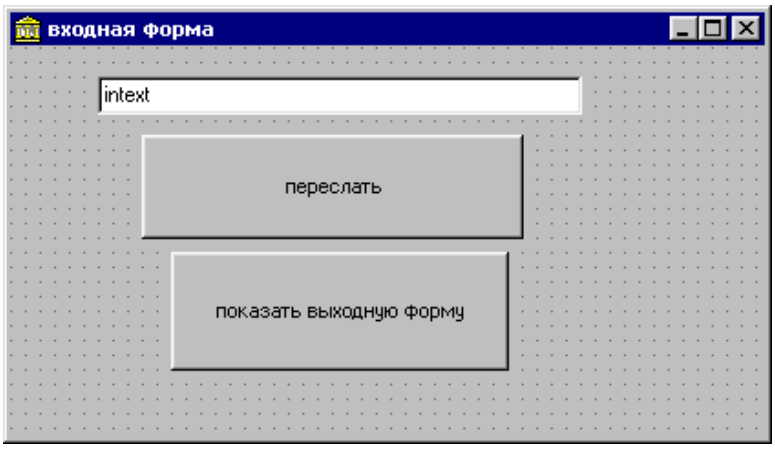


Рис.1.2. Входная форма

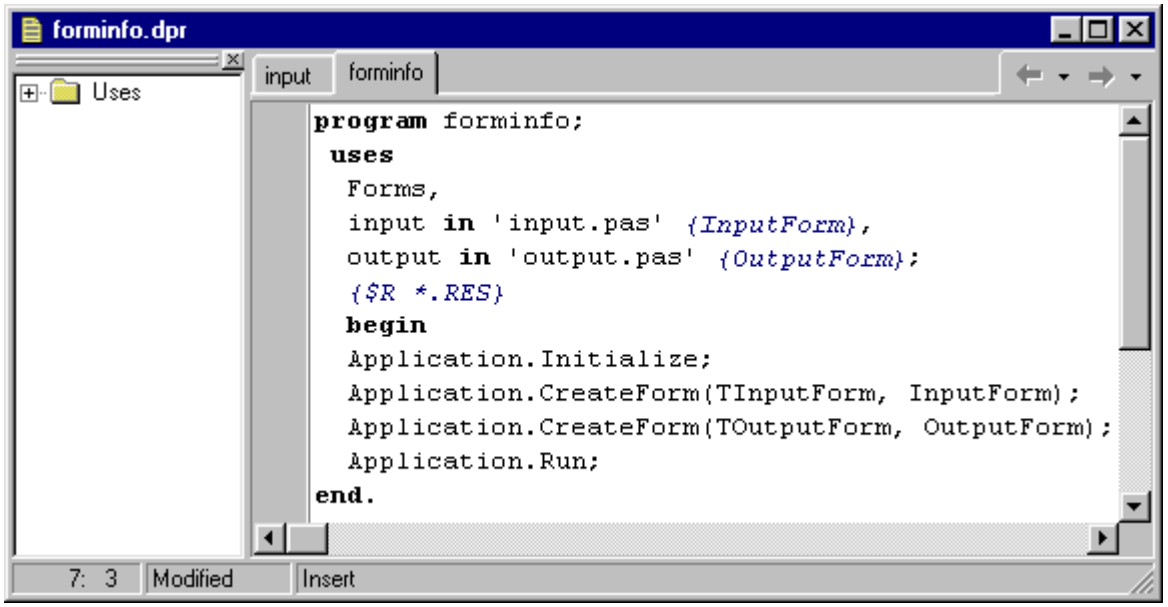


Рис.1.3. Код проекта Forminfo

Модуль формы Input

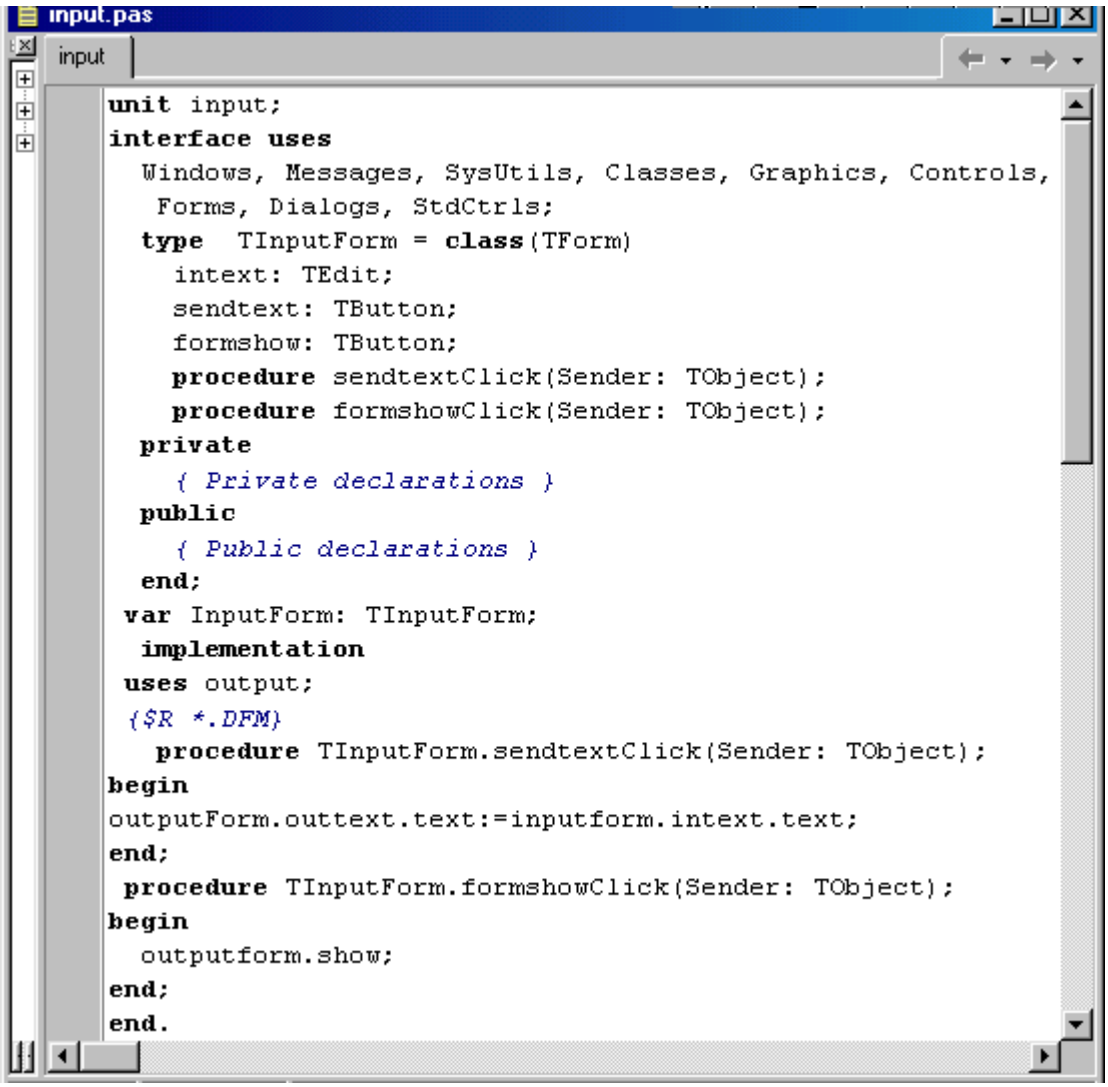


Рис. 1.3. Код модуля формы Input

14. Протестируйте программу, вводя текст во входной форме и нажимая кнопку “Переслать”. Текст, который вы ввели, должен появиться в текстовом окне «выходной формы».

Для просмотра содержимого файла проекта вызовите: Project → View Source (рис.1.3).

Для просмотра кода модуля входной формы, дважды щелкните мышью на входной форме, либо выберите View →Unit, либо View →Unit Dialog.

Лабораторная работа № 2
Использование редактора

Цель работы:
Создать приложение для представления о доступных возможностях Delphi.

Решаемые в работе задачи:
- Создать приложение, в котором вводят два числа, при нажатии на кнопку, программа вычисляет их сумму и выводит результат.

Задание:
1) Создать форму для вычисления суммы двух чисел;
2) При нажатии на кнопку «Сложение» вывести результат.

Порядок выполнения работы:

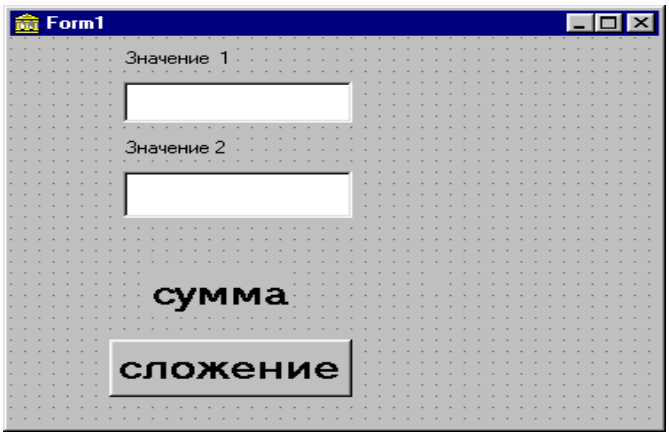


Рис. 2.1. Форма приложения Addition

1. На инструментальной панели Delphi выберите страницу с закладкой Standard.
2. В выбранной странице Standard щелкните на пиктограмме кнопку Button, а затем щелкните на форме. В результате этих действий на форме будет добавлена кнопка, размеры и положения которой вы можете изменить.
3. Измените значение свойства Caption на значение “сложение”, а name- “add “.

4. Используя ту же технику, разместите на форме две метки (Label) и два окна редактирования (Edit) по вертикали.
5. Используя Инспектор Объектов, измените свойства Caption Label1 на Значение 1, Label2 на Значение 2.
6. Измените свойства “Caption “ для формы на “Addition”.
7. Измените свойства text для Edit1, Edit2 на пустую строку.
8. Установите еще две кнопки Label3 и Label4.
9. Свойству Caption для Label3 на “Сумма” и Label4 на пустую строку.
10. При двойном щелчке Button1 (Сложение) появляется редактор кода.

Добавьте следующий текст:

```
procedure TForm1.Button1Click(Sender: TObject);
var a,b:integer;
begin
  a:=strtoint(edit1.text);
  b:=strtoint(edit2.text);
  label4.caption:=inttostr(a+b);
end;
end.
```

11. Чтобы сохранить файл (программный код):
FILE ⟶ SAVE AS, открыть каталог.
12. **C:\program files\delphi 7.0\mydir\add.pas,** и нажать кнопку [Сохранить].
13. Чтобы сохранить проект, в том же каталоге mydir\addition.dpr. и нажать кнопку [Сохранить].
14. Нажав клавишу [F9] или на пиктограмме Run, выполните программу.
15. Когда ваша форма всплывет на экране, и введите числа, затем, нажав на кнопке «сложение», вы увидите результат суммирования.
16. Закройте ваш проект, FILE → CLOSE ALL.

Лабораторная работа № 3.
Эксперт кода. Отладка. Использование отладчика

Цель работы:
- Работа с ошибками.

Решаемые в работе задачи:
- При намеренно допущенных ошибках пробовать компилировать программу.

Задание:
1)При намеренно допущенных ошибках в лабораторной работе №2 использовать окно отладчика содержащего две таблицы.
2)Используя окна сообщений компилятора отлаживать программу.

Порядок выполнения работы:
Эксперт кода един в трех лицах:
- эксперт шаблонов кода;

- эксперт завершения кода;
- эксперт программных параметров;

1. Первым в списке идет эксперт шаблонов кода. Предположим, вы набираете код какой-то программы и в следующей строке должны использовать оператор if. Вы можете набрать только if и затем нажать комбинацию клавиш [Ctrl+J], после чего выбрать из всплывающего меню нужную вам разновидность оператора if , дважды щелкнув на нем или нажав [Enter].

   Также можете редактировать, добавлять или удалять шаблоны выполнив:

   Tools → Code Templates (Инструменты→ Шаблоны кода)

2. Эксперт завершения кода заканчивает код за вас, когда вы вводите имя класса и после него ставите точку (.). Этот эксперт может быть полезен при написании сложных компонентов программ и частом использовании одних и тех же объектов, например, в классе TApplication.
3. Эксперт программных параметров отображает параметры функций, процедур и методов в виде подсказки. Чтобы активизировать эксперт программных параметров, вам нужно ввести имя процедуры, функции или метода, затем открывающуюся круглую скобку; после этого на экране появиться подсказка со списком параметров.
4. Например, Fileopen(…

Использование точек прерывания

Воспользуемся проектом Addition и внесем в него несколько изменений, одно из которых приведет к ошибке в прикладной программе намеренно). Добавим следующую процедуру в раздел implementation:

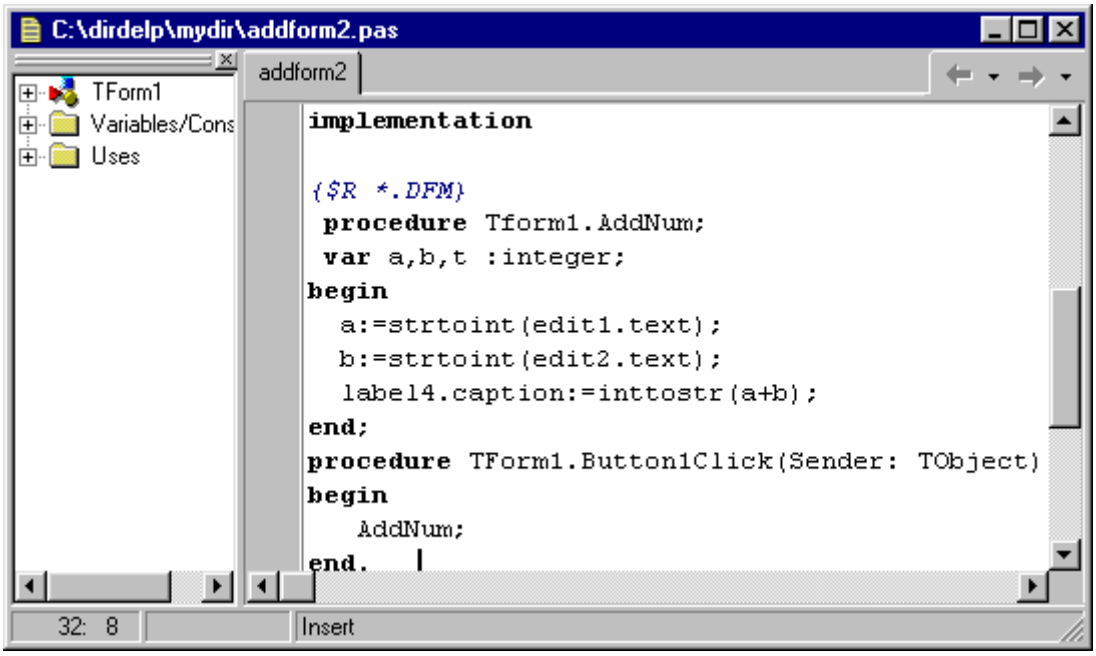


Рис. 3.1. Фрагмент кода приложения Addition

Выполнив программу, обратите внимание на полученный результат. Выдаваемый результат должен быть неправильным. Чтобы найти причину неправильного результата, воспользуемся отладчиком.

Установите точку прерывания. Для этого, щелкните на сером поле окна редактирования слева от текста программы напротив той строки, на которой вы хотите установить точку прерывания. В данном случае это будет строка:

a:= StrtoInt (Edit1.Text).

Строка должна выделиться красным цветом и большой красной точкой.

Выполнив программу еще раз, введите новые значения. Вслед за этим появится окно редактора кода, где будет отмечено местоположение точки прерывания. Появившаяся зеленая стрелка указывает на следующую, подлежащую выполнению строку.

Теперь, когда вы нашли некоторые ошибки, применив для этой цели отладчик, вы можете внести изменение в код программы, чтобы удалить ошибки и получить правильно работающую программу.

Лабораторная работа № 4
Отладка DLL (динамически присоединяемые библиотеки)

Цель работы:

- Используя встроенный отладчик Delphi, научиться быстро определять и исправлять ошибки в программе.

Решаемые в работе задачи:

- Создается новое приложение Delphi;
- Создается программа, вызывающая функцию из DLL.

Задание:

1) Создать приложение в каталоге, где находится модуль DLL.
2) Используя DLL, найти и исправить ошибки.

