Санкт-Петербургский Государственный морской
технический университет
Кафедра компьютерной графики и информационного обеспечения

Программирование в Delphi

Методические указания к лабораторным и курсовым работам

Санкт-Петербург
2001

Методические указания предназначено для студентов специальностей 0101,5101,0104,0105 и 1208 Санкт-Петербургского Государственного морского технического университета, выполняющим лабораторные и курсовые работы в рамках дисциплины «Компьютерные науки». Лабораторные работы предназначены для освоения основных приемов конструирования окна Windows-приложения, изучения стандартных объектов палитры компонентов и создания на их основе различных типов приложений с однооконным и многооконным интерфейсом, применение таймера для реализации программ с использованием отсчета реального времени, построения и обработки статических и движущихся изображений.Курсовые работы выполняются в соответствии с индивидуальным заданием.

ДМИТРИЕВ
Сергей Александрович

КОРОТЯЕВА
Татьяна Леонидовна

Программирование в Delphi

Методические указания к лабораторным и курсовым работам

Ответственный редактор С.А. Дмитриев
Редактор Т.А. Канн

Среда быстрой разработки приложений Delphi

Delphi – самое популярное в нашей стране современное средство разработки приложений разнообразного назначения. Основной упор в Delphi делается на максимальном повторном использовании кода. Это позволяет разработчикам строить приложения весьма быстро из заранее подготовленных объектов, а также дает им возможность создавать собственные объекты для среды Delphi. В стандартную поставку Delphi входят основные объекты, которые образуют удачно подобранную иерархию более чем из 300 базовых классов.

Главная программная единица в Delphi – это проект. Основными составными частями проекта являются формы и модули. Формы – это объекты, в которые помещаются другие объекты для создания пользовательского интерфейса приложения. Модули состоят из кода, который реализует функционирование приложения с помощью обработчиков событий для форм и их компонентов.

Традиционно в среде Windows было достаточно сложно реализовывать пользовательский интерфейс. Событийная модель в Windows всегда была сложна для понимания и отладки. Но именно разработка интерфейса в Delphi является самой простой задачей для программиста. Среда Delphi включает в себя полный набор визуальных инструментов для быстрой разработки приложений (RAD – rapid application development), поддерживающий разработку пользовательского интерфейса и подключение к корпоративным базам данных. VCL – библиотека визуальных компонентов, включает в себя стандартные объекты построения пользовательского интерфейса, объекты управления данными, графические объекты, объекты мультимедиа, диалоги и объекты управления файлами, управление DDE и OLE.

Рассмотрим основные составные части Delphi.

Главное окно – управление проектом и его файлами, запуск и отладка проекта, настройка. Delphi обладает мощнейшим встроенным графическим отладчиком, позволяющим находить и устранять ошибки в коде. Вы можете установить точки останова, проверить и изменить переменные, при помощи пошагового выполнения в точности понять поведение программы.

Дизайнер форм (**Form Designer**) – создание интерфейса (окна) проекта и визуальный контроль расположения компонентов.Дизайнер форм первоначально показывает заготовку пустого окна, которое за-

полняется всевозможными объектами, выбранными из *Палитры компонентов*. Эти компоненты имеют графическое представление в поле формы, для того чтобы можно было ими соответствующим образом оперировать. Но для работающей программы видимыми остаются только визуальные компоненты. Компоненты сгруппированы на страницах палитры по своим функциям.

Окно *Редактора исходного текста* (**Code Editor**) – запись кода программы, обрабатывающей события компонентов. В окне Редактора на объектно-ориентированном языке программирования Object Pascal кодируются логика программы и конкретный алгоритм обработки данных. Набор текста программы облегчают шаблоны и списки названий свойств и методов.

Палитра компонентов (**Component Palette**) – набор вкладок для выбора и установки компонентов на форму. Для использования Палитры компонентов нужно первый раз щелкнуть кнопкой мыши (здесь и далее имеется в виду левая кнопка мыши) на одном из объектов, второй раз – на *Дизайнере форм*. Выбранный объект появится в проектируемом окне, и им можно манипулировать с помощью мыши.

Инспектор объектов (**Object Inspector**) – изменение свойств компонентов **Properties** и задание их реакции на события **Events**. Если дважды щелкнуть кнопкой мыши в пустой клетке рядом с названием системного события на странице **Events** в Инспекторе объектов, Delphi создаст процедуру и заголовок метода, куда вы можете добавить код обработки этого события:

```
procedure TForm1.имя(Sender: TObject);
  begin

  <место расположения кода на Object Pascal>

  end;
```

Эта процедура включается в состав модуля формы. Информация о формах хранится в двух типах файлов – *.dfm* и *.pas*, причем первый тип файла хранит образ формы и ее свойства, второй тип описывает функционирование обработчиков событий и поведение компонентов. Оба файла автоматически синхронизируются Delphi, так что если добавить новую форму в ваш проект, связанный с ним файл модуля с расширением *.pas* автоматически будет создан, и его имя будет добавлено в проект.

Лабораторная работа 1

Знакомство с основами визуального программирования. Размещение компонентов на форме и задание их свойств

Создать программу для Windows , подсчитывающую сумму двух чисел.

1.Установить на форму два компонента SpinEdit (вкладка **Samples**) для ввода исходных данных – целых чисел: SpinEdit1 и SpinEdit2.

2.Установить над ними два компонента StaticText (вкладка **Additional**) для надписей.

3.Установить два компонента Label (вкладка **Standard**) для надписи “Результат” (Label1) и вывода самого результата (Label2).

4.Установить кнопку Button (вкладка **Standard**) для запуска расчета результата.

5.Установить свойства компонентов в окне **Object inspector** на вкладках **Properties**, выделяя предварительно соответствующий компонент на форме либо в верхнем раскрывающемся списке.

- Значение свойства Caption (надпись) для компонентов Label, Button и StaticText изменить – вместо стандартных (повторяющих название компонента) занести необходимые надписи на русском языке. Для всей формы это свойство установит заголовок окна программы.
- Установить значения свойств, обеспечивающих появление всплывающей подсказки ShowHint – True (выбрать из раскрывающегося списка) и Hint (текст всплывающей подсказки) для компонентов SpinEdit, Label и Button.

6.Создать обработчики событий. В окне **Object inspector** на вкладках **Events** выполнить два щелчка кнопкой мыши на поле события. В окне кода программы автоматически сгенерируется заготовка процедуры обработки события: заголовок и конструкция Begin … End.

- Для SpinEdit создать обработку события OnChange (реакция на изменение числа в поле). В окно кода между Begin … End вставить следующие строки кода программы:

  f1:=SpinEdit1.Value; (для SpinEdit2 заменить цифру 1 на 2).

  Label2.Caption:=’’; (сброс результата при изменении числа);

- Для кнопки Button1 создать обработку события OnClick (реакция на щелчок мыши) – вычисление и вывод результата:
  Sum:=f1+f2; Label2.Caption:='Сумма='+ IntToStr(Sum);
  (функция перевода из числовой в символьную форму).
- В секцию объявления переменных Var добавить описание целых чисел: f1,f2, Sum:integer.

7.Сохранить проект. Запустить программу и проверить ее работу. Самостоятельно добавить кнопки для вычитания, умножения и деления. Внести необходимые изменения в типы данных.Показать результат преподавателю.

Знакомство с программированием процессов реального времени. Визуализация течения процессов

Создать программу, показывающую текущее время.

