Министерство образования Республики Беларусь
БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Программное обеспечение вычислительной техники
и автоматизированных систем»

Н.Н. Гурский

РАЗРАБОТКА ПРИЛОЖЕНИЙ В ВИЗУАЛЬНЫХ СРЕДАХ

Лабораторный практикум
по дисциплине «Разработка приложений в визуальных средах»
для студентов специальностей
1-40 01 01 «Программное обеспечение информационных технологий»,
1-40 01 02 «Информационные системы и технологии (по направлениям)»

В 2 частях

Часть 1

Учебное электронное издание

М и н с к 2 0 1 0

УДК 681.324(076.5)

Автор:
Н.Н. Гурский

Рецензенты:
Р.И. Фурунжиев, к.т.н., профессор БГАТУ;
В.А. Казакевич, к.ф-м.н., доцент БНТУ

В пособии представлен комплекс заданий для выполнения лабораторных работ по первой части дисциплины "Разработка приложений в визуальных средах", посвященной изучению основ разработки приложений в визуальной среде Delphi; рассмотрены принципы построения приложений, связанные с использованием основных компонентов, приведен список учебной литературы.

Белорусский национальный технический университет
пр-т Независимости, 65, г. Минск, Республика Беларусь
Тел.(017)292-77-52 факс(017)292-91-37
Регистрационный № БНТУ/ФИТР49-10.2010

СОДЕРЖАНИЕ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

При выполнении лабораторных работ необходимо:

1. В соответствии с целью работы сформулировать задачу, которая должна быть решена с помощью приложения.
2. Разработать алгоритм решения задачи.
3. Разработать приложение, включающее интерфейс, программные модули вычислительных процедур, формы представления результатов.
4. Выполнить компьютерное моделирование.
5. Произвести тестирование алгоритма и приложения.
6. Сделать выводы и обобщения.
7. Составить электронный вариант отчета с результатами выполнения приложения.

Образец оформления титульного листа приведен в приложении 1. При выполнении работ рекомендуется обратиться к литературе [1 - 9].

Лабораторная работа № 1
РАЗРАБОТКА ПРОСТЕЙШЕГО ПРИЛОЖЕНИЯ В ВИЗУАЛЬНОЙ СРЕДЕ DELPHI

Цель лабораторной работы: изучить основы среды Delphi и составить простейшую программу.

1. Краткие сведения

1.1. Интегрированная среда разработчика Delphi

Среда Delphi визуально реализуется в виде нескольких одновременно раскрытых на экране монитора окон. Количество, расположение, размер и вид окон может меняться программистом в зависимости от его текущих нужд, что значительно повышает производительность работы. При запуске Delphi на экране картинку появляется главное окно, приведенное на рис. 1.1.

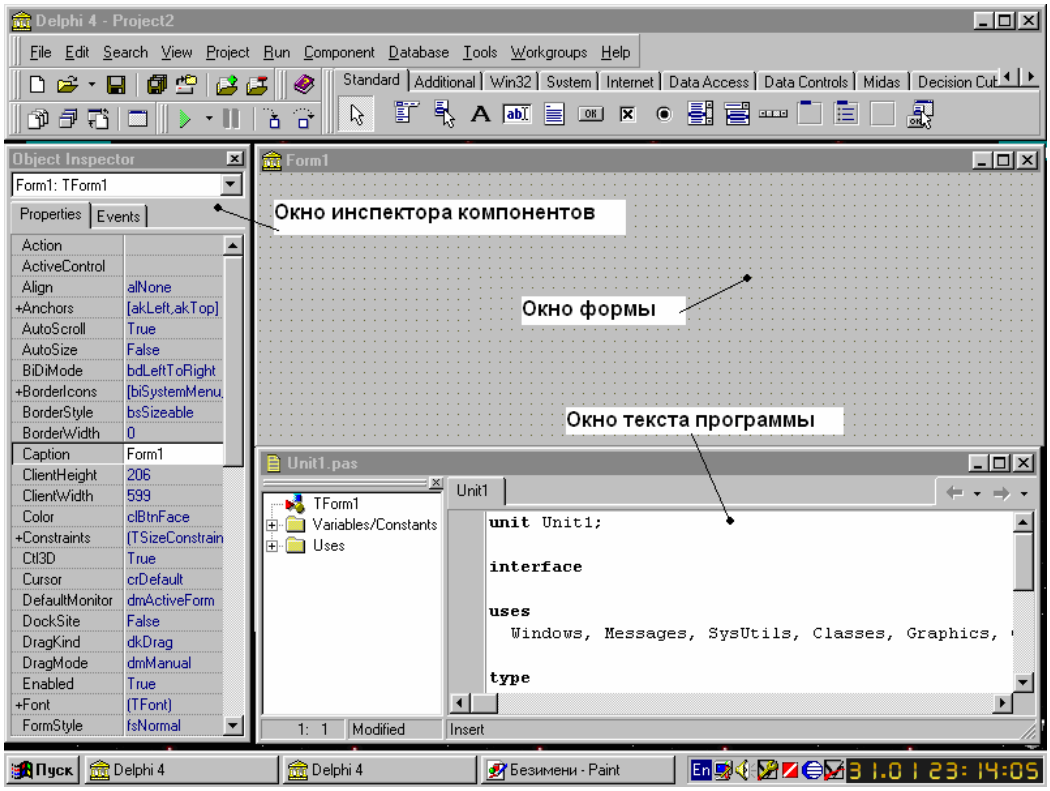


Рис. 1.1. Главное окно Delphi

Главное окно всегда присутствует на экране и предназначено для управления процессом создания программы. Основное меню (см. приложение 2) содержит все необходимые средства для управления проектом. Пиктограммы облегчают доступ к наиболее часто применяемым командам основного меню. Через меню компонентов (см. приложение 3) осуществляется доступ к наборам их свойств, которые описывают некоторый визуальный элемент (компонент), помещенный программистом в окно формы. Каждый компонент имеет определенный набор свойств (параметров), которые программист может задавать. Например, цвет, заголовок окна, надпись на кнопке, размер и тип шрифта и др.

Окно инспектора компонентов (вызывается с помощью клавиши F11) предназначено для изменения свойств компонента - закладка *Properties* (свойства) и создания обработчиков (процедур) при активизации тех или иных событий - страница *Events* (события).

Окно формы представляет собой внешний вид создаваемого Windows - приложения. В это окно в процессе проектирования интерфейса программы помещаются необходимые компоненты. Причем при выполнении программы большинство из помещенных компонент будут иметь тот же вид, что и на этапе проектирования.

Окно текста программы предназначено для просмотра, написания и редактирования текста программы. В системе Delphi используется язык программирования Object Pascal. При первоначальной загрузке в окне текста программы находится текст, содержащий минимальный набор операторов для нормального функционирования пустой формы в качестве Windows - приложения. При размещении некоторого компонента на форме происходит автоматическая фиксация его имени в коде программы.

Программа в среде Delphi представляется набором процедур, вызываемых при наступлении того или другого события и, таким образом, реализует событийно управляемую модель программируемого процесса обработки данных. Для каждого возникающего на форме события, с помощью страницы Events инспектора объектов в тексте программы организуется процедура (procedure), между ключевыми словами begin и end которой необходимо записать на языке Object Pascal требуемый алгоритм.

Переключение между окном формы и окном текста программы осуществляется с помощью клавиши F12.

1.2. Структура приложения в Delphi

Приложение в Delphi состоит из файла проекта (*.dpr),* одного или нескольких файлов исходного текста (*.pas),* файлов с описанием компонентов, расположенных на форме, (*.dfm*).

В ***файле проекта*** находится информация о модулях, составляющих данный проект. Файл проекта автоматически создается и редактируется средой Delphi.

Файл исходного текста - программный модуль (Unit) предназначен для размещения текста программы на языке Object Pascal.

Модуль состоит из 2-х разделов: интерфейсного (interface) и реализации (implementation). В интерфейсном разделе описываются типы, переменные, заголовки процедур и функции, которые могут быть использованы другими модулями. В разделе реализации располагаются тела процедур и функции, описанных в разделе объявлений, а также типы переменных, процедуры и функции, которые будут функционировать только в пределах данного модуля.

Структура модуль имеет вид:

```
Unit Unit1;
  interface
    //Раздел объявлений
  implementation
    //Раздел реализации
begin
    //Раздел инициализации
end.
```

При компиляции программы Delphi создает файл с расширением *.dcu,* содержащий в себе результат перевода в машинные коды содержимого файлов с расширением *.pas* и *.dfm.* Компоновщик преобразует файлы с расширением *.dcu* в единый загружаемый файл с расширением *.exe.* В файлах, имеющих расширение *.-df, .-dp, .-pa,* хранятся резервные копии файлов с образом формы, проекта и исходного текста соответственно.

1.3. Пример написания программы

Задание: считая заданными значения переменных x, y, z, составить программу вычисления значения арифметического выражения:

$$u = tg^2(x+y) - e^{y-z}\sqrt{\cos x^2 + \sin z^2}\ .$$

Форму приложения организовать, как показано на рис. 1.2.

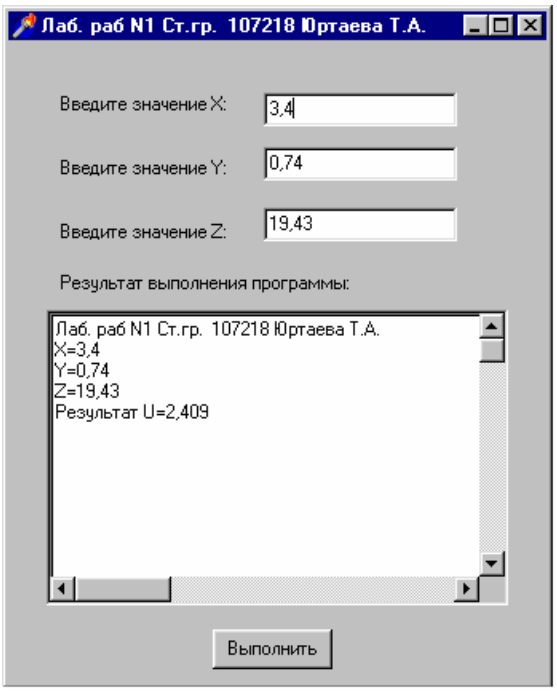


Рис. 1.2. Форма приложения

1.3.1. Настройка формы

Пустая форма в правом верхнем углу имеет кнопки управления, которые предназначены: для свертывания формы в пиктограмму, для разворачивания формы на весь экран и возвращения к исходному размеру и для закрытия формы.

1.3.2. Изменение заголовка формы

Новая форма имеет одинаковые имя (Name) и заголовок (Caption) - Form1. Для изменения заголовка формы щелкните кнопкой мыши на форме, затем вызовите окно

инспектора объектов (F11). В форме инспектора объектов найдите и щелкните мышью на свойстве Caption страницы Properties. В выделенном окне наберите ” Ла6.раб.№1. Ст. гр. 107218 Юртаева Т.А.”.

1.3.3. Размещение строки ввода (Edit)

Если необходимо ввести или вывести информацию, которая вмещается в одну строку, используют окно однострочного редактора текста, представляемого компонентом Edit. В данной программе с помощью однострочного редактора будут вводиться переменные x, y, z типа real.

Выберите на закладке компонентов Standard пиктограмму  , щелкните мышью в том месте формы, где вы хотите ее поставить. Вставьте три компонента Edit в форму. Захватывая их "мышью" отрегулируйте размеры окошек компонент и их положение.

Обратите внимание на то, что в тексте программы появились три новых однотипных переменных Edit1, Edit2, Edit3. В каждой из этих переменных с расширением Text будет содержаться строка символов (тип String) и отображаться в соответствующем окне Edit.

Так как численные значения переменных x, y, z имеют действительный тип для преобразования строковой записи числа, находящегося в переменной Edit1.Text, в действительное, используется стандартная функция

```
X := StrToFloat(Edit1.Text);
```

Если исходные данные имеют целочисленный тип, например ***integer***, то используется стандартная функция:

```
X := StrToInt(Editl.Text);
```

При этом в записи числа не должно быть пробелов, а действительное число записывается с десятичной запятой или точкой (это зависит от настроек ОС Windows конкретного компьютера).

С помощью инспектора объектов установите шрифт и размер символов отражаемых в строке Edit (свойство Font).

1.3.4. Размещение надписей (Label)

На форме рисунка 1.2 имеются четыре надписи. Для нанесения таких надписей на форму используется компонент Label. Выберите в палитре компонентов закладки Standard пиктограмму  , щелкните на ней мышью, появится надпись Label, Проделайте это для четырех надписей. Для каждой надписи, щелкнув на ней мышью, отрегулируйте размер и, изменив свойство Caption инспектора объектов, введите строку, например “Введите значение X”, а также выберите размер символов (свойство Font).

Обратите внимание, что в тексте программы автоматически появились четыре новых переменных типа TLabel. В них хранятся пояснительные строки, которые можно изменять в процессе работы программы.

1.3.5. Размещение многострочного окна вывода (Memo)

Для вывода результатов работы программы обычно используется текстовое окно, которое представлено компонентом Memo. Выберите на закладке Standard пиктограмму  и поместите компоненту Memo на форму. С помощью мыши отрегулируйте его размеры и местоположение. После установки с помощью инспектора свойства ScrollBars - SSBoth в окне появятся вертикальная и горизонтальная полосы прокрутки.

В тексте программы появилась переменная Memo1 типа TMemo. Информация, которая отображается построчно в окне типа TMemo, находится в массиве строк Memo1.Lines. Каждая строка имеет тип String.

Для очистки окна используется метод Memo1.Clear. Для того чтобы добавить новую строку в окно, используется метод Memo1.Lines.Add (переменная типа String).

Если нужно вывести число, находящееся в переменной действительного или целого типа, то его надо предварительно преобразовать к типу ***String*** и добавить в массив Memo1.Lines. Например, если переменная u:= 100 целого типа, то оператор Memo1.Lines.Add (‘Значение u = ’ + IntToStr(u)) сделает это и в окне появится строка «Значение u = 100». Если переменная u:=-256,38666 действительная, то при использовании метода Memo1.Lines.Add (‘Значение u= ’ + FloatToStrF(u, ffFixed, 8, 2))

будет выведена строка «Значение u= -256,39». При этом под все число отводится восемь позиций, из которых две позиции занимает его дробная часть.

Если число строк в массиве Memo1 превышает размер окна, то для просмотра всех строк используется вертикальная полоса прокрутки. Если длина строки Memo1 превосходит количество символов в строке окна, то в окне отображается только начало строки. Для просмотра всей строки используется горизонтальная полоса прокрутки.

1.3.6. Написание программы обработки события создания формы (FormCreate)

При запуске программы возникает событие “создание формы” (OnCreate). Создадим обработчик этого события, который заносит начальные значения переменных x, y, z в соответствующие окна TEdit, а в окне TMemo помешает строку с указанием номера группы и фамилию студента. Для этого дважды щелкнем мышью на любом свободном месте формы. На экране появится текст, в котором автоматически внесен заголовок процедуры - обработчика (процедуры) события создания формы:

Procedure TForm1.FormCreate(Sender:TObject);

Между begin...end записывается текст программы (см. пример, расположенный ниже).

1.3.7. Написание программы обработки события нажатия кнопки (ButtonClick)

Поместите на форму кнопку Button, которая находится на странице Standard панели стандартных компонентов. С помощью инспектора объектов измените заголовок (Caption) – Button1 на слово “Выполнить” или другое по вашему желанию. Отрегулируйте положение и размер кнопки.

После этого два раза щелкните мышью на кнопке, появится текст процедуры, которая будет каждый раз вызываться при возникновении события - нажатия кнопки:

Procedure TForm1.ButtonClick(Sender: TObject);

1.3.8. Запуск программы

Запустить программу можно нажав Run в главном меню Run, или клавишу F9, или пиктограмму  . При этом происходит трансляция и, если нет ошибок,

компоновка программы и создание единого загружаемого файла с расширением .exe. На экране появляется активная форма программы (рисунок 1.2).

Работа с программой происходит следующим образом. Нажмите (щелкните мышью) кнопку “Выполнить”. В окне Memo1 появляется результат. Измените исходные значения x, y, z в соответствующих окнах Edit и снова нажмите кнопку “Выполнить” - появятся новые результаты. Завершить работу программы можно нажимая кнопку  на форме или выполнив команду Program Reset, вызываемую из главного меню с пунктом Run.

Текст программы:

```
Unit LabRab_1;
interface
uses
  Windows, Messages, SysUlits, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls;
type
   TForm1=class(TForm)
    Edit1:TEdit;
    Edit2: TEdit;
    Edit3: TEdit; Label1: TLabel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Memo1:TMemo;
    Button1:TButton;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject):
    private
      {Private declarations}
    public
      { Public declarations}
  end;
var
  Form1: TForrn1;

implementation
{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
begin
  Edit1.Text := '3,4';     // Начальное значение X
  Edit2.Text := '0,74';   // Начальное значение Y
  Edit3.Text := '19,43'; // Начальное значение Z
```

```
  Memo1.Clear;          //Очистка окна редактора Memo1
  // Вывод строки в многострочный редактор Memol
  Memo1.Lines.Add('Ла6.раб.№1. Ст. гр. 107218 Юртаева Т.А.');
end;

procedure TForm1.Button1Click(Sender: TObject);
var
   x, y, z : real;
   a, b, c, u : real;
begin
  x : = StrToFloat(Edit1.Text);   // Считывается значение x
// Вывод x в окно Memo1
  Memo1.Lines.Add('x = ' + Edit1.Text);
   y := StrToFloat(Edit2.Text);   // Считывается значение y
// Вывод y в окно Memol
  Memo1.Lines.Add('y =' + Edit2.Text);
   z := StrToFloat(Edit3.Text);  // Считывается значение z
 // Вывод z в окно Memo1
   Memo1.Lines.Add('z = ' + Edit3.Text);
 // Вычисляем арифметическое выражение
   a := Sqr(Sin(x+y) / Cos(x+y));
   b := Exp(y-z);
   c := Sqrt(Cos(Sqr(x))+Sin(Sqr(z)));
   u:= a – b*c;
 //Выводим результат в окно Memo1
   Memo1.Lines.Add('Результат u:= ' + FloatToStrF(u, ffFixed, 8, 3));
 end;

 end.
```

2. Постановка задачи

Разработать линейное приложение с использованием компонент TLabel, TEdit, TMemo, TButton в соответствии с индивидуальным вариантом задания.

3. Задания

1. $t = \dfrac{2\cos\left(x - \dfrac{\pi}{6}\right)}{0.5 + \sin^2 y}\left(1 + \dfrac{z^2}{3 - z^2/5}\right).$

2. $u = \dfrac{\sqrt[3]{8 + |x - y|^2 + 1}}{x^2 + y^2 + 2} - e^{|x-y|}\left(tg^2 z + 1\right)^x.$

3. $v = \dfrac{1+\sin^2(x+y)}{\left|x - \dfrac{2y}{1+x^2y^2}\right|} x^{|y|} + \cos^2\left(arctg\,\dfrac{1}{z}\right).$

4. $w = \left|\cos x - \cos y\right|^{\left(1+2\sin^2 y\right)}\left(1 + z + \dfrac{z^2}{2} + \dfrac{z^3}{3} + \dfrac{z^4}{4}\right).$

5. $\alpha = \ln\left(y^{-\sqrt{|x|}}\right)\left(x - \dfrac{y}{2}\right) + \sin^2 \text{ arctg(z)}.$

6. $\beta = \sqrt{10\left(\sqrt[3]{x} + x^{y+2}\right)}\left(\arcsin^2 z - |x-y|\right).$

7. $\gamma = 5\text{arctg}(x) - \dfrac{1}{4}arctg(x)\dfrac{x + 3|x-y| + x^2}{|x-y|z + x^2}.$

8. $\varphi = \dfrac{e^{|x-y|}|x-y|^{x+y}}{arctg(x) + arctg(z)} + \sqrt[3]{x^6 + \ln^2 y}\ .$

9. $= \left|x^{\frac{y}{x}} - \sqrt[3]{\dfrac{y}{x}}\right| + (y-x)\dfrac{\cos y - \dfrac{z}{(y-x)}}{1 + (y-x)^2}.$

10. $a = 2^{-x}\sqrt{x + \sqrt[4]{|y|}}\sqrt[3]{e^{x-1/\sin z}}\ .$

11. $b = y^{\sqrt[3]{x}} + \cos^3(y)\dfrac{|x-y|\left(1 + \dfrac{\sin^2 z}{\sqrt{x+y}}\right)}{e^{x-y} + \dfrac{x}{2}}.$

12. $c = 2^{(y)} + \left(3^x\right)^y - \dfrac{y\left(arctgz - \dfrac{\pi}{6}\right)}{|x| + \dfrac{1}{y^2+1}}.$

13. $f = \dfrac{\sqrt[4]{y + \sqrt[3]{x-1}}}{|x-y|\left(\sin^2 z + tgz\right)}.$

14. $g = \dfrac{y^{x+1}}{\sqrt[3]{|y-2|} + 3} + \dfrac{x + \dfrac{y}{2}}{2|x+y|}(x+1)^{-1/\sin z}\ .$

15. $h = \dfrac{x^{y+1} + e^{y-1}}{1 + x|y - tgz|}\left(1 + |y-x|\right) + \dfrac{|y-x|^2}{2} - \dfrac{|y-x|^3}{3}.$

16. Вывести на экран 1 или 0 в зависимости от того, имеют ли три заданных целых числа одинаковую четность или нет.