Порядок выполнения работы:

1. File →New
2. Дважды щелкните на пиктограмме DLL.
3. Создайте приведенный ниже модуль DLL (рис.4.1).

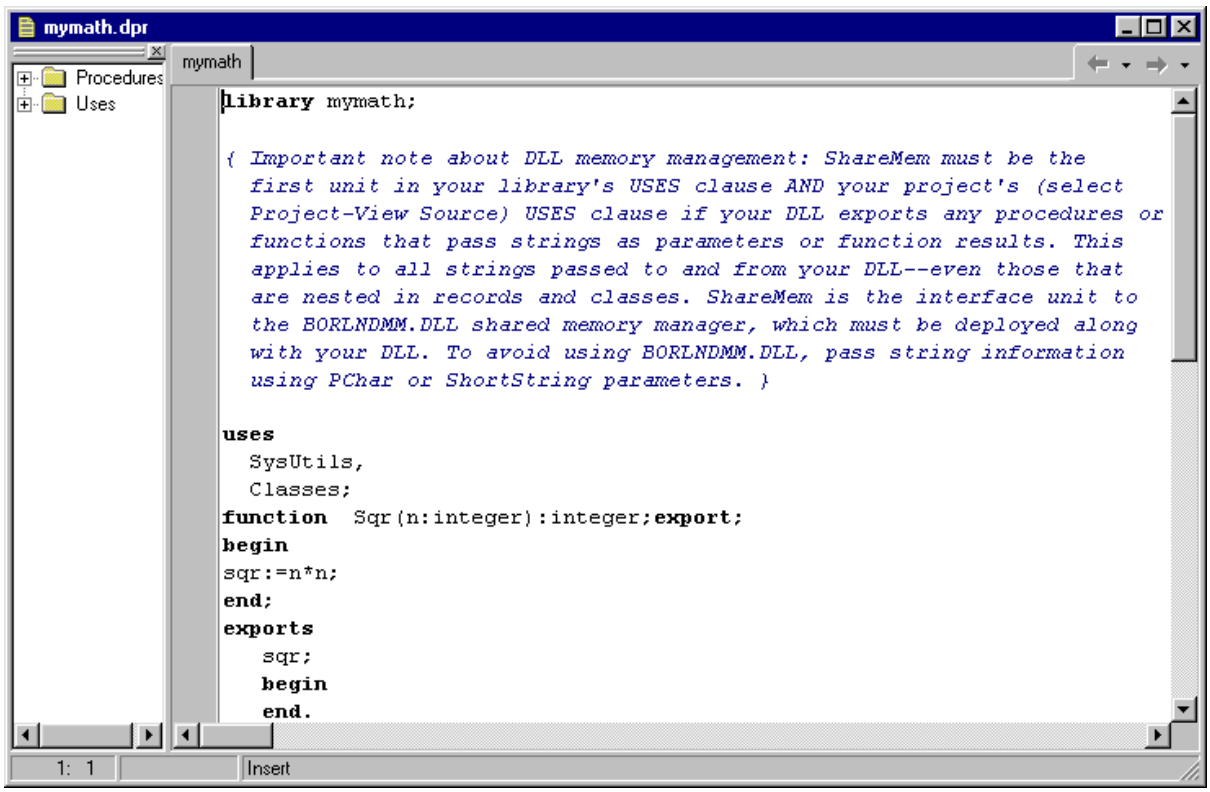


Рис.4.1. Математический модуль DLL для примера отладки

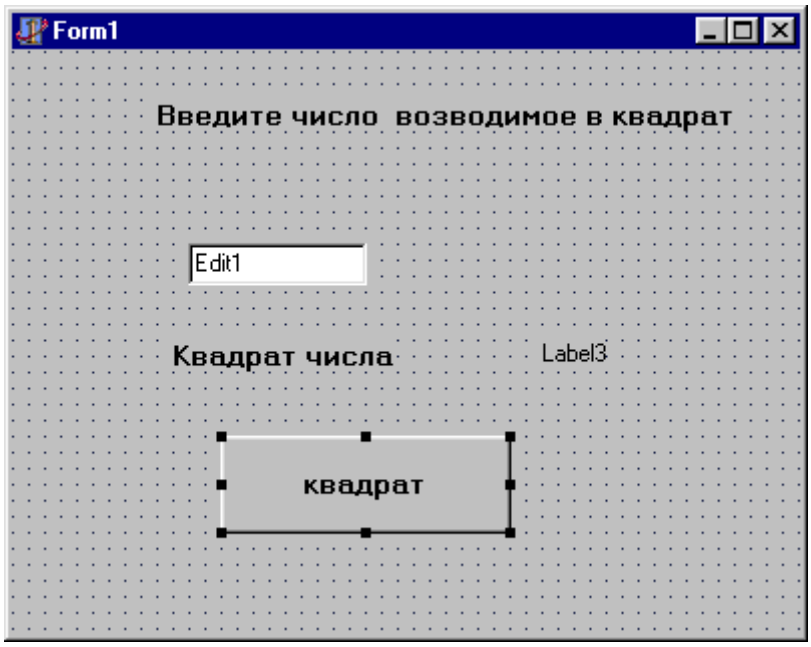


Рис. 4.2. Приложение TestSqr для тестирования DLL

4. Сохраните файл с именем Mymath.dpr.
5. Project→Build All (Проект→Собрать все).

Теперь создайте главную программу, чтобы проверить модуль DLL и отладить его.

6. File → New Application.

7. Создайте форму и код по следующему образцу (рис.4.2 и рис.4.3):

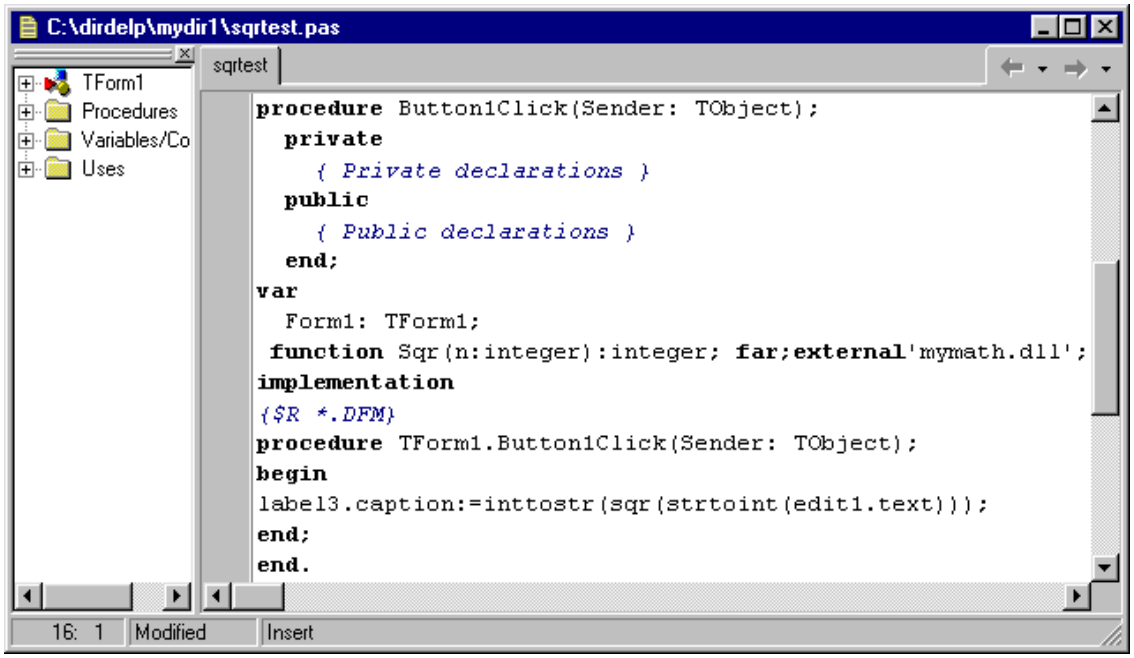


Рис.4.3. Приложение TestSqr для примера отладки DLL

8. Сохраните эти объекты с именами Sqrtest.pas и Testsqr.dpr в своем каталоге, где находится модуль DLL.

9. PROJECT→Build All.
Эта программа вызывает функцию sqr из DLL. Число, введенное в Edit. Text передается модулю DLL при вызове функции sqr, значение возвращаемое DLL помещается в Label3.Caption.
10. Компилируйте эти два модуля.

11. После появления на экране окна, введите число и нажмите на кнопку “квадрат”, будет выведено квадрат данного числа.

Примечание.
1. Откройте исходный текст DLL (mymath.dpr)
2. Run→Parameters.
3. Введите имя и путь к главной программе, в нашем случае **C:\mydir\test sqr.exe.**
12. Нажмите F9 (или пиктограмму Run).
Delphi будет запускать главную программу, чтобы вызвать DLL, и при этом будет останавливать выполнение в любых указанных точках прерывания. С этой точки зрения вы можете продолжать процесс отладки, как если бы ваш модуль DLL был обычным приложением Delphi.

Лабораторная работа № 5.
Работа с VCL (Библиотека визуальных компонентов).
Работа с компонентами страницы Standard

Цель работы:

- Получить представление о свойствах и обработке событий компонентов страницы Standard.

Решаемые в работе задачи:

- Добавляя компоненты страницы Standard в форму, познакомиться со свойствами и обработкой событий.

Задание:

1. Добавляя компоненты страницы Standard в форму, измените ее свойства.
2. Освоить использование компонентов.
3. Узнать о предоставляемых возможностях.

Порядок выполнения работы:

1. Откройте новое приложение.
2. Измените свойство Name на StandardTab.
3. Добавьте компонент Panel вверхнюю часть экрана и растяните эту панель как газетную шапку, для использования заголовка формы.
4. Для Panel (контейнер) свойство Caption измените на “Демонстрация страницы Standard”.
5. Добавьте в форму метку Label, и задайте ее свойство Caption- “Имя”.
6. Для Label -“Имя” установите свойство- AutoSize- true.
7. Под эту метку добавьте компонент Edit.
8. С помощью Инспектора Объектов удалите все символы из свойства Text этого компонента.

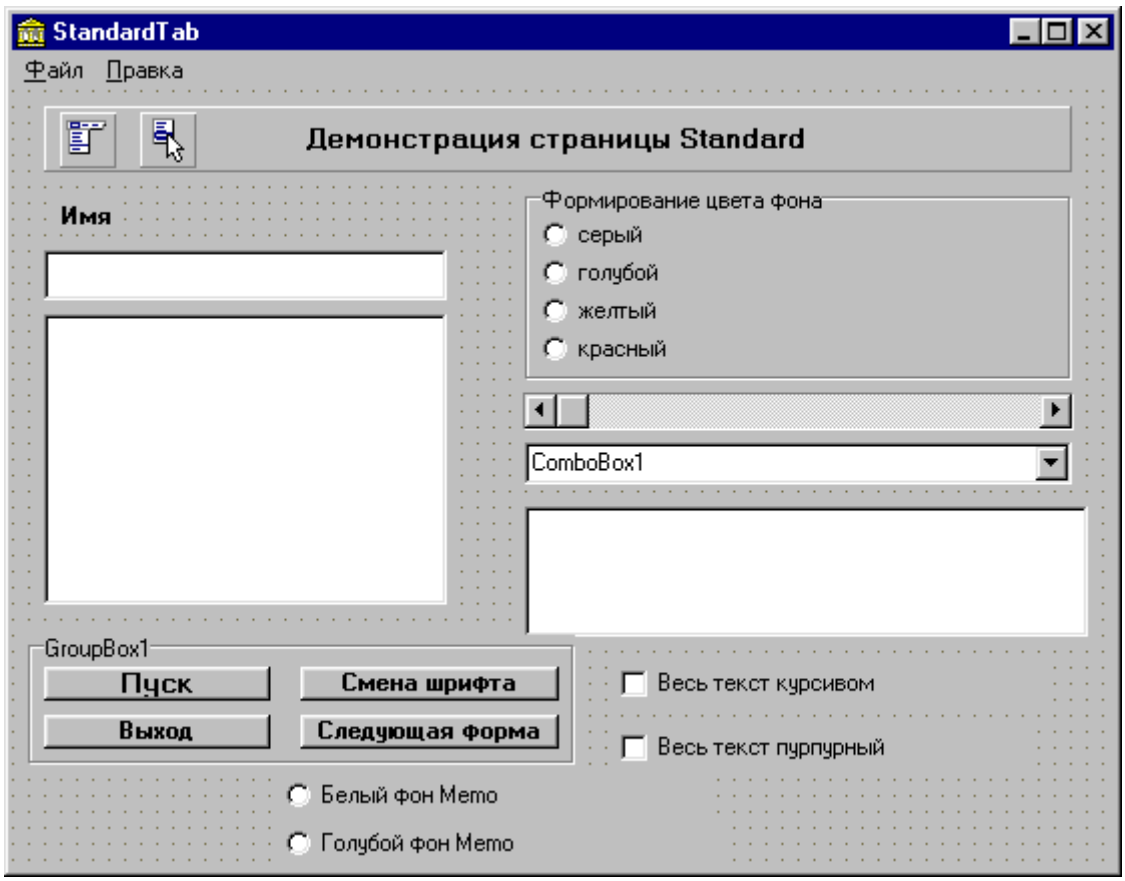


Рис.5.1. “Страница Standard”

9. Добавьте в форму MainMenu и PopupMenu.
10. Добавьте компонент Memo.

(Компонент Memo - достаточно мощное средство, позволяющее построить собственный текстовый редактор).

11. В окне Инспектора Объектов двойным щелчком откройте свойство Lines (строки) и удалите текст ”Memo”.
12. Четыре раза нажмите [Enter]. Этим вы отводите место в текстовом буфере компонента Memo, (в процессе работы нашей демонстрации понадобиться модифицировать 3-4 строки, и сейчас мы их создали). Если этого не сделать, демонстрация работать не будет.
13. Добавьте в форму компоненты RadioGroup, Scrollbar, Combobox и ListBox.
14. Дважды щелкнув на помещенном компоненте RadioGroup добавьте следующий код:

```
procedure TStandardTab.RadioGroup1Click(Sender: TObject);
begin
if RadioGroup1.ItemIndex=0 then StandardTab.Color:=clSilver;
if RadioGroup1.ItemIndex=1 then StandardTab.Color:=clBlue;
if RadioGroup1.ItemIndex=2 then StandardTab.Color:=clYellow;
if RadioGroup1.ItemIndex=3 then StandardTab.Color:=clRed;
end;
```

15. Дважды щелкните на компоненте ScrollBar и добавьте код в событие Onchange:

```
procedure TStandardTab.ScrollBar1Change(Sender: TObject);
begin
RadioGroup1.ItemIndex:=ScrollBar1.Position;
end;
```

16. Поместите в форму компонент **GroupBox** и убедитесь, что на этой панели достаточно места для нескольких компонентов.
17. Добавьте в группу четыре кнопки **Button**, две **RadioButton** и два **CheckBox .**
18. Для каждой из четырех кнопок измените свойство **Caption**:
    Button 1 - ”Пуск”
    Button 2 - ”Смена шрифта”
    Button 3 - ”Выход”
    Button 4 - ”Следующая форма“.
19. Добавьте коды для компонента Button1:

```
procedure TStandardTab.Button1Click(Sender: TObject);
begin
 Memo1.Clear;
 Memo1.Lines.Add(Edit1.Text);
 Memo1.Lines.Add(ComboBox1.Text);
 Memo1.Lines.Add('Строка ListBox #' +IntToStr(ListBox1.ItemIndex+1));
 if RadioButton1.Checked then Memo1.Color:=clWhite;
 if RadioButton2.Checked then Memo1.Color:=clAqua;

end;
```

20. Добавьте код для события OnClick компонента Button2.

```
procedure TStandardTab.Button2Click(Sender: TObject);
begin
if CheckBox1.State=cbChecked then  StandardTab.Font.Style:=[fsItalic]
else   StandardTab.Font.Style:=[ ];
if CheckBox2.State=cbChecked then  StandardTab.Font.Color:=clPurple
else StandardTab.Font.Color:=clBlack;
end;
```

21. Добавьте код для события OnClick компонента Button3:

```
procedure TStandardTab.Button3Click(Sender: TObject);
begin
    Close;
end;
```

22. Добавьте код в компоненты Main Menu и PopupMenu.
23. Дважды щелкнув на значении свойства Items, запустите Конструктор Меню. (Можно щелкнуть на компоненте MainMenu на форме и воспользоваться всплывающим меню, щелкнув правой кнопкой мыши (рис. 5.2)).

Рис. 5.2. Конструктор Меню с выпадающим меню

24. Задайте свойство Caption равным заголовку добавляемого пункта меню. Запишите его как “&Файл”. & символ сообщает Delphi, что следующую за ним букву надо показывать подчеркнутой и включить ее в комбинацию быстрой клавиши, т.е. при нажатии Alt+F вы перейдете к пункту меню “Файл”. После добавления в меню пункта “Файл”, вы увидите, что справа от него появляется окошко. Оно позволяет вам после щелчка на нем добавить следующий пункт меню, аналогичным способом, установив, свойство Caption в Инспекторе Объектов.

25. Для добавления пунктов в выпадающее меню можно щелкнуть на имеющемся пункте верхнего меню.

а) щелкните на пункте “Файл” и под ним откроется окно.

б) щелкните на нем: окно подсветиться и Инспектор Обьектов приготовится к вводу заголовка.

в) добавьте пункт “Выход” в раздел “Файл”.

г) измените свойство Caption в Инспекторе Объектов на “Выход”, и вы увидите “Выход” добавленным под пунктом “Файл” главного меню.

Обратите внимание на пустые графы ниже пункта “Выход” и вправо от пункта “Файл” в окне меню. На них щелкают, когда требуется добавить новые пункты. Если этого не делать, они в вашем меню не появятся.

26. Добавьте в раздел меню “Справка” с подразделами выпадающего меню “Справка” и “О программе”, разделительной линией между ними.

27. Установите соответствующие значения Caption - такие же.

Заполните свойств Caption и Name -присваивая имя пунктам-File1, Exit1, Help1, Help , About1.

28. Закройте окно конструктора меню.

29. Просмотрите пункты вашего меню в форме, проводя под ними мышь с нажатой кнопкой.

30. В вашем меню Файл→ Выход можете добавить код, закрывающий приложение. Воспользуемся кнопкой Button3.

а) в Инспекторе Объектов выберите в выпадающем списке “Выход”.

б) перейдите на Events (Инспектор Объектов) и щелкните в OnClick.

в) выберите Button3.Click.

31. Щелкните на компоненте Popup Меню, помещенном в форму ранее.

32. Щелкните правой кнопкой линии для вызова, всплывающего меню.