1.Установить на форму компонент Timer (вкладка **System**) для отсчета интервалов времени (Timer1).

2.Установить компонент Progress Bar (вкладка **Win32**) для показа десятых долей секунды.

3.Установить компонент Label (вкладка **Standard**) для вывода времени (Label1).

4.Установить свойства компонентов в окне **Object inspector** на вкладках **Properties**, выделяя предварительно соответствующий компонент на форме либо в верхнем раскрывающемся списке:

- убрать содержимое свойства Caption (надпись) для компонента Label1;
- увеличить размер шрифта надписи и изменить цвет букв на красный (в диалоговом окне свойства Font компонента Label1);
- установить значение свойства Interval для компонента Timer1 равным 100, что обеспечивает генерацию события OnTimer примерно через 0,1 секунду.

5.Добавить переменную i : integer для подсчета десятых долей секунды в секцию Var модуля Unit1.

6.Создать обработчики событий:

- для запуска таймера создать обработчик события OnCreate главной формы Form1. В окно кода между Begin … End вставить следующую строку кода программы: *Timer1timer(Timer1);i:=0;*

- для таймера создать обработку события OnTimer –вывод времени через каждую секунду (10 интервалов по 0,1 с) и вывод процесса подсчета десятых долей секунды:

  if (i mod 100) = 0 then Label1.Caption:=TimeToStr(Now);
  ProgressBar1.Position:=i;
  i:=i+10; if i>100 then i:=0;

  TimeToStr(Now) – функция перевода текущего времени в символьную форму.

7.Сохранить проект. Запустить программу.

Самостоятельное задание. Добавить компонент Gauge (вкладка **Samples**), который также может показывать течение процесса и отобразить с его помощью ход часов. Показать результат преподавателю.

Лабораторная работа 2
Вычисления с использованием действительных чисел. Маски для контроля ввода. Использование функций модуля Math

Создать программу, обеспечивающую ввод массива действительных чисел и работу с его значениями.

1.Создать форму (Form1) с тремя кнопками (Button), полем для ввода действительных чисел (компонент MaskEdit) и пятью полями для вывода надписей и чисел (Label).

2. Создать для компонента MaskEdit маску контроля ввода числа:

- щелкнуть по кнопке мыши в поле свойства EditMask и открыть диалоговое окно Input Mask Editor;
- в поле Input Mask ввести маску 000.00, разрешающую ввод в поле пяти цифр, две последние – дробная часть числа;
- в поле Character For Blanks ввести символ маски #, который будет показывать позиции для ввода цифр.

3.Написать обработчики событий для кнопок:

- «Ввод» – событие OnClick(после ввода 10 чисел кнопка исчезает):

  if** i<11 **then
  begin
  if** i<10 **then
  Label1.Caption:='Введите'+ IntToStr(i+1)+'-й элемент массива'
  ***else begin** Label1.Caption:= 'Все элементы введены';*

Button1.Enabled:=false; Button1.Visible:=false
end;
val(MaskEdit1.EditText,m[i],cod);
inc(i);
Label2.Caption:= Label2.Caption + ' ' + MaskEdit1.EditText;
*MaskEdit1.SetFocus;**end**;*

- «Максимум» – событие OnClick(вывод надписи и максимального значения):
  Label4.Caption:= floatToStr(MaxValue(m));.
- «Среднее» – событие OnClick:
  Label4.Caption:= floatToStr(Mean(m));

4.Для события формы OnCreate создать обработчик, задающий начальные установки:

i:=1;
Label1.Caption:='Введите '+ IntToStr(i)+'-й элемент массива';

5.Добавить в модуль формы Form1 ссылку для использования модуля Math:

Uses Math;

6.Описать массив и переменные в секции Var:

*i,cod:integer; m:**array** [1..10] **of** Double;*

7.Сохранить проект. Проверить его работу.

Самостоятельная работа. Добавить возможность изменения значений отдельных элементов массива и кнопку поиска минимального значения. Добавить кнопку, обеспечивающую деление элементов массива на максимальное число с выводом результирующего массива. Показать результат преподавателю.

Печать на принтере результатов. Использование модуля Printers

Создать программу, обеспечивающую вывод на принтер массива действительных чисел и его элемента, имеющего максимальное значение. Открыть проект, созданный в лабораторной работе 2, и добавить на форму кнопку «Печать»(Button5), и компонент диалога печати (PrintDialog1).

1.Добавить в список подключаемых модулей Uses модуль Printers.

2.Написать обработчик событий для кнопки «Печать»(Button5):

```
var f:Textfile; j:byte;s:string; //Объявление файловой переменной,
//счетчика и строки
begin
if PrintDialog1.Execute then //Вызов диалога печати
begin
AssignPrn(f); //Связывание файловой переменной и принтера
Try //Защищенный блок печати
Rewrite(f); //Открыть файл принтера на печать
Writeln(f,'Результат работы'); //Вывод строк на принтер
Writeln(f,'Массив');
for j:=1 to 10 do
s:= s + ' ' +floatToStr(M[j]);
Writeln(f,s);
Writeln(f,'Максимум');
s:= floatToStr(MaxValue(m));
Writeln(f,s)
finally
CloseFile(f); //Закрытие файла принтера
end;
end;
end;
```

3.Сделать кнопку «Печать» первоначально недоступной (Свойство *Enabled –false)* до ввода массива.

4.Сделать кнопку «Печать» доступной после ввода всего массива: поставить строку *Button5.Enabled:=true;* в обработчик кнопки «Ввод» рядом с выводом надписи «Все элементы введены».

5.Сохранить проект. Проверить его работу.

Самостоятельная работа. Добавить вывод фамилии разработчика. Показать напечатанный результат преподавателю.

Лабораторная работа 3
Работа с файлами. Размещение компонентов диалога на форме и задание их свойств

Создать программу для Windows, открывающую и корректирующую файл на диске.

1.Установить на форму компонент Memo (вкладка **Standartd**) для вывода и корректировки текста файла (Memo1).

2.Установить сверху два компонента Button (вкладка **Standard**) для открытия файла – “Открыть” (Button1) и сохранения результата – “Сохранить” (Button2).

3.Установить на форму компонент OpenDialog (вкладка **Dialogs**) для организации диалога открытия файла (OpenDialog1). Визуально этот компонент на форме работающей программы не виден, но его значок виден на макете формы при конструировании. Может быть в любом месте.

4.Установить свойства компонентов в окне **Object inspector** на вкладках **Properties**:

- значение свойства Caption (надпись) для компонентов Button изменить – вместо стандартных (повторяющих название компонента) занести необходимые надписи на русском языке. Для всей формы это свойство установит заголовок окна программы;
- значения свойств, обеспечивающих появление всплывающей подсказки ShowHint – True (выбрать из раскрывающегося списка) и Hint (текст всплывающей подсказки) для компонентов.

5.Очистить поле компонента Memo – два щелчка кнопкой мыши на поле свойства Lines. Откроется окно текста String list editor. Его содержимое нужно стереть.

6.Создать обработчики событий в окне **Object inspector** на вкладках **Events**.