17. Найти сумму цифр заданного четырехзначного числа.

18. Определить число, полученное выписыванием в обратном порядке цифр заданного трехзначного числа.

19. Вывести на экран 1 или 0 в зависимости от того, равна ли сумма двух первых цифр заданного четырехзначного числа сумме двух его последних цифр.

20. Вывести на экран 1 или 0 в зависимости от того, равен ли квадрат заданного трехзначного числа кубу суммы цифр этого числа.

21. Вывести на экран 1 или 0 в зависимости от того, есть ли среди первых трех цифр дробной части заданного положительного вещественного числа цифра нуль.

22. Вывести на экран 1 или 0 в зависимости от того, есть ли среди цифр заданного трехзначного числа одинаковые.

23. Присвоить целой переменной k третью от конца цифру в записи положительного целого числа n.

24. Присвоить целой переменной k первую цифру из дробной части положительного вещественного числа.

Лабораторная работа № 2
РАЗРАБОТКА ПРИЛОЖЕНИЯ, РЕАЛИЗУЮЩЕГО РАЗВЕТВЛЯЮЩИЙСЯ ВЫЧИСЛИТЕЛЬНЫЙ ПРОЦЕСС

Цель лабораторной работы: научиться пользоваться стандартными компонентами организации переключений (TCheckBox, TRadioGroup и др.). Используя компоненты организации переключений разработать интерфейс и программу для заданного разветвляющегося алгоритма.

1. Краткие сведения

1.1. Операторы if и case языка Pascal

Для программирования разветвляющихся алгоритмов в языке Pascal используются переменные типа ***boolean***, которые могут принимать только два значения - **true** и **false** (да, нет), а также операторы ***if*** и ***case***. Оператор **if** проверяет результат логического выражения или значение переменной типа ***boolean*** и организует разветвление вычислений. Оператор ***case*** организует разветвления в зависимости от селектора - значения некоторой переменной, например, n целого типа:

```
case n of
    0:  u:= x + y;
    1:  u:= x - y;
    2:  u:= x * y;
  else u:= 0;
end;
```

В соответствии со значением ***n*** вычисляется значение переменной u. При этом, если n=0, то u=x+y, если n=1, то u=x-y, если n=2, то u=x*y и, наконец, u=0 при любых значениях n, отличных от 0, 1 или 2.

1.2. Кнопки-переключатели в Delphi

При создании программ в Delphi для организации разветвлений часто используются компоненты в виде кнопок-переключателей. Состояние такой кнопки

(включено - выключено) визуально отражается на форме. На форме (см. рисунок 2.1) представлены кнопки-переключатели двух типов (TCheckBox и TRadioGroup).

Компонент **TCheckBox** организует кнопку независимого переключателя, с помощью которой пользователь может указать свое решение типа да/нет. В программе состояние кнопки связано со значением булевской переменной, которая проверяется с помощью оператора if.

Компонент **TRadioGroup** организует группу кнопок – зависимых переключателей. При нажатии одной из кнопок группы все остальные кнопки |отключаются. В программу передается номер включенной кнопки (0, 1, 2,..), который анализируется с помощью оператора ***case.***

1.3. Пример программы

Задание: ввести три числа x, y, z. Вычислить по усмотрению или u=sin(x), или u=cos(x), или u=tg(x). Найти по желанию максимальное из трех чисел: max(u, y, z) или max(|u|, |y|, |z|).

Создать форму, представленную на рисунке 2.1, и написать соответствующую программу.

1.3.1. Создание формы

Создайте форму такую же, как в первом задании, скорректировав текст надписей и положение окон TEdit.

1.3.2. Работа с компонентом TCheckBox

Выберите в палитре компонентов Standard пиктограмму  и поместите ее в нужное место формы. С помощью инспектора объектов измените заголовок (Caption) на "Вычисление MaxAbs". В тексте программы появилась переменная ***CheckBox1:TCheckBox***. Теперь в зависимости от того, нажата или нет кнопка, булевское свойство **CheckBox1.Checked** будет принимать значения ***true*** или ***false***.

1.3.3. Работа с компонентом TRadioGroup

Выберите в палитре компонентов Standard пиктограмму  и поместите ее в нужное место формы. На форме появится окаймленный линией чистый прямоугольник с заголовком RadioGroup1. Замените заголовок (Caption) на U(x). Для того чтобы разместить на компоненте кнопки, необходимо свойство Columns установить равным единице. Дважды щелкните по правой части свойства Items мышью, появится строчный редактор списка заголовков кнопок. Наберите три строки с именами: в первой cos(x), во второй – sin(x), в третьей – tg(x), нажмите ОК.

Обратите внимание на то, что в тексте программы появилась переменная RadioGroupl типа TRadioGroup. Теперь при нажатии одной из кнопок группы в переменной целого типа **RadioGroup1.Itemlndex** будет находиться номер нажатой клавиши, что используется в приведенной программе.

1.3.4. Создание обработчиков событий FormCreate и ButtonlClick

Процедуры - обработчики событий FormCreate и ButtonlClick создаются аналогично тому, как и в первой теме. Тексты процедур приведены ниже. Запустите программу и убедитесь в том, что все ветви алгоритма выполняются правильно.

Форма приложения приведена на рис. 2.1.

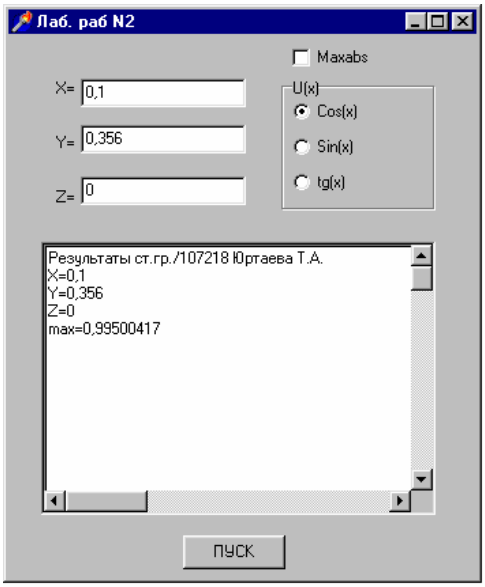


Рис. 2.1. Форма приложения

Текст программы:

```
Unit LabRab_2;
Interface;
Uses
  Windows, Messages, SysUtils Classes, Graphics, Controls,
  Forms, Dialogs, SidClrls, ExtCtrIs;
Type
  TForm1 = class(TForm)
   CheckBoxl: TCheckBox;
   RadioGroupl: TRadioGroup;
   Memol: TMemo;
   Button1: TButton;
   Edit1: TEdit;
   Label1: TLabel;
   Label2: TLabel;
   Edit2: TEdit;
   Label3: TLabel;
   Edit3: TEdil;
   procedure FormCreate(Sender: TObject);
   procedure ButtonlClick(Sender: TObject);
private
 { Private declarations}
public
{ Public declarations}
 end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.FormCreate(Sender:TObject);
begin
    Edit1.Text:='0.1';
    Edit2.Text:='0.356';
    Edit3.Text:='0.0';
    Memo1.Clear;
    Memo1.Lines.Add('Рез-ты ст. гр. 107217 Юртаева Т.А.');
 end;

procedure TForm1.Button1Click(Sender: TObject);
var
  x, y, z, u, ma : extended;
begin
// Ввод исходных данных и их вывод в окно Memo1
   x:= StrToFloat(Edit1.Text);
   Memo1.Lines.Add('x=' + Edit1.Text);
   y:= StrToFloat(Edit2.Text);
   Memol.Lines.Add('y=' + Edit2.Text);
   z:= StrToFloat(Edit3.Text);
   Memol.Lines.Add(‘z=' + Edit3.Text);
 // Проверка номера нажатой кнопки и выбор соответствующей ей функции
```

```
  case RadioGroupl.ItemIndex of
    0: u:= cos(x);
    1: u:= sin(x);
    2: u:= sin(x)/cos(x);
  end;
// Проверка состояния кнопки CheckBox1
  if CheckBox1.Checked then
  begin
    u:= abs(u);
    y:= abs(y);
    z:= abs(z)
  end;
// Нахождение максимального из трех чисел
     if u > y then ma := u else ma := y;
     if z > ma then ma := z;
     if CheckBox1.Checked then
     Memo1.Lines.Add('MaxAbs=' + FloatToStrF(ma, ffFixed, 8, 2))
     else
     Memo1.Lines.Add('max=' + FloatToStrF(ma, ffFixed, 8, 2));
 end;
 end.
```

2. Постановка задачи

Разработать приложение, реализующее разветвляющийся вычислительный процесс в соответствии с индивидуальным заданием.

3. Задания

По указанию преподавателя выберите индивидуальное задание из нижеприведенного списка. В качестве f(x) использовать по выбору: sh(x), x^2; e^x. Отредактируйте вид формы и текст программы, в соответствии с полученным заданием.

1. $a=\begin{cases}(f(x)+y)^2-\sqrt{f(x)y}, & xy>0\\ (f(x)+y)^2-\sqrt{|f(x)y|}, & xy<0\\ (f(x)+y)^2+1, & xy=0\end{cases}$

2. $b=\begin{cases}\ln(f(x))+(f(x)^2+y)^3, & x/y>0\\ \ln|f(x)/y|+(f(x)+y)^3, & x/y<0\\ (f(x)^2+y)^3, & x=0\\ 0, & y=0\end{cases}$

3. $c = \begin{cases} f(x)^2 + y^2 + \sin y, & x - y = 0 \\ (f(x) - y)^2 + \cos y, & x - y > 0 \\ (y - f(x))^2 + tg(y), & x - y < 0. \end{cases}$

4. $d = \begin{cases} (f(x) - y)^3 + arctg(f(x)), & x > y \\ (y - f(x))^3 + arctg(f(x)), & y > x \\ (y + f(x))^3 + 0.5, & y = x \end{cases}$

5. $e = \begin{cases} i\sqrt{f(x)}, i - \textit{нечетное}, & x > 0 \\ i/2\sqrt{|f(x)|}, i - \textit{четное}, & x < 0 \\ \sqrt{|if(x)|}, & \textit{иначе}. \end{cases}$

6. $g = \begin{cases} e^{f(x) - |b|}, & 0.5 < xb < 10 \\ \sqrt{|f(x) + b|}, & 0.1 < xb < 0.57 \\ 2f(x)^2, & \textit{иначе}. \end{cases}$

7. $s = \begin{cases} e^{f(x)}, & 1 < xb < 10 \\ \sqrt{|f(x) + 4 * b|}, & 12 < xb < 40 \\ bf(x)^2, & \textit{иначе}. \end{cases}$

8. $j = \begin{cases} \sin(5f(x) + 3m|f(x)|), & -1 < m < x \\ \cos(3f(x) + 5m|f(x)|), & x > m \\ (f(x) + m)^2, & \textit{иначе}. \end{cases}$

9. $l = \begin{cases} 2f(x)^3 + 3p^2, & x > p \\ |f(x) - p|, & 3 < x < |p| \\ (f(x) - p)^2, & x = p. \end{cases}$

10. $k = \begin{cases} \ln(|f(x)| + |q|), & |xq| > 10 \\ e^{f(x)+q}, & |xq| < 10 \\ f(x) + q, & |xq| = 10. \end{cases}$

11. $m = \dfrac{\max(f(x), y, z)}{\min(f(x), y)} + 5.$

12. $n = \dfrac{\min(f(x) + y, y - z)}{\max(f(x), y)}.$

13. $p=\frac{\left|\min\left(f(x),y\right)-\max\left(y,z\right)\right|}{2}$.

14. $q=\frac{\max\left(f(x)+y+z,xyz\right)}{\min\left(f(x)+y+z,xyz\right)}$.

15. r=max(min(f(x),y),z).

16. Известно, что из четырех чисел a_1, a_2, a_3, a_4 одно отлично от трех других, равных между собой. Присвоить номер этого числа переменной n.

17. По номеру n (n>0) некоторого года определить с - номер его столетия (учесть, что, к примеру, началом XX столетия был 1901, а не 1900 год).

18. Значения переменных a, b и c поменять местами так, чтобы оказалось $a \le b \le c$.

19. Дано целое k от 1 до 180. Определить, какая цифра находится в k-й позиции последовательности 10111213…9899, в которой выписаны подряд все двузначные числа.

20. Дано натуральное k. Определить *k-ю* цифру в последовательности 110100100010000100000..., в которой выписаны подряд степени 10.

21. В старояпонском календаре был принят 60-летний цикл, состоявший из пяти 12-летних подциклов. Подциклы обозначались названиями цвета: green (зеленый), red (красный), yellow (желтый), white (белый), black (черный). Внутри каждого подцикла годы носили названия животных: крысы, коровы, тигра, зайца, дракона, змеи, лошади, овцы, обезьяны, курицы, собаки и свиньи. (1984 год- *год зеленой крысы* -был началом очередного цикла). Разработать программу, которая вводит номер некоторого года нашей эры и выводит его название по старояпонскому календарю.

22. Если сумма трех попарно различных действительных чисел *x*, *y*, *z* меньше единицы, то наименьшее из этих трех чисел заменить полусуммой двух других; в противном случае заменить меньшее из *x* и *y* полусуммой двух оставшихся значений.

23. Для целого числа k от 1 до 99 вывести фразу “мне k лет”, учитывая при этом, что при некоторых значениях k слово “лет” надо заменить на слово “год” или “года”.

24. Для натурального числа k вывести фразу “мы выпили k бутылок пива”, согласовав слово “бутылка” с числом k.

Лабораторная работа № 3
РАЗРАБОТКА ПРИЛОЖЕНИЯ, РЕАЛИЗУЮЩЕГО ЦИКЛИЧЕСКИЙ ВЫЧИСЛИТЕЛЬНЫЙ ПРОЦЕСС

Цель лабораторной работы: изучить средства отладки программ в среде Delphi. Составить и отладить программу для циклического вычислительного процесса.

1. Краткие сведения

1.1. Операторы организации циклов Repeat, While, For

Под циклом понимается многократное выполнение одних и тех же операторов при различных значениях промежуточных данных. Число повторений может быть задано в явной или неявной форме. Для организации повторений в языке Pascal предусмотрены операторы Repeat, While и For.

Оператор Repeat имеет форму:

```
Repeat
  <операторы>
 until <условие>;
```

и организует повторение операторов, помещенных между ключевыми словами repeat и until, до тех пор, пока не выполнится <условие>=true, после чего управление передается следующему за циклом оператору.

Оператор **While** имеет форму:

```
While<условие >do
begin
   <операторы>
end;
```

и организует повторение операторов, помещенных между begin и end, до тех пор, пока не выполнится <условие>=false. Заметим, что если <условие>=false при первом входе в цикл, то <операторы> не выполнятся ни разу, в отличие от оператора Repeat, в котором хотя бы один раз они выполнятся.

Оператор **For** имеет форму:

```
For i := i1 to i2 do
```

```
begin
  <операторы>
end;
```

и организует повторное вычисление операторов при нарастающем изменении переменной цикла i от начального значения i1 до конечного i2 с шагом, равным единице. Заметим, что если i2>il, то <операторы> не выполнятся ни разу.

Модификация оператора имеет вид:

```
For i := i2 downto i1 do
begin
  <операторы>
end;
```

и организует повторение вычислений при убывающем изменении i на единицу.

1.2. Средства отладки программ в Delphi

В написанной программе после ее запуска, как правило, обнаруживаются ошибки. Ошибки первого уровня (ошибки компиляции) связаны с неправильной записью операторов (орфографические и/или синтаксические ошибки). При обнаружении ошибки компилятор Delphi останавливается напротив первого оператора, в котором обнаружена ошибка. В нижней части экрана появляется текстовое окно, содержащее сведения обо всех ошибках, найденных в проекте. Каждая строка этого окна содержит имя файла, в котором найдена ошибка, номер строки с ошибкой и характер ошибки. Для быстрого перехода к интересующей ошибке необходимо дважды щелкнуть на строке с ее описанием.

Для получения более полной информации о характере ошибки необходимо обратится к HELP нажатием клавиши F1. Следует обратить внимание на то, что одна ошибка может повлечь за собой другие, которые исчезнут при ее исправлении. Поэтому следует исправлять ошибки последовательно, сверху вниз и, после исправления каждой ошибки компилировать программу снова.

Ошибки второго уровня - ошибки времени выполнения. Они связаны с ошибками выбранного алгоритма решения или с неправильной программной реализацией алгоритма. Эти ошибки проявляются в том, что результат расчета оказывается неверным, либо происходит переполнение (деление на нуль) и др. Поэтому перед использованием отлаженной программы ее необходимо протестировать, т.е. сделать просчеты при таких комбинациях исходных данных, для которых заранее

известен результат. Если тестовые расчеты указывают на ошибку, то для ее поиска следует использовать встроенные средства отладки среды Delphi.

В простейшем случае для локализации места ошибки рекомендуется поступать следующим образом. В окне редактирования текста установить курсор в строке перед проверяемым участком, выделить этот оператор, нажатием мышью на полосе слева от текста программы, нажать клавишу F4 (выполнение до курсора). При этом выполнение программ будет остановлено на строке, содержащей курсор. Теперь можно увидеть, чему равны значения интересующих переменных. Для этого нужно поместить на переменную курсор и в качестве подсказки на экране будет высвечено ее значение. В другом варианте требуется нажатие комбинации клавиш Ctrl-F7 и в появившемся диалоговом окне указать интересующую переменную (с помощью данного окна можно также изменить значение переменной во время выполнения программы).

Нажимая клавишу F7 (пошаговое выполнение), можно построчно выполнять программу, контролируя изменение тех или иных переменных и правильность вычислений. Если курсор находится внутри цикла, то после нажатия F4 расчет останавливается после одного выполнения тела цикла. Для продолжения расчетов следует нажать мышью на команде <Run> меню Run.

Нажимая клавишу F8 можно продолжать отладку не заходя внутрь процедур и функций.

1.3. Пример выполнения задания

Задание: написать и отладить программу, которая выводит таблицу значений функций S(x) для x изменяющихся в интервале от x1 до x2 с шагом h.

$$S(x)=\sum_{k=1}^{N}(-1)^k\frac{x^k}{k}.$$

Форма приложения приведена на рис. 3.1.

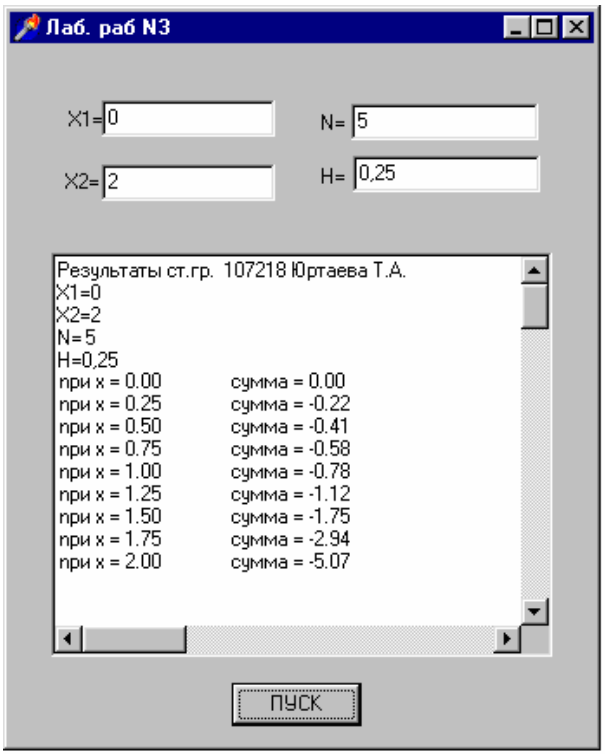


Рис. 3.1. Форма приложения

Текст программы:

```
Unit LabRab_3;
Interface
Uses
    Windows, Messages, SysUtils, Classes, Graphics, Controls,
    Forms, Dialogs, StdCtrls, ExtCtrls;


Type
    TForm1=class(TForm)
       Memo1:TMemo;
       Button1: TButton;
       Label1: TLabel;
       Label2: TLabel;
       Label3: TLabel;
       Label4: TLabel;
        Edit1:TEdit;
        Edit2:TEdit;
        Edit3:TEdit;
        Edit4:TEdit;
   procedure FormCreate(Sender: TObject);
   procedure Button1Click(Sender: TObject);
private
  {Private declarations}
 public
```

```
  {Public declarations}
 end;
 var
    Form1:TForm1;
 Implementation
 Uses Math;
{$R *.dfm}

  procedure TForm1.FormCreate(Sender: TObject);
 begin
   Edit1.Text := ’0’;
   Edit2.Text : = '2';
   Edit3.Text := '5';
   Edit4.Text := '0,25';
   Memo1.Clear;
   Memo1.Lines.Add('Результаты ст. гр. 107218 Юртаева Т.А.');

Procedure TForm1.Button1Click(Sender: TObject);
Var
    x1, x2, x, h, a, s  : extended;
    N, k, c : integer;
begin
   xl := StrToFloal(Edit1.Text);
   Memo1.Lines.Add('x1 = ' + Edit1.Text);
   x2 := StrToFloat(Edit2.Text);
   Memo1.Lines Add('x2 = ' + Edit2.Text);
   N := StrToInt(Edit3.Text);
   Memo1.Lines.Add(‘N = ' + Edit3.Text);
   h := StrToFloat(Edit4.Text);
   Memo1.Lines.Add(‘H = ' + Edit4.Text);
  x := x1;
  Repeat
     s := 0;
     for k :=1 to N do
     begin
        if (k mod 2) = 0 then c:= 1 else c:= -1;
        a := c * Power(x, k) / k;
        s := s + a;
     end;
  Memo1.Lines.Add('при x = ' + FloatToStrF(x, ffFixed, 6, 2) + #9 +
                          ' сумма = ' + FloalToStrF(s, ffFixed, 6, 2));
    x := x+h;
 until x>x2;
 end;
 end.
```

После отладки программы составьте тест (N=2, X1=0, X2=1**,** h=3), установите курсор на первый оператор (N:=), нажмите клавишу F4. После этого нажимая клавишу

F7, выполните пошаговую отладку программы и проследите, как меняются все переменные в процессе выполнения.