33. Выберите в нем Menu Designer (Конструктор Меню).

а) добавьте два раздела “Пуск” и “Шрифт”.

б) в Инспектора Объектов name- Go, UpdateFonts1 соответственно.
в) закройте Конструктор Меню.

34. В Инспектора Объектов выберите Go, затем выберите Button1.Click в событии OnClick.
35. Выполните проект, нажав на пиктограмме Run или клавиши F9.
36. Протестируйте Ваше приложение.

Лабораторная работа № 6
VCL Additional (Дополнительная)

Цель работы:

Создать проект для ознакомления с компонентами **VCL (Visual Component Library) Additional.**

Решаемые в работе задачи:

- Создается проект с компонентами траницы **Additional.**
- Подробное знакомство со свойствами и обработкой событий компонентами системы**.**

Задание:

1) Создать проект с компонентами **VCL Additional.**
2) Установить связь между компонентами.

Порядок выполнения работы:

1. Измените имя формы (Name) на AdditionalTab.
2. Свойство Caption- “Страница Additional”.
3. File→Save, сохраните новую форму под именем addition.pas.

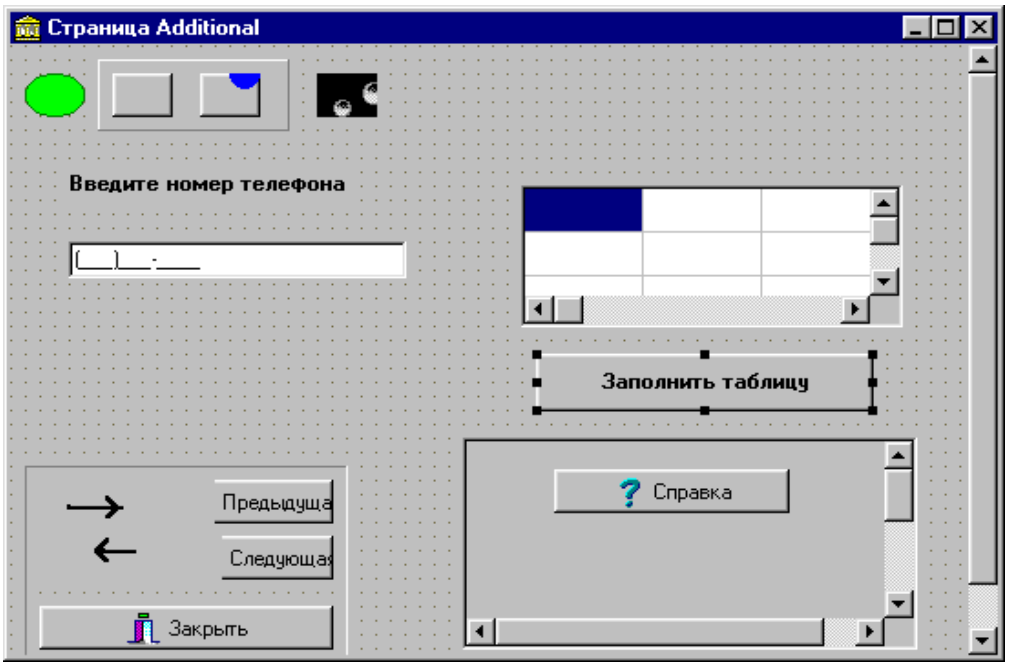


Рис. 6.1. Форма Additional для приложения VCL

4. Чтобы форма появлялась в середине экрана, задайте свойство Position. Щелкните poDesigned, и откроется окно списка. Выберите в нем poScreenCenter.
5. View→Forms, выберите в нем StandardTab.
6. Дважды щелкнув на кнопке Button4 (следующая) введите для ее события Click следующий код:

```
procedure TStandardTab.Button4Click(Sender:
begin
StandardTab.Hide;
AdditionalTab.Show;
end;
```

7. Три кнопки BitBtn поместите в нижний левый угол формы.
8. Для BitBtn2 установите в свойстве Kind значение bkCustom.
9. Установите для свойства Caption BitBtn1-“Предыдущая”, а для BitBtn2-“Следующая”.
10. Для BitBtn1 дважды щелкните на Tbitmap в свойстве Glyph. При этом запускается окно редактора картинок.
11. Щелкните в окне редактора на кнопке Load.
12. Перейдите bmp – файл в каталоге
C:\programfiles\borland\delphi7.\images\buttons и выберите arrow1L.bmp.
13. Свойство Enabled поставьте в состояние True.
14. Повторите те же шаги BitBtn2 и установите для него гравировку.
15. Для BitBtn3 щелкните на свойстве Kind и выберите вкClose.
16. Измените свойство Caption на “Закрыть”.

17. Для придания кнопкам BitBtn1, BitBtn2 и BitBtn3 аккуратного вида добавьте компонент Bevel и прорисуйте рамку вокруг кнопок. Ваши кнопки окажутся расположенными на блоке в виде утопленной панели.
18.Дважды щелкнув на BitBtn1, в событие Click добавьте:

```
procedure TAdditionalTab.BitBtn1Click(Sende
begin
StandardTab.Show;
AdditionalTab.Hide;
end;
```

19. К коду для BitBtn2 обратимся позднее, пока добавьте в событие Click для BitBtn3 код:

```
procedure TAdditionalTab.BitBtn3Click(Sende
begin
StandardTab.Close;
end;
```

20. Для правильной работы кнопки BitBtn3 в событие Close формы добавьте следующий код:

```
procedure TAdditionalTab.FormClose(Sender:
  var Action: TCloseAction);
begin
Application.Terminate;
end;
```

21. (Если этого не сделать, при нажатии кнопки BitBtn3 форма будет закрываться, но приложение останется в памяти, впустую расходуя ресурсы).
22. На левом верхнем углу формата разместите компонент Panel (из страницы Standard) достаточного размера для установления двух квадратных кнопок.
23. На компоненте Panel установите две кнопки Speed Button.
24. Speed Button может находиться в состояниях Up (вернее, отжатое), Disabled (отключенное), Down (нижнее, нажатое) и Stay Down (прижатое). Для каждого состояние отображается соответствующий фрагмент гравировки.

а) Tools→Configure Tools.

б) нарисуйте что-нибудь.

в) File→ Save as, установите для файла тип BMP и сохраните файл именем Grnred. Bmp

25. Для компонента SpeedButton1 дважды щелкните на Bitmap в свойстве Glyph.
26. При запуске окна редактора картинок выберите имя только что созданного файла GRNRED.BMP и нажмите на кнопку [OK].
27. Установите в свойстве Glyph компонента SpeedButton2 файл:
C:\program files\borland\delphi7.0\images\buttons\globe. bmp;
28. Убедитесь, что свойство NumGlyps для SpeedButton1 установлено равным 4 (4 по умолчанию). Этим компонент извещают, что для упомянутых ранее состояний имеется 4 графировки.

29. Установите 1 для свойства GroupIndex. Тем самым кнопке сообщается, что она принадлежит к группе номер 1. В любой момент времени только одна кнопка в группе может быть в состоянии Down (нижнее).
30. Для компонента SpeedButton2 установить оба свойства –Num Glyphs и GroupIndex равными 2.
31. Свойство Visual (видимый) для SpeedButton2 установите равным False.
32. Дважды щелкните на SpeedButton1 и введите код для ее события Click:

```
procedure TAdditionalTab.SpeedButton1Click(Sender: TObject);
begin
if speedButton1.down=true then
   begin
   image1.visible:=false;
   Shape1.Brush.Color:=clRed;
    end;
end;
```

33. Код события для SpeedButton2 должен выглядеть так:

```
procedure TAdditionalTab.SpeedButton2Click(Sender: TObject);
begin
  if speedButton2.down=true then
   begin
   image1.visible:=true;
   Shape1.Brush.Color:=clLime;
    end;  end;
```

34. Добавьте в левую часть линейки инструментов компонент Shape (фигура).Он должен быть приблизительно того же размера, что быстрые кнопки линейки .
35. Установите в ее свойстве Shape значение stEllipse.
36. Дважды щелкните на свойстве Pen (перо), чтобы увидеть его вложенные свойства.
    а) установите свойство Color равным clGreen.
    б) дважды щелкните на свойстве Brush (кисть) и в его вложенных свойствах установите свойство color равным clLime.

Свойства Pen определяют атрибуты контура фигуры (напр., цвет линии и т.д). Свойства Brush используются для установки атрибутов ее заполнения (закраски).
37. Добавьте компонент Image справа от линейки инструментов, того же размера, что и кнопки.
38. Дважды щелкнув на свойстве Picture, вы попадаете в редактор изображений. Загрузите файл пиктограммы **\delphi7 \images\icons\earth.ico.** Код, который вы введете для быстрых кнопок, используется для воздействия на внешний вид компонентов Shape и Image.
39. Для компонента SpeedButton1 используйте еще одно свойство, имеющееся у всех визуальных компонентов: свойство Hint (подсказка).
40. В свойстве Hint хранится строка текста, которая высвечивается рядом с курсором мыши, когда пользователь задерживает его на некоторое время над

соответствующим компонентом. Измените свойство Hint на ”Подсказка к быстрой кнопке”. Установите свойство Showtlinbs в True.

41. Добавьте в форму компонент StaticText, его свойство Caption измените на “Введите номер телефона”.

42. Затем прямо под этой меткой добавьте компонент MaskEdit .

а) дважды щелкните на свойстве EditMask и запустите редактор маски ввода;

б) щелкните на Phone;

в) нажмите на [OK];

43. Таким образом, компонента MaskEdit будет настроена только на прием телефонных номеров в форме: номер телефона.

44. Добавьте в форму компонент StringGrid (сетку строк).

а) для StringGrid задайте свойствам RowCount и ColCount значения 3;

б) установите в 0 свойства FixedCols FixedRows;

в) подберите размер сетки так, чтобы видны были ровно девять ячеек.

45. Добавьте в форму Button и измените свойство Caption на «Заполнить таблицу».

46. Дважды щелкнув на этой кнопке, добавьте следующий код:

```
procedure TAdditionalTab.Button1Click(Sender: TObject);
var x,y:integer;
begin
with StringGrid1  do
for  x:=0 to ColCount-1 do
for y:=0 to RowCount-1 do
Cells[x,y]:='Коорд.' +IntToStr(x)+'-'+ IntToStr(y);
end;
```

47. Нарисуйте ScrollBox в правом нижнем углу формы.

а) для демонстрации способности этого компонента показывать фрагменты, которые не помещаются в зоне, где их надо использовать, добавим в него несколько компонентов. Поместите на ScrollBox кнопку BitBtn.

б) установите ее свойство Kind равным bkHelp и измените Caption на «Справка».

в) добавьте на ScrollBox панель. Обратите внимание, что если панель опустить ниже нижнего края ScrollBox, появляется линейка прокрутки;

г) временно растяните ScrollBox в высоту в два-три раза больше, чем требуется на самом деле;

д) поместите кнопку в верхнюю, а панель- в нижнюю часть растянутого компонента;

е) установите свойство Caption панели «Панель в ScrollBox»;

ж) теперь восстановите высоту Scrollbox до соответствующего размера;

з) дважды щелкните по кнопке BitBtn4-«Справка» и добавьте код в ее событие Click:

```
procedure TAdditionalTab.BitBtn4Click(Sender: TObject);
begin
ShowMessage('Проверки кнопки "Справка" в ScrollBox!');
end;
```

48. Дважды щелкните на BitBtn2 и добавьте следующий код:

```
procedure TAdditionalTab.BitBtn2Click(Sender: TObject);
begin
ShowMessage('Эта возможность отсутствует');
end;
```

49. Сохраните проект.
50. Протестируйте.

Лабораторная работа №7
Демонстрация компонент смешанных страниц

Цель работы:

- Построить приложение, которое демонстрирует некоторые компоненты из разных страниц.

Решаемые в работе задачи:

- Используя компоненты страниц Win32, Dialog, System и Standard построить проект.

Задание:

1) Создать проект MOREVCL.DPR
2) Строить форму приложения, а также ее модуль – MORE.PAS.

Порядок выполнения работы:

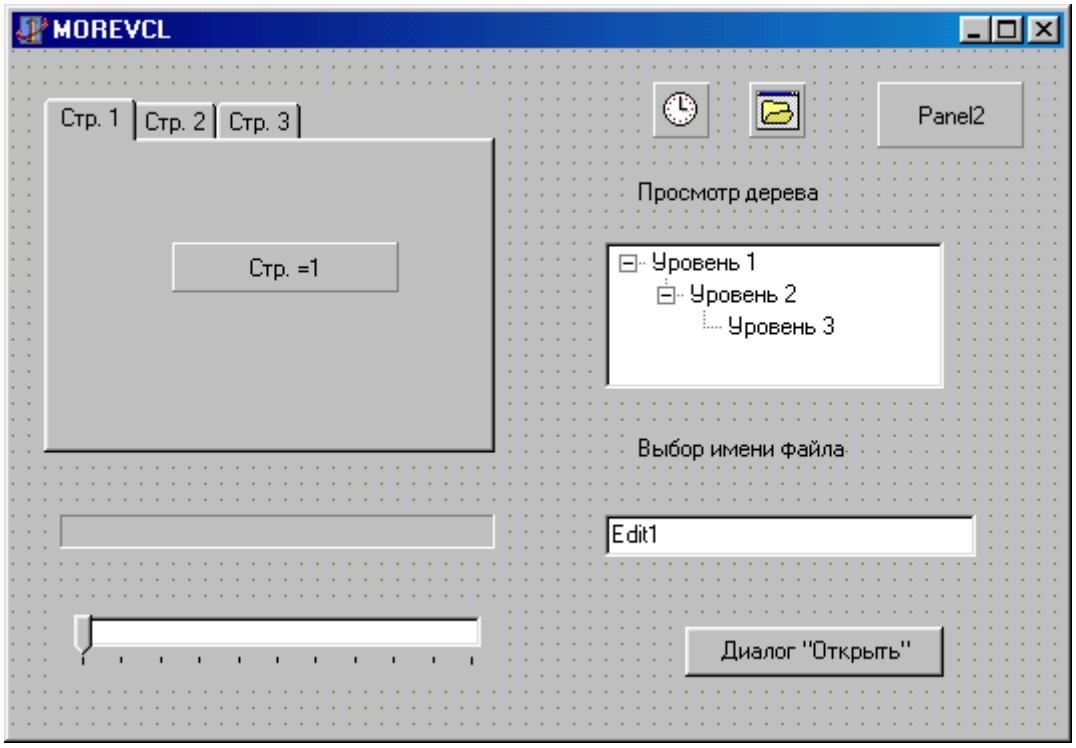


Рис. 7.1. Демонстрационная форма MOREVCL

1. На форме с левой стороны разместите;

а) компонент TabControl (из страницы Win32);

в) на нем поместите панель Panel;

с) для TabControl дважды щелкните на свойстве Tabs и в нем введите названия закладок. Каждое название начинается с новой строки. Введите «Стр.1», «Стр.2» и «Стр.3»;

d) для Panel1 свойство Caption измените на «Стр.1»;

e) Для Panel1 добавьте код в событие Change;

```
procedure TForm1.TabControl1Change(Sender: TObject);
begin
if TabControl1.TabIndex=0 then Panel1.Caption:='Стр.=1';
if TabControl1.TabIndex=1 then Panel1.Caption:='Стр.=2';
if TabControl1.TabIndex=2 then Panel1.Caption:='Стр.=3';
end;
```

2. Поместите на форму компоненты ProgressBar и TrackBar (из страницы Win32).
3. Установите свойства Max для ProgressBar и TrackBar равными 10.
4. Введите следующий код в событие OnChange компонента TrackBar;

```
procedure TForm1.TrackBar1Change(Sender: TObject);
begin
 ProgressBar1.Position:=TrackBar1.Position;
end;
```

5. Поместите на форму Timer (из страницы System) и Open Dialog (из страницы Dialogs, размещая их где угодно, поскольку эти компоненты не визуальные).
6. В правом верхнем углу нарисуйте Panel.
7. Ниже разместите метку Label, TreeView (из страницы Win32), снова Label, окно редактирования Edit и кнопку Button.
8. Дважды щелкните Timer и добавьте следующий код к событию OnTimer:
9. Измените свойство Caption для Label1 «Просмотр дерева», для Label2 на

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
Panel2.Caption:=TimeToStr(Time);
end;
```

«Выбор имени файла».
10. Для компонента TreeView дважды щелкните на поле справа от свойства Items, где указано Tree Nodes, чтобы вызвать соответствующий редактор свойства.

11. Добавьте к дереву узел с именем «Уровень 1», затем добавьте подуровень для узла «Уровень 1» - узел второго уровня с именем «Уровень 2» и для последнего - узел следующего уровня с именем «Уровень 3».

12. Дважды щелкнув кнопке Button1 (Диалог «Открыть»), добавьте следующий код:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 OpenDialog1.FileName:='*.*';
if OpenDialog1.Execute then
Edit1.Text:= OpenDialog1.FileName;
end;
```

13. Сохраните модуль с именем more.pas, проект с именем MOREVCL.DPR
14. Запустите программу.
15. Протестируйте.

а) Вы должны увидеть при передвижении ползунок TrackBar, как изменяется индикатор ProgressBar;

б) Выбирая один из ярлычков TabControl, проследите как изменяется надпись на панели;

в) При щелчке на дереве TreeView, оно должно развертывать и свертывать свои узлы;

г) Нажатие кнопки должно отображать на экране диалог OpenFile и имя выбранного файла должно появляться в окне Edit;

Лабораторная работа № 8
Атрибуты файлов

Цель работы:
Создать небольшую программу, позволяющую просматривать и переключать четыре файловых атрибута, которые были описаны (read-only, system, hidden, archive) для указанного файла.