- Для кнопки Button1 создать обработку события OnClick (реакция на щелчок мыши) – запуск диалога открытия файла и чтение текста в поле Lines компонента Memo:

  If** OpenDialog1.Execute **then

  Memo1.Lines.LoadFromFile(OpenDialog1.FileName); (метод для чтения информации). Для автоматизации набора в окне кода программы можно использовать темплеты редактора (список вызывается комбинацией клавиш *Ctrl+J,* а вставка из списка – выделить и нажать *Enter*) и список методов и полей (вызывается *Ctrl+пробел*).
- Для кнопки Button2 создать обработку события OnClick–сохранение файла из поля Lines компонента Memo в файл:

  Memo1.Lines.SaveToFile(OpenDialog1.FileName);.
- Для главной формы Form1 создать обработчик,закрывающий файл при окончании работы формы OnDestroy: *Close*.

7.Сохранить проект. Создать с помощью стандартной программы «Блокнот» тестовый файл из нескольких строк, который будет открывать программа. Запустить программу.Открыть с ее помощью тестовый файл и внести в него коррективы. Показать результат преподавателю.

Усовершенствование программы. Дополнить проект следующими свойствами и компонентами.

1.Для вывода в заголовке окна имени обрабатываемого файла в обработчик события OpenDialog1Close добавить строку кода:

Caption:='Редактор файлов'+ExtractFileName(OpenDialog1.FileName);

2.Увеличить поле Memo на все окно формы. Для этого выбрать из списка значение свойства Align – alClient (вместо alNone). Кнопки управления окажутся внутри поля Memo.

3.Чтобы выделить для кнопок специальную зону вверху добавить в окно формы компонент Bevel (вкладка **Additional**) и установить для его свойства Align значение alTop.

4.Включить для поля Memo полосы прокрутки, установив для свойства ScrollBars значение ssBoth.

5.Добавить третью кнопку сохранения файла с другим именем "Сохранить как…". Для обеспечения диалога сохранения добавить компонент SaveDialog. Для третьей кнопки написать обработчик события OnClick:

```
if SaveDialog1.Execute then
begin
Memo1.Lines.SaveToFile(SaveDialog1.FileName);
Caption:='Редактор файлов' +
ExtractFileName(SaveDialog1.FileName);
end;
```

Код включает строку изменения имени в заголовке окна.Проверить работу и сохранить тестовый файл с другим именем.

6.Добавить четвертую кнопку изменения шрифта "Шрифт". Для подключения диалога изменения шрифта добавить компонент FontDialog. Для кнопки обработчик события OnClick будет:

```
if FontDialog1.Execute then
Memo1.Font:=FontDialog1.Font;
```

а для обработчика события OnApplay компонента FontDialog

```
Memo1.Font:=FontDialog1.Font;
```

Проверить работу и показать результат преподавателю.

Самостоятельное задание. Разработать необходимые изменения для режима “Новый файл”: ввод текста без открытия существующего файла. Добавить запрос на сохранение файла при закрытии окна, если в текст были внесены изменения. Для этого создать обработчик события формы CloseQuery, вызывающий окно диалога в случае внесения изменений в текст:

canclose:=messagedlg (“Сохранить?”,mtConfirmation,[mbYes,mbNo] ,0)=mrNo;

и обработчик события поля Memo Change, фиксирующий такое событие. Предусмотреть сброс фиксации изменения при сохранении. Показать результат преподавателю.

Создание меню и связывание пунктов меню с обработчиками событий

Добавить к программе, разработанной на прошлом занятии, верхнее меню. Меню должно содержать пункты: «*Файл*» и «*Редактирование*».Пункт «*Файл*» должен содержать подпункты: «*Новый*», «*Открыть*», «*Сохранить*», «*Сохранить как ...*» и «*Закрыть редактор*».

Пункт «*Редактирование*» должен содержать подпункты: «*Копировать*», «*Вырезать*», «*Вставить*» и «*Шрифт*».

1. Установить на форму компонент MainMenu (вкладка **Standard**) (MainMenu1). Визуально этот компонент при конструировании на форме может быть в любом месте. Само меню появится сверху, когда будут описаны его пункты.

2.Щелкнуть кнопкой мыши на значении свойства Items [Menu] компонента MainMenu1и на […] (в конце строки значения) для открытия вспомогательного окна, в котором создаются пункты меню.

3.Набрать в окне названия всех пунктов меню согласно перечню и закрыть вспомогательное окно. Для установки значений «горячих» клавиш (Hot key) перед первыми буквами добавить знак &. В код программы добавятся объекты – пункты меню, имеющие по умолчанию имена, состоящие из буквы N и номера (в порядке задания пунктов) для нашего случая от N1 до N11 (эти имена можно увидеть и выбрать из списка в окне **Object inspector**).

4.Установить в окне **Object inspector** для пунктов меню, совпадающих по назначению с кнопками («*Новый*», «*Открыть*» и т.д.)на вкладках **Events** ссылки на имеющиеся обработчики событий кнопок, выбирая их из раскрывающегося списка события OnClick.

5.Создать обработчики событий для пунктов меню «*Редактирование*» **:**

- для пункта меню «*Копировать*»: *Memo1.CopyToClipboard*;
- для пункта меню «*Вырезать*»: *Memo1.CutToClipboard*;
- для пункта меню «*Вставить*»: *Memo1.PasteFromClipboard*;
- для общего пункта «*Редактирование*» создать обработку события OnClick–установку активности пунктов меню:
  n8.Enabled:=Memo1.SelLength>0; (пункт «*Копировать*»)
  n9.Enabled:=n8.Enabled; (пункт «*Вырезать*»)
  n11.Enabled:=n8.Enabled; (пункт «*Шрифт*»)
  n10.Enabled:=clipboard.hasformat(cf_text); (пункт «*Вставить*»)
- для всех подпунктов меню «*Редактирование*» установить начальные значения свойств Enable - False.

6.Подключить в строку Uses кода программы модуль работы с буфером обмена. Его имя - ClipBrd.

7.Сохранить проект. Запустить программу. Открыть с ее помощью тестовый файл и проверить работу меню. Компонент Memo автоматически подключает и стандартное для текстового редактора контекстное всплывающее меню. Попробовать, как оно работает. Показать результат преподавателю.

Самостоятельное задание. Добавить в меню «*Редактирование*» пункт «*Выделить все*». Написать для него обработчик события, самостоятельно найдя соответствующий метод для компонента Memo. Показать результат преподавателю.

Самостоятельная работа. Создать программу просмотра изображений.

По аналогии с прграммой – редактором текста создать программу, обеспечивающую просмотр графических изображений, хранящихся в файлах. Вывод изображений производится с помощью компонента Image. Программа должна иметь верхнее меню:

- открыть рисунок;
- сохранить рисунок с другим именем
- конец работы.

Для инструментальных кнопок, повторяющих пункты меню, использовать компоненты SpeedButton с пиктограммами, которые устанавливаются в качестве свойства Gluph и выбираются из папки *Program Files\Common Files\Borland Shared\Images\Buttons*.

В обработчиках событий меню и кнопок использовать компоненты работы с графическими файлами OpenPictureDialog и SavePictureDialog.

Для загрузки и извлечения графической информации использовать соответствующие методы компонента Image.

Сохранить проект. Запустить программу. Открыть с ее помощью графический файл и проверить работу меню. Показать результат преподавателю.

Печать содержимого текстового поля компонента Memo

Добавить к проекту ЛР3, кнопку печати текста, набранного в поле ввода компонента Memo.

1.Добавить на форму две кнопки: (Button) «*Печать*» и «*Настройка печати*».

2.Добавить на форму два невизуальных компонента PrintDialog и PrintSetupDialog.