2. Постановка задачи

Разработать приложение с реализацией циклических вычислений в соответствии с индивидуальным заданием.

3. Задания

В заданиях с № 1 по № 15 (табл. 3.1) необходимо вывести на экран таблицу значений функции Y(x) и ее разложения в ряд S(x) для x изменяющихся от x_n до x_k с шагом $h = (x_k - x_n)/n$. Близость значений S(x) и Y(x) во всем диапазоне значений x указывает на правильность вычисления S(x) и Y(x).

Таблица 3.1

№	x_n	x_k	S(x)	n	Y(x)
1	2	3	4	5	6
1	0.1	1	$x - \frac{x^3}{3!} + ... + (-1)^n \frac{x^{2n+1}}{(2n+1)!}$	16	$\sin x$
2	0.1	1	$1 + \frac{x^2}{2} + .. + \frac{x^{2n}}{(2n)!}$	10	$\frac{e^x + e^{-x}}{2}$
3	0.1	1	$1 + \frac{\cos\frac{\pi}{4}}{1!}x + ... + \frac{\cos n\frac{\pi}{4}}{n!}x^n$	12	$e^{x\cos\frac{\pi}{4}}\cos\left(x\sin\frac{\pi}{4}\right)$
4	0.1	1	$1 - \frac{x^2}{2!} + ... + (-1)^n \frac{x^{2n}}{(2n)!}$	8	$\cos x$
5	0.1	1	$1 + 3x^2 + ... + \frac{2n+1}{n!}x^{2n}$	14	$\left(1 + 2x^2\right)e^{x^2}$
6	0.1	1	$x + \frac{x^3}{3!} + ... + \frac{x^{2n+1}}{(2n+1)!}$	8	$\frac{e^x - e^{-x}}{2}$
7	0.1	1	$\frac{x^3}{3} - \frac{x^5}{15} + ... + (-1)^{n+1}\frac{x^{2n+1}}{4n^2 - 1}$	12	$\frac{1+x^2}{2}arctgx - \frac{x}{2}$
8	0.1	1	$1 + \frac{2x}{1!} + ... + \frac{(2x)^n}{n!}$	10	e^{2x}

Окончание табл. 3.1

1	2	3	4	5	6
9	0.1	1	$x-\frac{x^3}{3}+\ldots+(-1)^n\frac{x^{2n+1}}{2n+1}$	14	$\left(\frac{x^2}{4}+\frac{x}{2}+1\right)e^{\frac{x}{2}}$
10	0.1	0.5	$x-\frac{x^3}{3}+\ldots+(-1)^n\frac{x^{2n+1}}{2n+1}$	15	arctgx
11	0.1	1	$1-\frac{3}{2}x^2+\ldots+(-1)^n\frac{2n^2+1}{(2n)!}x^{2n}$	10	$\left(1-\frac{x^2}{2}\right)\cos x-\frac{x}{2}\sin x$
12	0.1	1	$-\frac{(2x)^2}{2}+\frac{(2x)^4}{24}-\ldots+(-1)^n\frac{(1+x)^n}{n}$	8	$2\left(\cos^2 x-1\right)$
13	-2	-0.1	$-(1+x)^2+\frac{(1+x)^4}{2}+\ldots+(-1)^n\frac{(1+x)^{2n}}{n}$	16	$\ln\frac{1}{2+2x+x^2}$
14	0.2	0.8	$\frac{x}{3!}+\frac{4x^2}{5!}+\ldots+\frac{n^2}{(2n+1)!}x^n$	12	$\frac{1}{4}\left(\frac{x+1}{\sqrt{x}}sh\sqrt{x}-ch\sqrt{x}\right)$
15	0.1	0.8	$\frac{x^2}{2}-\frac{x^4}{12}+\ldots+(-1)^{n+1}\frac{x^{2n}}{2n(2n-1)}$	18	$xarctgx-\ln\sqrt{1+x^2}$

16. Посчитать k–количество цифр в десятичной записи целого неотрицательного числа n.
17. Переменной t присвоить значение 1 или 0 в зависимости от того, является ли натуральное число k степенью 3.
18. Дано n вещественных чисел. Вычислить разность между максимальным и минимальным из них.
19. Дана непустая последовательность различных натуральных чисел, за которой следует ноль. Определить порядковый номер наименьшего из них.
20. Даны целое n>0 и последовательность из n вещественных чисел, среди которых есть хотя бы одно отрицательное число. Найти величину наибольшего среди отрицательных чисел этой последовательности.
21. Дано n вещественных чисел. Определить, образуют ли они возрастающую последовательность.
22. Дана последовательность из n целых чисел. Определить, со скольких отрицательных чисел она начинается.
23. Определить k — количество трёхзначных натуральных чисел, сумма цифр которых равна n ($1 \le n \le 27$). Операции деления (**/, div, mod**) не использовать.
24. Вывести на экран в возрастающем порядке все трёхзначные числа, в десятичной записи которых нет одинаковых цифр (операции деления не использовать).

Лабораторная работа № 4
РАЗРАБОТКА ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ МАССИВОВ

Цель лабораторной работы: изучить свойства компоненты TStringGrid. Написать программу с использованием массивов.

1. Краткие сведения

1.1. Работа с массивами

Массив есть упорядоченный набор однотипных элементов, объединенных под одним именем. Каждый элемент массива обозначается именем, за которым в квадратных скобках следует один или несколько индексов, разделенных запятыми, например: a[1], bb[I], c12[I, j*2], q[l, 1, I*j-l]. В качестве индекса можно использовать любые порядковые типы за исключением LongInt.

Тип массива или сам массив определяются соответственно в разделе типов (Type) или переменных (Var) с помощью ключевого слова **Array** следующим образом:

Array [описание индексов] **of** <тип элементов массива >

Примеры описания массивов:

```
Const N=20;  // Задание максимального значения индекса;
Type
   TVector = array [1..N] of real;        // Описание типа одномерного массива;
Var
  A : TVector;                    //A - массив типа TVector;
   Ss  : array[1..10] of integer;  //Ss - массив из десяти целых чисел;
   Y : array[1..5, 1..10] of char; //Y -двумерный массив символьного типа.
```

Элементы массивов могут использоваться в выражениях так же, как и обычные переменные, например:

```
F:= 2*a[3] + a[ss[l] + 1]*3;
A[n]:= 1+sqrt(abs(a[n-1]));
```

1.2. Компонент TStringGrid

При работе с массивами ввод и вывод информации на экран удобно организовывать в виде таблиц, используя компонент TStringGrid. Последний предназначен для отображения информации в виде двумерной таблицы, каждая ячейка

которой представляет собой окно однострочного редактора (аналогично окну TEdit). Доступ к информации осуществляется с помощью свойства

Cells[ACol, ARow : integer] : string;

где ACol, ARow - индексы элементов двумерного массива. Свойства ColCount и RowCount устанавливают количество строк и столбцов в таблице, а свойства FixedCols и FixedRows задают количество строк и столбцов фиксированной зоны. Фиксированная зона выделена другим цветом, и в нее запрещен ввод информации с клавиатуры.

1.3. Пример выполнения задания

Задание: создать программу для определения вектора:

$$\vec{Y} = A \times \vec{B},$$

где A – квадратная матрица; а Y, B – одномерные массивы.

Элементы вектора Y определяются по формуле:

$$Y_i = \sum_{j=1}^{N} A_{ij} \times B_i .$$

Значения N вводить в компонент TEdit, A и B – в компонент TStringGrid. Результат, после нажатия кнопки типа TButton, вывести в компонент типа TStringGrid.

Форма приложения приведена на рис. 4.1.

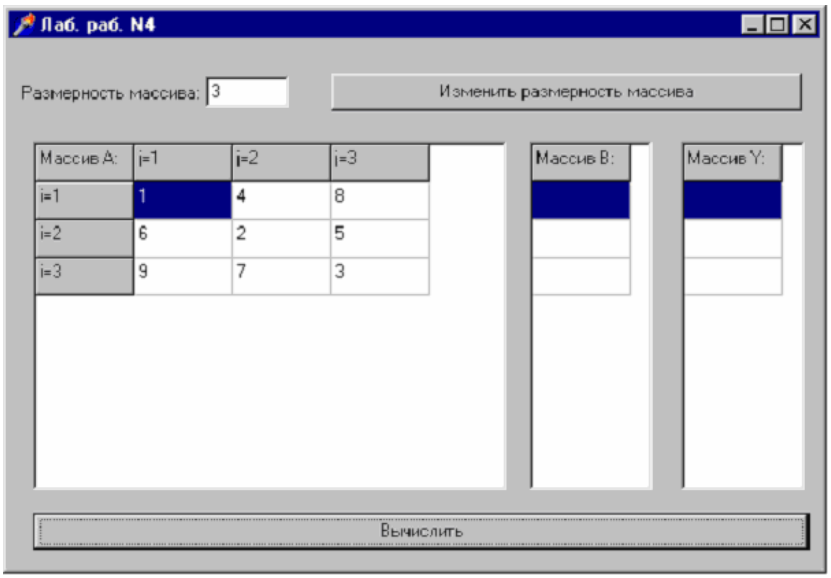


Рис. 4.1. Форма приложения

1.3.1. Настройка компонента TStringGrid

Для установки компонента TStringGrid на форму необходимо на странице Additional палитры компонентов щелкнуть мышью по пиктограмме  . После этого щелкните мышью в нужном месте формы. Захватывая кромки компонента, отрегулируйте его размер. В инспекторе объектов значения свойств ColCount и RowCount установите 2, а FixedCols и FixedRows установите 1. Так как компоненты StringGrid2 и StringGrid3 имеют только один столбец, то у них: ColCount=1, RowCount=2, FixedCols=0 и FixedRows=1. По умолчанию в компонент TStringGrid запрещен ввод информации с клавиатуры, поэтому необходимо свойство Options goEditing для компонентов StringGrid1 и StringGrid2 установить в положение True.

Текст программы:

```
Unit LabRab_4;
Interface
Uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, ExtCtrls, Grids;

type
  TInputForm = class(TForm)
    StringGrid1: TStringGrid;
    StringGrid2: TStringGrid;
    StringGrid3: TStringGrid;
    Label1: TLabel;
    Edit1: TEdit;
    Button1: TButton;
    Button2: TButton;
    Procedure FormCreate(Sender: TObject);
    Procedure Button1Click(Sender: TObject);
    Procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
  const
    Nmax = 10; // Максимальная размерность массива
  type
    TMas1 = array[1..Nmax] of Extended; // Объявление типа одномерного массива
    TMas2 = array[1..Nmax, 1..Nmax] of Extended; // Объявление типа двумерного
массива
  var
    InputForm: TInputForm;
    A : TMas2;       // Объявление двумерного массива
    B, Y : TMas1;  // Объявление одномерных массивов
    N, i, j : integer;

  Implementation
  {$R *.dfm}

  Procedure TInputForm.FormCreate(Sender: TObject);
  begin
    N := 3; // Размерность массива
    Edit1.Text := IntToStr(N);
  // Задание числа строк и столбцов
    StringGrid1.ColCount   := N + 1;
    StringGrid1.RowCount := N + 1;
    StringGrid2.RowCount := N + 1;
    StringGrid3.RowCount := N + 1;
  // Ввод в левую верхнюю ячейку таблицы названия массива
    StringGrid1.Cells[0, 0] := 'Массив A';
    StringGrid2.Cells[0, 0] := 'Массив B';
    StringGrid3.Cells[0, 0] := 'Массив Y';
  // Заполнение верхнего и левого столбцов поясняющими подписями
    for i:=1 to N do
    begin
      StringGrid1.Cells[0, i] := 'i='+ IntToStr(i);
      StringGrid1.Cells[i, 0] := 'j='+ IntToStr(i);
    end;
  end;

  Procedure TInputForm.Button1Click(Sender: TObject);
  begin
    N := StrToInt(Edit1.Text);
  // Задание числа строк и столбцов в таблицах
    StringGrid1.ColCount := N + 1;
    StringGrid1.RowCount := N + 1;
```

```
  StringGrid2.RowCount := N + 1;
  StringGrid3.RowCount := N + 1;
// Заполнение верхнего и левого столбцов поясняющими подписями
  for i:=1 to N do
  begin
    StringGrid1.Cells[0, i] := 'i=' + IntToStr(i);
    StringGrid1.Cells[i, 0] := 'j=' + IntToStr(i);
  end;
end;

Procedure TInputForm.Button2Click(Sender: TObject);
var
  s : extended;
begin
// Заполнение массива A элементами из таблицы StringGrid1
  for i:=1 to N do
  for j:=1 to N do
    A[i, j] := StrToFloat(StringGrid1.Cells[j, i]);
// Заполнение массива B элементами из таблицы StringGrid2
  for i:=1 to N do
    B[i] := StrToFloat(StringGrid2.Cells[0, i]);
// Умножение массива A на массив B
  for i:=1 to N do
  begin
    s := 0;
    for j:=1 to N do
    s := s + A[i, j]*B[j];
    Y[i] := s;
// Вывод результата в таблицу StringGrid3
    StringGrid3.Cells[0, i] := FloatToStrF(Y[i], ffFixed, 6, 2);
  end;
end;

end.
```

2. Постановка задачи

Разработать приложение обработки и представления информации в табличной форме в сооветствии с индивидуальным заданием.

3. Задания

Во всех заданиях скалярные переменные вводить с помощью компонента типа TEdit с соответствующим пояснением в виде компонента типа TLabel. Скалярный результат выводить в виде компонента TLabel. Массивы представлять на форме в виде компонентов TStringGrid, в которых 0-й столбец и 0-ю строку использовать для отображения индексов массивов. Вычисления выполнять после нажатия кнопки типа TButton.

1. Задана матрица размером $N \times M$. Получить массив B, присвоив его *k*-му элементу значение 0, если все элементы *k*-го столбца матрицы нулевые, и значение 1 в противном случае.
2. Задана матрица размером $N \times M$. Получить массив B, присвоив его *k*-му элементу значение 1, если элементы *k*-ой строки матрицы упорядочены по убыванию и значение 0, в противном случае.
3. Задана матрица размером $N \times M$. Получить массив B, присвоив его *k*-му элементу значение 1, если *k*-ая строка матрицы симметрична, и значение 0 в противном случае.
4. Задана матрица размером $N \times M$. Определить *k* - количество "особых" элементов матрицы, считая элемент "особым", если он больше суммы остальных элементов своего столбца.
5. Задана матрица размером $N \times M$. Определить *k* - количество "особых" элементов матрицы, считая элемент “особым”, если в его строке слева от него находятся элементы меньшие его, а справа – большие.
6. Задана символьная матрица размером $N \times M$. Определить *k* - количество различных элементов матрицы (т. е. повторяющиеся элементы считать один раз).
7. Дана матрица размером $N \times M$. Упорядочить ее строки по неубыванию их первых элементов.
8. Дана матрица размером $N \times M$. Упорядочить ее строки по неубыванию суммы их элементов.
9. Дана матрица размером $N \times M$. Упорядочить ее строки по неубыванию их наибольших элементов.
10. Определить, является ли заданная квадратная матрица n-го порядка симметричной относительно побочной диагонали.

11. Для матрицы размером $N \times M$ вывести на экран все ее седловые точки. Элемент матрицы называется седловой точкой, если он является наименьшим в своей строке и одновременно наибольшим в своем столбце или, наоборот.
12. В матрице n-го порядка переставить строки так, чтобы на главной диагонали матрицы были расположены элементы, наибольшие по абсолютной величине.
13. В матрице n-го порядка найти максимальный среди элементов, лежащих ниже побочной диагонали, и минимальный среди элементов, лежащих выше главной диагонали.
14. В матрице размером $N \times M$ поменять местами строку, содержащую элемент с наибольшим значением со строкой, содержащей элемент с наименьшим.
15. Из матрицы *n*-го порядка получить матрицу порядка *n-1* путем удаления из исходной матрицы строки и столбца, на пересечении которых расположен элемент с наибольшим по модулю значением.
16. Дан массив из *k* символов. Вывести на экран сначала все цифры, входящие в него, а затем все остальные символы, сохраняя при этом взаимное расположение символов в каждой из этих двух групп.
17. Дан массив, содержащий от 1 до *k* символов, за которым следует точка. Вывести этот текст в обратном порядке.
18. Дан непустой массив из цифр. Вывести на экран цифру, наиболее часто встречающуюся в этом массиве.
19. Отсортировать элементы массива *X* по возрастанию.
20. Элементы массива *X* расположить в обратном порядке.
21. Элементы массива *X* циклически сдвинуть на *k* позиций влево.
22. Элементы массива *X* циклически сдвинуть на *n* позиций вправо.
23. Преобразовать массив *X* по следующему правилу: все отрицательные элементы массива перенести в начало, а все остальные – в конец, сохраняя исходное взаимное расположение, как среди отрицательных, так и среди остальных элементов.
24. Элементы каждого из массивов *X* и *Y* упорядочены по неубыванию. Объединить элементы этих двух массивов в один массив *Z* так, чтобы они снова оказались упорядоченными по неубыванию.

Лабораторная работа № 5
РАЗРАБОТКА ПРИЛОЖЕНИЯ ОБРАБОТКИ СТРОКОВОЙ ИНФОРМАЦИИ

Цель лабораторной работы: изучить методы программирования с использованием строк и правила работы с компонентами TListBox и TComboBox. Написать программу работы со строками.

1. Краткие сведения

1.1. Типы данных для работы со строками

Короткие строки типа ShortString и String[N]

Короткие строки имеют фиксированное количество символов. Строка ShortString может содержать 255 символов. Строка String[N] может содержать N символов, но не более 255. Первый байт этих переменных содержит длину строки.

Длинная строка типа String

При работе с этим типом данных память выделяется по мере необходимости (динамически) и может занимать всю доступную программе память. Вначале компилятор выделяет для переменной 4 байта, в которых размещается номер ячейки памяти, начиная с которой будет располагаться символьная строка. На этапе выполнения программа определяет необходимую длину цепочки символов и обращается к ядру операционной системы с требованием выделить необходимую память.

Процедуры и функции для работы с короткими и длинными строками представлены в приложении.

Широкая строка типа WideString

Введена для обеспечения совместимости с компонентами, основанными на OLE-технологии. От типа String отличается только тем, что для представления каждого символа используется не один, а два байта.

Нуль-терминальная строка типа PChar

Представляет собой цепочку символов, ограниченную символом #0. Максимальная длина строки ограничена только доступной программе памятью. Нуль-терминальные строки широко используются при обращениях к *API*-функциям *Windows (API - Application Program Interface* - интерфейс прикладных программ**)**.

Представление строки в виде массива символов

Строка может быть описана как массив символов. Если массив имеет нулевую границу, он совместим с типом PChar.

```
Var
   MasS : array[1..100] of Char;
```

В отличие от нуль-терминальной строки здесь длина имеет фиксированное значение и не может меняться в процессе выполнения программы.