Решаемые в работе задачи:

- Создание проекта “Диспетчера файловых атрибутов” с единственной формой с именем Fileattr.dpr. Имя модуля: Attrscr.pas.

Задание:

- Проект мог читать скрытые и системные файлы и отображать их.

Порядок выполнения работы:

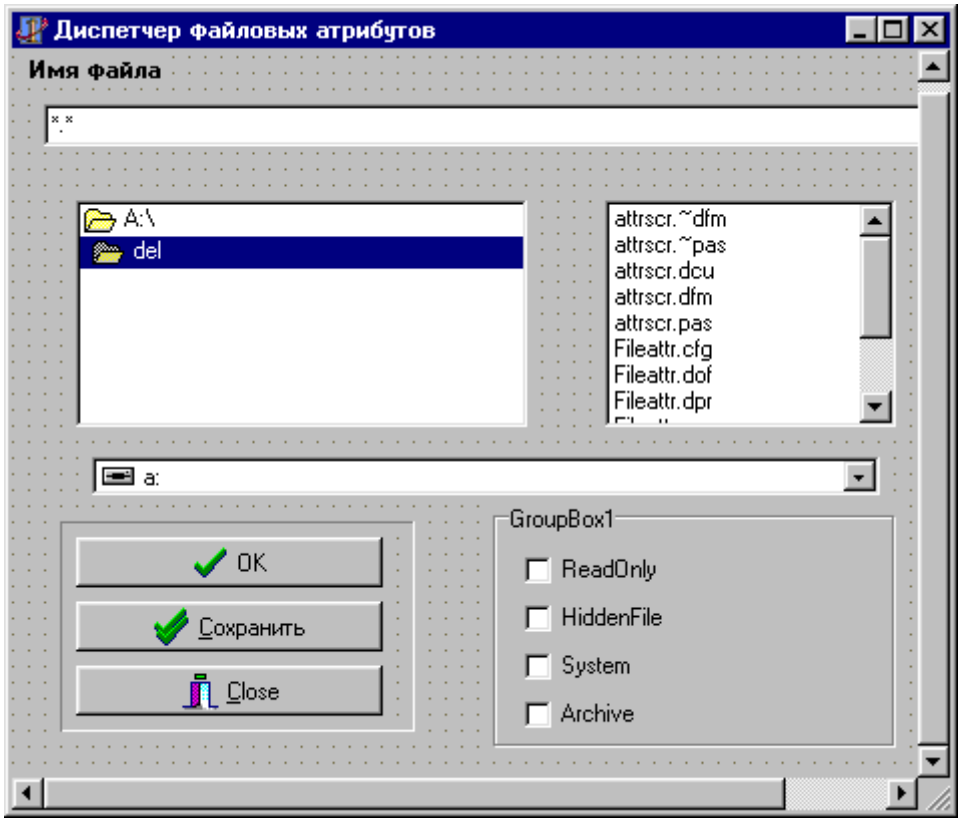


Рис. 8.1. Диалоговое окно «Диспетчера файловых атрибутов»

1. Измените свойство формы Caption на «Диспетчер файловых атрибутов».
2. Разместите Label, измените свойство Caption на «Имя файла».
3. Поместите окно редактирования Edit и очистите окно из свойства Text.
4. Слева Edit ниже расположите DirectoryListBox, а справа- FileListBox.
5. Ниже их поместите DriveComboBox.

6. В нижней части формы поместите Bevel с тремя компонентами BitBtn внутри.
7. Справа Bevel поместите GroupBox с четерьмя компонентами CheckBox внутри.
8. Свяжите между собой DirectoryListBox, FileListBox, DriveComboBox.

а) измените свойства FileList в DirectoryListBox на FileListBox1. Заметьте, что FileListBox1 находится в выпадающем списке значений, из которого можно выбрать. Это делает соединение компонентов очень простым.

б) В FileListBox измените свойство FileEdit на Edit1.

Эти изменения приводят к тому, что компоненты работают вместе, как в диалоговом окне File.

9. Для компонента FileListBox установите следующее свойство в True: ftReadOnly, ftSystem, ftHidden, ftArchive, ftNormal.
10. Два оставшихся значения (ftVolumID и ftDirectory) должны быть установлены в False.
11. Измените свойства Caption для четырёх флажков CheckBox на ReadOnly, Hidden File, System и Archive соответственно.
12. Измените имена трёх компонентов BitBtn на BitBtnOK, BitBtnSave, BitBtnClose.
13. В BitBtn OK измените свойство Kind на bkOK
14. Измените свойство Kind для BitBtnSave и BitBtnClose на bkAll и bkClose соответственно.
15. Измените свойство Caption кнопки bkSave на &Сохранить, а кнопки bkClose на &Закрыть.
16. Добавьте код к кнопке BitBtnOK: (Рис.8.2)
17. Код кнопки BitBtnSave: (Рис.8.3)
18. Код кнопки BitBtnClose:

```
procedure TForm1.BitBtn3Click(Sender: TObject);
begin
Application.Terminate;
end;
```

19. Добавьте также к разделу Implementation следующее:

```
implementation

{$R *.DFM}
    var fname:string;
    AttrByte:integer;
```

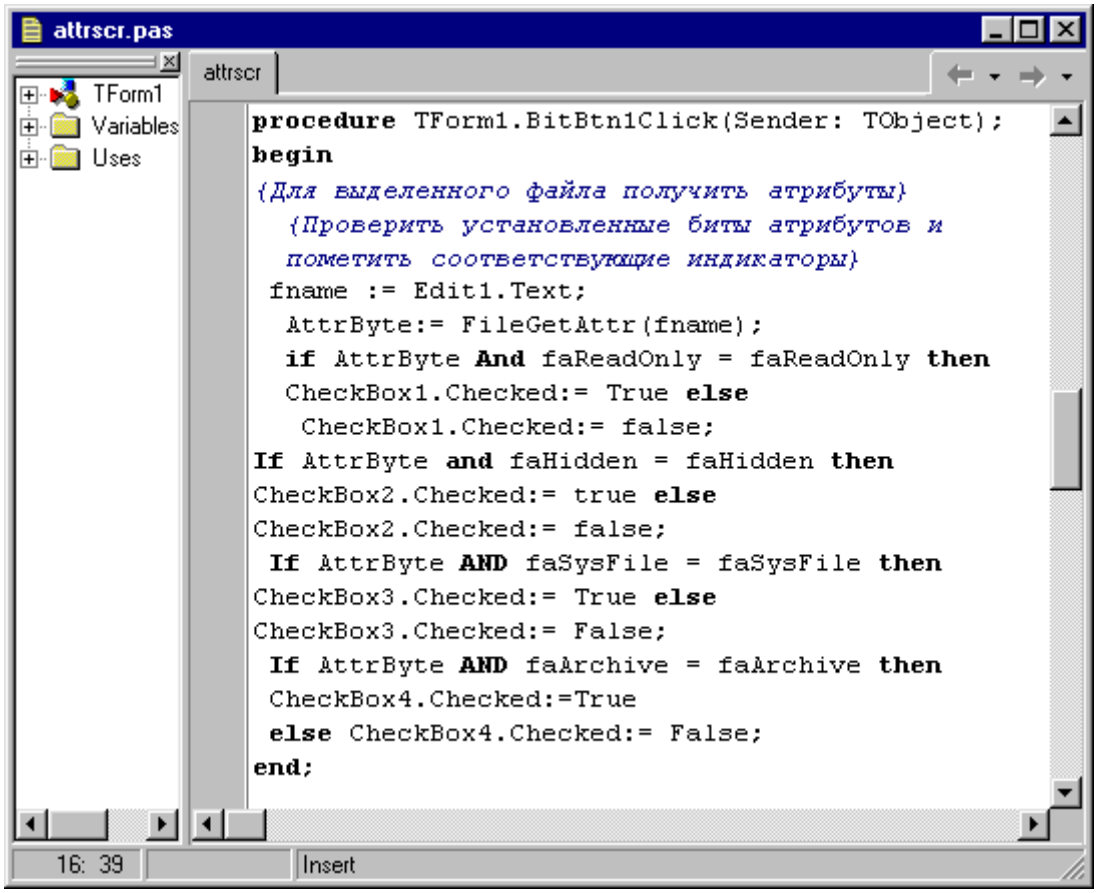


Рис.8.2. Код для кнопки BitBtn.

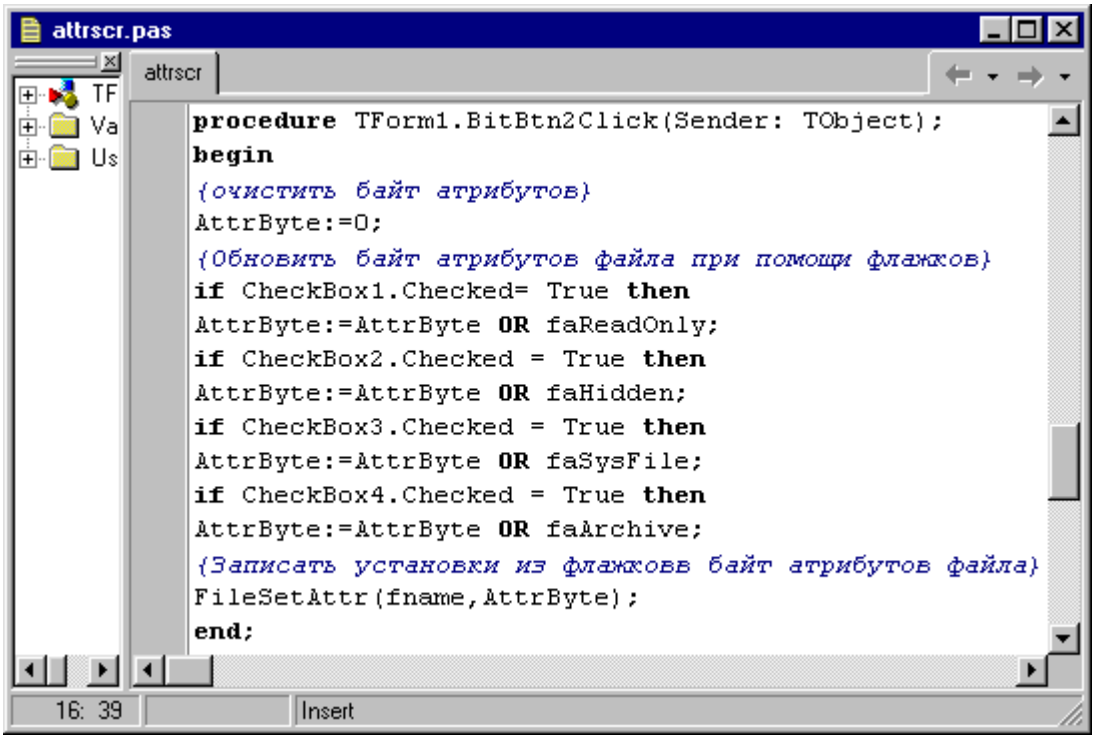


Рис. 8.3. Код для кнопки BitBtn2

20. Код обработчика событий OnDblClick для FileListBox1:

```
procedure TForm1.FileListBox1DblClick(Sender: TObject);
begin
Form1.BitBtn1Click(self);
end;
```

21. Выполните программу.
22. Протестируйте.

Код «Диспетчера файловых атрибутов» имеет два раздела. Они содержаться в обработчиках событий кнопок OK (BitBtnOkClick) и сохранить (BitBtnSaveClick). Код кнопки Ok используется для чтения атрибутов файла, выбранного в окне списка файлов, и отображения их во флажках формы. Код кнопки «Сохранить» используется для присваивания атрибутов, установленных флажками формы выбранному файлу.

Для тестирования этой программы создайте временный файл и запустите программу. При нажатии кнопки Ok для этого файла флажок Archive должен быть помечен, а все другие флажки – нет. Пометьте флажок Hidden File и щелкните на кнопке «Сохранить». Попробуйте теперь найти имя этого файла в листинге каталога. Оно не должно появляться в списке (если только вы не используете программу, отображающую скрытые файлы). Отмените установку флажка Hidden File, щелкните на «Сохранить». Теперь имя этого файла появится в списке каталога.

Контрольные вопросы:

- Что такое атрибуты файлов?
- В чем разница между атрибутами?
- Виды атрибутов.
- Приведите пример атрибутов файла?

Лабораторная работа № 9
Работа с текстовыми файлами

Цель работы:
Создать программу для чтения и записи простого текстового файла.

Решаемые в работе задачи:

- **Создание проекта с формой “Демонстрация текстовых файлов”.**

Задание:
1) Создайте проект, который читает текст из файла и отображает его в нижнем окне редактирования.
2) Установить связи между файлами.

Порядок выполнения работы:

1. Начиная сверху, поместите элементы Label, Edit1, Label2, Edit2.
2. Слева направо в нижней части формы расположите 3 компонента Button.
3. Задайте заголовки каждого компонента в соответствии с рис. 9.1.
4. Добавьте код для кнопки “Сохранить”: (рис.9.2).
5.Добавьте код для кнопки “Загрузить”: (рис.9.3)

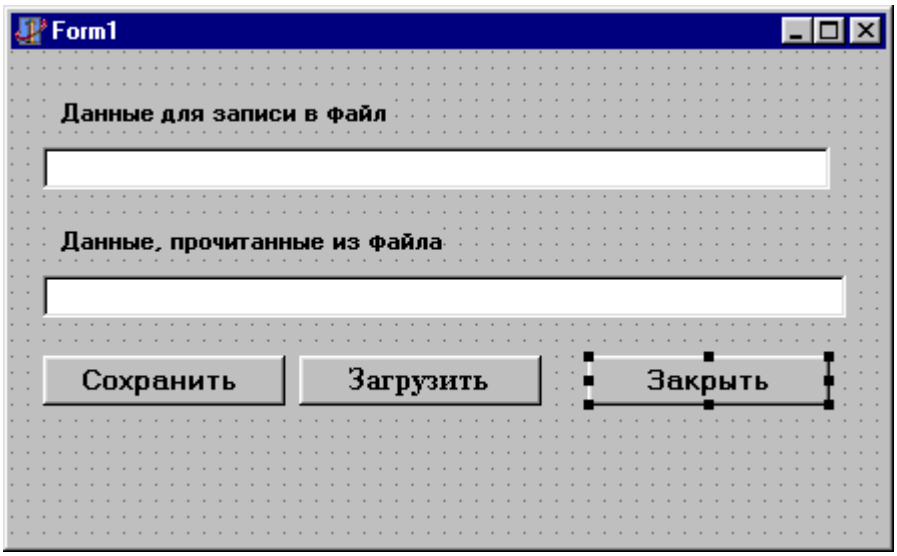


Рис. 9.1. Приложение “Демонстрация текстовых файлов”

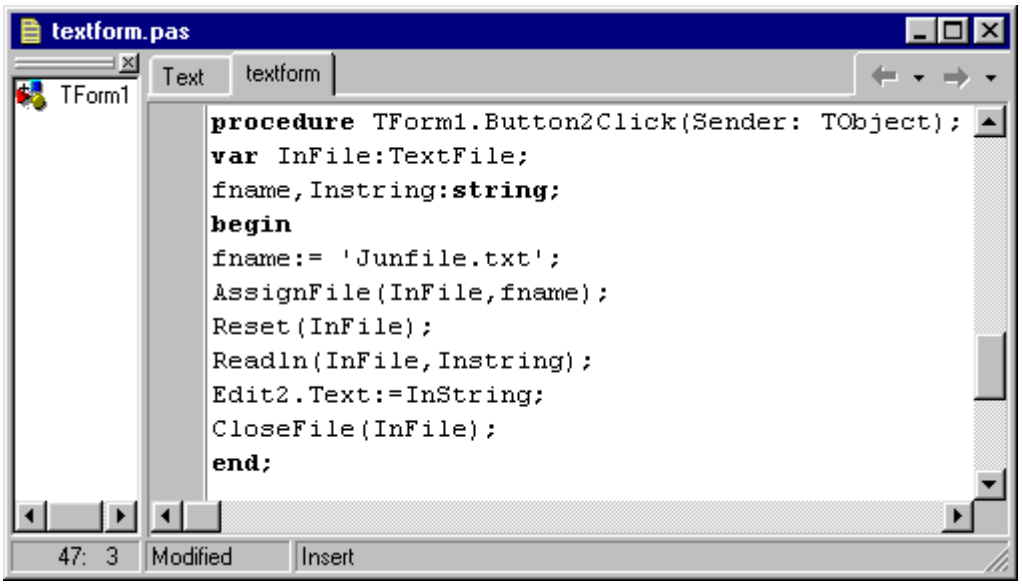


Рис.9.2. Код для кнопки «Сохранить»

6. Добавьте код для кнопки “Закрыть”:

```
procedure TForm1.Button3Click(Sender: TObject);
begin
Appication.Terminate;
end;
```

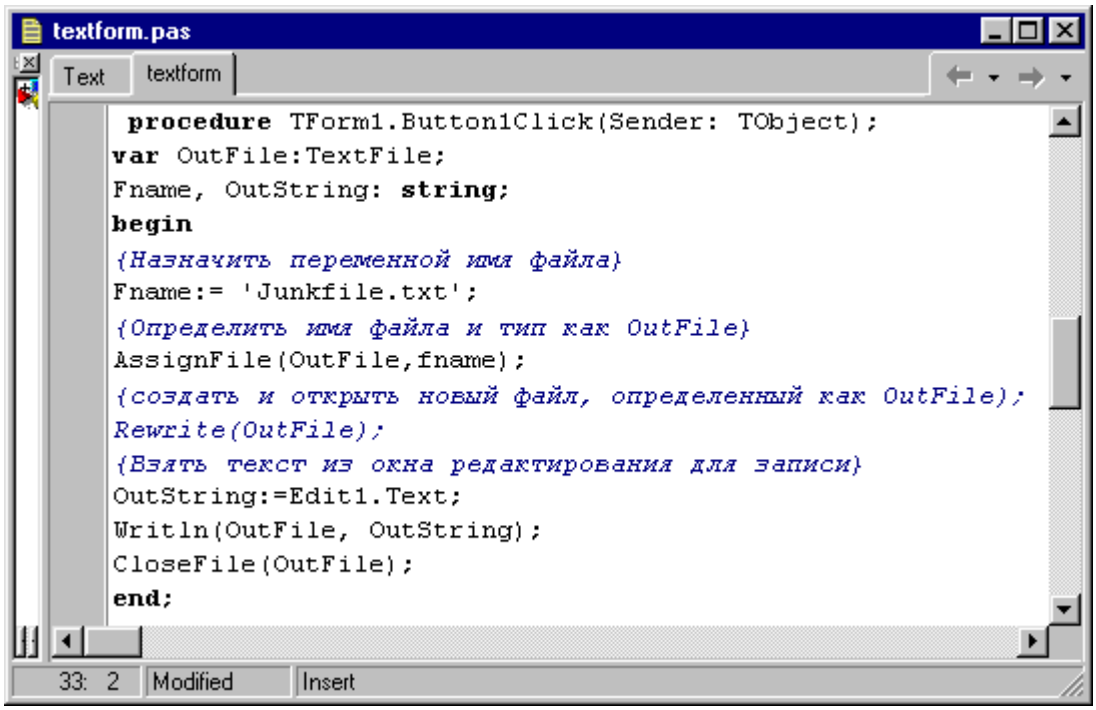


Рис.9.3. Код для кнопки «Загрузить»

Это небольшое приложение позволяет вам ввести текст в верхнее окно редактирования, нажать кнопку “Сохранить” и в результате записать этот текст в файл с именем “JunkFile.txt. Вы можете прочитать этот текст из файла и отобразить его в нижнем окне редактирования.