3.Подключить модуль Printers, добавив его имя в объявление Uses.

4.Создать обработчики событий для кнопок **:**

- «*Настройка печати*»:
  PrinterSetupDialog1.Execute;
- «*Печать*»:

```
var f:Textfile; j:integer;
begin
if PrintDialog1.Execute then
begin
AssignPrn(f);
try
Rewrite(f);
printer.canvas.Font:=Memo1.Font;
for j:=0 to Memo1.Lines.Count -1 do
Writeln(f,Memo1.Lines[j]);
finally
```

```
 CloseFile(f);
 end;
end;
end;
```

Сохранить проект. Запустить программу. Открыть с ее помощью тестовый файл инапечатать его. Показать результат преподавателю.

Самостоятельное задание. Добавить в меню лабораторной работы 3 пункты, повторяющие кнопки печати. Показать результат преподавателю.

Автоматизация создания форм приложения с главным меню и панелью инструментов (использование Application Wizard)

Система программирования Delphi имеет специальное средство Application Wizard, позволяющее легко создать форму и подключить к ней верхнее меню и панель инструментов, дублирующую пункты меню. Для запуска Application Wizard нужно выбрать пункт меню *File->New* и в появившемся окне New Items перейти на вкладку Projects, выделить пиктограмму Application Wizard и щелкуть по кнопке OK. Повторим создание текстового редактора лабораторной работы 3.

В первом диалоговом окне установить, какие пункты меню нужно включить в создаваемое приложение (File и Edit) и щелкуть по кнопке Next.

Во втором диалоговом окне предлагается задать фильтры для отбора определенных типов файлов по их расширению (в лабораторной работе они не устанавливались). Задать два расширения :

- *.txt –текстовые файлы;
- *.* - все файлы.

В третьем диалоговом окне можно выбрать кнопки для панели инструментов. Для включения выделить нужную кнопку и щелкуть по кнопке Insert. Для отделения функциональных групп кнопок используется кнопка Space.

В четвертом окне задается имя создаваемого файла приложения, папка для его размещения, подключение всплывающих подсказок. Щелкуть по кнопке Finish и получить заготовку модуля с комментариями о том, какие обработчики и куда вставить. Кроме того, в папке проекта появляется файл с расширением .txt, в который Application

Wizard помещает текст установок свойств и описания объектов, созданных автоматически и включенных в код проекта для реализации спроектированной формы.

Добавить необходимые обработчики самостоятельно. Также установить компонент Memo. Внести коррективы в название окна формы. Задать тексты всплывающих подсказок на русском языке. Сохранить проект и проверить его работоспособность.

Еще более быстро можно получить заготовку текстового редактора, выбрав на вкладке Projects пиктограмму Win95/98 Logo Application. Это готовый шаблон, который использует компонент RichEdit, имеющий большие возможности, чем Memo.

Лабораторная работа 4
Работа со строками. Создание пользовательского модуля

Создать программу для Windows, которая вводит фразу и формирует список слов, из которых она состоит.

1.Установить на форму компонент Memo (вкладка **Standard**) для вывода и корректировки текста (Memo1).

2.Установить компонент Button (вкладка **Standard**) для переноса слов в список (Button1).

3.Установить на форму компонент ListBox (вкладка **Standard**) для организации списка слов (ListBox1).

4.Установить свойства компонентов в окне **Object inspector** на вкладках **Properties**.

- Изменить стандартное значение свойства Caption (надпись) для формы (повторяющее название компонента Form1) на русское “Список слов”.
- Установить значения свойств, обеспечивающих появление всплывающей подсказки ShowHint – True (выбрать из раскрывающегося списка) и Hint (текст всплывающей подсказки) для компонентов.
- Очистить поле компонента Memo1– два щелчка кнопкой на поле свойства Lines. Перейти в окно текста String list editor и стереть содержимое.

5.Добавить к проекту новый модуль: выбрать из меню пункт *New*… и в диалоговом окне на вкладке New щелкнуть по пиктограмме

Unit. В окне Code editor появится новая вкладка Unit2, и в текст проекта будут автоматически вставлены ссылки на этот модуль. Если имеется уже готовый модуль, его можно подключить к проекту, выбрав из меню *Project -> Add to project*…

6.В модуль Unit2 поместить код функции, выделяющей из текста слово с заданным номером:

```
interface
     //описание функций и процедур доступных
     // функциям и процедурам, использующим этот модуль
     // выделяет из строки слово с указанным номером
    // слово - последовательность символов между пробелами
Function GetSubStr3(st:string;  { строка }
                    n:integer) { номер слова }
                    :string;     { слово или ' ' }
 implementation // реализация процедур и функций модуля
Function GetSubStr3(st:string;n:integer):string;
var
  p:integer; i:integer;
begin
For i:=1 to n-1 do
begin
p:=pos(" ",st);
if (p>0)        then
st:=copy(st,p+1,Length(st)-p)
                else st:='';
//если в начале оставшейся части строки есть пробелы, удалим их
while (pos(" ",st)=1)and (length(st)>0) do
delete(st,1,1);
end;
p:=pos(" ",st);
if  p <> 0
    then result:=copy(st,1,p-1)
    else  result:=st ;
end;
end.
```

7.Для заполнения списка в момент нажатия кнопки создать обработчик события OnClick:

```
var i:integer;s:string;
begin
for i:=1 to 10000 do begin
s:=Getsubstr3(Memo1.text,i);
if  s='' then  Break
        else
ListBox1.Items.Add(s);
end;
```

8.Сохранить проект. Запустить программу.

Самостоятельное задание. Изменить функцию модуля так, чтобы она находила и другие разделители слов: точку, запятую ит.п. Добавить в модуль новую функцию, обеспечивающую переворот слова (последняя буква становится первой, а первая – последней). Используя эту функцию, доработать программу так, чтобы при выборе слова в списке его перевернутое написание выводилось ниже списка. Показать результат преподавателю.

Лабораторная работа 5
Создание программы построения изображений. Использование мыши при работе программы

Создать программу, обеспечивающую построение графических примитивов: линий, поямоугольников и эллипсов с помощью мыши. Предусмотреть изменение толщины линии и цвета закраски фона.

1.Установить на форму компоненты: PaintBox (вкладка **System**), RadioGroup с тремя кнопками RadioButton (вкладка **Standard**) для выбора примитива, два компонента Label (вкладка **Standard**) для вывода надписей “Толщина линии” и “Цвет FG – линии; BG – фон”, SpinEdit для ввода толщины линии и ColorGrid для выбора цвета (вкладка **Samples**). Цвет линии (ForeGround – FG) выбирается щелчком левой кнопки мыши, а фона (BackGround – BG) – правой кнопкой.

2.Установить свойства SpinEdit MaxValue=20 и MinValue=1, определяющие пределы изменения толщины линии (1 – 20 пикселов). Начальное значение толщины линии - свойство Value установить равным 1.

3.Установить для свойства Color компонента PaintBox значение clWhite для начального вывода белого фона рисунка.

4.Описать в секции Var переменные: nt:Tpoint - для запоминания координат курсора мыши; i: integer – для установки типа выбранного графического примитива (0 – линия, 1 - эллипс, 2 – прямоугольник); f – логическую переменную для начального вывода белого поля рисунка.