1.2. Компонент TListBox

Компонент TListBox представляет собой список, элементы которого выбираются при помощи клавиатуры или мыши. Список элементов задается свойством Items, методы Add, Delete и Insert которого используются для добавления, удаления и вставки строк. Объект Items (TString) хранит строки, находящиеся в списке. Для определения номера выделенного элемента используется свойство ItemIndex.

1.3. Компонент TComboBox

Комбинированный список TComboBox представляет собой комбинацию списка TListBox и редактора TEdit, поэтому практически все свойства заимствованы у этих компонентов. Для работы с окном редактирования используется свойство Text как в TEdit, а для работы со списком выбора - свойство Items как в TListBox. Существует. Пять модификаций компонента, определяемых его свойством Style. В модификации csSimple список всегда раскрыт, в остальных он раскрывается после нажатия кнопки справа от редактора.

1.4. Компонент TBitBtn

Компонент TBitBtn расположен на странице ***Additional*** палитры компонентов и представляет собой разновидность стандартной кнопки TButton. Его отличительная

особенность - наличие растрового изображения на поверхности кнопки, которое определяется свойством Glyph. Кроме того, имеется свойство Kind, которое задает одну из 11 стандартных разновидностей кнопок. Нажатие любой из них, кроме bkCustom и bkHelp закрывает модальное окно и возвращает в программу результат `mr***` (например, bkOk - mrOk). Кнопка bkClose закрывает главное окно и завершает работу программы.

1.5. Обработка событий

Обо всех происходящих в системе событиях таких, как создание формы, нажатие кнопки мыши или клавиатуры и т. д., ядро *Windows* информирует окна путем посылки соответствующих сообщений. Среда Delphi позволяет принимать и обрабатывать большинство таких сообщений. Каждый компонент содержит обработчики сообщений на странице Events инспектора объектов.

Для создания обработчика события необходимо раскрыть список компонентов в верхней части окна инспектора объектов и выбрать необходимый компонент. Затем, на странице Events, нажатием левой клавиши мыши выбрать обработчик и дважды щелкнуть по его левой (белой) части. В ответ Delphi активизирует окно текста программы и покажет заготовку процедуры обработки выбранного события.

Каждый компонент имеет свой набор обработчиков событий, однако некоторые из них присущи большинству компонентов. Наиболее часто применяемые события представлены в табл. 5.1.

Таблица 5.1

Наиболее часто генерируемые события

Событие	Описание события
1	2
OnActivate	Возникает при активации формы
OnCreate	Возникает при создании формы (компонент TForm). В обработчике данного события следует задавать действия, которые должны происходить в момент создания формы, например установка начальных значений

Окончание табл. 5.1

1	2
OnKeyPress	Возникает при нажатии кнопки на клавиатуре. Параметр Key имеет тип Char и содержит ASCII-код нажатой клавиши (клавиша Enter клавиатуры имеет код #13, клавиша Esc - #27 и т.д.). Обычно это событие используется в том случае, когда необходима реакция на нажатие одной из клавиш
OnKeyDown	Возникает при нажатии клавиши на клавиатуре. Обработчик этого события получает информацию о нажатой клавише и состоянии клавиш Shift, Alt и Ctrl, а также о нажатой кнопке мыши. Информация о клавише передается параметром Key, который имеет тип Word
OnKeyUp	Является парным событием для OnKeyDown и возникает при отпускании ранее нажатой клавиши
OnClick	Возникает при нажатии кнопки мыши в области компонента
OnDblClick	Возникает при двойном нажатии кнопки мыши в области компонента

1.6. Пример выполнения задания

Задание: написать программу подсчета числа слов в произвольной строке. В качестве разделителя может быть любое число пробелов. Для ввода строк и работы с ними использовать компонент типа TComboBox. Ввод строки заканчивать нажатием Enter. Для выхода из программы использовать кнопку Close.

Форма приложения приведена на рис. 5.1.

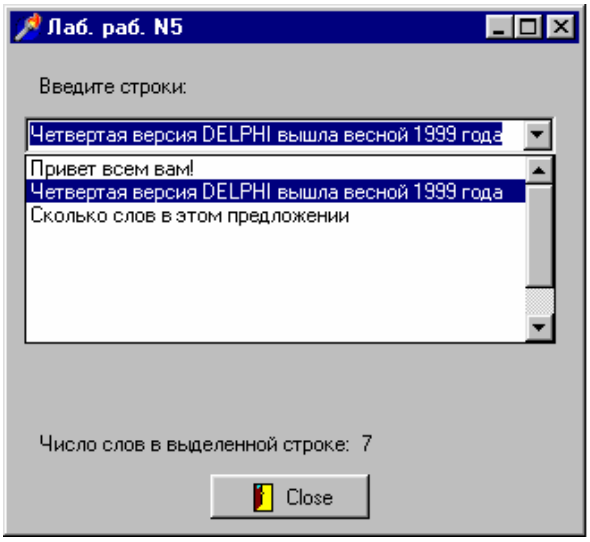


Рис. 5.1. Форма приложения

Текст программы:

```
Unit LabRab_5;
Interface
```

```
Uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons;
type
  TUnit_Lab5 = class(TForm)
    ComboBox1: TComboBox;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    BitBtn1: TBitBtn;
    Button1: TButton;
    Procedure FormActivate(Sender: TObject);
    Procedure ComboBox1KeyPress(Sender: TObject; var Key: Char);
    Procedure ComboBox1Click(Sender: TObject);
  Private
    { Private declarations }
  Public
    { Public declarations }
  end;
var
  Unit_Lab5: TUnit_Lab5;

Implementation
{$R *.dfm}

// Обработка события активизации формы
Procedure TUnit_Lab5.FormActivate(Sender: TObject);
begin
  ComboBox1.SetFocus; //Передача фокуса ComboBox1
end;

// Обработка события нажатия левой клавиши мыши
Procedure TUnit_Lab5.ComboBox1KeyPress(Sender: TObject; var Key: Char);
begin
  if Key =#13 then //Если нажата клавиша Enter то...
  begin
//  Строка из окна редактирования заносится в список выбора
    ComboBox1.Items.Add(ComboBox1.Text);
    ComboBox1.Text := ''; //Очистка окна редактирования
  end;
end;

Procedure TUnit_Lab5.ComboBox1Click(Sender: TObject);
var
  St : string;
  n, i, nst, ind  : integer;
begin
  n   := 0; //Содержит число слов выбранной строки
  ind := 0; //Содержит число слов
//Определение номера выбранной строки
```

```
  nSt := ComboBox1.ItemIndex;
//Занесение выбранной строки в переменную st
  St := ComboBox1.Items[nst];
//Просмотр всех символов строки st
  for i := 1 to Length(St) do
  begin
    Case ind of
      0: if St[i] <>'' then
         begin
            //Если встретился символ после пробела
            //число слов увеличивается на единицу
            ind := 1;
           inc(n);
         end;
// Если встретился пробел после символов
       1: if St[i] ='' then ind := 0;
     end; //Case
   Label3.Caption := IntToStr(n); //Вывод числа слов в Label3
end;

end.
```

2. Постановка задачи

Разработать приложение с использованием компонент, управляющих представлением строковой информации в соответствии с индивидуальным заданием.

3. Задания

Во всех заданиях исходные данные вводить с помощью компонента TEdit в компонент TListBox либо с помощью свойства Text в свойство Items компонента строки заканчивать нажатием клавиши Enter. Для выхода из программы использовать кнопку Close. Для расчетов вводить несколько различных строк.

1. Дана строка, состоящая из групп нулей и единиц. Каждая группа отделяется от другой одним или несколькими пробелами. Найти количество групп с пятью символами.
2. Дана строка, состоящая из групп нулей и единиц. Найти и вывести на экран самую короткую группу.
3. Дана строка, состоящая из групп нулей и единиц. Подсчитать количество символов в самой длинной группе.
4. Дана строка, состоящая из групп нулей и единиц. Найти и вывести на экран группы

с четным количеством символов.

5. Дана строка, состоящая из групп нулей и единиц. Подсчитать количество единиц в группах с нечетным количеством символов.

6. Дана строка, состоящая из букв, цифр, запятых, точек, знаков “+”, “-“ . Выделить подстроку, которая соответствует записи целого числа (т.е. начинается со знака “+” или “-” и внутри подстроки нет букв, запятых и точек).

7. Дана строка символов, состоящая из букв, цифр, запятых, точек, знаков ”+" и "-". Выделить подстроку, которая соответствует записи вещественного числа с фиксированной точкой.

8. Дана строка символов, состоящая из букв, цифр, запятых, точек, знаков ”+”и "-". Выделить подстроку, которая соответствует записи вещественного числа с плавающей точкой.

9. Дана строка символов, состоящая из произвольных десятичных цифр, разделенных пробелами. Вывести на экран числа этой строки в порядке возрастания их значений.

10. Дана строка символов, состоящая из произвольных десятичных цифр, разделенных пробелами. Вывести четные числа этой строки.

11. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Вывести на экран слова этого текста в порядке, соответствующем латинскому алфавиту.

12. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Вывести на экран порядковый номер слова, накрывающего *k*-ю позицию (если на *k*-ю позицию попадает пробел, то номер предыдущего слова).

13. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Разбить исходную строку на две подстроки, причем первая длиной *k* символов (если на *k*-ю позицию попадает слово, то его следует отнести ко второй строке, дополнив первую пробелами до *k* позиций).

14. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Вывести на экран порядковый номер слова максимальной длины и номер позиции строки, с которой оно начинается.

15. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Вывести на экран порядковый номер слова минимальной длины и количество символов в этом слове.

16. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. В каждом слове заменить первую букву на прописную.

17. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Удалить первых *k* слов из строки, сдвинув на их место последующие слова строки.

18. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Поменять местами *i*-е и *j*-е слова.

19. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Поменять местами первую и последнюю буквы каждого слова.

20. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Заменить буквы латинского алфавита на соответствующие им буквы русского алфавита.

21. Дана строка символов S1 S2 …Sm, в которой могут встречаться цифры, пробелы, буква "E" и знаки "+", "-". Известно, что первый символ S1 является цифрой. Из данной строки выделить подстроки, разделенные пробелами. Определить, является ли первая подстрока числом. Если да, то выяснить: целое или вещественное число, положительное или отрицательное.

22. Дана строка символов, содержащая некоторый текст на русском языке. Разработать программу форматирования этого текста, т.е. его разбиения на отдельные строки (по *k* символов в каждой строке) и выравнивания по правой границе путем вставки между отдельными словами необходимого количества пробелов.

23. Дана строка символов, содержащая некоторый текст на русском языке. Заменить буквы русского алфавита на соответствующие им буквы латинского алфавита.

24. Дана строка символов, содержащая некоторый текст. Разработать программу, которая определяет, является ли данный текст палиндромом, т.е. читается ли он слева направо так же, как и справа налево (например, “А роза упала на лапу Азора”).

Лабораторная работа № 6

РАЗРАБОТКА ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ ЗАПИСЕЙ И ФАЙЛОВ

***Цель лабораторной работы**:* изучить правила работы с компонентами TOpenDialog и TSaveDialog. Написать программу с использованием файлов и данных типа запись.

1. Краткие сведения

1.1. Программирование с использованием переменных типа запись

Запись - это структура данных, объединяющая элементы одного или различных типов, называемыми полями. Записи удобны для создания структурированных баз данных с разнотипными элементами, например:

```
Type
 TStudent = record  //Объявление типа запись
            Fio     : string[20];                 //Поле ФИО
            Group : integer;                      //Поле номера студ. группы
            Ocn     : array[1..3] of integer; //Поле массива оценок
   end:
 Var
      Student: TStudent;          //Объявление переменной типа запись
```

Доступ к каждому полю осуществляется указанием имени записи и поля, разделенных точкой, например:

```
Student.Fio    := 'Иванов А.И.';  //Внесение данных в поля записи
Student.Group := 107218;
```

Доступ к полям можно осуществлять также при помощи оператора With:

```
 With Student do
 begin
   Fio      := 'Иванов А.И.';
   Group := 107218;
 end;
```

1.2. Работа с файлами

Файл - это именованная область данных на внешнем физическом носителе. В *Object Pascal* различают три вида файлов в зависимости от способа их организации и

доступа к элементам: *текстовые, типизированные и нетипизированные.*

Текстовой файл - это файл, состоящий из строк. Примером текстового файла может служить файл исходного текста программы в Delphi (расширение *.pas). Для работы с текстовым файлом должна быть описана соответствующая файловая переменная:

```
var F : TextFile;
```

Типизированные файлы имеют строго заданную их описанием структуру, когда все элементы имеют фиксированный и одинаковый размер. Это свойство типизированных файлов позволяет получить доступ к любому компоненту файла по его порядковому номеру. Элементами такого файла являются, как правило, записи. В описании файловой переменной указывается ее тип:

```
Var F: TStudent;
```

Нетипизированный файл – это файл, в котором данные не имеют определенного типа и рассматриваются, как последовательность байт. Файловая переменная объявляется:

```
Var F: File;
```

Порядок работы с файлами следующий:

```
AssignFile(F, 'FileName.txt');  //Связывание файловой переменной F
                                //с именем дискового файла "FileName.txt"
Rewrite(F);                     //Создание нового файла
Reset(F);                       //Открытие уже существующего файла
Read(F, Stud);                  //Чтение данных из файла
Write(F, Stud)                  //Запись данных в файл
CloseFile(F);                   //Закрытие файла
```

1.3. Процедуры работы с файлами

AssignFile(var F; FileName: string) - связывает файловую переменную `F` и файл с именем FileName.

Reset(var F[: File; RecSize: word]) - открывает существующий файл. При открытии нетипизированного файла `RecSize` задает размер элемента файла.

Rewrite(var F[: File; RecSize: word]) - создает и открывает новый файл.

Append(var F: TextFile) - открывает текстовый файл для дописывания текста в конец файла.

Read(F, v1[, v2...vn]) - чтение значений переменных, начиная с текущей позиции для типизированных файлов, и строк для текстовых.

Write(F, v1[, v2,...vn]) - запись значений переменных начиная с текущей позиции для типизированных файлов и строк для текстовых.

CloseFile(F) - закрывает ранее открытый файл.

Rename(var F; NewName: string) - переименовывает неоткрытый файл любого типа.

Erase(var F) - удаляет неоткрытый файл любого типа.

Seek(var F; NumRec: Longint) - для нетекстового файла устанавливает указатель на элемент с номером NumRec.

SetTextBuf(var F: TextFile; var Buf[;Size: word]) - для текстового файла устанавливает новый буфер ввода-вывода объема Size.

Flush(var F: TextFile) - немедленная запись в файл содержимого буфера ввода-вывода.

Truncate(var F) - урезает файл, начиная с текущей позиции.

LoResult: integer - код результата последней операции ввода-вывода.

FilePos(var F): longint - для нетекстовых файлов возвращает номер текущей позиции. Отсчет ведется от нуля.

FileSize(var F): longint - для нетекстовых файлов возвращает количество компонентов в файле,

Eoln(var F: TextFile): boolean - возвращает True, если достигнут конец строки.

Eof(var F)): boolean - возвращает True, если достигнут конец файла.

SeekEoln(var F: TextFile): boolean - возвращает True, если пройден последний значимый символ в строке или файле, отличный от пробела или знака табуляции.

SeekEoln(var F: TextFile): boolean - то же, что и SeekEoln, но для всего файла.

BlockRead(var F: File; var Buf; Count: word[; Result: word])

BlockWrite(var F : File; var Buf; Count: word[; Result: word]) - соответственно процедуры чтения и записи переменной Buf с количеством Count блоков,

1.4. Компоненты TOpenDialog и TSaveDialog

Компоненты TOpenDialog и TSaveDialog находятся на странице DIALOGS. Все компоненты этой страницы являются невизуальными, т.е. не видны в момент работы

программы. Поэтому их можно разместить в любом удобном месте формы. Оба рассматриваемых компонента имеют идентичные свойства и отличаются только внешним видом. После вызова компонента появляется диалоговое окно, с помощью которого выбирается имя программы и путь к ней. В случае успешного завершения диалога имя выбранного файла и маршрут поиска содержится в свойстве FileName. Для фильтрации файлов, отображаемых в окне просмотра, используется свойство Filter, а для задания расширения файла, в случае, если оно не задано пользователем - свойство DefaultExt. Если необходимо изменить заголовок диалогового окна – используется свойство Title.

1.5. Пример выполнения задания

Задание: написать программу, вводящую в файл или читающую из файла ведомость абитуриентов, сдавших вступительные экзамены. Каждая запись должна содержать фамилию, а также оценки по физике, математике и сочинению. Вывести список абитуриентов, отсортированный в порядке уменьшения их среднего балла и записать эту информацию в текстовой файл.

Настройка компонентов TOpenDialog a TSaveDialag

Для установки компонент TOpenDialog и TSaveDialog на форму необходимо, на странице Dialogs палитры компонентов щелкнуть мышью соответственно по пиктограммам  или  и поставить их в любое свободное место формы. Установка фильтра производится следующим образом. Выбрав соответствующий компонент, дважды щелкнуть по правой части свойства Filter инспектора объектов. Появится окно Filter Editor, в левой части которого записывается текст, характеризующий соответствующий фильтр, а в правой части - маску.

Для OpenDialog1 установим значения маски как показано на рис. 6.1. Формат *.dat означает что, будут видны все файлы с расширением dat, а формат *.* - что будут видны все файлы (с любым именем и с любым расширением).

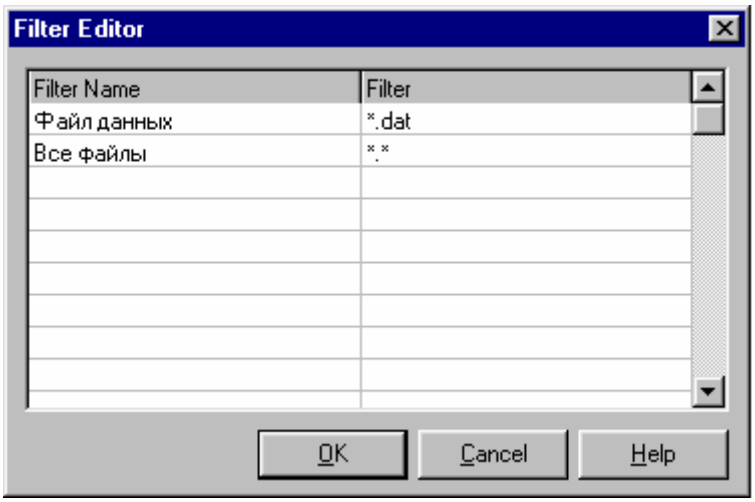


Рис. 6.1. Установка фильтра для файлов

Для того чтобы файл автоматически записывался с расширением .dat в свойстве DefaultExt запишем требуемое расширение - .dat. Аналогичным образам настроим SaveDialog1 для текстового файла (расширение *.txt).

Работа с программой

После запуска программы на выполнение появится диалоговое окно программы. Кнопка “Ввести запись” видна не будет. Необходимо создать новый файл записей, нажав на кнопку "Создать" или открыть ранее созданный, нажав кнопку "Открыть".

После этого станет видна кнопка "Ввести запись" и можно будет вводить записи. При нажатии на кнопку "Сортировать" будет проведена сортировка ведомости по убыванию среднего балла и диалоговое окно примет вид, как на рис. 6.2. Затем при нажатии на кнопку “Сохранить” будет создан текстовой файл, содержащий отсортированную ведомость. Файл записей закрывается одновременно с программой при нажатии на кнопку "Close".