Лабораторная работа №10
Работа с текстовыми файлами (с указанием конца)

Цель работы:

- Создать приложение с кодами, использующими цикл для загрузки текста построчно (пока не достигнет конца файла).

Решаемые в работе задачи:

- Создание проекта подобному стандартному текстовому редактору, но с меньшим количеством опций и возможностей.

Задание:

1) Создать проект подобный стандартному текстовому редактору, но с меньшим количеством опций и возможностей.
2) Создать файл в любом другом редакторе.
3) Тестировать файл с помощью созданного проекта.

Порядок выполнения работы:

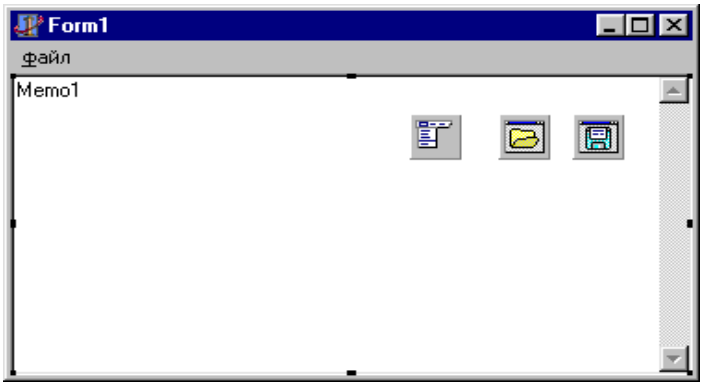


Рис.10.1. Приложение текстовый редактор WINEDIT

1. Поместите на форму компоненты Memo, OpenDialog, SaveDialog и MainMenu.
2. Щелкните на компоненте OpenDialog и дважды щелкните свойство Filter для вызова редактора фильтра.
3. Добавьте следующие два фильтра к свойству Filter: *.txt, *.*.
4. Щелкните на компоненте SaveDialog и дважды щелкните свойство Filter, и добавьте следующие два фильтра к свойству Filter:

Название фильтра	**Фильтр**
Текстовые файлы	*.txt
Все файлы	*.*

5. Используя компонент MainMenu, добавьте меню Файл с подразделами Открыть, Сохранить, /, Выход.
6. Измените свойство Align компонента Memo на alClient. Это приводит к тому, что Memo автоматически изменит размеры и заполнит всю область клиента формы.
7. Установите для компонента Memo свойства ScrollBar в ssVertical.
8. Добавьте код к разделам меню Файл:

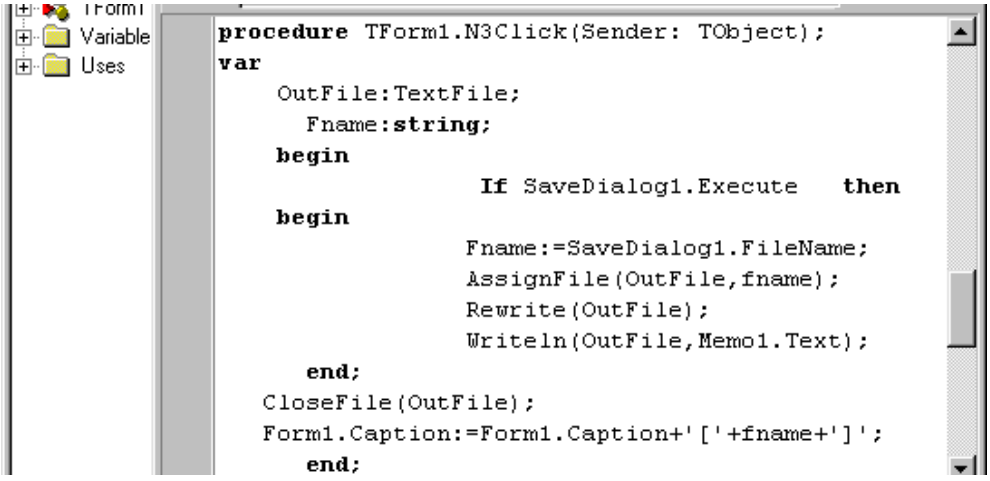


Рис. 10.2. Код для меню «Сохранить»

а) код разделу меню «Открыть» (рис. 10.3).

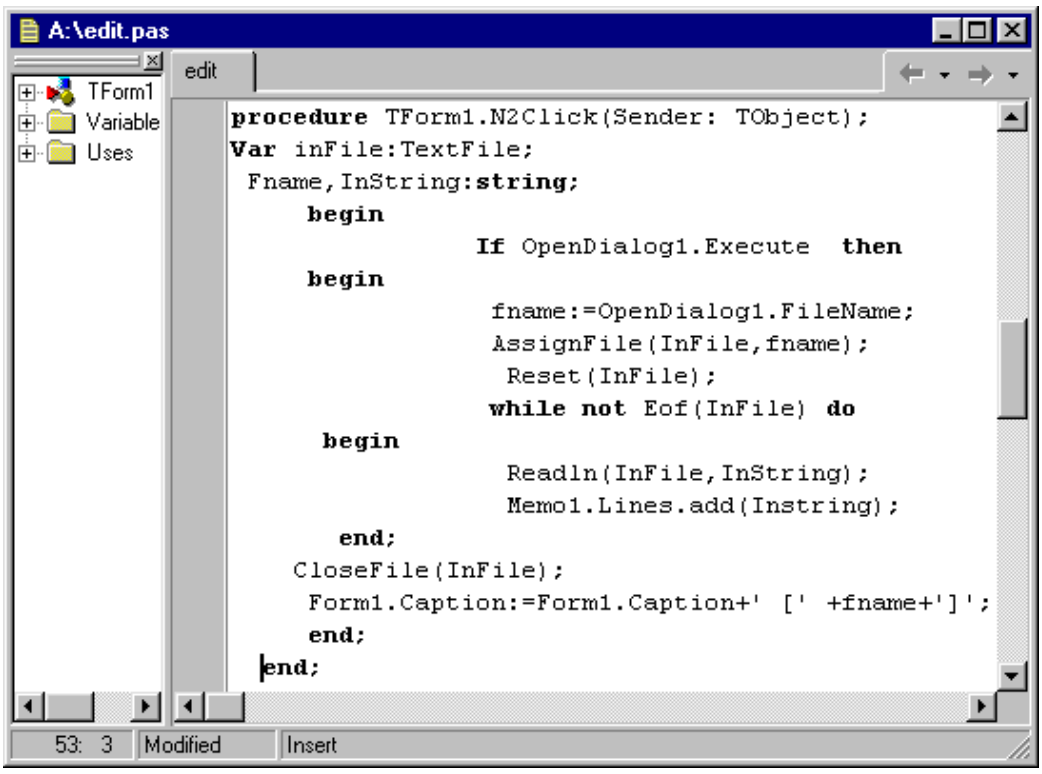


Рис. 10.3. Код для меню «Открыть»

б) код к разделу меню «Сохранить»: (рис.10.2)

в) код к разделу меню «Выход»:

```
procedure TForm1.N5Click(Sender: TObject);
begin
Application.Terminate;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
 {Очистить окно Memo  }
 Memo1.Text:=' ';
end;
```

9. Модуль сохранить под именем Edit.pas, а проект WinEdit.dpr.
10. Запустить программу.
11. Загрузите текстовый файл для тестирования.
12. Внесите некоторые изменения в этот файл и сохраните его новым именем.

Лабораторная работа № 11
Работа с типизированными файлами

Цель работы:

- Создать приложение - базу данных адресно-телефонной книги без использования инструментальных средств баз данных.

Решаемые в работе задачи:

1. Создание собственного типа данных с именем Address. Сохраняя предыдущие формы записей выполнить очищение поля для ввода новых записей.
2. Применение от одной записи к другой.
3. Выход из приложения.

Задание:

1) Создать базу данных (на основе индивидуального задания), содержащую некоторые данные;
2) Установить связи между компонентами.

Порядок выполнения работы:

1. Создайте проект с именем Address.dpr, и модуль на нем Address.pas.
2. В результате в Implementation поместите следующий код:

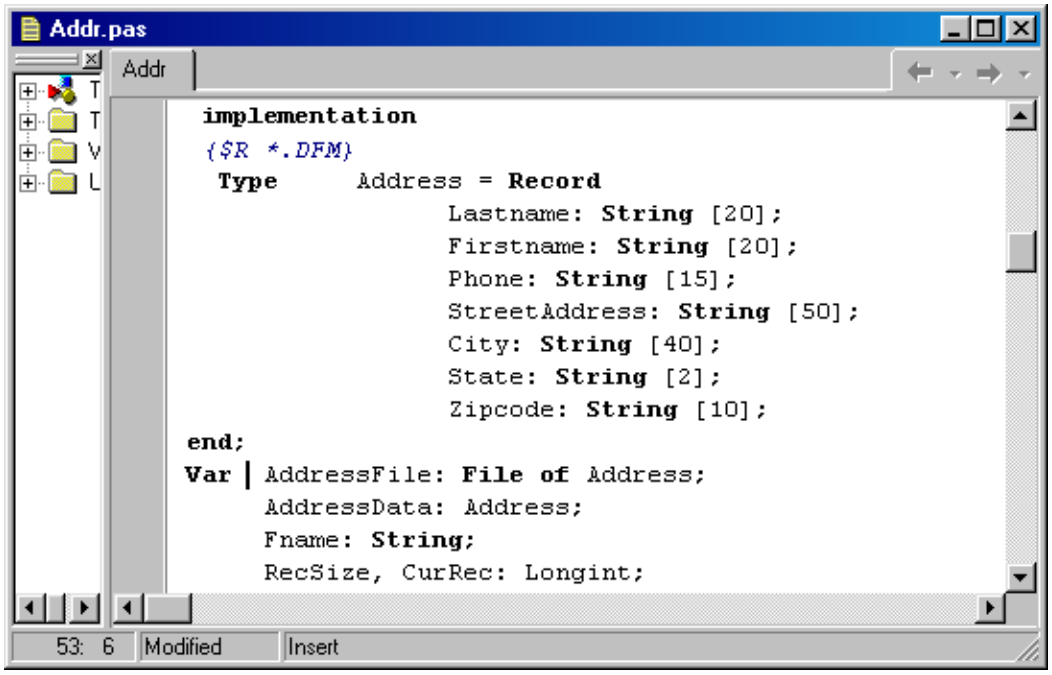


Рис. 11.1. Код описания записи

3. Добавьте в форму семь окон редактирования, семь меток, четыре кнопки Button и одну кнопку BitBtn.

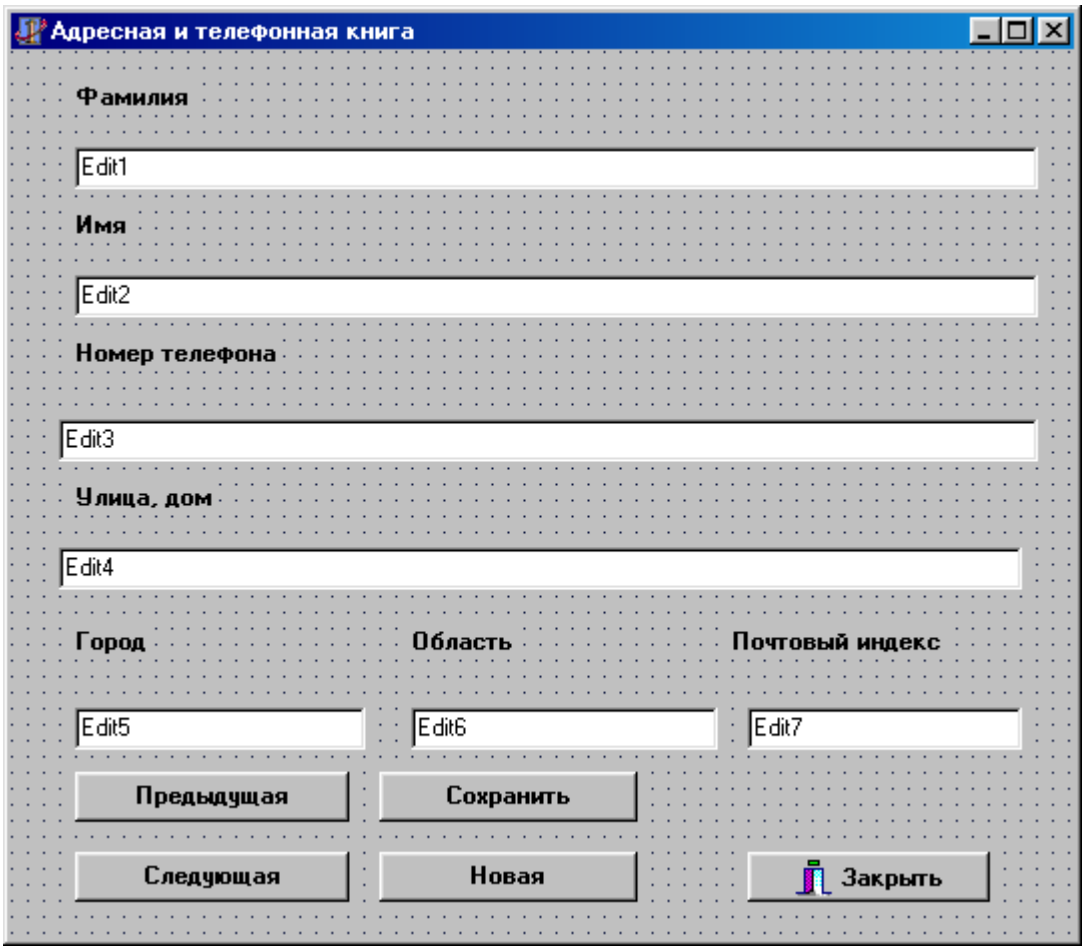


Рис. 11.2. Проектирование формы «Адресной и телефонной книги»

4. Измените свойство заголовков и кнопки BitBtn.
5. Добавьте код для кнопки « Предыдущая»:

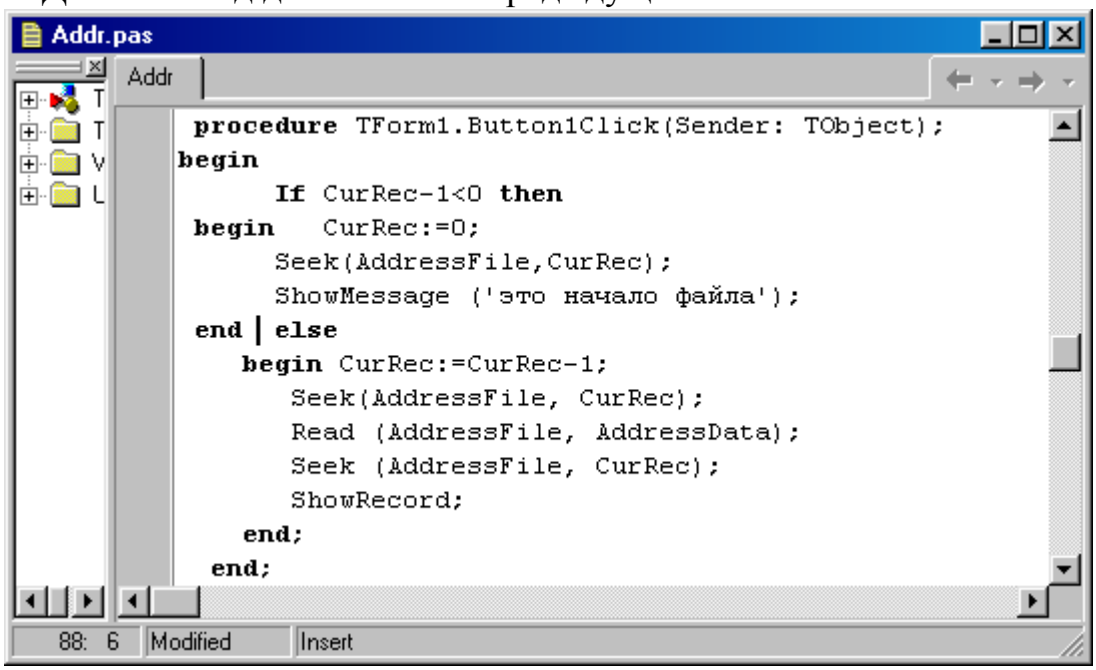


Рис.11.3. Код для кнопки «Предыдущая»