5.Создать обработчики событий:

- для главной формы OnActivate – начальная установка логической переменной

  f:=true;
- для кнопок выбора графического примитива событие OnClick с присвоением переменной i соответствующего значения: *i:=0; i:=1; i:=2;*
- для компонента PaintBox: событие OnPaint – начальный вывод поля рисунка

  if f **then**
  with *SENDER* ***AS*** *TPaintBox, Canvas* ***do***
  begin
  Brush.Color:=Color;
  FillRect(ClientRect);
  f:=false;
  end*;*
- для события OnMouseDown – нажатие кнопки мыши в начальной точке примитива и запоминание координат этой точки (в конечную точку мышь перемещаем, не отпуская нажатой кнопки):

  nt.x:=x; nt.y:=y;
  with *SENDER* ***AS*** *TPaintBox, Canvas* ***do***
  PaintBox1.Canvas.MoveTo(x,y);
- для события OnMouseUp – отпускание кнопки мыши в конечной точке графического примитива и вывод его:

  with *SENDER* ***AS*** *TPaintBox, Canvas* ***do begin***
  case *i* ***of***
  0 :PaintBox1.Canvas.LineTo(x,y) ;
  1 :PaintBox1.Canvas.Ellipse(nt.x,nt.y,x,y);
  2: PaintBox1.Canvas.Rectangle(nt.x,nt.y,x,y);
  end*;*
  end*;*

- для компонента ColorGrid обработчик события OnClick – изменение установки цвета:

  PaintBox1.Canvas.Pen.Color:=ColorGrid1.ForegroundColor;

  PaintBox1.Canvas.Brush.Color:=ColorGrid1.BackgroundColor;

  Не анализируется, какой кнопкой мыши был сделан щелчок, а переставляются оба значения.
- для компонента SpinEdit – изменение значения толщины линии OnChange:

  PaintBox1.Canvas.Pen.Width:= SpinEdit1.Value;

6.Сохранить проект. Проверить его работу. Показать результат преподавателю.

Самостоятельная работа. Заменить компонент PaintBox на компонент Image и добавить к программе кнопки сохранения файла рисунка и чтения файла. Компонент Image также имеет свойство Canvas, позволяющее рисовать, как и при использовании компонента PaintBox. Но компонент PaintBox не имеет методов чтения – записи файла. Показать результат преподавателю.

Лабораторная работа 6
Создание программы построения графиков

Создать программу, обеспечивающую построение графиков.

1.Установить на форму компоненты: Chart (вкладка **Additional**) и Button (вкладка **Standard**) для запуска построения графика.

2.Изменить свойства кнопки и главной формы для получения надписей “Нарисовать” и “Построение графиков”.

3.Вызвать правой кнопкой мыши контекстное меню компонента Chart и выбрать пункт меню Edit Chart. В открывшемся диалоговом окне Editing Chart1 на вкладке Series щелкнуть по кнопке Add для добавления серии данных Series1 и во вспомогательном окне TeeChartGallery выбрать для этих данных тип графика Line. С помощью кнопки Title задать надписи на поле чертежа, характеризующие данные (так называемые Legends – “Легенды”). К объектам программы автоматически добавятся три набора Series1, Series2, Series3 для построения трех кривых на одном графике.

4.Создать обработчики событий:

- для кнопки рисования графика OnClick , описав переменную i в секции Var

```
With Series1 do
Begin
    i:=1;While i<= 12 do begin
   Add(  i, IntToStr(i) ,  clBlue); inc(i);
   Add(  i-5,IntToStr(i) ,  clBlue );inc(i) ;
   Add(  2*i,IntToStr(i) ,clBlue  ) ;inc(i);
end;
end;
With Series2 do
Begin
   for i:=1 to 10 do
   Add(  i*i, IntToStr(i) , clRed ) ;
end;
With Series3 do
Begin
   for i:=1 to 10 do
   Add( 100-10*i, IntToStr(i) , clGreen ) ;
end;
```

Первая серия строит ломаную линию, меняя формулу для трех соседних точек, вторая – $\mathbf{i^2}$ и третья – **100-10*i.** Для добавления данных используется метод Add компонента Series.

Первый параметр – значение функции, второй – текст, который пишется около делений горизонтальной оси, третий задает цвет линии графика.

5.Сохранить проект. Проверить его работу.

Самостоятельная работа. Изменяя свойства компонента Chart, изучить их влияние на представление данных. Построить самостоятельно график математической функции, заданной преподавателем. Показать результат преподавателю.

Лабораторная работа 7
Создание программы, включающей анимационный компонент изображения. Использование шкалы и дополнительного окна для установки параметров программы

Создать программу, обеспечивающую управление анимацией с помощью мыши (запуск и установка числа повторений).

1.Установить на форму компоненты: Animate и TrackBar (вкладка **Win32**), две кнопки Button (вкладка **Standard**) для запуска анимации и вызова диалогового окна, три компонента Label (вкладка **Standard**) для вывода надписей по краям шкалы, указывающих границы изменения числа повторений, и выбранного движком действующего значения.

2. Установить начальные значения свойств компонента TrackBar Max=20 и Min=1, определяющие пределы изменения величины свойства Position при перемещении движка шкалы. Свойство Frequency =1 определяет цену деления шкалы.

3.Установить для компонента Animate свойство FileName: *c:\Program Files\Delphi5\Demos\Coolstuf\cool.avi* – файл анимации из демонстрационных примеров, входящих в Delphi5 (при отсутствии этого файла выбрать один из стандартных клипов Windows, изменив значение свойства CommonAVI с *aviNone* на любое другое из раскрывающегося списка возможных значений и оставив пустым поле свойства FileName).

4.Скорректировать свойства других компонентов, обеспечив соответствующие надписи.

5.Добавить второе диалоговое окно в программу.

- Выбрать пункт меню *File->New…*
- В диалоговом окне File Items перейти на вкладку Dialogs, и выделить пиктограмму стандартного диалога Standard Dialog.
- Проверить установку режима Copy (переключатель внизу), чтобы не испортить шаблон и щелкнуть ОК. В проект добавится вторая форма с именем OKBottomDlg и второй модуль Unit2.pas.
- Для использования этого модуля в код модуля главного окна Unit1 нужно добавить ссылку на подчиненный модуль *Uses Unit2* (иначе свойства компонентов вспомогательного окна будут недоступны из обработчиков событий модуля Unit1).
- Установить на диалоговое окно два компонента SpinEdit (вкладка **Samples**) для ввода значений границ диапазона и два компонента StaticText (вкладка **Additional**) для вывода текста.

6.Создать обработчики событий.

6.1.Для компонентов диалогового окна (в модуле Unit2):

- для кнопки ОК событие OnClick:

  ModalResult := mrOK;
- для кнопки Cansel событие OnClick:

ModalResult := mrCancel;

устанавливающие код возврата.

6.2. Для компонентов главной формы (в модуле Unit1):

- для кнопки запуска анимации «*Пуск*» событие OnClick:

  Animate1.Repetitions:=TrackBar1.Position;

  Animate1.Active:=true;
- для компонента TrackBar событие OnChange– изменение числа повторов анимации:

  Label3.Caption:='Повторов '+ IntToStr(TrackBar1.Position);
- для кнопки «*Изменить*», вызывающей диалоговое окно изменения диапазона, и изменяющей диапазон, если нажата кнопка ОК при выходе из диалогового окна, событие OnClick:

```
If OkBottomDlg.ShowModal = mrOK then begin
TrackBar1.Min:=OkBottomDlg.SpinEdit1.Value;
TrackBar1.Max:=OkBottomDlg.SpinEdit2.Value;
Label1.Caption:=IntToStr(TrackBar1.Min);
Label2.Caption:=IntToStr(TrackBar1.Max);
Label3.Caption:='Повторов '+ IntToStr(TrackBar1.Position);end;
```

Метод *ShowModal* не дает продолжить работу программы без закрытия окна.