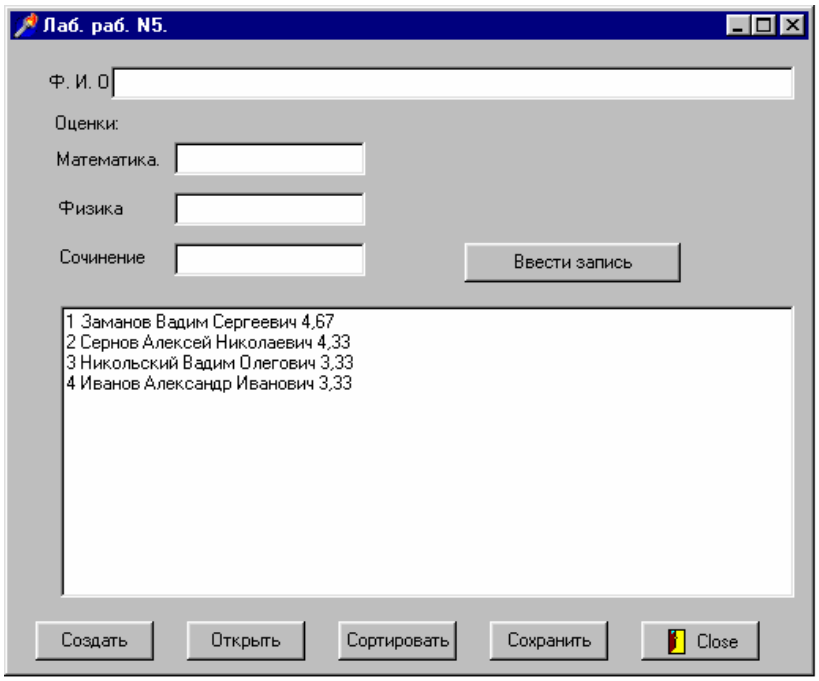


Рис. 6.2. Форма приложения

Текст программы:

```
Unit LabRab_6;
Interface
Uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, Buttons, StdCtrls;

Type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    Button5: TButton;
    BitBtn1: TBitBtn;
    OpenDialog1: TOpenDialog;
    SaveDialog1: TSaveDialog;
    Memo1: TMemo;
    Procedure FormCreate(Sender: TObject);
    Procedure Button1Click(Sender: TObject);
```

```
    Procedure Button2Click(Sender: TObject);
    Procedure Button3Click(Sender: TObject);
    Procedure Button4Click(Sender: TObject);
    Procedure Button5Click(Sender: TObject);
    Procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

Type
  TStudent = record
    FIO  : string[40];              //Поле Ф.И.О.
    otc  : array[1..3] of word;  //Поле массива оценок
    sball  : extended;              //Поле среднего балла
  end;

var
  Fz   : file of TStudent;                //Файл типа запись
  Ft   : TextFile;                        //Текстовой файл
  Stud : array[1..100] of TStudent; //Массив записей
  Nzap : integer;                         //Номер записи
  FileNameZ, FileNameT : string; //Имена файлов

var
  Form1: TForm1;

Implementation
{$R *.dfm}

Procedure TForm1.FormCreate(Sender: TObject);
begin
  Edit1.Text := '';
  Edit2.Text := '';
  Edit3.Text := '';
  Edit4.Text := '';
  Memo1.Clear;
  Button1.Hide; //Сделать невидимой кнопку "Ввести запись"
  Nzap := 0;
end;

Procedure TForm1.Button1Click(Sender: TObject);
begin
  nzap := nzap + 1;
  with stud[nzap] do
  begin
    FIO    := Edit1.Text;
    otc[1] := StrToInt(Edit2.Text);
    otc[2] := StrToInt(Edit3.Text);
```

```
    otc[3] := StrToInt(Edit4.Text);
    sball  := (otc[1] + otc[2] + otc[3])/3;
    Memo1.Lines.Add(fio + '' + IntToStr(otc[1 ]) +
      ' ' + IntToStr(otc[2]) + '' + IntToStr(otc[3]));
  end;

  Write(fz, Stud[nzap]);   //Запись в файл
  Edit1.Text := '';
  Edit2.Text := '';
  Edit3.Text := '';
  Edit4.Text := '';
end;

Procedure TForm1.Button2Click(Sender: TObject);
begin
   OpenDialog1.Title := 'Создать новый файл';
// Изменение заголовка окна диалога
   if OpenDialog1.Execute then
// Выполнение стандартного диалога выбора имени файла
  begin
    FileNameZ := OpenDialog1.FileName;
//  Возвращение имени дискового файла
    AssignFile(Fz, FileNameZ);
//  Связывание файловой переменной Fz с именем файла
    Rewrite(Fz);  //Создание нового файла
    Button1.Show; //Сделать видимой кнопку "Ввести запись"
  end;
end;

Procedure TForm1.Button3Click(Sender: TObject);
begin
  if OpenDialog1.Execute then
//Выполнение стандартного диалога выбора имени файла
  begin
    FileNameZ := OpenDialog1.FileName;
//  Возвращение имени дискового файла
    AssignFile(Fz, FileNameZ);
//  Связывание файловой переменной Fz с именем файла
    Reset(Fz); //Открытие существующего файла
  end;

  While not EOF(fz) do
  begin
    nzap := nzap + 1;
    Read(fz, stud[nzap]);  //Чтение записи из файла
    with stud[nzap] do
      Memo1.Lines.Add(fio + '' + IntToStr(otc[1]) +
      ' ' + IntToStr(otc[2]) + '' + IntToStr(otc[3]));
    end;
    Button1.Show; //Сделать видимой кнопку "Ввести запись"
```

```
  end;

Procedure Tform1.Button4Click(Sender: TObject);
// Сортировка записей
var
  i, j : word;
  st   : Tstudent;
begin
  for i := 1 to nzap-1 do
  for j := i + 1 to nzap do
  if Stud[i].sball < Stud[j].sball then
  begin
    st := Stud[i];
    Stud[i] := Stud[j];
    Stud[j] := st;
  end;

  Memo1.Clear;
  for i := 1 to nzap do
// Вывод в окно Memo1 отсортированных записей
  with stud[i] do
    Memo1.Lines.Add(IntToStr(i) + " +
    fio + " + FloatToStrF(sball, ffFixed, 4, 2));
end;

Procedure Tform1.Button5Click(Sender: TObject);
// Сохраниение результатов сортировки в текстовом файле
var
  i : word;
begin
  if SaveDialog1.Execute then
// Выполнение стандартного диалога выбора имени файла
  begin
    FileNameT := SaveDialog1.FileName;
//  Возвращение имени дискового файла
    AssignFile(Ft, FileNameT);
//  Связывание файловой переменной Ft с именем файла
    Rewrite(Ft); //Открытие нового текстового файла
  end;

  for i:=1 to nzap do
  with stud[i] do
    Writeln(Ft, i:4, '.', fio, sball:8:2);
//  Запись в текстовой файл
    CloseFile(Ft); //Закрытие текстового файла
end;

Procedure TForm1.BitBtn1Click(Sender: TObject);
begin
  CloseFile(fz);
```

```
// Закрытие файла записей при нажатии на кнопку "Close"
end;
{
Procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  CloseFile(fz);
// Закрытие файла записей при нажатии на кнопку
end;
}
end.
```

2. Постановка задачи

Разработать приложение, поддерживающее чтение и сохранение информации в файлах в соответствии с индивидуальным заданием.

3. Задания

В программе предусмотреть сохранение вводимых данных в файле и возможность чтения из ранее сохраненного файла. Результаты выводить в окно просмотра и в текстовой файл.

1. В магазине формируется список лиц, записавшихся на покупку товара повышенного спроса. Каждая запись этого списка содержит: порядковый номер, Ф.И.О., домашний адрес покупателя и дату постановки на учет. Удалить из списка все повторные записи, проверяя Ф.И.О и домашний адрес.
2. Список товаров, имеющихся на складе, включает в себя наименование товара, количество единиц товара, цену единицы и дату поступления товара на склад. Вывести в алфавитном порядке список товаров, хранящихся больше месяца, стоимость которых превышает 1000000 руб.
3. Для получения места в общежитии формируется список студентов, который включает Ф.И.О. студента, группу, средний балл, доход на члена семьи. Общежитие в первую очередь предоставляется тем, у кого доход на члена семьи меньше двух минимальных зарплат, затем остальным в порядке уменьшения среднего балла. Вывести список очередности предоставления мест в общежитии.
4. В справочной автовокзала хранится расписание движения автобусов. Для каждого рейса указаны его номер, тип автобуса, пункт назначения, время отправления и прибытия. Вывести информацию о рейсах, которыми можно воспользоваться для

прибытия в пункт назначения раньше заданного времени.

5. На междугородной АТС информация о разговорах содержит дату разговора, код и название города, время разговора, тариф, номер телефона в этом городе и номер телефона абонента. Вывести по каждому городу общее время разговоров с ним и сумму.
6. Информация о сотрудниках фирмы включает: Ф.И.О., табельный номер, количество проработанных часов за месяц, почасовый тариф. Рабочее время свыше 144 часов считается сверхурочным и оплачивается в двойном размере. Вывести размер заработной платы каждого сотрудника фирмы за вычетом подоходного налога, который составляет 12 % от суммы заработка.
7. Информация об участниках спортивных соревнований содержит: наименование страны, название команды, Ф.И.О. игрока, игровой номер, возраст, рост, вес. Вывести информацию о самой молодой, рослой и легкой команде.
8. Для книг, хранящихся в библиотеке, задаются: регистрационный номер книги, автор, название, год издания, издательство, количество страниц. Вывести список книг с фамилиями авторов в алфавитном порядке, изданных после заданного года.
9. Различные цеха завода выпускают продукцию нескольких наименований. Сведения о выпущенной продукции включают: наименование, количество, номер цеха. Для заданного цеха необходимо вывести количество выпущенных изделий по каждому наименованию в порядке убывания количества.
10. Информация о сотрудниках предприятия содержит: Ф.И.О., номер отдела, должность, дату начала работы. Вывести списки сотрудников по отделам в порядке убывания стажа.
11. Ведомость абитуриентов, сдавших вступительные экзамены в университет, содержит: Ф.И.О., адрес, оценки. Определить количество абитуриентов, проживающих в г. Минске и сдавших экзамены со средним баллом не ниже 4.5, вывести их фамилии в алфавитном порядке.
12. В справочной аэропорта хранится расписание вылета самолетов на следующие сутки. Для каждого рейса указаны: номер рейса, тип самолета, пункт назначения, время вылета. Вывести все номера рейсов, типы самолетов и времена вылета для заданного пункта назначения в порядке возрастания времени вылета.
13. У администратора железнодорожных касс хранится информация о свободных местах в поездах дальнего следования на ближайшую неделю в следующем виде:

дата выезда, пункт назначения, время отправления, число свободных мест. Оргкомитет международной конференции обращается к администратору с просьбой зарезервировать *m* мест до города *N* на *k-й* день недели с временем отправления поезда не позднее *i* часов вечера. Вывести время отправления или сообщение о невозможности выполнить заказ в полном объеме.

14. Ведомость абитуриентов, сдавших вступительные экзамены в университет, содержит: Ф.И.О. абитуриента, оценки. Определить средний балл по университету и вывести список абитуриентов, средний балл которых выше среднего балла по университету. Первыми в списке должны идти студенты, сдавшие все экзамены на 5.
15. В радиоателье хранятся квитанции о сданной в ремонт радиоаппаратуре. Каждая квитанция содержит следующую информацию: наименование группы изделий (телевизор, радиоприемник и т.п.), марка изделия, дата приемки в ремонт, состояние готовности заказа (выполнен, не выполнен). Вывести информацию о состоянии заказов на текущие сутки по группам изделий.
16. Разработать программу формирования ведомости об успеваемости студентов. Каждая запись этой ведомости должна содержать: номер группы, Ф.И.О. студента, оценки за последнюю сессию. Вывести списки студентов по группам. В каждой группе Ф.И.О. студентов должны быть расположены в порядке убывания среднего балла.
17. В исполкоме формируется список учета нуждающихся в улучшении жилищных условий. Каждая запись этого списка содержит: порядковый номер, Ф.И.О., величину жилплощади на одного члена семьи и дату постановки на учет. По заданному количеству квартир, выделяемых по данному списку в течение года, вывести весь список с указанием ожидаемого года получения квартиры.
18. Имеется список женихов и список невест. Каждая запись списка содержит пол, имя, возраст, рост, вес, а также требования к партнеру: наименьший и наибольший возраст, наименьший и наибольший вес, наименьший и наибольший рост. Объединить эти списки в список пар с учетом требований к партнерам без повторений женихов и невест.
19. В библиотеке имеется список книг. Каждая запись этого списка содержит: фамилии авторов, название книги, год издания. Вывести информацию о книгах, в названии которых встречается некоторое ключевое слово (ввести с клавиатуры).

20. В магазине имеется список поступивших в продажу автомобилей. Каждая запись этого списка содержит: марку автомобиля, стоимость, расход топлива на 100 км, надежность (число лет безотказной работы), комфортность (отличная, хорошая, удовлетворительная). Вывести перечень автомобилей, удовлетворяющих требованиям покупателя, которые вводятся с клавиатуры в виде некоторого интервала допустимых значений.
21. Каждая запись списка вакантных рабочих мест содержит: наименование организации, должность, квалификация (разряд или образование), стаж работы по специальности, заработная плата, наличие социального страхования (да/нет), продолжительность ежегодного оплачиваемого отпуска. Вывести список рабочих мест в соответствии с требованиями клиента.
22. В технической службе аэропорта имеется справочник, содержащий записи следующей структуры: тип самолета, год выпуска, расход горючего на 1000 км. Для определения потребности в горючем техническая служба запрашивает расписание полетов. Каждая запись расписания содержит следующую информацию: номер рейса, пункт назначения, дальность полета.

    Вывести суммарное количество горючего, необходимое для обеспечения полетов на следующие сутки.

23. Список группы студентов содержит следующую информацию: Ф.И.О., рост и вес. Вывести Ф.И.О студентов, рост и вес которых являются в списке уникальными.
24. Для участия а конкурсе на замещение вакантной должности сотрудника фирмы желающие подают следующую информацию: Ф.И.О., год рождения, образование (среднее, специальное, высшее), знание иностранных языков (английский, немецкий, французский, владею свободно, читаю и перевожу со словарем), владение компьютером (MSDOS, Windows), стаж работы, наличие рекомендаций. Вывести список претендентов в соответствии с требованиями руководства фирмы.

Лабораторная работа № 7
РАЗРАБОТКА ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ ПОДПРОГРАММ И МОДУЛЕЙ

Цель лабораторной работы: изучить возможности Delphi для написания подпрограмм и создания модулей. Составить и отладить программу, использующую внешний модуль Unit с подпрограммой.

1. Краткие сведения

1.1. Общие сведения

Подпрограмма - это именованная, определенным образом оформленная группа операторов, которая может бить вызвана любое количество раз из любой точки основной программы. Подпрограммы используется в том случае, когда одна и та же, последовательность операторов в тексте программы повторяется несколько раз. Эта последовательность заменяется вызовом подпрограммы, содержащей необходимые операторы. Подпрограммы также применяются для создания специализированных библиотечных модулей, содержащих набор подпрограмм определенного назначения, для использования их другими программистами.

Подпрограммы подразделяются на процедуры и функции. Процедура имеет следующую структуру:

Procedure <имя процедуры> ([список формальных параметров]);
Const [описание используемых констант];
Type [описание используемых типов];
Var [описание используемых переменных];
begin
<операторы>
end;

Процедуры и функции могут быть использованы в качестве формальных параметров подпрограмм. Для этого определяется тип:

Type <имя>= function ([список формальных параметров]):<тип результата>;

или

Type <имя>= procedure ([список формальных параметров]);

Имя процедуры или функции должно быть уникальным в пределах программы. Список формальных параметров необязателен и может отсутствовать. Если же он есть, то в нем перечисляются через точку с запятой имена формальных параметров и их типы. Имеется три вида формальных параметров параметры-значения, параметры-переменные, параметры-константы. При вызове подпрограммы передача данных для этих видов осуществляются по-разному. Параметры-значения копируются, и подпрограмма работает с их копией, поэтому при вызове на месте такого параметра можно ставить арифметическое выражение. При использовании параметров-переменных (в описании перед ними ставится `Var`) и параметров-констант в подпрограмму передается адрес, и она работает с самой переменной. С помощью параметров-переменных подпрограмма передает результаты своей работы вызывающей программе.

В функциях используется специальная переменная `Result`, интерпретируемая как значение, которое вернет в основную программу по окончании работы функции.

В язык Object Pascal встроен ряд наиболее часто употребляемых процедур и функций, которые являются частью языка и вызываются без предварительного определения в разделе описаний.

1.2. Использование модулей

Модуль - автономно компилируемая программная единица, включающая в себя процедуры, функции, а также различные компоненты раздела описаний. Структура модуля содержит следующие основные части: заголовок, интерфейсная часть, исполняемая, инициирующая и завершающая.

Заголовок состоит из зарезервированного слова Unit и следующего за ним имени модуля, которое должно совпадать с именем дискового файла. Использование имени модуля в разделе Uses основной программы приводит к установлению связи модуля с основной программой.

Интерфейсная часть расположена между зарезервированными словами Interface и Implementation и содержит объявление тех объектов модуля, которые должны быть доступны другим программам.

Исполняемая часть начинается зарезервированным словом implementation и содержит описание процедур и функций, объявленных в интерфейсной части. Она может также содержать вспомогательные типы, константы, переменные, процедуры и функции которые, будут использоваться только в исполняемой части и не будут доступны внешним программам.

Инициирующая часть начинается зарезервированным словом Initialization и содержит операторы, которые исполняются до передачи управления основной программе.

Завершающая часть начинается зарезервированным словом Finalization и выполняется в момент окончания работы программы. Инициализирующая и завершающая части модуля используются крайне редко.

1.3. Пример выполнения задания

Задание: написать программу вывода на экран таблицы функции, которую, оформить в виде процедуры. В качестве функции использовать по выбору Tg(x), Ch(x) и Sin(x).

Создание модуля

Создавая модуль, следует обратить внимание на то, что он не должен иметь своей формы. Система Delphi при начальной загрузке автоматически создает шаблон программы, имеющий в своем составе форму, файл проекта и т.д. Так как модуль состоит только из одного файла, то необходимо перед его созданием уничтожить заготовку файла проекта и форму. Для этого в меню File выбрать Close All, файл проекта не сохранять.

Для создания модуля в меню File выбрать File New, и затем в репозитории – пиктограмму 

. В результате будет создан файл с заголовком Unit Unit1. Имя модуля можно сменить на другое, отвечающее внутреннему содержанию модуля, например Unit MatFunc. Затем необходимо сохранить файл с именем, совпадающим с именем заголовка модуля: MatFunc.pas. Следует обратить внимание на то, что имя файла должно совпадать с именем модуля, иначе Delphi не сможет подключить его к другой программе.

Подключение модуля

Для того чтобы подключить модуль к проекту необходимо в меню Project выбрать опцию Add to Project... и выбрать файл, содержащий модуль. После этого в разделе Uses добавить имя подключаемого модуля - MatFunc. Теперь в проекте можно использовать функции, содержащиеся в модуле.

Форма приложения приведена на рис. 7.1.

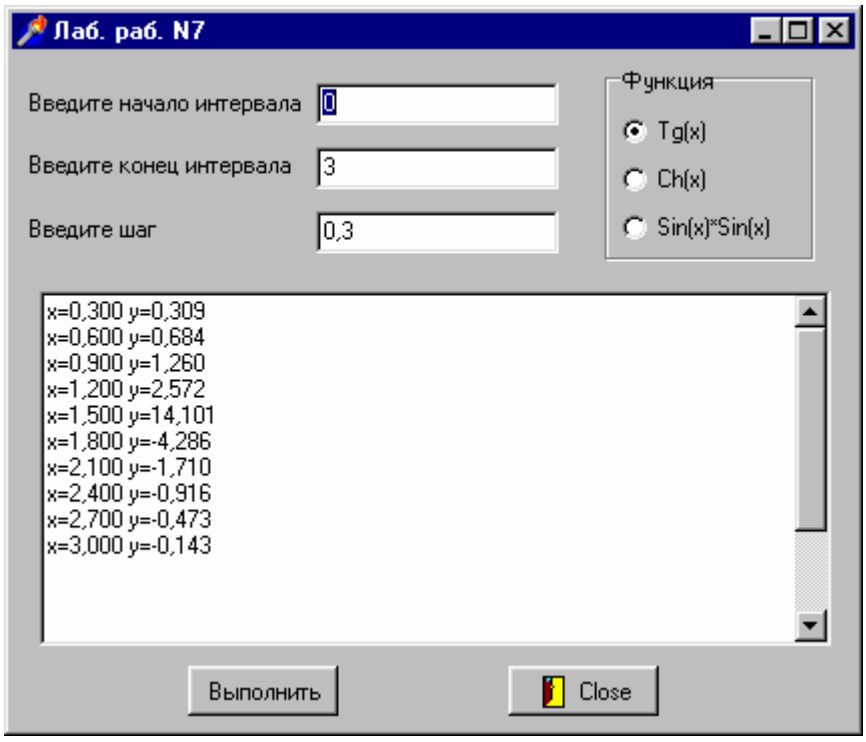


Рис. 7.1. Форма приложения

Тексты модуля:

```
Unit MatFunc;
Interface
  Function Tg(x: extended) : extended;  //Функция для вычисления тангенса
  Function Ch(x: extended) : extended;  //Функция для вычисления гиперболического
                                        //синуса
  Function Sin2(x: extended) : extended;//Функция для вычисления квадрата синуса

Implementation

Function Tg(x: extended) : extended;
begin
  Result := sin(x)/cos(x);
end;
```

```
Function Ch(x: extended) : extended;
begin
  Result := (exp(x)-exp(-x))/2;
end;

Function Sin2(x: extended) : extended;
begin
  Result := sqr(sin(x));
end;

end.
```

Текст вызывающей программы:

```
Unit LabRab_7;
Interface
Uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls, MatFunc;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Memo1: TMemo;
    Button1: TButton;
    Button2: TButton;
    RadioGroup1: TRadioGroup;

    Procedure FormCreate(Sender: TObject);
    Procedure Button1Click(Sender: TObject);
    Procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

type
  func = function(x:extended) : extended;

var
  Form1: TForm1;

implementation
```

```
{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
  Edit1.Text := '0';
  Edit2.Text := '2';
  Edit3.Text := '0.2';
  Memo1.Clear;
  RadioGroup1.ItemIndex := 0;
end;

procedure Tabl(f: func; xn, xk, h: extended);
var
  x, y : extended;
begin
  x := xn;
  Repeat
    y := f(x);
    Form1.Memo1.Lines.Add('x=' + FloatToStrF(x,ffFixed, 8, 2) +
                    '  y=' + FloatToStrF(y, ffFixed, 8, 2));
    x := x + h;
  Until (x > xk);
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  xn, xk, h : extended;
begin
  xn := StrToFloat(Edit1.Text);
  xk := StrToFloat(Edit2.Text);
  h  := StrToFloat(Edit3.Text);

  Case RadioGroup1.ItemIndex of
    0 : Tabl(tg, xn, xk, h);
    1 : Tabl(ch, xn, xk, h);
  end;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  Halt; //Завершение приложения
end;

end.
```

2. Постановка задачи

Разработать приложение, состоящее из нескольких Unit в соответствии с индивидуальным заданием.