6. Добавьте код для кнопки «Следующая»:

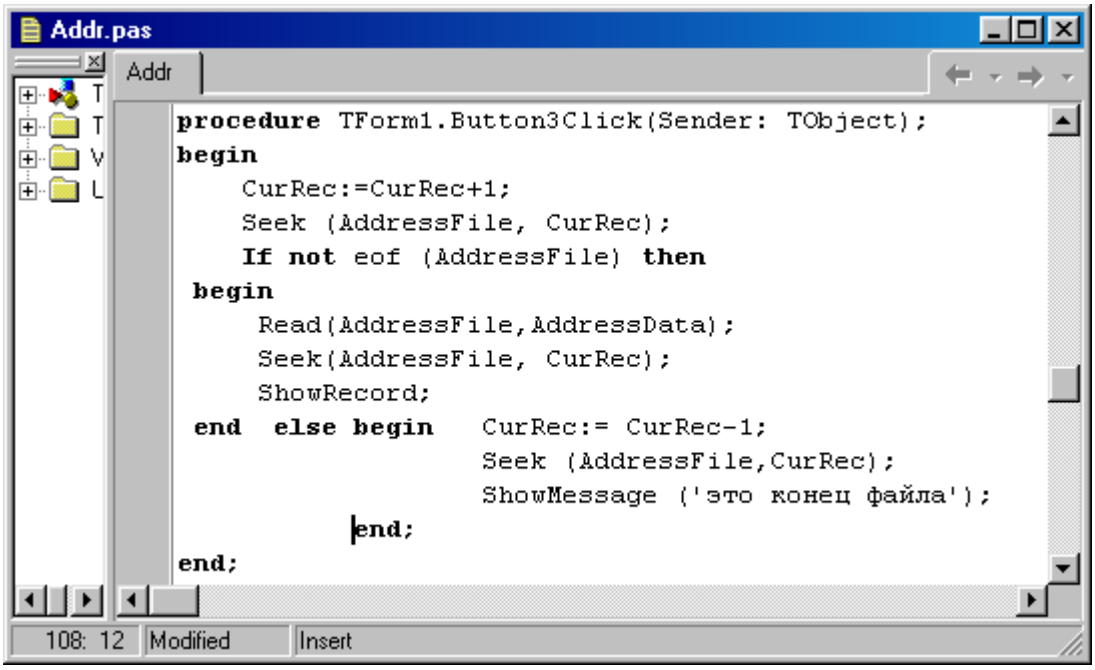


Рис. 11.4. Код для кнопки «Следующая»

7. Добавьте код для кнопки «Сохранить»:

```
procedure TForm1.Button2Click(Sender: TObject);
begin
      SaveRecord;
      ShowRecord;
end;
```

8. Добавьте код для кнопки «Новая»:

```
procedure TForm1.Button4Click(Sender: TObject);
begin
  Repeat  CurRec:=CurRec+1;
   Seek(AddressFile, CurRec);
 Until Eof(AddressFile);
   ClearData;
   SaveRecord;
   Seek(AddressFile, CurRec);
  end;
```

9. Добавьте код для кнопки «Закрыть»:

```
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
SaveRecord;
CloseFile(AddressFile);
Application.Terminate;
end;
```

10. Добавьте код для загрузки записи:

```
 procedure TForm1.LoadRecord;
begin
     Read(AddressFile,AddressData);
     ShowRecord;
end;
```

11. Код для очистки данных:

```
   Procedure TForm1.ClearData;
begin Edit1.Text:= '  ';
      Edit2.Text:= '  ';
      Edit3.Text:= '  ';
      Edit4.Text:= '  ';
      Edit5.Text:= '  ';
      Edit6.Text:= '  ';
      Edit7.Text:= '  ';
end;
```

12. Код для сохранения записи:

```
   Procedure TForm1.SaveRecord;
 begin
  AddressData.Lastname:=Edit1.Text;
  Addressdata.Firstname:=Edit2.Text;
  Addressdata.Phone:= Edit3.Text;
  Addressdata.StreetAddress:=Edit4.Text;
  Addressdata.City:=Edit5.Text;
  Addressdata.State:=Edit6.Text;
  Addressdata.ZipCode:=Edit7.Text;
Write(AddressFile, Addressdata);
end;
```

13. Код для редактирования записи:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
Cleardata;
CurRec:=0; fname:='Address.Dat';
AssignFile(AddressFile,Fname);
RecSize:= Sizeof (AddressData);
If FileExists(fname) then
     begin Reset(AddressFile);
           If not Eof(AddressFile) then
              begin Read(AddressFile,AddressData);
     ShowRecord;
              end;
     End  else begin ClearData;
                 Rewrite(AddressFile);
     end;
     end;
```

14. Код для отображения записи:

```
Procedure TForm1.ShowRecord;
begin
Form1.Edit1.Text:=Addressdata.lastname;
Form1.Edit2.Text:=Addressdata.Firstname;
Form1.Edit3.Text:=Addressdata.Phone;
Form1.Edit4.Text:=Addressdata.StreetAddress;
Form1.Edit5.Text:=Addressdata.City;
Form1.Edit6.Text:=Addressdata.State;
Form1.Edit7.Text:=Addressdata.Zipcode;
  end;
```

15. Протестируйте приложение.
16. Запустите это приложение. Вы увидите рис. 11.5.

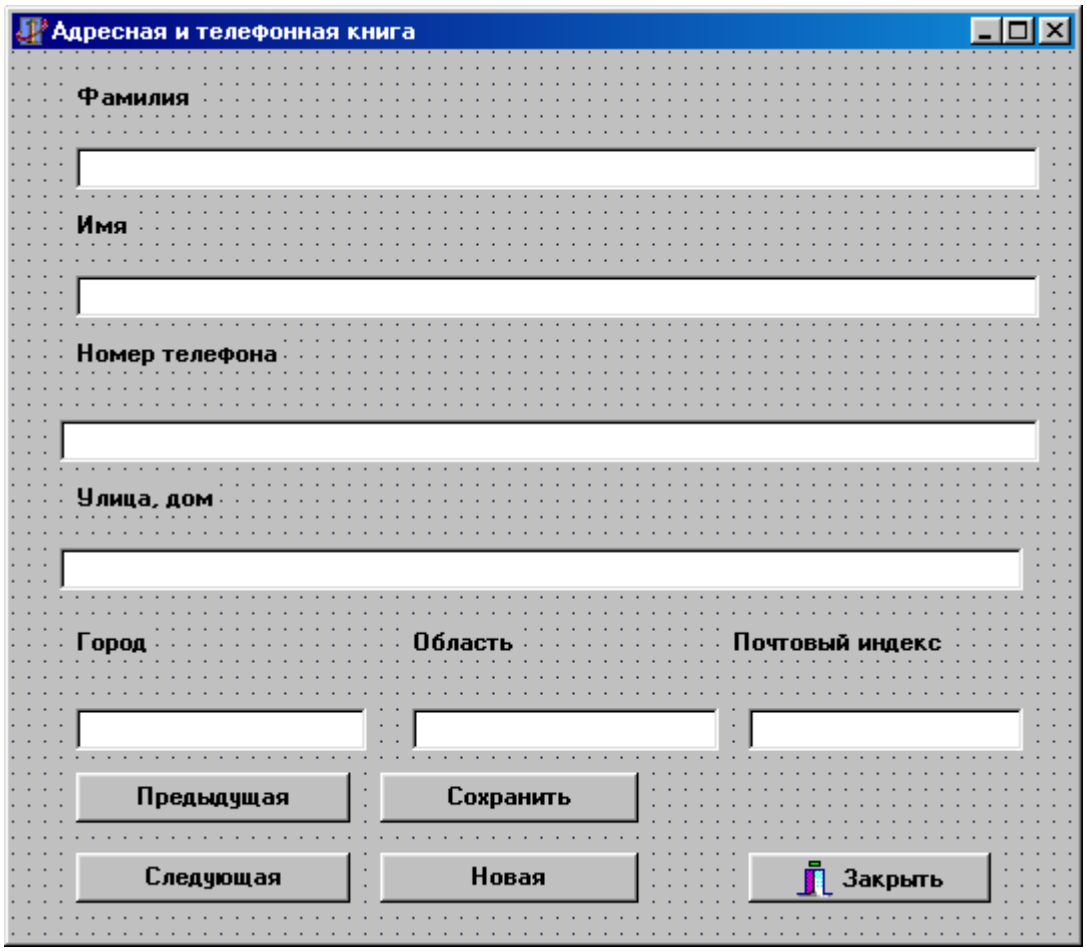


Рис. 11.5. Форма приложения «Адресная и телефонная книга»

Когда программа запускается в первый раз, она определяет присутствует ли файл данных. В данном случае она не должна найти этот файл, поэтому создает его (Address.Dat)

1. Введите первую запись: напишите свое имя, адрес и номер телефона в окнах редактирования и нажмите « Сохранить».
2. Для ввода следующей записи нажмите кнопку «Новая». Окна очищаются и готовы для ввода следующей записи.

3. Введите еще несколько записей таким же образом, нажимая «Сохранить» после введения каждой очередной записи.
4. Теперь протестируйте приложение, перемещаясь по записям с использованием кнопок «Предыдущая» и «Следующая». Заметьте, что, когда вы достигаете начала или конца файла появляется окно сообщения, указывающее на окончание записей. Очевидно, что важной задачей является перехват ошибок, а также предотвращение попыток проскочить начало или конец файла.
5. Покиньте приложение и запустите его снова. На этот раз приложение обнаруживает файл Address.Dat и загружает его, отображая первую запись. Как и раньше, вы должны иметь возможность сохранять и добавлять новые записи.
6. Изучите код приложения.

Лабораторная работа № 12
Работа с нетипизированными файлами

Нетипизированные файлы дают больше возможностей в работе с файлами. Можно перемещаться в любое место файла, изменять один байт или весь блок, сохранять данные и закрывать этот файл. Только в этом случае не нужно заботиться о строгой структуре. При написании кода должны писать код, определяющий позицию в файле, с которой можно работать в данный момент. Это делается с помощью файловых указателей:

In File - переменная типа File;

Fbuffer - это массив типа Byte размером 1 КБ, который можно использовать для хранения данных, считываемых из файла;

Fpointer - используются для хранения значения файлового указателя (позиции в файле), который нужен для того, чтобы можно было возвращаться в это же место после операции чтения или записи.

С функцией AssignFile вы знакомы, но заметьте, что вы добавляете еще один параметр к вызову функции Reset. Это размер записи равный по умолчанию 128 байтам. Для данного примера установите его равным 1.

Когда файл открывается, вам необходимо загружать его в память по одному КБ за раз. Обратите внимание на цикл While который выполняется до тех пор, пока не будет достигнут конец файла. Заметьте так же, что в каждой итерации цикла вы сохраняете позицию файлового указателя. Вы считываете блок, используя процедуру BlockRead для размещения блока размером 1 КБ в буфере.

Затем скопируйте эти данные в буфер побайтно, заменяя все пробелы на запятые. После обработки буфера используйте процедуру Seek для установки файлового указателя на начало записи. Это необходимо, потому что после операции чтения или записи файловый указатель перемещается к началу

следующего блока данных. Затем используйте BlockWrite для записи буфера на диск. Если еще имеются данные для обработки, выполните этот цикл снова, в противном случае этот файл обновляется и закрывается с помощью CloseFile. После этого отображается сообщение «Процесс завершен!».

Приведенный код работает очень быстро по двум причинам. Процедуры BlockRead и BlockWrite загружают весь блок данных одной операции. Затем обработка происходит в памяти (буфере), что гораздо быстрее, чем побайтовая считывание записи.

Цель работы:

- Создать проект, который в заданном файле все пробелы заменяет на запятые.

Решаемые в работе задачи:

1. Создать простую программу на Delphi, которая читает файл в буфере блоками.
2. Сканирует буфер на наличие пробелов и заменяет их на запятые.
3. Сохраняет изменения на диске.

Задание:

1) Создать приложение, которое выбирает заданный файл и в этом файле все пробелы заменяет на запятые.

Порядок выполнения работы:

1. Создайте новый проект Delphi и сохраните его модуль, как SP2CMA.pas.
2. Сохраните проект под именем SP2COMMA.DPR.

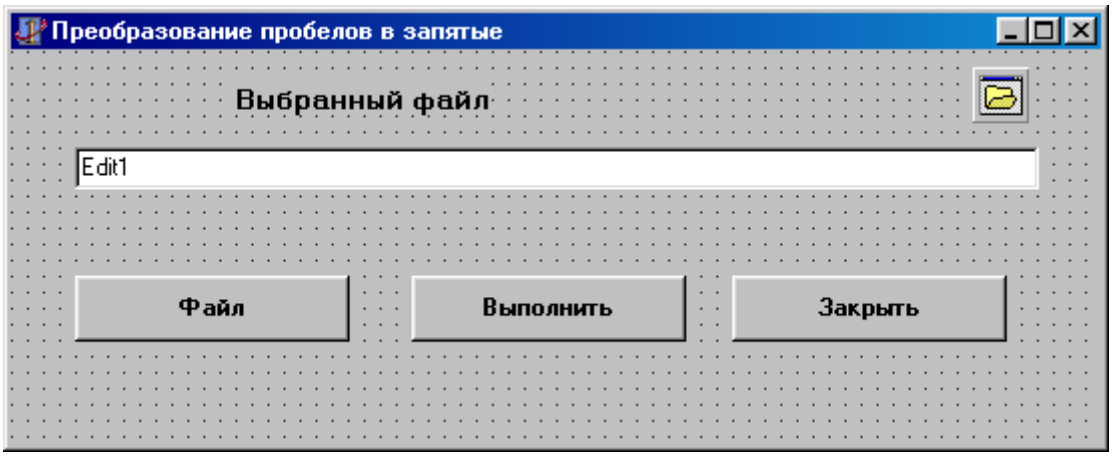


Рис.12.1. Программа преобразования пробелов в запятые

3. Расположите в форму три кнопки, окно редактирования и одну метку.
4. Измените заголовок формы «Преобразование пробелов на запятые».
5. Добавьте компонент OpenDialog.

6. Щелкните дважды на свойстве Filter и используете редактор фильтра для задания имени фильтра «Все файлы», а самого фильтра - «*. *».
7. Установите свойства Visible окна редактирования и метки равными False.
8. Щелкните на кнопку «Выполнить» и добавьте код: (рис.12.2)
9. Добавьте код для кнопки «файл»:

```
 procedure TForm1.Button1Click(Sender: TObject);
begin
       OpenDialog1.FileName:='*.*';
    If OpenDialog1.Execute then
       Fname:=OpenDialog1 .FileName;
       Edit1.Text:=Fname;
       Edit1.Visible:=true;
       Label1.Visible:=true;
 end;
```

10. Добавьте код для кнопки «Закрыть»:

```
 procedure TForm1.Button3Click(Sender: TObject);
begin
  Application.Terminate;
end;
```

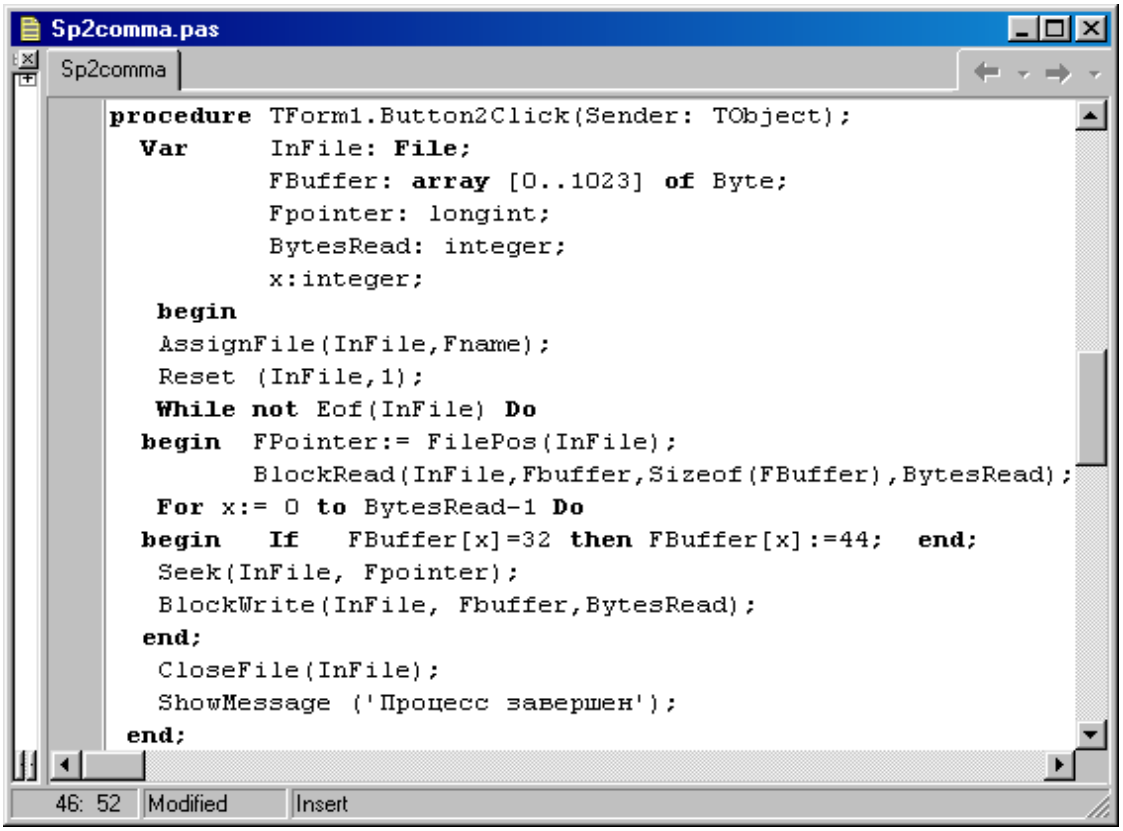


Рис.12.2. Код кнопки «Выполнить»

11. Протестируйте вашу работу.

Программа читает текстовый файл выбранный пользователем Fbuffer блоками по 1024 байта. Когда Вы нажимаете кнопку «Выполнить», содержимое буфера сканируется и пробелы заменяются на запятые.

Файловый указатель устанавливается на начало блока, так как он изменялся после ветвления оператора BlockRead и модифицированные данные снова записываются в файл. После выполнения этого процесса программа отображает на экране сообщение «Процесс завершен!». Это указывает на то, что пользователь может повторить эту операцию с другим файлом или выйти из программы.

1. Итак, сохраните этот код, откомпилируйте и выполните его.
2. Перейдите в редактор NotePad и создайте текстовый файл с одним –двумя предложениями для тестирования полученного предложения.
3. Переключитесь на ваше приложение в Delphi и нажмите кнопку «файл».
4. Используя диалоговое окно, выберите раннее созданный текстовый файл.
5. Нажмите кнопку «Выполнить».
6. Появится сообщение о том, что обработка завершена.

Выйдите из программы, снова войдите в редактор NotePad и загрузите свой файл.

7. Вы увидите запятые, там, где раньше были пробелы.

Лабораторная работа № 13
Вывод файла на печать

Это работа рассматривает печатающие объекты, доступные в Delphi. Используя печатающий объект, вы можете печатать текст. Простейшая форма печати просто подразумевает создание файловой переменной, о которой вы уже знаете, и связывание ее с принтером. Затем вы используете команду Writeln, чтобы послать текст на принтер.

Следующая программа использует для печати команду Writeln:

```
  Var
      P:TextFile;
begin
      Assign Prn(P);
      Rewrite(P);
      Writeln(P,'это тест печати);
      CloseFile(P);
end;
```

В этом фрагменте, объявляется переменная, названная P и имеющая тип TextFile. Далее используется команда AssignPrn. Она настраивает переменную на порт принтера, обращаясь с ним, как с файлом. Затем порт принтера должен быть открыт, что делается командой Rewrite. Текст передается в принтер процедурой Writeln, а закрывается порт принтера командой CloseFile. Важно закрыть порт принтера для завершения операции печати. По этой команде, как и в случае файла, любой текст, остающийся в памяти, пересылается в принтер, и порт закрывается.

Цель работы:
Создать простое приложение, которое печатает текстовый файл.

Решаемые в работе задачи:
- Создание программы, которая выбирает текстовый файл и его печатает.

Задание:
1)Создать текстовый файл в приложении NotePad.
2)Печатать ваш текстовый файл с помощью вашего проекта.

Порядок выполнения работы:

1. Разместите на форме две кнопки Button, компонент OpenDialog и метку.
2. Используйте свойство Filter компонента OpenDialog, чтобы создать фильтр *.txt.

Рис.13.1. Приложение печати текстового файла.

3. Введите коды для кнопки Button1:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  If  OpenDialog1.Execute  then
   begin Fname := OpenDialog1.FileName;
         Label1.Caption:='готов  для  печати';
   end;
end;
```

4. Код для кнопки Button2:

```
 procedure TForm1.Button2Click(Sender: TObject)
Var    P,F:TextFile;
       TempStr:String;
Begin       AssignFile(F,Fname);
            Reset(F);
            AssignPrn(P);
            Rewrite(P);
            Label1.Caption:=Fname +'печатается';
While  NotEoF(F) do
begin    Readl(F,TempStr);
         Write(P,TempStr);
end;
         CloseFile;
         CloseFile(P);
Label1.Caption:='Печать  завершена';
end;
```

5.Сохраните проект как File2PRN.dpr, а модуль как FilePRN.pas.
6. Выполните и протестируйте ее, печатая какие-нибудь текстовые файлы.

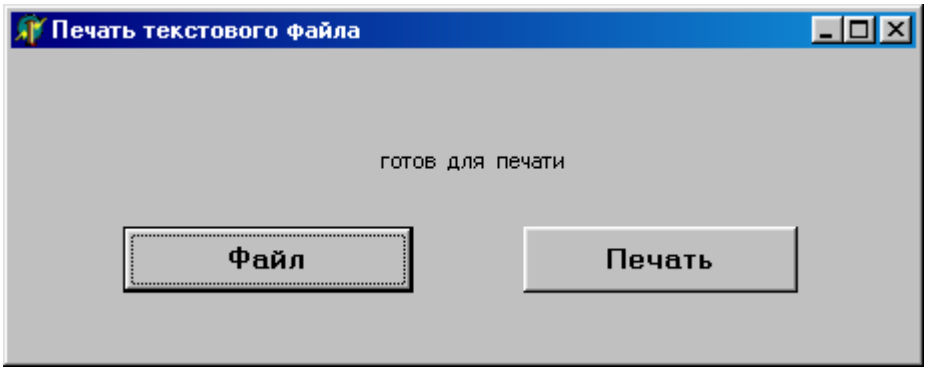


Рис. 13.2. Форма после выбора файла

Лабораторная работа № 14
Печать файла с помощью объекта Tprinter

Цель работы:

- Разработать программу, которая продемонстрирует использование объекта TPrinter.

Решаемые в работе задачи:

1. Используя печатающий объект, напечатать объект.
2. Используя проект, разработанный в лабораторной работе 13, создать программы печати.
3. Отредактировать код кнопки «Печать» в лабораторной работе 13.

Задание:

Создать программу, которая читает файл построчно, как и ранее, но теперь она вычисляет позицию на канве, куда должен быть послан текст, и помещает его там.

Порядок выполнения работы:

Рис.14.1. Приложение печати текстового файла объекта TPrinter

1. Код для кнопки файл-Button1:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 if OpenDialog1.Execute then
begin
Fname:=OpenDialog1.FileName;
Label1.Caption:='Готов к печати'+Fname;
end;
end;
```

2. Код для кнопки печать –Button2:

```
procedure TForm1.Button2Click(Sender: TObject);
var F:TextFile; TempStr, PageNum: string;
    Ctr, x, PHeight, LineSpace: integer;
Begin  Ctr:=1;
{Открытие текстового файла который надо печатать}
                        AssignFile(F,Fname); Reset(F);
{Начало печати}         Printer.BeginDoc;
{Извлечение высоты страницы } Pheight:=Printer.PageHeight;
{ Вычисление расстояний между строками исходя из 60 строк на странице}
LineSpace:=PHeight Div 60;
{Извлечение номера текущей печатаемой страницы}
PageNum:=IntToStr(Printer.PageNumber);
{Обновление метки с номером текущей страницы}
Label1.Caption:='Теперь печатается '+Fname +'Страница'+PageNum;
While Not Eof (F) do begin
{Чтение строки текста из файла в TempStr }
                        Readln (F, TempStr);
{ Содержимое TempStr посылается на принтер}
                        Printer.CanVas.TextOut (0,x,TempStr);
{Увеличение x на состоящее число для печати следующей печати этой стро
                        x:=x+LineSpace;
{Подсчет числа напечатанных строк}
Ctr:=Ctr+1;
{Если напечатано 60 строк, то начало новой страницы ,извлечение номера
If Ctr>59 then
Begin
Printer.NewPage;
     X:=0;  Ctr:=0;
   PageNum:= IntTOStr (Printer.PageNumber);
   Label1.Caption:='Теперь печатается  '+Fname+'Страница'+PageNum;
  end;
 end;
{Закрытие текстового файла и запуск вывода текста на принтер}
  CloseFile (F);
  Printer.EndDoc;
  Label1.Caption:='Печать закончена'+' Число страниц =  '+PageNum;
 end;
 end.
```

3. Сохраните повторно модуль под именем FilePOBJ.Pas, а проект File3PRN.DPR.
4. Выполните и протестируйте вашу работу.
5. Программа ведет себя во многом так же, как и предыдущая, за исключением того, что теперь она печатает через печатающий объект. Есть так же несколько изменений в сообщениях о состоянии.

Лабораторная работа № 15
Печать файла с использованием компонентов TPrinterDialog и TPrinterSetupDialog.

Цель работы:

- Используя форму лабораторной работы 14, напечатать файл с помощью компонентов TprinterDialog и TprinterSetupDialog.
-

Решаемые в работе задачи:

1. На форму, полученную в лабораторной работе 14 добавьте компонент TprinterDialog и соответствующий код.
2. При появлении стандартного диалогового окна печати, сделать выбор таких опций печати, как количество копий, упорядочивание, ориентация страницы.

Задание:

В проекте File2POB.dpr, добавляя в форму компонент TprinterDialog, просмотреть особенности печати.

Порядок выполнения работы:

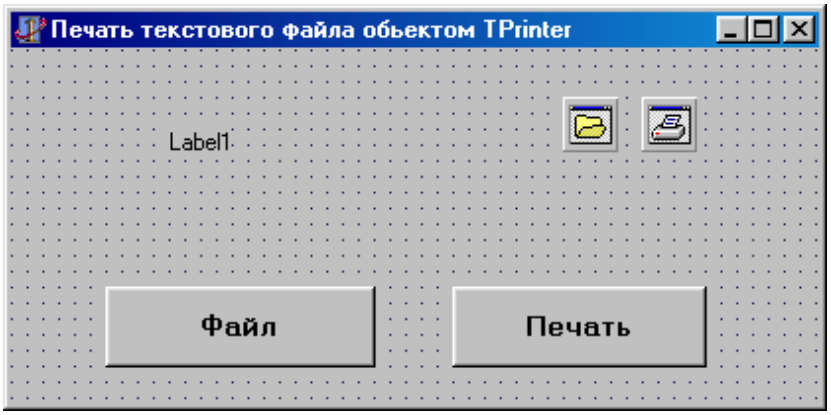


Рис.15.1. Приложение «Печать текстового файла объектом Tpinter и SetupDialog

1. Загрузите проект File2PRN.dpr (лаб. 14), и сохраните его как File2POB.drp, а модуль его PRNDLG.pas.
2. Со страницы Dialogs возьмите компонент PrinterDialog и поместите его на форму.
3. Добавьте код кнопки «Печать».
4. Запустите программу и задайте печать, появится диалоговое окно печати. В этом окне можете установить требуемые операции, прежде чем начнете печатать.
5. Можете вызвать диалоговое окно свойств - операции печати, нажав соответствующую кнопку.

Лабораторная работа № 16
Шрифты и их размеры

Цель работы:
Создать приложение, которое, меняя шрифты и их размеры, учитывая между строчное пространство, напечатает предложенный файл.

Решаемые в работе задачи:
1. Изменить размер шрифта.
2. Изменить шрифт.
3. Вычислять количество строк текста, умещающегося на странице, и начинать в надлежащем месте новую страницу.
4. Для печати указать число копий.

Задание:
Создание проекта, который обеспечивает для выбранного файла печати, с использованием заданного шрифта, заданным размером и числом копий.

Порядок выполнения работы:

Изменение шрифтов и их размеров просто. Предположим, например, что вы хотите изменить размер шрифта на 18, а сам шрифт не Times New RomanCyr, Свойство Font канвы позволяет вам это сделать. Добавляя следующие строки к стандартной программе печати, вы легко можете изменить шрифт:

```
Printer.Canvas.Font. Size :=18;
Printer.Canvas.Font.Name := 'TimesNewRomanCyr';
```

Добавляя, оператор, который выдаст число строк на странице, исходя из размера шрифта, оставим междустрочное пространство:

переменные **LinesPerPage** – число строк на странице.
Line Space – расстояние между строками.

```
LinesPerPage := PHeight  Div  (Font Size +10);
LineSpace := PHeight  Div  LinesPerPage;
```

Добавим команды, обеспечивающие печать числа копий, заданное заказчиком. Оператор

```
Printer.Canvas.Font.Size := Font.Dialog1.Font.Size;
```

устанавливает размер шрифта по информации, полученной из диалогового окна **«Шрифт».**

Оператор

Printer. Canvas. Font. Name:=’Times NewRomanCyr’;

устанавливает тип шрифта **Times New Roman Cyr.**
А оператор

Printer.Canvas.Font.Type := Font.Dialog1.Font.Type;

устанавливает тип шрифта из диалогового окна **«Шрифт».**
Для печати нескольких копий надо создать цикл, заключающий в себе операторы печати и выполняющийся столько раз, сколько указано пользователем. Чтобы узнать заданное число копий, напишите следующий оператор:

Num.Copies :=Print.Dialog1.Copies;

Число копий будет хранится в переменной NumCopies.

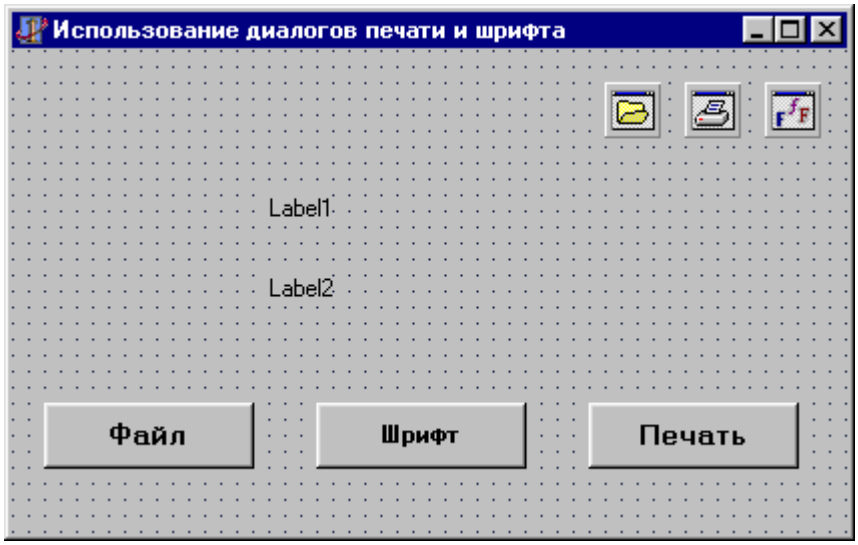


Рис.16.1. Приложение, исполняющее диалоговые окна печати и шрифтов

1. В форму добавьте компоненты OpenDialog, Шрифт (FontDialog), Печать (PrintDialog), две метки и три кнопки Button.
2. Кнопка «Шрифт» позволяет выбрать используемый при печати шрифт с помощью стандартного для Windows диалогового окна «Шрифт».
3. Кнопка «Печать» высвечивает на экране стандартной в Windows диалог «Печать».

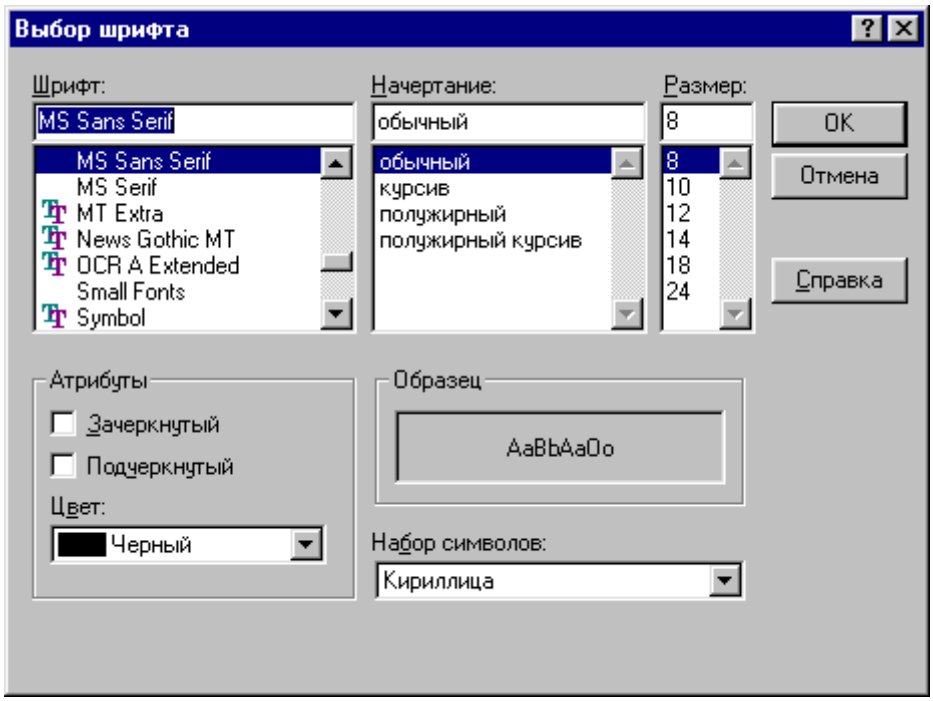


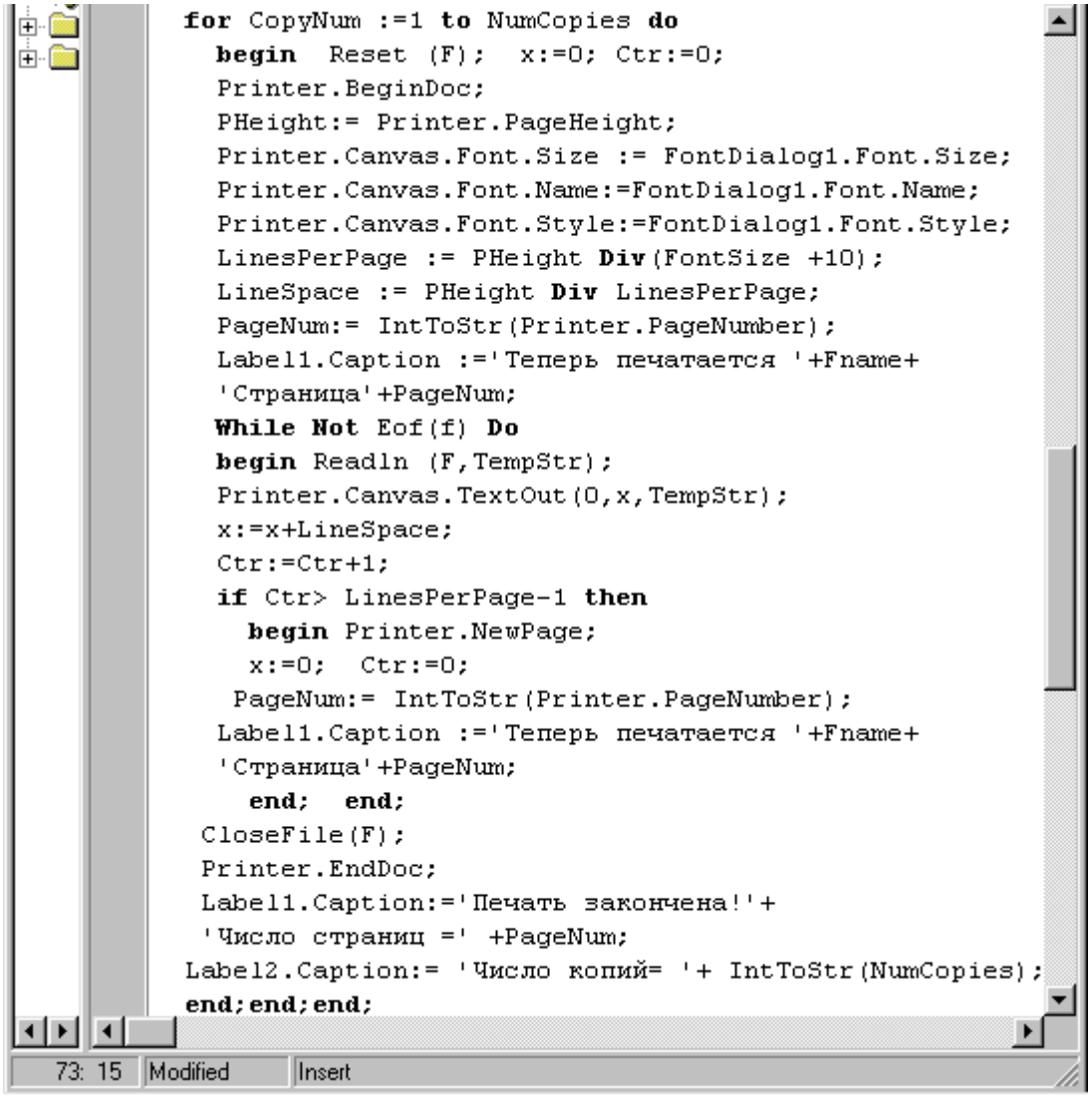
Рис. 16.2. Диалоговое окно «Шрифт»

4. Добавьте код для кнопки «Файл»:

```
  procedure TForm1.Button1Click(Sender: TObject);
begin
if OpenDialog1.Execute then
  begin Fname :=OpenDialog1.FileName;
  Label1.Caption := 'готов к  печати ' +Fname;
  end;
end;
```

5. Добавьте код для кнопки «Шрифт»:

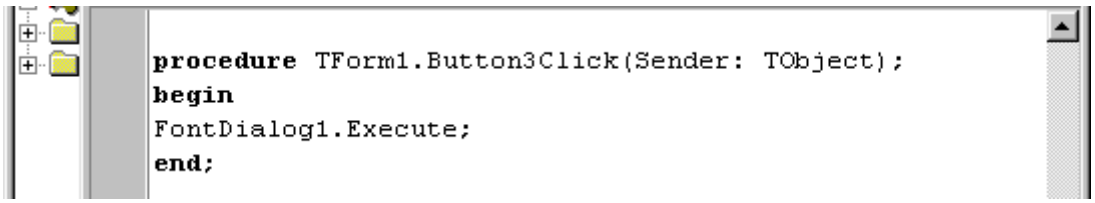
```
procedure TForm1.Button2Click(Sender: TObject);
var  F:TextFile;
     TempStr,PageNum :string;
     Ctr, x, PHeight,LineSpace,LinesPerPage,
     FontSize, CopyNum, NumCopies:  integer;
begin
  if OpenDialog1.Execute then
  begin  ctr:=1;
  AssignFile (F,Fname);
  NumCopies :=PrintDialog1.Copies;
```



```
for CopyNum :=1 to NumCopies do
  begin  Reset (F);  x:=0; Ctr:=0;
  Printer.BeginDoc;
  PHeight:= Printer.PageHeight;
  Printer.Canvas.Font.Size := FontDialog1.Font.Size;
  Printer.Canvas.Font.Name:=FontDialog1.Font.Name;
  Printer.Canvas.Font.Style:=FontDialog1.Font.Style;
  LinesPerPage := PHeight Div(FontSize +10);
  LineSpace := PHeight Div LinesPerPage;
  PageNum:= IntToStr(Printer.PageNumber);
  Label1.Caption :='Теперь печатается '+Fname+
  'Страница'+PageNum;
  While Not Eof(f) Do
  begin Readln (F,TempStr);
  Printer.Canvas.TextOut(0,x,TempStr);
  x:=x+LineSpace;
  Ctr:=Ctr+1;
  if Ctr> LinesPerPage-1 then
    begin Printer.NewPage;
    x:=0;  Ctr:=0;
   PageNum:= IntToStr(Printer.PageNumber);
  Label1.Caption :='Теперь печатается '+Fname+
  'Страница'+PageNum;
    end;  end;
 CloseFile(F);
 Printer.EndDoc;
 Label1.Caption:='Печать закончена!'+
 'Число страниц =' +PageNum;
Label2.Caption:= 'Число копий= '+ IntToStr(NumCopies);
end;end;end;
```

73: 15 Modified Insert

6. Добавьте код для кнопки «Печать»:



```
procedure TForm1.Button3Click(Sender: TObject);
begin
FontDialog1.Execute;
end;
```

Эта программа дает возможность пользователю выбрать файл для печати с помощью диалогового окна «Открыть», имеющегося в Windows. Кнопка «Шрифт» позволяет выбрать используемый при печати шрифт с помощью стандартного для Windows диалогового окна «Шрифт». Кнопка «Печать» высвечивает на экране стандартный в Windows диалог «Печать». Как видите, большая часть кода связан с обработкой события OnClick кнопки «Печать».

Лабораторная работа № 17
Работа с графическими изображениями

Цель работы:
Напечатать графическое изображение.

Решаемые в работе задачи:
1. Создание программы, которая даст возможность использованием кнопки открыть диалоговое окно “Открыть” для выбора графического файла, выводимого на экран и на принтер.
2. Использование метода CopyRect для переноса изображения на канву принтера, которая посылает изображение на принтер.
3. Выбор коэффициента масштабирования.
4. Создание формы с нужными компонентами.
5. Создание проекта и его модуля.

Задание:
Создание проекта, печатающего графику.

Порядок выполнения работы:

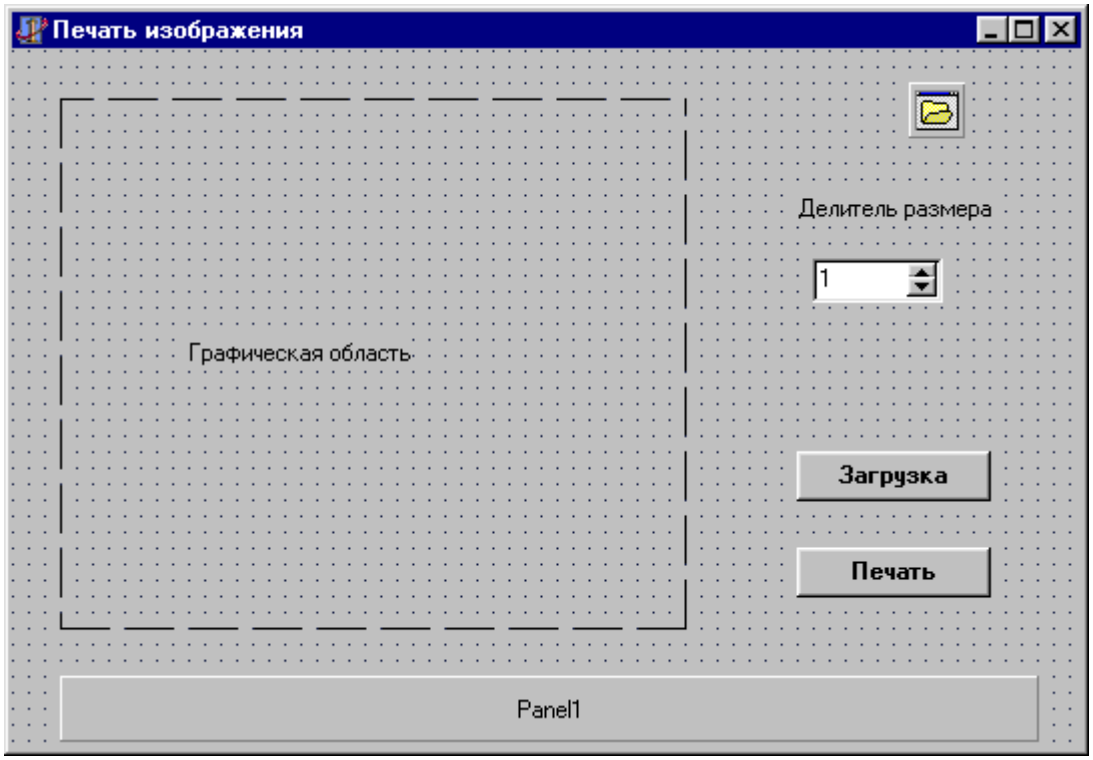


Рис. 17.1. Форма печати изображений

1. Создайте новый проект.

Разместите компоненты в форму как показано на рис. 17.1.

КОЛИЧЕСТВО	КОМПОНЕНТ
1	компонент Image
1	компонент OpenDilog
2	Button
1	панель
2	метка
1	компонент SpinEdit со страницы Samples

2. Замените заголовок формы на “Печать изображения”.
3. Замените названия кнопок на “Загрузка” и “Печать”.
4. Добавьте фильтры к компоненту OpenDialog для графических файлов (*.bmp.и *. wmf.), а так же и для всех других графических файлов.
5. Для компонента SpinEdit установите MinValue на 1 и MaxValue на 10.
6. Замените заголовок Label1 на “Делитель размера” и заголовок Label2 на “Графическая область”.
7. Измените значение свойства Image1, Stretch на True, чтобы размер изображения подгонялся, если возможно, под окно размеры изображения при загрузке.
8. Установите свойство AutoSize на False.
Чтобы загрузить изображение, используйте метод LoadFromFile для панели изображения, которую вы поместили в форму.
В этом случае используйте имя файла из OpenDialog как параметр метода LoadFromFile. Изображение загрузится и будет отображено на экране.

```
If OpenDialog 1.Execute then
Image 1. Picture.LoadFromFile
(OpenDialog 1.FileName);
```

Чтобы напечатать изображение вам надо знать, где центр канвы, чтобы совместить с ним центр изображения. Следующие операторы получают координаты x и y центра делением на 2 высоты и ширины страницы печати. Они сохраняются в переменных CenterX и CenterY.
Определение центра канвы:

```
PHeight : = Printer.PageHeight;
Pwidth  : =  Printer.PageWidth;
CenterX : = Pwidht  Div  2;
CenterY : = PHeight  Div 2;
```

Вам надо вычислить коэффициенты масштабирования, исходя из заданного пользователем масштабного коэффициента. Прежде всего, получите величину этого коэффициента из свойства Value компонента Spin Edit. Чтобы разместить изображение вам нужно знать координаты X и Y верхнего и нижнего правого углов. Соответствующие переменые показаны в следующей таблице:

ПЕРЕМЕННАЯ	ПОЗИЦИЯ
X1	левая
У1	верхняя
X2	правая
У2	нижняя

Следующие операторы вычисляют координаты на основании коэффициента масштабирования, заданного пользователем.

{Расчет центра на основании масштабного коэффициента пользователя}

```
Sdiv:=SpinEdit1.Value;
X1:= Centerx- (Pwidth Div (SDiv*2));
Y1:= Centery- (Pheight Div (SDiv*2));
X2:= Centerx +(Pwidth Div (SDiv*2)) ;
Y2:= Centery +(Pheight Div (SDiv*2));
```

Чтобы поместить графику на канву принтера, надо использовать метод CopyRect копирующий прямоугольную область с одной канвы на другую:

procedure CopyRect (Dest: Trect; Canvas: TCanvas; Source: Trect);

Исходный и конечный прямоугольники передаются как переменные типа Trect. В данном случае исходной канвой является канва Form1.

Вы помещаете окно изображения в точку (0,0) потому, что оператор CopyRect берет изображение из канвы формы, а не непосредственно из панели Image. Если вы установите панель изображений в координаты (0,0) области клиента, то вам не придется беспокоиться о смещениях при вычислениях. Если бы вы сдвинули панель изображения в другое место, не сопровождая это вычислением величины смещения, то могли бы потерять часть или все изображение, потому, что оно оказалось бы вне прямоугольной области, которую вы копируете.

Так как CopyRect использует две переменные типа Trect , вы должны объявить две переменные этого типа и ввести в них данные. Объявление переменных имеет вид:

var RrnRect, ImgRect: Trect;

Trect определена в модуле Windows как запись следующего вида:

```
TRect = record
    case Integer of
    0:    ( Left, Top, Right, Bottom : Integer)
    1:    ( TopLeft, BottomRight: Tpoint);
end;
```

Вы засылаете в нее четыре величины. Затем используете функцию Rect для сохранения данных новых переменных. Параметрами функции Rect являются координаты X1, Y2, X2, Y1 - Left , Top, Right, Bottom, а возвращает эта функция запись Trect

Например: {сохранение желательного размера канвы компонента Printer в переменной PrnRect}

```
PrnRect := Rect (x1,y1,x2,y2);
```

{сохранение размера канвы компонента Image в переменной ImgRect}

ImgRect := Rect (0, 0, Image2.Wigth, Image2.Height);

После того как вы задали значение переменных PrnRect и ImgRect, можно использовать эти переменные в операторе метода CopyRect для копирования изображения на канву принтера:

Printer.Canvas.CopyRect(PrnRect,Form1.Canvas,ImgRect);

PicPrint.pas

PicPrint

```
unit PicPrint;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, ExtCtrls, StdCtrls, Spin, Printers;
type
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Image1: TImage;
    OpenDialog1: TOpenDialog;
    SpinEdit1: TSpinEdit;
    Label1: TLabel;
    Label2: TLabel;
    Panel1: TPanel;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

Рис.17.2. Листинг проекта программы «Печати изображения»

9. Сохраните модуль под именем PicPrint.pas, а проект PRINTPIC.DPR.
10. Испытайте программу.

Программа позволяет пользователю выбирать графический файл с помощью кнопки “Загрузка”. Изображение отображается в форме в компоненте Image. Затем изображение может быть напечатано с помощью кнопки «Печать». Когда нажимается эта кнопка, программа копирует изображение из канвы и посылает его на принтер.

```
    procedure TForm1.Button2Click(Sender: TObject);
begin
if OpenDialog1.Execute then
begin
      Label2.Visible:=False;
      Image1.Picture.LoadFromFile(OpenDialog1.FileName);
end;
end;
end.
```

49: 32 Modified Insert

Средства для выполнения работы:
Delphi 7.0 / Совместимый с IBM PC персональный компьютер 486/ Pentium.

Форма отчета и порядок защиты:

Студент должен:

1) Представить конспект по домашней подготовке;
2) Ответить на контрольные вопросы;
3) Представить компьютерный отчет по проделанной работе;
4) Выполнить дополнительное задание по указанию преподавателя;
5) Представить письменный отчет, содержащий следующие пункты:

- Название лабораторной работы;
- Средства выполнения работы;
- Задание;
- Порядок выполнения работы;

Литература

1. Баас Р., Фервай М., Гюнтер Х. Delphi 4: полное руководство. Учебное пособие. Киев, 1998.
2. Архангельский А.Я. Delphi 7. Справочное пособие. – М.: БИНОМ, 2004.
3. Гофман В., Хомоненко А. Работа с базами данных в Delphi.
4. Сорокин А.В. Delphi разработка баз данных. – СПб.: Питер, 2005.