7.Сохранить проект. Проверить его работу. Показать результат преподавателю.

Самостоятельная работа. Создать обработчик события, который выводит по окончании анимации вместо числа повторений текст «Анимация окончена. Выберите новое число повторов». Проверить работу измененной программы. Показать результат преподавателю.

Лабораторная работа 8
Знакомство с программированием движущихся объектов – анимация

Создать программу для Windows, показывающую изображение, движущееся по форме. Движение имитируется циклом: нарисовать – стереть – нарисовать с небольшим сдвигом и т.д. Окно формы во время создания пустое. Изображение появляется в процессе выполнения программы.

1.Установить на форму компонент Timer (вкладка **System**) для отсчета интервалов времени: Timer1.

2.Установить заголовок формы *«плывущий кораблик»* в окне **Object inspector** на вкладках **Properties**.

3.Добавить переменные x y: integer для сохранения координат опорной точки рисунка в секцию Var модуля Unit1.

4.Создать обработчики событий.

- Для запуска таймера создать обработчик события FormActivate главной формы Form1. В окно кода между Begin … End вставить следующие строки кода программы: *x:=0; y:=100;*

form1.color:=clNavy{цвет формы};

timer1.Interval:=50;{скорость движения, чем меньше интервал, тем быстрее движется кораблик}

- Для таймера создать обработку события OnTimer – вывод изображения:

ris(x,y,form1.color); {*стереть существующий на экране рисунок, рисуя его еще раз поверх исходного цветом фона*}

x:=x+5;{*переместить изображение по оси x*}

ris(x,y,clWhite); {*нарисовать в новой точке белым цветом*}

if *x > ClientWidth* ***then*** *x:=0;* {*изображение ушло за границу формы*}

5.Написать процедуру рисования в секции **implementation** модуля

```
procedure ris(x,y: integer;   {координаты базовой точки}
              color: TColor); {цвет корабля}
var  buf: TColor;
begin
with form1.canvas do
begin
   buf:=pen.Color;   {сохраним текущий цвет}
   pen.Color:=color; {установим нужный цвет}
   { рисуем ...}    {корпус}
   MoveTo(x,y);{переход в точку}
   LineTo(x+50,y);{линия из текущей в заданную точку}
   LineTo(x+55,y-10);
   LineTo(x,y-10);
   LineTo(x,y);
   { надстройка }
   Rectangle(x+15,y-15,x+45,y-10);{прямоугольник задается диагональными точками}
```

```
        {капитанский мостик}
        Rectangle(x+40,y-20,x+45,y-15);
        { труба}
        Rectangle(x+25,y-25,x+30,y-15);
        { иллюминатор}
       Ellipse(x+42,y-6,x+47,y-1);{Окружность(эллипс) задается точками описанного прямоугольника}
        { мачта }
        MoveTo(x+40,y-20);
        LineTo(x+40,y-35);
        pen.Color:=buf; {восстановим старый цвет карандаша}
    end;
    end;
```

6.Сохранить проект. Запустить программу. Показать результат преподавателю.

Самостоятельная работа. Сделать так, чтобы кораблик ходил туда и обратно. Показать результат преподавателю.

Лабораторная работа 9
Знакомство с программированием движущихся объектов – мультипликация

Создать программу для Windows, показывающую изображение, состоящее из ряда ранее подготовленных кадров в формате bmp (файл film.bmp получить у преподавателя). В картинке bmp кадры одинакового размера ставятся один за другим слева направо по оси X. По оси У картинка имеет высоту кадра. Движение имитируется циклом: нарисовать кадр – стереть повторным выводом пустого кадра – нарисовать следующий кадр т.д. Окно формы содержит кнопку запуска. Изображение появляется после щелчка кнопкой мыши по ней.

1.Установить на форму компоненты Timer для отсчета интервалов времени: Timer1 и кнопку запуска Button1 (вкладка **Standard**).

2.Установить свойства компонента в окне **Object inspector** на вкладках **Properties**:

- для формы, заголовок – *Фильм***;**
- для кнопки надпись *Пуск;*
- для таймера свойство *Interval* установить 100 (скорость воспроизведения; она тем меньше, чем больше интервал).

3.Задать константу числа кадров:

```
const
  N_KADR=5; // кадров в фильме
```

4.Добавить переменные в декларацию Var модуля Unit1:

```
Film: TBitMap;       // фильм - все кадры
Kadr: TBitMap;       // текущий кадр
WKadr,HKadr: integer; // ширина и высота кадра
CKadr: integer;      // номер текущего кадра
RectKadr: TRect;     // положение и размер кадра в фильме
Back: TBitmap;//пустой кадр
WorkBitMap: Tbitmap;//рабочие области
```

5.Создать обработчики событий.

- Обработчик события FormActivate главной формы Form1:

```
Film:=TBitMap.Create;
Film.LoadFromFile("film.bmp");
WKadr:=Round(Film.Width/N_Kadr);
HKadr:=Film.Height;
Kadr:=TBitMap.Create;
Kadr.Transparent:=True;
Kadr.Width:=WKadr;
Kadr.Height:=HKadr;
 Back:=TBitMap.Create;
Back.Width:=WKadr;
Back.Height:=HKadr;
WorkBitMap:=TBitMap.Create;
WorkBitMap.Width:=WKadr;
WorkBitMap.Height:=HKadr;
Ckadr:=0;
```

- Для таймера обработчик события OnTimer –вывод изображения:

```
DrawKadr;
```

- Обработчик события OnClick кнопки:

```
Back.Canvas.CopyRect
(Bounds(0,0,WKadr,HKadr),Form1.canvas,Bounds(0,0,WKadr,HKadr));
    Form1.Timer1.Enabled:=True;
```

6.Написать процедуру рисования кадра в секции **implementation** модуля:

```
procedure DrawKadr;
begin
```

```
WorkBitmap.Canvas.Draw(0,0,Back);//пустой кадр в рабочую область
// определим положение текущего кадра в фильме
RectKadr:=Bounds(WKadr*CKadr,0,WKadr,HKadr);
// выделим кадр из фильма
Kadr.Canvas.CopyRect(Rect(0,0,WKadr,HKadr),Film.Canvas,RectKadr);
// отрисуем кадр на рабочей поверхности
WorkBitmap.Canvas.Draw(0,0,Kadr);
// выведем кадр на форму
Form1.Canvas.Draw(10,10,WorkBitmap);
// подготовимся к выводу следующего кадра
CKadr:=CKadr+1;
if CKadr = N_KADR
then CKadr:=0;
end;
```

7.Сохранить проект. Запустить программу. Показать результат преподавателю.

Самостоятельная работа. Сделать в редакторе Paint другой набор кадров для своего фильма. Показать результат преподавателю.

Лабораторная работа 10

Создание программы, включающей многостраничный и табличный компоненты. Сохранение и чтение с диска данных для расписания

Создать программу, обеспечивающую заполнение, просмотр и сохранение расписания на вкладках, соответствующих дням недели.