3. Задания

По указанию преподавателя выберите вариант задачи из заданий, приведенных в работе 3. Предусмотрите возможность выбора функции, для которой будет рассчитываться таблица. Функции поместите в отдельный модуль. Вызывать выбранную функцию должна процедура, использующая в качестве входного параметра имя соответствующей функции.

Лабораторная работа № 8

РАЗРАБОТКА ПРИЛОЖЕНИЯ С ВЫДАЧЕЙ РЕЗУЛЬТАТОВ ВЫЧИСЛЕНИЙ В ВИДЕ ГРАФИКОВ

Цель лабораторной работы: изучить возможности построения графиков с помощью компонента отображения графической информации TChart. Написать и отладить программу построения на экране графика заданной функции.

1. Краткие сведения

1.1. Построение графика с помощью компонента TChart

Обычно результаты расчетов представляются в виде графиков и диаграмм. Система Delphi имеет мощный пакет стандартных программ вывода на экран и редактирования графической информации, который реализуется с помощью визуально отображаемого на форме компонента TChart (рис. 8.1).

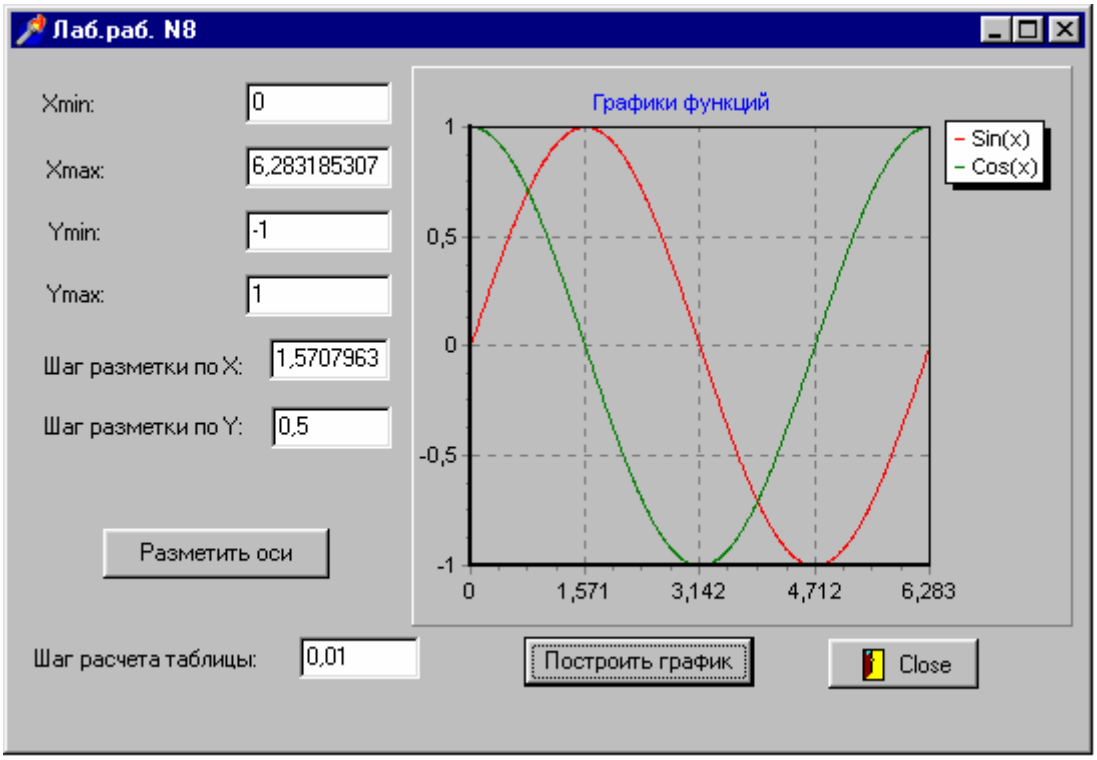


Рис. 8.1. Форма приложения

Построение графика (диаграммы) производится после вычисления таблицы значений функции y = f(x) на интервале [Xmin, Xmax] с данным шагом. Полученная таблица передается в специальный двумерный массив Seriesk (k-номер графика)

компонента TChart с помощью метода Add. Компонент TChart осуществляет всю работу по отображению графиков, переданных в объект Seriesk: строит и размечает оси, рисует координатную сетку, подписывает название осей и самого графика, отображает переданную таблицу в виде всевозможных графиков или диаграмм. При необходимости, с помощью встроенного редактора EditingChart в компоненту TChart передаются данные о толщине, стиле и цвете линий, параметрах шрифта подписей, шагах разметки координатной сетки и другие настройки. В процессе работы программы изменение параметров возможно через обращение к соответствующим свойствам компонента TChart. Так, например, свойство TChart.BottomAxis содержит значение максимального предела нижней оси графика и при его изменении во время работы автоматически изменяется изображение графика (см. ниже).

1.2. Пример выполнения задания

Задание: составить программу, отображающую графики функций sin(x) и cos(x) на интервале [Xmin..Xmax]. Предусмотреть возможность изменения разметки координатных осей, а также шага построения таблицы.

Настройка формы

Панель диалога программы организуется в виде, представленном на рисунке 8.1. Для ввода исходных данных используются окна TEdit. Компонент TChart вводится в форму путем нажатия пиктограммы  ,расположенной на закладке Additional палитры компонент.

Работа с компонентом TChart

Для изменения параметров компонента TChart необходимо дважды щелкнуть по нему мышью в окне формы. Появится окно редактирования EditingChart (см. рис. 8.2). Для создания нового объекта Series1 нужно щелкнуть по кнопке Add на странице Series. В появившимся диалоговом окне TeeChartGalleri выбрать пиктограмму с надписью Line (график выводится в виде линий). Если нет необходимости представления графика в трехмерном виде, отключить независимый переключатель 3D. После нажатия на кнопку OK появится новая серия с название Seriesl. Для изменения названия cktletn нажать кнопку Title.

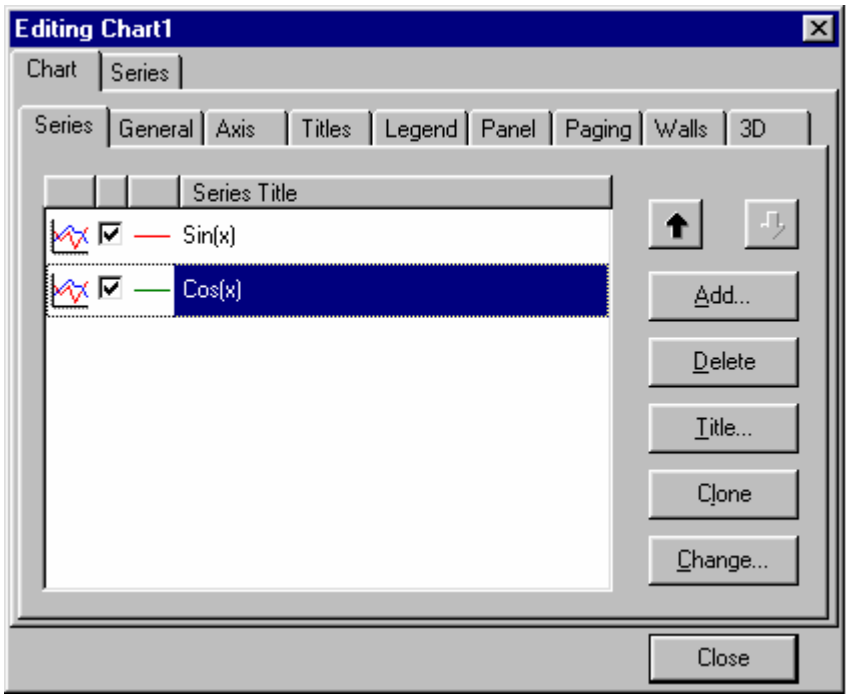


Рис. 8.2. Задание параметров компонента TChart

В появившимся однострочном редакторе набрать имя отображаемой функции "sin(x)". Аналогичным образом создать объект Series2 для функции cos(x).

Для изменения надписи над графиком на странице Titles в многострочном редакторе набрать: “Графики функций".

Для разметки осей выбрать страницу Axis и научиться устанавливать параметры настройки осей. Нажимая различные кнопки меню, познакомиться с другими возможностями EditingChart.

Написание процедуры обработки события создания формы

В данном месте программы устанавливаются начальные пределы и шаг разметки координатных осей. Когда свойство Chart1.BottomAxis Automatic имеет значение False, автоматическая установка параметров осей не работает.

Написание процедур обработки событий нажатия на кнопки

Процедура TForm1.Button1Click обрабатывает нажатие кнопки *"Установить оси"*. Процедура TForm1.Button2Click обрабатывает нажатие кнопки *"Построить график"*. Для добавления координат точек `(X, Y)` из таблицы значений в двумерный массив объекта `Seriesk` используется процедура Series1.AddXY(Const AXValue, AYValue: Double; Const AXLabel: String; AColor: TColor): Longint, где AXValue, AYValue - координаты точки по осям X и Y; AXLabel может принимать значение ‘’;

AColor задает цвет линий (если равен clTeeColor, то принимается цвет, определенный при проектировании формы).

Текст программы:

```
Unit LabRab_8;
Interface
Uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, TeeProcs, TeEngine, Chart, Series;

type
  TForm1 = class(TForm)
    Chart1: TChart;
    Button1: TButton;
    Button2: TButton;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Edit6: TEdit;
    Edit7: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Series1: TLineSeries;
    Series2: TLineSeries;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  Xmin, Xmax, Ymin, Ymax, Hx, Hy, h : extended;

Implementation
{$R *.dfm}

Procedure TForm1.FormCreate(Sender: TObject);
begin
  Xmin := 0;
```

```
  Xmax := 2 * Pi;
  Ymin := -1;
  Ymax := 1;
  Hx   := pi/2;
  Hy   := 0.5;
  h    := 0.01;
  Edit1.Text := FloatToStr(Xmin);
  Edit2.Text := FloatToStr(Xmax);
  Edit3.Text := FloatToStr(Ymin);
  Edit4.Text := FloatToStr(Ymin);
  Edit5.Text := FloatToStr(Hx);
  Edit6.Text := FloatToStr(Hy);
  Edit7.Text := FloatToStr(h);

  Chart1.BottomAxis.Automatic := False;
  Chart1.BottomAxis.Minimum := Xmin;
  Chart1.BottomAxis.Maximum := Xmax;

  Chart1.LeftAxis.Automatic := False;
  Chart1.LeftAxis.Minimum   := Ymin;
  Chart1.LeftAxis.Maximum   := Ymax;
  Chart1.BottomAxis.Increment := Hx;
  Chart1.LeftAxis.Increment   := Hy;
end;

Procedure TForm1.Button1Click(Sender: TObject);
var
  x, y1, y2 : extended;
begin
  Series1.Clear;
  Series2.Clear;
  Xmin := StrToFloat(Edit1.Text);
  Xmax := StrToFloat(Edit2.Text);
  h := StrToFloat(Edit7.Text);
  x := Xmin;
  Repeat
    y1 := sin(x);
    Series1.AddXY(x, y1, '', clTeeColor);
    y2 := cos(x);
    Series2.AddXY(x, y2, '', clTeeColor);
    x := x + h;
  Until (x > Xmax);
end;

Procedure TForm1.Button2Click(Sender: TObject);
begin
  Halt //Exit
end;

end.
```

2. Постановка задачи

Разработать приложение отображения результатов вычислений в виде графика

3. Задания

Построить графики функций для соответствующих вариантов из работы №1. Таблицу данных получить путем изменения параметра X с шагом h. Вывод исходных данных организовать через окна TEdit. Самостоятельно выбрать удобные параметры настройки.

Лабораторная работа № 9
РАЗРАБОТКА ПРИЛОЖЕНИЯ, СОСТОЯЩЕГО ИЗ НЕСКОЛЬКИХ ФОРМ

***Цель лабораторной работы*:** изучить основные свойства и методы, связанные с созданием и активизацией форм.

1. Краткие сведения

Сложное приложение может состоять более чем из одной форы. Для добавления в проект новой формы необходимо выполнить команду New Form. В дальнейшем показать нужную форму можно с помощью методов ShowModal – модальный режим, либо Show – немодальный режим. Оба этих режима отличаются тем, что в модальном режиме невозможно одновременно выполнять действия более, чем на одной форме. Напротив, в немодальном режиме имеется возможность свободно переходить с формы на форму без закрытия остальных.

Для реализации визуальных способов изменения данных в программе используются различные компоненты. Одной из часто используемых компонент, является компонента TScrollBar.

Компонент TScrollBar

Это управляющий элемент, расположенный на стандартной закладке палитры компонент с пиктограммой , похожий на полосу скроллинга окна и используемый для изменения числовой величины визуальным изменением положения бегунка компоненты.

Свойствами этого компонента являются:

- Kind = (sbHorisontal, sbVertical) – для задания расположения TScrollBar;
- Min, Max : Integer – для установки минимального и максимального значений изменяемой величины;
- Position : Integer – в этом свойстве находится текущее значение числа;
- LargeChange : TScrollBarInc, SmallChange :TScrollBarInc - с помощью этих свойств соответственно можно устанавливать малый и большой сдвиг бегунка.

При изменении положения бегунка возникает событие OnScroll. Чтобы программным образом устанавливать положение бегунка в заданное место используется метод: SetParams(Aposition, Amin, Amax : integer).

Смешивание цветов

Для получения составного цвета можно использовать смешивание трех составляющие цветов при вызове функции RGB (red,green,blue), например,

Color:= RGB(255, 0, 0); //ярко – красный цвет.

2. Постановка задачи

Разработать приложение, состоящее из нескольких форм и поддерживающее визуальное изменение данных.

3. Задание

Взять за основу задание к лабораторной работе № 7. Добавить к проекту дополнительную форму, на которой реализовать смешивание цветов. При этом должен быть организован диалог приложения, как показано на рис. 9.1.

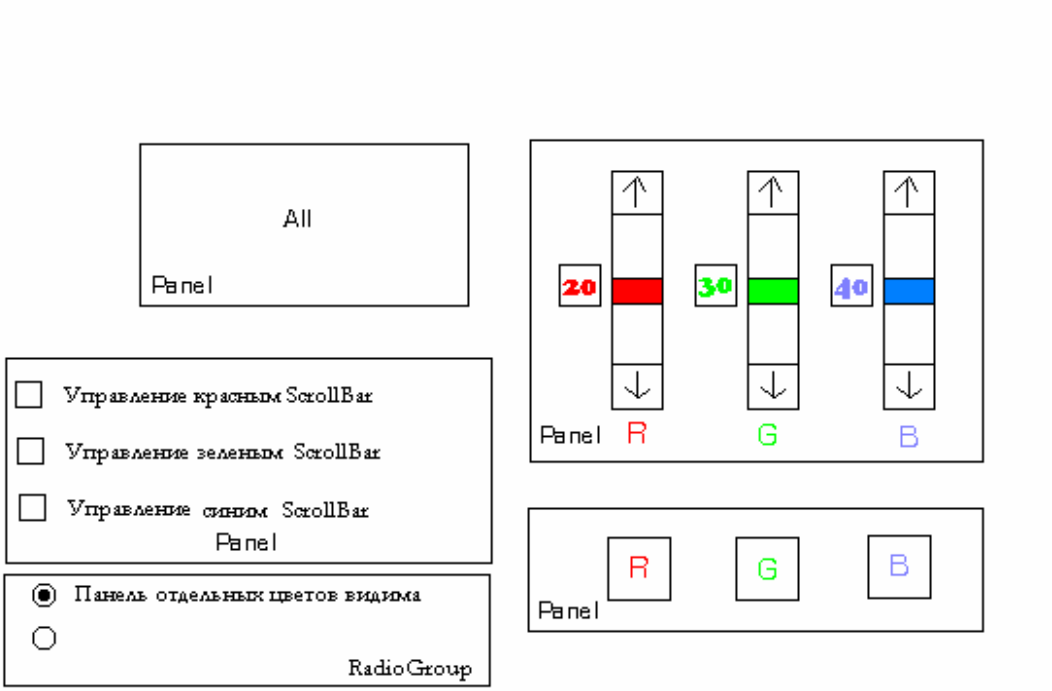


Рис. 9.1. Форма приложения

Выбранный цвет необходимо применить к одной из компонент главной формы.
Замечание. Вызов формы смешивания цветов реализовать в модальном и немодальном режимах.

Лабораторная работа № 10
РАЗРАБОТКА ПРИЛОЖЕНИЯ С СОХРАНЕНИЕМ ПАРАМЕТРОВ И УСТАНОВОК В INI-ФАЙЛАХ

***Цель лабораторной работы*:** изучить возможности автоматического сохранения параметров и установок, принятых в программе.

1. Краткие сведения

Удобным средством запоминания текущих настроек приложения являются ini-файлы. Ini-файлы - это текстовые файлы, предназначенные для хранения информации о формах или о настройках различных приложений. Информация в файле логически группируется в разделы, каждый из которых начинается оператором заголовка, заключенным в квадратные скобки, например, [Desktop]**.** В строках, следующих за заголовком, содержится информация, относящаяся к данному разделу, в виде:

<ключ>=<значение>

Любое приложение можно зарегистрировать в системном реестре и зафиксировать там же текущие настройки приложения (в 32-разрядных Windows и выше). Для 32-разрядных приложений Microsoft не рекомендует работать с Ini-файлами. Ннесмотря на это, и 32-разрядные приложения, наряду с реестром, часто используют эти файлы. Да и разработки Microsoft не обходятся без этих файлов.

Ini-файлы, как правило, хранятся в каталоге Windows, который можно найти с помощью функции GetWindowsDirectory**.**

В Delphi работу с Ini-файлами проще всего осуществлять с помощью создания в приложении объекта типа TIniFile. Этот тип описан в модуле IniFiles**,** который надо подключать к приложению оператором uses (автоматически это не делается).

Создается объект типа TIniFile методом Create(<имя файла>), в который передается имя Ini-файла**,** с которым он связывается.

Для записи значений ключей существует несколько методов: WriteString, WriteInteger, WriteFloat, WriteBool и др. Каждый из них записывает значение соответствующего типа. Объявления всех этих методов очень похожи. Например:

```
procedure WriteString (const Section, Ident, Value: string);
procedure WriteInteger(const Section, Ident: string; Value: Longint);
```

Здесь Section — раздел Ini-файла, Ident — имя ключа, Value — значение ключа. Если соответствующий раздел или ключ отсутствует в файле, он автоматически создается.

Имеются аналогичные методы чтения значений ключей: ReadString, ReadInteger, ReadFloat, ReadBool и др. Например:

```
function ReadString(const Section, Ident, Default: string) : string;
function ReadInteger(const Section, Ident: string; Default: Longint) : Longint;
```

В этих примерах методы чтения возвращают значение ключа Ident раздела Section**.** Параметр Default определяет значение, возвращаемое в случае, если в файле не указано значение соответствующего ключа.

Проверить наличие значения ключа можно методом ValueExists**,** в который передаются имена раздела и ключа. Метод DeleteKey удаляет из файла значение указанного ключа в указанном разделе.

Проверить наличие в файле необходимого раздела можно методом SectionExists. Метод EraseSection удаляет из файла указанный раздел вместе со всеми его ключами. Имеется еще ряд методов, которые можно посмотреть во встроенной справке Delphi.

Рассмотрим на примере, как запоминать и удалять настройки программы. Создадим простое тестовое приложение. Поместим на форму три компонента Button и компонент FontDialog**.**

Первой кнопку присвоим имя (Name) BInst и зададим Install в ее свойстве Caption. Эта кнопка будет имитировать установку программы. Точнее, не саму установку, поскольку копировать файлы с установочной дискеты мы не будем, а только создание Ini-файла в каталоге Windows.

Вторую кнопку назовем BUnInst и зададим UnInstall в ее свойстве Caption. Эта кнопка будет имитировать удаление программы. Здесь мы не будем удалять программу с диска, а только удалим из каталога Windows наш ini-файл**.**

Третью кнопку назовем BFont и зададим Font в ее свойстве Caption. С помощью этой кнопки будем менять имя шрифта, используемого в форме. Имя этого шрифта надо будет запоминать в Ini-файле**,** чтобы в дальнейшем при запуске приложения можно было читать эту настройку и задавать ее форме.

Текст программы:

```
unit LabRab_10;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
  Controls, Forms, Dialogs, StdCtrls, IniFiles;

type
  TForm1 = class(TForm)
    BInst: TButton;
    BUnInst: TButton;
    BFont: TButton;
    FontDialog1: TFontDialog;
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure BInstClick(Sender: TObject);
    procedure BUnInstClick(Sender: TObject);
    procedure BFontClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  Ini  : TIniFile;
  sFile: string;

implementation
{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
var
  APchar: array[0..254] of Char;
begin
//Формирование имени каталога Windows
  GetWindowsDirectory(APchar, 255);
//Формирование имени ini-файла в каталоге Windows
  sFile := string(APchar) + '\My.ini';
  if FileExists(sFile) then
  begin
```

```
    Ini := TIniFile.Create(sFile);
//  Чтение в имя шрифта формы значения ключа Шрифт
    Font.Name := Ini.ReadString('Параметры', 'Шрифт',
                          'MS Sans Serif');
  end;
end;

procedure TForm1.BInstClick(Sender: TObject);
var
  f: File;
begin
//Проверка существования ini-файла
  if (not FileExists(sFile)) then
  begin
//  Создание ini-файла
    AssignFile(f, sFile);
    Rewrite(f);
    CloseFile(f);
    Ini := TIniFile.Create(sFile);
  end;
//Создание раздела Files, ключа main и запись в него имени
//выполняемого файла вместе с путем
  Ini.WriteString('Files','main', ParamStr(0));
//Создание раздела Параметры, ключа Шрифт и
//запись в него имени шрифта формы
  Ini.WriteString('Параметры', 'Шрифт', Font.Name);
end;

procedure TForm1.BUnInstClick(Sender: TObject);
var
  F: File;
begin
//Удаление с диска ini-файла, если он существует
  if FileExists(sFile) then
  begin
    AssignFile(F, sFile);
    Erase(F);
  end;
end;

procedure TForm1.BFontClick(Sender: TObject);
begin
//Выбор шрифта
  if (FontDialog1.Execute) then
  begin
//  Присваивание выбранного шрифта форме
    Font.Assign(FontDialog1.Font);
    if (Ini<>nil) and Ini.ValueExists('Параметры','Шрифт')
      then
//  Запись шрифта в ключ "Шрифт" раздела "Параметры"
```

```
    Ini.WriteString('Параметры','Шрифт', Font.Name);
  end;
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
  if (Ini = nil) then Exit;
//Очистка буфера и запись файла на диск
  Ini.UpdateFile;
//Освобождение памяти
  Ini.Free;
end;

end.
```

В начале текста следует подключение к приложению модуля IniFiles и объявление переменной Ini типа TIniFile, а также переменной sFile**,** в которой будет формироваться имя файла и путь к нему.

```
  Ini  : TIniFile;
  sFile: string;
```

При создании формы приложения в процедуре TForm1.FormCreate формируется имя файла **(**My.Ini**)** вместе с путем к нему - каталогом Windows. Путь Windows определяется функцией GetWindowsDirectory**.**

Далее функцией FileExists проверяется, существует ли этот файл, т.е. проведена ли уже установка программы. Если существует, то создается объект Ini, связанный с этим файлом, и значение MS Sans Serif ключа Шрифт раздела Параметры читается в имя шрифта формы. Тем самым читается настройка, произведенная при предыдущем выполнении приложения.

Теперь рассмотрим процедуру TForm1.BInstClick**,** имитирующую установку программы. В этой процедуре сначала функцией FileExists проверяется, существует ли в каталоге Windows файл My.Ini**.** Если не существует, то последовательным применением функций AssignFile, Rewrite и CloseFile этот файл (пока пустой) создается. Затем создается связанный с этим файлом объект Ini**.** Последующие операторы записывают в этот файл два раздела Files и Параметры с соответствующими ключами.

В ключ main записывается имя приложения с путем к нему. Для этого используется функция ParamStr(0)**.**

В результате в созданный файл записывается, например, такой текст:

```
[Files]
main = D:\DEMO Delphi\DEMO1_INI.EXE

[Параметры]
Шрифт = MS Sans Serif
```

Процедура TForml.BUnInstClick имитирует удаление приложения и его файла настройки. В данном случае просто удаляется Ini-файл, но в настоящем приложении надо было бы прочитать имя файла (или файлов) приложения из раздела Files и удалить их с диска.

Процедура TForm1.FormDestroy, срабатывающая при закрывании формы приложения, сначала методом UpDateFile переписывает содержимое объекта Ini в файл на диске, а затем методом Free удаляет из памяти этот временный объект.

Процедура TForml.BFontClick вызывает стандартный диалог выбора шрифта и если пользователь выбрал шрифт, то он присваивается форме и его имя заносится в ini-файл.

Сохраните свое приложение и запустите его на выполнение. Нажмите кнопку Install. После этого убедитесь в наличии файла My.Ini в каталоге Windows. Можете воспользоваться для этого программой Windows «Проводник» или любой другой. В частности, можно открыть этот файл просто из среды Delphi. При нажатии кнопки UnInstall файл должен удаляться с диска. Проверьте запись в файл настройки шрифта и чтение ее при последующих запусках. Для этого опять нажмите кнопку Install, а затем нажмите кнопку Font и выберите шрифт с каким-нибудь другим именем. Затем закройте свое приложение и запустите его повторно. Вы увидите, что на форме применен тот шрифт, который вы зарегистрировали в файле настройки. Таким образом, приложение проимитировало установку программы, удаление программы и запоминание ее текущих настроек.

После запуска программы и нажатия затем кнопки Install, используя проводник, в директории c:\windows можно увидеть файл MyIni.ini. Содержание файла:

```
[Files]
main = D:\DEMO DELPHI EXAMPLES\
       DEMO1 INIFILES\PROJECTDEMO1INIFILES.EXE

[Параметры]
Шрифт = Times New Roman
```

После запуска программы и нажатия кнопки UnInstall в упомянутой директории файл MyIni.ini отсутствует.

2. Постановка задачи

Разработать приложение, поддерживающее чтение и сохранение настроек в Ini – файлах.

3. Задание

Реализовать чтение и сохранение параметров при запуске и завершении программы. В качестве задания использовать лабораторную работу № 5.

Лабораторная работа № 11
РАЗРАБОТКА ПРИЛОЖЕНИЯ, ПОДДЕРЖИВАЮЩЕГО СОЗДАНИЕ ГРАФИЧЕСКИХ ИЗОБРАЖЕНИЙ

Цель лабораторной работы: изучить основные графические компоненты, их свойства и методы.

1. Краткие сведения

Любая Windows-программа осуществляет вывод информации на экран с помощью GDI (*Graphic Device Interface*). Функции, реализованные в GDI, являются *аппаратно независимыми*. Эти функции взаимодействуют с конкретным устройством не напрямую, а через специальную программу, которая называется *драйвером устройства*. Для любых устройств (мониторов, принтеров, плоттеров и т.д.) используется соответствующий драйвер.

Функции GDI взаимодействуют с драйвером устройства через специальную структуру, называемую *контекстом устройства (Device Context)*. В качестве контекста в Delphi выступает объект Canvas.

В Delphi имеется несколько *независимых классов*, которые определяют средства создания изображений. К ним можно отнести TCanvas - холст, TPen - перо, TBrush - кисть, TFont - шрифт. Данные классы Delphi иногда называют *классами-надстройками*, так как связанные с ними объекты самостоятельно в программе не используются, а выступают как свойства того или иного элемента управления (Form, Edit, …). Рассмотрим основные свойства этих классов.

Класс TPen

С помощью этого класса производится рисование линий и контуров различных геометрических фигур. Перо характеризуется цветом, стилем и толщиной.

Основные свойства класса:

Color: TColor - для задания конкретного цвета. Цвет в Windows задается в формате RGB, т.е. тройкой чисел, определяющих степени интенсивности трех его цветовых составляющих – красной, зеленой и синей. Для задания конкретного цвета используется тип TColor, описанный в Unit Graphics как:

Type TColor = -$7FFFFFFF..$7FFFFFFF,

т.е. для задания конкретного цвета выделяется целое число в 4 байта. Самый крайний байт определяет интенсивность красной составляющей. В шеснадцатиричной системе счисления соответствующие составляющие изменяются в диапазонах:

$00 00 00 00 - $00 00 00 FF - красная составляющая,

$00 00 00 00 - $00 00 FF 00 – зеленая составляющая,

$00 00 00 00 - $00 FF 00 00 – синяя составляющая.

Левый байт задает палитру.

Для наиболее часто используемых цветов определены соответствующие константы. Они разбиваются на 2 группы:

1. Цвета, безотносительно, к какому элементу они применяются, например: clBlack .. clWhite, clNone.
2. Цвета, предназначенные для окрашивания каких-либо деталей изображения: полос скроллинга, фона рабочего окна Windows, фона меню и т.д. Это такие цвета как: clWindows, clMenu и т.д.

Цвета второй группы могут меняться в зависимости от настроек Windows.

Замечания. Получить составной цвет можно также смешав три составляющие при вызове функции RGB:

Color:= RGB(255, 0, 0); //ярко – красный цвет.

Если требуется выделить из смешанного цвета одну из его составляющих, то это можно сделать функциями GetRValue, GetGValue, GetBValue, например: RedValue:= GetRValue(Color).

Style : TPenStyle - задает тип линии путем использования констант:

psSolid,	————
psDash,	- - - - - - - -
psDot,	.....
psDashDot,	—·—·—·—
psDashDotDot	··—··—··—
psClean;	

Width: Integer - задает толщину линий.

Класс TBrush

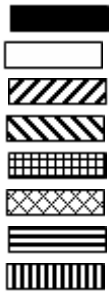
С помощью этого класса задаются характеристики кисти.

Основные свойства, определенные в классе:

Color: TColor - задает цвет кисти. По умолчанию clWhite.

Style: TBrushStyle - определяет стиль кисти. Для задания стиля используются константы:

bsSolid
bsClear
bsBDiagonal
bsFDiagonal
bsCross
bsDiagCross
bsHorizontal
bsVertical



Класс TFont

С помощью этого класса задаются характеристики текста с помощью свойств:

Color: TColor - - задает цвет шрифта. По умолчанию clBlack.

Name: TFontName - - задает название шрифта, например: ‘Arial’.

Size: Integer - задает размер букв.

Style: TFontStyle - задает стиль букв. Для задания стиля используются константы: [fsBold], [fsItalic] [fsUnderline], [fsStrikeOut].

Способы отображения графики

Delphi предоставляет программисту 4 способа отображения графики:

- использование заранее созданных графических изображений;
- создание изображений с помощью графических компонентов;
- создание изображений с помощью примитивов (линия, круг и т.д.) непосредственно во время работы программы.
- представление информации в виде графиков.

1-й способ. Компонента TImage

Если графическое изображение уже создано, например, с помощью графического редактора (например, Paint), то его можно показать с помощью компоненты TImage. В Delphi с помощью этого компонента можно отобразить следующие графические изображения:

1) растровое (*.bmp);
2) пиктограммы (*.ico);
3) типа метафайла (*.wmf);
4) курсора (*.cur).

Вместе с тем известны и другие способы хранения изображений (*.pcx, *.gif, *.tiff, *.jpeg, *.dwg). Для того, чтобы включить изображения других форматов их нужно перевести в формат *.bmp.

Основные свойства компонента TImage:

Canvas - содержит канву для прорисовки изображения;

Center - указывает, надо ли центрировать изображение в границах компонента. Игнорируется, если: AutoSize := True; или Stretch := True; и изображение не является пиктограммой (ICO);

Increment - разрешает/запрещает показ большого изображения по мере его загрузки;

Picture - центральное свойство класса. Служит контейнером изображения TPicture;

Proportional - разрешает/запрещает пропорционально уменьшать высоту и ширину изображения, если оно не может целиком уместиться в рабочей зоне компонента;

Stretch - разрешает/запрещает изменять размер изображения так, чтобы оно целиком заполнило клиентскую область компонента;

Transparent - **з**апрещает/разрешает накладывать собственный фон изображения на фон компонента.

Компонент TImage позволяет поместить графическое изображение в любое место на форме. Собственно картинку можно загрузить во время дизайна в редакторе свойства Picture (Инспектор Объектов). Картинка должна храниться в файле в формате BMP (*bitmap*), WMF (*Windows Meta File*) или ICO (*icon*). При проектировании следует помнить, что изображение, помещенное на форму во время дизайна, включается в файл .DPR и затем прикомпилируется к EXE-файлу. Поэтому такой EXE-файл может получиться достаточно большой. Как альтернативу можно рассмотреть загрузку картинки во время выполнения программы, для этого у свойства Picture (которое является объектом со своим набором свойств и методов) есть специальный метод LoadFromFile.

Пример. По нажатию кнопки необходимо загрузить в компоненту TImage изображение.

Обработчик нажатия кнопки Button1Click выглядит следующим образом:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if OpenPictureDialog1.Execute then
  begin
    Image1.Picture.LoadFromFile(OpenPictureDialog1.FileName);
    Image1.Stretch := True;
```

```
  end;
end;
```

Если изображение, находящееся в TImage, нужно сохранить в файле, можно применить метод SaveToFile, который также принадлежит свойству Picture.

2-й способ. Компоненты TShape, TBevel

С помощью этого способа имеется возможность рисовать простейшие геометрические фигуры (прямоугольник, квадрат, скругленный прямоугольник, скругленный квадрат, эллипс, окружность). Фигура полностью занимает пространство компонента. Если задан квадрат или круг, а размеры элемента по горизонтали и вертикали отличаются, фигура чертится с размером меньшего измерения. Для создания таких фигур используется компонента **TShape**, расположенная на закладке Additional под пиктограммой  .

Могут быть использованы следующие свойства компонента:

Shape : TShapeType = (stRectangle, stSquare, stRoundRect, stRoundSquare, stEllipse, stCircle) – тип геометрической фигуры,

где

stRectangle – прямоугольник,
stSquare- квадрат,
stRoundRect- скругленный прямоугольник,
stRoundSquare- скругленный квадрат,
stEllipse- эллипс,
stCircle- окружность.

Выбранная фигура рисуется на весь экран компонента TShape. Изменение свойства Shape приводит к немедленной перерисовке изображения.

Brush : TBrush – используется для заливки области;
Pen : TPen -- используются для изменения параметров рамки.

Пример:

```
Procedure TForm1.FormCreate( );
begin
  with Shape1 do
  begin
    Shape := stRectangle;    //Фигура - прямоугольник
    Brush.Color := clRed;   //Красный цвет заливки
    Pen.Color := Blue;        //Синий цвет рамки
    Brush.Style := bsHorizontal; //Дискретная заливка в виде горизонтальных линий
    Pen.Style := psSolid;      //Сплошной тип линии рамки
    Pen.Width := 2;            //Толщина линии рамки
  end;
end;
```

Bitmap : TBitmap - позволяет в качестве закраски или заливки использовать растровое изображение, например: Shape1.Brush.Bitmap := Image1.Picture.Bitmap;.

Компонент TBevel

Этот компонент  используется для выделения группы элементов или отделения их друг от друга. Компонент TBevel служит для украшения программ и может принимать вид рамки или линии. Объект предоставляет меньше возможностей по сравнению с TPanel, но не занимает ресурсов. Компонент класса TBevel используют для оформительского дизайна, выделяя группу элементов или отделяя их друг от друга.

Изменения внешнего вида компонента осуществляется с помощью свойств:

Shape : TBevelShape = (bsBox, bsFrame, bsTopLine, bsBottomLine, bsLeftLine, bsRightLine) – геометрия компонента;

Style :TBevelStyle = (bsLowered, bsRaised) – вид (вдавленный, выпуклый) компонента.

3-й способ. Поддержка графических операций низкого уровня

Для создания графических изображений в области некоторых компонент (TForm, TImage, TPaintBox, TPrinter, TListBox, TComboBox, TDrawGrid), используется свойство Canvas. С каждым из перечисленных компонент связано событие OnPaint. Это событие возникает, когда ядру Windows необходимо перерисовать содержимое компонента (например, при активизации формы, когда один из перечисленных компонентов становится видимым). Чтобы отрисовать графическое изображение внутри рабочей области перечисленных компонент нужно обработать событие OnPaint, т.е. записать соответствующий обработчик.

Можно воспроизвести на соответствующих компонентах любые графические объекты без использования компонент TImage, TShape, TLabel.

Класс TCanvas

Класс TCanvas имеет свойства:

Pen: TPen - - устанавливает *цвет*, *толщину*, *стиль* линий и границ геометрических фигур, например:

with Canvas do

```
begin
  Pen.Color := clBlue;
  Pen.Width := 2;
  Pen.Style := psDash;
end;
```

Brush: TBrush – это свойство позволяет устанавливать цвет и шаблон кисти;

Font: TFont – это свойство позволяет устанавливать параметры текста;

PenPos: TPoint - выдает текущую позицию пера;

Pixels : TColor - двухмерный массив, содержащий цвета пикселей, например:

```
Procedure TForm1.Button1Click( );
Var
  i, j : LongInt;
  begin
    Button1.Visible := false;
    with Canvas do
    begin
      for i:=1 to Width do
        for j:=1 to Height do
        Pixels[i,j] := i*j;
        Button1.Visible := true;
    end;
  end;
```

Методы класса TCanvas

Большое количество методов класса TCanvas позволяют отображать различные геометрические фигуры с помощью свойства Pen. Если фигура замкнута, то ее поверхность закрашивается Brush. Все тексты изображаются шрифтом Font.

В процессе работы программы эти характеристики можно изменять. Так:

Arc(x1, y1, x2, y2, x3, y3, x4, y4) - рисует дугу (рис. 11.1).

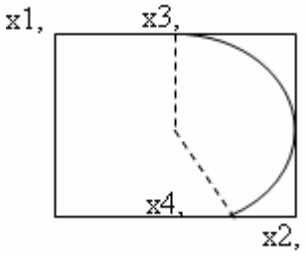


Рис. 11.1

Chord(x1, y1, x2, y2, x3, y3, x4, y4) - рисует сегмент из дуги эллипса и хорды (рис. 11.2).

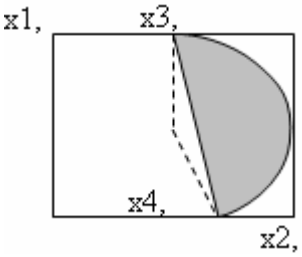


Рис. 11.2

Ellipse(x1, y1, x2, y2) - рисует эллипс;

FillRect(Rect) - закрашивание прямоугольника;

MoveTo (x, y) - - перемещает перо в точку с координатами x, y;

LineTo(x, y) - рисует линию из текущего положения пера в точку с координатами x и y;

Pie(x1, y1, x2, y2, x3, y3, x4, y4) - - рисует сектор эллипса;

Poligon(Point: array of TPoint) - вычерчивание заданного многоугольника.

Пример.

```
var
  P: array[1..3] of TPoint;
begin
  P[1].x :=  10;    P[1].y := 300;
  P[2].x := 200;   P[2].y := 300;
  P[3].x := 100;   P[3].y :=  20;
  Canvas.Poligon(P);
end;
```

Poliline(Point: array of TPoint) - рисует ломаную;

RoundRect(x1, y1, x2, y2, x3, y3) - вычерчивание и заполнение прямоугольника со скругленными углами (рис. 11.3);

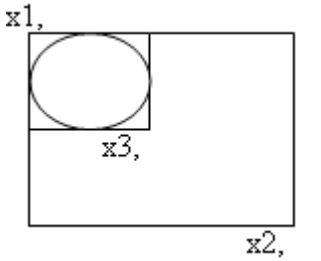


Рис. 11.3

TextOut(x, y, S: String) - осуществляет вывод строки;

Метод **TextRec** выводит текст только внутри указанного прямоугольника. Длину и высоту текста можно узнать с помощью функций *TextWidth* и *TextHeight;*

Draw(x, y, Graphic: TGraphic) - прорисовка графического объекта Graphic так, чтобы левый верхний угол располагался в (x, y). Объект Graphic может быть типа Bitmap, Icon и Metafile;

StretchDraw(Rect: TRect; Graphic: TGraphic) - вычерчивание и масштабирование объекта Graphic до полного заполнения Rect.

Пример. На форме имеется Image1. С помощью свойства Picture в нее помещена картинка. Требуется переместить эту картинку в другое положение.