Установить на форму компоненты: TabControl (вкладка **Win32**), а на него StringGrid (вкладка **Additional**) для таблицы расписания. Добавить компоненты OpenDialog и SaveDialog (вкладка **Dialogs**) для открытия и сохранения на диске файла расписания.

1.Установить для компонента TabControl начальное значение свойства Tabs – набрать строки названий дней недели для ярлычков в окне String List Editor (окно появляется при щелчке по кнопке с многоточием в поле данных Tabs).

2.Изменить заголовок окна формы на «Расписание».

3.Свойство Align для компонентов – *alClient*, чтобы расписание занимало все поле формы.

4.Установить для компонента StringGrid свойства:

- ColCount (число столбцов) = 4 и RowCount (число строк) =7;
- Группа свойств Options (раскрыть щелчком кнопки мыши на знаке (+) – свойство goEditing –True (можно редактировать содержание ячеек), свойство goColSizing – True (можно менять размеры колонок, перетаскивая границы мышью).

5.Описать в секции **Type** модуля программы тип массива для размещения текущего состояния расписания и сохранения его на диске – трехмерный массив *r=**array** [0..6,1..7,1..3] **of string**[40];* первый индекс – день недели, соответствует номеру вкладки (свойство TabIndex компонента TabControl, нумерация начинается с 0), второй – номер занятия и третий – нумерация ячеек в строке (1 – предмет, 2 – аудитория, 3 – фамилия преподавателя).

6.Добавить в секцию **Var** описание массива типа расписание*: rasp:r* и файловой переменной *f: **file of** r* ;.

7.Создать обработчики событий.

7.1.Главная форма:

- для вывода заголовков строк и столбцов при создании формы – обработчик события OnCreate:

```
with StringGrid1 do begin
Cells[1,0]:='Предмет';
Cells[2,0]:='Аудитория';
Cells[3,0]:='Преподаватель';
Cells[0,1]:='8-30';
Cells[0,2]:='10-10';
……..
Cells[0,7]:='18-30';
end;
```

- для чтения с диска расписания – обработчик события OnActivate, организующий диалог запроса наличия файла, открытие файла и чтение расписания с диска в массив *rasp*:

```
var i,j:integer;
begin
SaveDialog1.FileName:='Расписание.ras';
if MessageDlg
('Естьфайл расписания?',mtConfirmation,[mbYes,mbNo],0)=mrYes
then
        begin
```

```
        if OpenDialog1.Execute then begin
        SaveDialog1.FileName:=OpenDialog1.FileName;
        AssignFile(f,OpenDialog1.FileName);
        Reset(F);
        read(f,rasp);
        CloseFile(f);
        with StringGrid1 do begin
        for i := 1 to 7 do
        for j := 1 to 3 do
        cells[j,i]:=rasp[TabControl1.TabIndex,i,j];
                        end;
                            end;
        end;
    end;
```

- для сохранения на диске расписания перед выходом из программы обработчик события OnCloseQuery:

```
    var i,j:integer;
    begin
    if MessageDlg
    ("Сохранить расписание?", mtConfirmation, [mbYes,
mbNo], 0)=mrYes
    then
        begin
        with StringGrid1 do begin
        for i := 1 to 7 do
        for j := 1 to 3 do
        rasp[TabControl1.TabIndex,i,j]:=cells[j,i];
                        end;
        if SaveDialog1.Execute then begin
        AssignFile(f,SaveDialog1.FileName);
        Rewrite(F);
        write(f,rasp);
                                end;
                        end;
    end;
```

7.2.Компонент TabControl:

- для сохранения содержимого страницы при смене вкладки перед переходом на новую страницу обработчик события OnChanging:

```
 var i,j:integer;
begin
with StringGrid1 do begin
        for i := 1 to 7 do
        for j := 1 to 3 do
        rasp[TabControl1.TabIndex,i,jj]:=cells[j,i];
                end;
end;
```

- для чтения содержимого очередной страницы из файла *rasp* обработчик события OnChange:

```
var i,j:integer;
begin
with StringGrid1 do begin
        for i := 1 to 7 do
        for j := 1 to 3 do
        cells[j,i] :=rasp[TabControl1.TabIndex,i,jj];
                end;
end;
```

8. Сохранить проект. Проверить его работу.

Самостоятельная работа. Исследовать другие свойства компонентов TabControl и StringGrid. Изменить цвет таблицы расписания и линии разграничения ячеек. Показать результат преподавателю.

Лабораторная работа 11

Работа со списком. Получение списка шрифтов, установленных на компьютере, и демонстрация их формы в текстовом поле

Создать программу для Windows, формирующую список установленных шрифтов. Выбор шрифта из списка меняет очертания букв в окне формы. При изменении размера должна изменяться высота букв.

1.Установить на форму компонент RichEdit (вкладка **Win32**) для вывода и корректировки текста (RichEdit1).

2.Установить два компонента Label (вкладка **Standard**) для надписей: “Выбор шрифта” (Label1) и “Высота шрифта” (Label2).

3.Установить на форму компонент ComboBox (вкладка **Standard**) для организации списка шрифтов (ComboBox1).

4.Установить на форму компонент SpinEdit(вкладка **Samples**) для изменения высоты шрифта (SpinEdit1).

5.Установить свойства компонентов в окне **Object inspector** на вкладках **Properties**.

- Изменить значение свойства Caption (надпись) для формы –- вместо стандартного (повторяющего название компонента Form1) задать русское “Демонстратор шрифта”.
- Установить значения свойств, обеспечивающих появление всплывающей подсказки ShowHint – True (выбрать из раскрывающегося списка) и Hint (текст всплывающей подсказки) для компонентов.
- Заполнить поле компонента RichEdit1 – два щелчка на поле свойства Lines. Перейти в окно текста String list editor и набрать содержимое: цифры, прописные и строчные буквы латинского и русского алфавита. Установить значение свойства Aline равным alTop, чтобы окно шрифта было прижато к верхнему краю формы.
- Установить для компонента SpinEdit1 значения: минимальное 8, максимальное 72, начальное 20 (свойства MinValue, MaxValue и Value).

6.Создать обработчики событий в окне **Object inspector** на вкладках **Events**.

- Для заполнения списка шрифтов в момент создания формы создать обработку события OnCreate для Form1:

  ComboBox1.Items := Screen.Fonts;

  RichEdit1.Font.size:=20;
- Для выбора в списке шрифта создать обработку события OnClick компонента Combobox1:

  RichEdit1.Font.Name := ComboBox1.Items[ComboBox1.ItemIndex];

  RichEdit1.Font.size:=SpinEdit1.Value;
- Для изменения размера шрифта создать обработку события OnChange компонента SpinEdit1:

  RichEdit1.Font.size:=SpinEdit1.Value;

  7.Сохранить проект.. Запустить программу.

Самостоятельное задание. Добавить кнопки демонстрации курсива и жирного начертания шрифта. Предусмотреть фиксацию и сброс фиксации начертания. Показать результат преподавателю.

Создание статусной строки и вывод в нее справочной информации

Создать для программы, разработанной в лабораторной работе 11, статусную строку, содержащую две панели с названием шрифта и выбранного стиля. Выбор шрифта из списка меняет надпись в левой части панели. При изменении стиля должна изменяться надпись в правой части панели

1.Установить на форму компонент StatusBar (вкладка **Win32**) статусную строку для вывода и корректировки справочного текста (StatusBar1).

2.Компонент Button1 (вкладка **Standard**) для переключения стиля текста “Полужирный” должна быть установлена при выполнении самостоятельного задания к лабораторной работе 11.

3.Установить свойства компонентов в окне **Object inspector** на вкладках **Properties**.

- Установить структуру панели StatusBar1, состоящей из двух частей: сделать два щелчка кнопкой мыши на поле свойства Panels, для того чтобы перейти в окно редактора панели Editing StatusBar1.Panels, и установить с помощью кнопки Add New две подпанели с номерами 0 и 1 типа TSatusPanel.
- Выделить подпанель 0 щелчком кнопки мыши и изменить ее свойство Width (ширина) на 250 (по умолчанию более узкая 50, а последняя панель занимает всю оставшуюся ширину статусной строки).

4.Добавить в обработчики событий, созданные в предыдущей лабораторной работе, следующие строки.

- Для установки надписи исходного стиля в момент создания формы в обработку события OnCreate:

  StatusBar1.Panels[1].Text:= ’Выбран стиль НОРМАЛЬНЫЙ’;
- Для события OnClick компонента Combobox1:

  StatusBar1.Panels[0].Text:= ’Выбран шрифт ’ + RichEdit1.Font.Name;
- Для компонента Button1 в обработку события OnClick или в соответствующие ветви программы, переключающие значения стиля текста:

  StatusBar1.Panels[1].Text:= ’Выбран стиль ПОЛУЖИРНЫЙ’;
  StatusBar1.Panels[1].Text:= ’Выбран стиль НОРМАЛЬНЫЙ’;

5.Сохранить проект. Запустить программу.

Самостоятельное задание. Добавить для кнопки демонстрации курсива третью подпанель и обеспечить вывод в нее соответствующего текста. Показать результат преподавателю.

Лабораторная работа 12
Создание программы MDI, имеющей несколько вторичных окон. Расположение нескольких изображений в разных окнах

Создать программу, обеспечивающую просмотр файлов bmp. Для каждой картинки создается свое окно.

1.Использовать заготовку Delphi для приложения MDI. Щелкнуть по кнопке New панели инструментов основного окна Delphi и в окне New Items на вкладке Projects выбрать пиктограмму MDI Application. Щелкуть по кнопке OK и получить заготовку, для копирования которой нужно выбрать свой рабочий каталог. Выполнить эту операцию под руководством преподавателя, так как можно испортить исходный шаблон. В рабочем каталоге появятся три формы и три модуля: Main – главного окна, Childwin – вторичного окна и About – вспомогательного окна справки о программе. Исходная заготовка на форме вторичного окна имеет компонент Memo.

2.Заменить на вторичной форме компонент Memo (удалив его) и поставить компонент Image (вкладка **Additional**), поскольку необходимо просматривать картинки .

3.На главной форме заменить компонент OpenDialog на OpenPictureDialog (вкладка **Dialogs**) для открытия графических файлов.

4. Установить для компонента Image начальное значение свойства Stretch – *true*, чтобы картинка растягивалась по размеру формы.

5.Исправить значение свойства Filter для компонента OpenPictureDialog, вызвав окно Filter Editor (щелкнув кнопкой мыши по кнопке с точками в поле значения этого свойства) и удалив в нем все форматы графических файлов, кроме *.bmp, *.ico и *.wmf. Другие форматы компонент Image не поддерживает.

6.Исправить для диалогового окна About тексты надписей компонентов Label, поместив сведения о назначении программы и авторах. Вставить в поле Image пиктограмму программы, которая будет

автоматически скопирована в рабочий каталог. Для этого изменить свойство Pictures, вызвав окно Picture Editor щелчком кнопки мыши на кнопке с точками в поле данных. В окне Picture Editor выбрать кнопку Load и в появившемся диалоговом окне выделить имя пиктограммы и щелкнуть по кнопке OK. Таким же образом можно загрузить любую картинку в поле Image на этапе конструирования формы, используя свойство Pictures.

7.Исправить сгенерированные автоматически обработчики событий, заменив имена удаленных компонентов на вновь установленные.

8.Сохранить проект. Проверить его работу. Для загрузки изображений можно использовать картинки из каталога Program Files\Common Files\BorlandShared\Images.

Самостоятельная работа. Создать обработчики события закрытия окна, которые обеспечат сохранение картинки с другим именем. Проверить работу измененной программы. Показать результат преподавателю.

Лабораторная работа 13
Создание программы SDI, имеющей несколько вторичных окон. Подключение к базе данных Paradox и просмотр данных

Создать программу, обеспечивающую просмотр файла демонстрационной базы данных country.db.

1.Создать форму с двумя кнопками вызова данных из БД.

2. Для создания формы представления данных (Form3) вызвать из верхнего меню мастера создания форм для БД *Database->Form Wizard.* В первом окне мастера оставить режимы (простая форма и на основе компонента Ttable) по умолчанию. Щелкнуть по кнопке Next. В следующем окне выбрать из раскрывающегося списка поля “Drive or Alias name“ значение DBDEMOS и выделить в списке “Table name“ имя БД *country.db*. Щелкнуть по кнопке Next. В следующем окне переместить поля таблицы Name, Area и Population из списка слева в поле справа (используя кнопку >). В следующем окне оставить режим (горизонтальная форма)по умолчанию. Щелкнуть по кнопке Next. В следующем окне выбрать режим создания отдельного модуля данных (Form

and Data Module) и снять флажок “Main form”. Щелкнуть по кнопке Finish. В рабочем каталоге появятся две дополнительные формы и два модуля: DataModule2 – вспомогательного окна для размещения невизуальных компонентов БД с модулем Unit2 и Form3 – вторичного окна просмотра выбранных данных с модулем Unit3.

3.Самостоятельно создать еще одну форму (Form4) вывода данных из той же БД в виде таблицы, содержащей все данные.

4.Написать обработчики событий для кнопок главной формы (Form1), вызывающие вторичные окна:

Form3.Show или *Form4.Show*.

5.Добавить в модуль формы Form1 ссылку на модули других форм:

Uses Unit3, Unit4.

6.Сохранить проект. Проверить его работу. Показать результат преподавателю.

Лабораторная работа 14
Создание консольного приложения

Создать программу, не имеющую графического интерфейса и окна. Такие программы часто создаются для серверов сети Internet. Этим методом можно обрабатывать старые Pascal – программы в среде Delphi.

1.Открыть проект и удалить главную форму Form1 и модуль, с ней связанный. Вызвать код главной программы в окно Code Editor (из верхнего меню *Project->View source*). Удалить из программы следующие строки:

Uses Forms;

Application.Initialize;

Application.Run;

2.Между *begin* и *end* вставить код программы на Pascal:

Writeln(’This program has not windowinterface’);

3.Добавить директиву компилятора: {$APPTYPE CONSOLE}.

4.Для проекта установить режим *Generate console application* (пункт меню *Project options*, вкладка *Linker*, раздел *EXE and DLL options*).

5.Запустить программу. Она будет выполняться в окне сеанса MS-DOS.

Самостоятельная работа. Добавить в программу ввод данных. Вычислить на их основе результат по формуле, предложенной преподавателем. Полученное значение вывести на экран.

Литература

1.*Дарахвелидзе П.*, *Марков Е.*, Delphi 5 Среда визуального программирования. СПб.: BHV, 1999.

2.*Гофман В.*, *Хомоненко А.* Delphi 6. СПб.: BHV, 2001.