```
Procedure TForm1.FormPaint( );
begin
  with Canvas do
  begin
    Draw (0, 0, Image1.Picture.Bitmap);
    StretchDraw (Rect(250,0,350,50), Image1.Picture.Bitmap);
  end;
end;
```

Как правило, все графические операции осуществляются не на форме, а посредством специальных графических компонент, например компонента Image, который позволяет разместить на экране растровое изображение, пиктограмму, метафайл, либо собственное изображение.

Для более простых графических операций используется компонент TPaintBox.

2. Постановка задачи

Разработать приложение, поддерживающее основные функции простейшего графического редактора.

3. Задание

Разработать приложение, содержащее три формы – три способа представления графической информации.

На 1-й форме продемонстрировать отображение графических картинок, созданных в других графических редакторах.

На 2-й форме с помощью кнопочного меню рисовать различные графические фигуры посредством компоненты класса TShape.

На 3-й форме реализовать рисование простейшими примитивами, типа линия, прямоугольник, эллипс и т.д.

Лабораторная работа № 12

РАЗРАБОТКА ПРИЛОЖЕНИЯ, УПРАВЛЯЕМОГО С ПОМОЩЬЮ ПАНЕЛИ ИНСТРУМЕНТОВ

***Цель лабораторной работы*:** научиться подключать инструментальную панель, изучить основные свойства и типы кнопок и их использование для управления вычислительным процессом

1. Краткие сведения

Для создания панели инструментов используется компонент TToolBar-инструментальная панель, пиктограмма которой имеет вид  .

Компонент TToolBar – это специальный контейнер для создания инструментальных панелей. В компонент TToolBar можно поместить любые другие компоненты. Как правило, он используется для расположения в ней кнопок, с помощью которых можно оперативно выполнить нужную команду. Кнопки можно группировать и располагать в несколько рядов.

Главная отличительная черта TToolBar - его способность гибкого управления дочерними элементами, которые он может группировать, выравнивать по размерам, располагать в несколько рядов. Компонент может манипулировать любыми вставленными в него дочерними элементами, но все его возможности в полной мере проявляются только при использовании специально для него разработанного компонента TToolButton (инструментальная кнопка). Этот компонент похож на кнопку TSpeedButton, но в палитре компонентов его нет.

Для того, чтобы вставить TToolButton в инструментальную панель TToolBar, необходимо правой кнопкой щелкнуть на компоненте TToolBar и в открывшемся окне выбрать NewButton или NewSeparator (новый сепаратор). Сепараторы предназначены для функционального выделения на инструментальной панели групп элементов и представляют собой разновидности кнопок TToolButton.

Хотя компонент TToolButton не имеет свойства, предназначенного для хранения картинки, однако он умеет использовать контейнер TImageList, чтобы извлечь из него нужную картинку и поместить ее на инструментальную кнопку.

2. Постановка задачи

Разработать приложение, вычислительный процесс которого управляется компонентами, расположенными в инструментальной панели.

3. Задание

Разработать программу, в которой предусмотреть управление вычислительным процессом с помощью кнопок, расположенных на инструментальной панели. Задание использовать из лабораторной работы № 8.

Лабораторная работа № 13
РАЗРАБОТКА ПРИЛОЖЕНИЯ, ПРЕДСТАВЛЕННОГО В ВИДЕ МНОГОСТРАНИЧНОГО ДОКУМЕНТА

Цель лабораторной работы: изучить компоненты TPageControl, TTabSheet.

1. Краткие сведения

Для создания многостраничных документов используются компоненты TTabControl и TPageControl.

Компонент TTabControl

Компонент TTabControl  (на странице Win32) представляет собой контейнер с закладками. Свойство Tabs определяет названия и количество закладок. Событие OnChange возникает при выборе новой закладки и позволяет управлять содержимым окна компонента.

Компонент TPageControl

Компонент TPageControl  (на закладке Win32) представляет собой контейнер с закладками, на каждой из которых содержаться панели класса TTabSheet. На каждой панели класса TTabSheet может содержаться свой набор помещенных на нее компонент.

Для того чтобы добавить новую панель и закладку, нужно щелкнуть правой кнопкой по компоненте PageControl и из локального меню выбрать команду NewPage.

Свойства:

ActivePage: TTabSheet - содержит активную панель. С помощью этого свойства можно установить активной нужную панель.

События:

OnChange - возникает при переключении панелей.

2. Постановка задачи

Разработать приложение в виде многостраничного документа.

3. Задание

Разработать программу ввода данных, выбора метода расчета, расчета и представления результатов в табличной и графической формах на примере лабораторной работы № 8, реализуя отдельные вычислительные шаги на различных закладках многостраничного документа.

Лабораторная работа № 14
РАЗРАБОТКА КОМПЛЕКСНОГО ПРИЛОЖЕНИЯ В DELPHI

***Цель лабораторной работы*:** показать умение создания современного приложения в визуальной среде.

1. Краткие сведения

При разработке приложения для данной лабораторной работы следует руководствоваться теоретическими сведениями всех предыдущих работ.

2. Постановка задачи

Разработать современное приложение, содержащее расширенный список компонентов управления программой и представления данных в различных видах.

3. Задание

Разработать приложение, поддерживающее различные способы управления вычислительным процессом с помощью TPageControl, TTabSheet, TChart**,** стандартных диалоговых компонентов, TPopupMenu**,** TMainMenu, TToolBar и др. на примере лабораторной работы № 8.

ЛИТЕРАТУРА

1. Архангельский, А.Я. Разработка прикладных программ для Windows в Delphi / А.Я. Архангельский. – М.: Изд. «Бином», 1999. – 256 с.
2. Бобровский, С. Delphi 5: учебный курс / С. Бобровский. – СПб.: Питер, 2000.- 640 с.
3. Марко, Кэнту. Delphi 5 для профессионалов / Кэнту Марко. – СПб.: Питер. 2001. – 944 с.
4. Подольский, С.В. Разработка интернет-приложений в Delphi 6 / С.В. Подольский, С.А. Скиба, О.А. Кожедуб. – СПб.: БХВ-Петербург, 2002. - 452 с.
5. Сван, Т. Delphi 4. Библия разработчика: Пер. с англ. / Том Сван.– СПб.: Диалектика, 1998. – 672 с.
6. Сухарев, М.В. Основы Delphi. Профессиональный подход / М.В. Сухарев. – СПб.: Наука и техника, 2004. – 600 с.
7. Тейксейра, С. Delphi 6. Руководство разработчика. Том 1. Основные методы и технологии. Пер. с англ. Уч. пос. / Стив Тейксейра, Ксавье Пачеко. – М.: Изд. дом «Вильямс», 2001. – 832 с.
8. Фаронов, В.В. Delphi 6. Учебный курс / В.В. Фаронов. – М.: Изд. Молгачева С.В., 2001. – 672 с.
9. Фаронов, В.В. Delphi 2005. Язык, среда, разработка приложений / В.В. Фаронов. – СПб.: Питер, 2005. - 560 с.

ПРИЛОЖЕНИЯ

Приложение 1

ОБРАЗЕЦ ТИТУЛЬНОГО ЛИСТА

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники (ФИТР)

Кафедра программного обеспечения вычислительной техники
и автоматизированных систем

О Т Ч Е Т

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 14

«Разработка комплексного приложения в Delphi»

по курсу:
"РАЗРАБОТКА ПРИЛОЖЕНИЙ В ВИЗУАЛЬНЫХ СРЕДАХ"

Выполнили: Студенты: Груша В.Н.
Геращенко С.И.
Гр. 107224

Проверил: доцент Гурский Н.Н.

Минск – 2010

Приложение 2

КОМАНДЫ ОСНОВНОГО МЕНЮ

В меню **File** расположены команды для выполнения операций с проектами, модулями и файлами (табл. П2.1).

Таблица П2.1

Команда	Описание
New Application	Создает новый проект, состоящий из формы, модуля и файла проекта
New Form	Создает новую форму и подключает ее к проекту
New Data Module	Создает новый модуль данных и подключает ее к проекту
Open	Открывает ранее созданный проект, модуль, форму или текстовой файл
Reopen	Вызывает список ранее загружавшихся проектов и форм для выбора и повторной загрузки
Save	Сохраняет текущую форму или модуль или файл
Save As	Сохраняет текущую форму с новым именем
Save Project As	Сохраняет текущий проект с новым именем
Save All	Сохраняет все открытые файлы, проект и используемые им модули
Close	Закрывает текущую форму
Close All	Закрывает все открытые файлы
|Use Unit	Добавляет имя указанного модуля в список используемых модулей (USES) текущего активного модуля
Add to Project	Добавляет файл к проекту
Remove FromProject	Удаляет файл из проекта
Print	Выводит содержимое активного файла на печать
Exit	Завершает работу Delphi

В меню **Edit** расположены команды, осуществляющие операции редактирования, работы с областью обмена данными, отмены действий и управления отображением компонентов (табл. П2.2).

Таблица П2.2

Команда	Описание
Undo	Отменяет ранее выполненные действия
Redo	Восстанавливает от
Cut	Вырезает выделенный объект и помещает его в буфер обмена данными
Copy	Копирует выделенный объект и/или фрагмент текста программы и помещает его в буфер обмена данными
Paste	Копирует содержимое буфера обмена данными в редактор или форму
Delete	Удаляет выбранный объект или фрагмент программы
Select All	Выделяет все компоненты формы или весь текст программы
Align to Grid	Выравнивает выбранный компонент по сетке
Bring to Front	Перемещает выбранный компонент поверх других компонентов
Send to Back	Перемещает выбранный компонент под другие компоненты
Align	Выравнивает компоненты
Size	Изменяет размер выделенных компонентов
Scale	Изменяет размер всех компонентов в форме
Tab Order	Изменяет порядок табуляции компонентов в активной форме
Creation Order	Задает порядок создания невизуальных компонентов
Lock Controls	Запрещает перемещение компонентов внутри формы
Add To Interface	Позволяет определить новую процедуру, функцию или свойство компонента ActiveX

Меню **Search** предоставляет команды для поиска и замены, а также команды для поиска указанных символов и строк, содержащих ошибки найденные компилятором (табл. П2.3).

Таблица П2.3

Команда	Описание
Find	Поиск указанного фрагмента текста
Find in files	Поиск указанного текста в нескольких файлах, задаваемых в диалоговой панели
Replace	Поиск указанного фрагмента текста и замена его новым
Search Again	Повторный поиск или повторная замена
Incremental Search	Поиск текста по мере его ввода
Go to Line Number	Перемещение курсора на строку с указанным номером
Show Last Compile Error	Перемещение курсора на строку, содержащую ошибку, найденную компилятором
Find Error	Поиск ошибки времени исполнения (run-time error)
Browse Symbol	Показывает характеристики указанного символа программы по его имени

В меню **View** содержатся команды для отображения различной информации и вызова менеджера проектов, инспектора объектов, браузера объектов и других информационных утилит (табл. П2.4).

Таблица П2.4

Команда	Описание
Project Manager	Менеджер проектов
Project Source	Отображает исходный текст файла проекта
Object Inspector	Инспектор объектов
Alignment Palette	Палитра выравнивания компонентов
Browser	Браузер объектов
Breakpoints	Список точек останова
Call Stack	Стек вызовов
Watches	Список точек слежения за переменными
Threads	Список потоков команд и их статус
Modules	Список модулей, загружаемых при выполнении данного проекта
Component List	Список компонентов
Window List	Список открытых окон
Toggle Form/Unit	Переключает активность из окна формы в окно текста программы и обратно
Unit	Показывает окно текста программы
Forms	Показывает окно формы
Type Library	Отображает содержимое библиотеки типов для компонентов ActiveX, серверов ActiveX и других COM-объектов
New Edit Window	Открывает новое окно с текстом текущей программы
SpeedBar	Отображает (прячет) панель быстрого доступа
Component Palette	Отображает (прячет) палитру компонентов

В меню **Project** содержатся команды для компиляции и сборки проектов, а также для установки опций текущего проекта (табл. П2.5).

Таблица П2.5

Команда	Описание
Add to Project	Добавляет файл к проекту
Remove from Project	Удаляет файл из проекта
Import Type Library	Импортирует в проект библиотеку типов элементов ActiveX
Add To Repository	Добавляет проект в репозитарий объектов
Compile	Компилирует модули, исходный текст которых изменился после последней компиляции
Build All	Компилирует все модули и создает исполняемую программу
Syntax Check	Проверяет синтаксическую правильность программы
Information	Отображает информацию о проекте
Web Deployment Options	Позволяет задать опции для внедрения компонента ActiveX или активной фирмы на Web-узел
Web Deploy	Внедряет компонент ActiveX или активную форму на Web-узел
Options	Задает опции компилятора и компоновщика, управляет рабочими каталогами

В меню **Run** расположены команды для отладки программ. Эти команды позволяют управлять различными функциями устроенного отладчика (табл. П2.6).

Таблица П2.6

Команда	Описание
Run	Компилирует и выполняет программу
Parameters	Задает параметры командной строки
Register ActiveX Server	Регистрирует сервер ActiveX в реестре Windows
Unregister ActiveX Server	Удаляет информацию о ранее зарегистрированном сервере ActiveX в реестре Windows
Step Over	Пошагово выполняет программу
Trace Into	Пошагово выполняет программу с заходом в подпрограммы
Trace To Next Source Line	Пошагово выполняет программу до следующей строки исходного текста
Run To Cursor	Выполняет программу до строки в окне редактора, на которой находится курсор
Show Execution Point	Отображает оператор, на котором было прервано выполнение программы
Program Pause	Приостанавливает выполнение программы
Program Reset	Завершает выполнение программы
Add Watch	Добавляет точку слежения за переменными
Add breakpoint	Добавляет точку останова
Evaluate/Modify	Позволяет узнать или изменить значение переменной

В меню **Component** содержатся команды для создания компонентов, установки новых компонентов, импорта компонентов ActiveX, создания нового компонента на базе существующего и установки пакетов (табл. П2.7).

Таблица П2.7

Команда	Описание
New Component	Вызывает окно эксперта компонентов
Install Component	Помещает компонент в существующий или новый проект
Import ActiveX Control	Импортирует компонент ActiveX
Create Component Template	Сохраняет компонент как шаблон для создания других компонентов
Install Package	Устанавливает пакеты, необходимые для прогона программы
Configure Palette	Вызывает диалоговую панель конфигурации палитры

Меню **Database** содержит средства для работы с базами данных (табл. П2.8).

Таблица П2.8

Команда	Описание
Explore	Вызывает инструмент исследования баз данных – Database Explorer или SQL Database (в зависимости от версии Delphi)
SQL Monitor	Вызывает инструмент запросов к БД - SQL Monitor
Form Wizard	Вызывает окно эксперта форм для создания формы, отображающей наборы данных из удаленных или локальных БД

Из меню **Tools** доступны средства настройки среды, дополнительные утилиты, входящие в состав Delphi, а также репозитарий объектов (табл. П2.9).

Таблица П2.9

Команда	Описание
Environment Options	Вызывает диалоговую панель настройки среды
Repository	Вызывает репозитарий
Configure Tools	Вызывает диалоговую панель редактирования опции Tools
Package Collection Editor	Вызывает окно редактора пакетов
Image Editor	Вызывает окно редактора графики
Database Desktop	Вызывает инструмент обслуживания БД – Database Desktop

Меню **Workgroups** содержит средства для работы с коллективными проектами (табл. П210.).

Таблица П2.10

Команда	Описание
Browse PVCS Projects	Показывает окно коллективной работы нескольких программистов над одним проектом программы
Mange Archive Directories	Показывает диалоговое окно управления архивом коллективного проекта программы
Add Project to Version Control	Сохраняет текущую версию коллективного проекта
Set Data Directories	Показывает диалоговое окно выбора каталогов для размещения версий коллективного проекта

В меню **Help** содержатся команды для вызова различных разделов справочной системы и отображения диалоговой панели “О программе” (табл. П2.11).

Таблица П2.11

Команда	Описание
Contents	Отображает содержание справочной системы
Keyword Search	Выполняет поиск справки по ключевому слову
What's New	Отображает справку по новым возможностям продукта
Getting Started	Выводит онлайновый вариант книги “Getting Started”
Using Object Pascal	Выводит онлайновый вариант книги “Using Object Pascal”
Developing Applications	Выводит онлайновый вариант книги “Developing
Object and Component Reference	Выводит онлайновый вариант книги “Object and Component Reference”
Borland Home Page	Соединяет с главной страницей Web-узла фирмы Borland
Delphi Home Page	Соединяет со страницей Web-узла фирмы Borland, посвященной Delphi
Borland Programs and Services	Соединяет со страницей Web-узла фирмы Borland, посвященной программам и сервисам
About	Отображает диалоговую панель “0 программе”

Приложение 3

СВОЙСТВА КОМПОНЕНТОВ

1. Общие свойства компонентов

Свойство Align

Задает способ выравнивания компонента внутри формы. Имеет одно из значений, приведенных в табл. П3.1.

Таблица П3.1

Значение	Описание
aINone	Выравнивание не используется. Компонент располагается на том месте, куда был помещен во время создания программы. Принимается по умолчанию
alTop	Компонент перемещается в верхнюю часть формы, и его ширина становится равной ширине формы. Высота компонента не изменяется
alBottom	Компонент перемещается в нижнюю часть формы, и его ширина становится равной ширине формы. Высота компонента не изменяется
alLeft	Компонент перемещается в левую часть формы, и его высота становится равной высоте формы. Ширина компонента не изменяется
alRight	Компонент перемещается в правую часть формы, и его высота становится равной высоте формы. Ширина компонента не изменяется
alClient	Компонент занимает всю рабочую область формы

Свойство Color

Задает цвет фона формы или цвет компонента или графического объекта. Может иметь одно из значений, приведенных в табл. П3.2.

Таблица П3.2

Значение	Цвет
clBlack	Черный (Black)
cIMaroon	Темно-красный (Maroon)
cIGreen	Зеленый (Green)
clOlive	Оливковый (Olive green)
cINavy	Темно-синий (Navy blue)
cIPurple	Фиолетовый (Purple)
cITeal	Сине-зеленый (Teal)
cIGray	Серый (Gray)
cISilver	Серебряный (Silver)
clRed	Красный (Red)
clLime	Ярко-зеленый (Lime green)
clBlue	Голубой (Blue)
clFuchsia	Сиреневый (Fuchsia)
clAqua	Ярко-голубой (Aqua)
clWhite	Белый (White)

Цвета, приведенные в табл. П3.3, являются системными цветами Windows и зависят от используемой цветовой схемы.

Таблица П3.3

Значение	Цвет
clBackGround	Текущий цвет фона окна
clActiveCaption	Текущий цвет заголовка активного окна
clInactiveCaption	Текущий цвет заголовка неактивного окна
cIMenu	Текущий цвет фона меню
clWindow	Текущий цвет фона Windows
clWindowFrame	Текущий цвет рамки окна
clMenuText	Текущий цвет текста элемента меню
clWindowText	Текущий цвет текста внутри окна
clCaptionText	Текущий цвет заголовка активного окна
clActiveBorder	Текущий цвет рамки активного окна
clInactiveBorder	Текущий цвет рамки неактивного окна
clAppWorkSpace	Текущий цвет рабочей области окна
clHighLight	Текущий цвет фона выделенного текста
clHightLightText	Текущий цвет выделенного текста
clBtnFace	Текущий цвет кнопки
clBtnShadow	Текущий цвет фона кнопки
clGrayText	Текущий цвет недоступного элемента меню
clBtnText	Текущий цвет текста кнопки

Помимо цветов, перечисленных в табл. П3.3, значение свойства Color может задаваться шестнадцатеричными значениями.

Свойство Ctl3D

Позволяет задать вид компонента. Если значение этого свойства равно False, компонент имеет двумерный вид, если True - трехмерный (значение по умолчанию).

Свойство Cursor

Позволяет определить вид курсора, который он будет иметь, находясь в активной области компонента. В Delphi предопределено большое количество стандартных курсоров. Кроме того, пользователь может создавать свои собственные курсоры или использовать созданные другими.

Свойство DrogCursor

Позволяет определить вид курсора, который будет отображаться, когда в компонент “перетаскивается” другой компонент. Значения этого свойства те же, что и у свойства Cursor.

Свойство DrogMode

Позволяет определить режим поддержки протокола drag-and-drop. Возможны значения, приведенные в табл. П3.4.

Таблица П3.4

Значение	Описание
dmAutomatic	Компонент можно “перетаскивать”, “зацепив” мышью
dmManual	Компонент не может быть “перетащен” без вызова метода BeginDrag

Свойство Enabled

Если это свойство имеет значение True, компонент реагирует на сообщения от мыши, клавиатуры и таймера. В противном случае (значение False) эти сообщения игнорируются.

Свойство Font

Многие визуальные компоненты используют шрифт по умолчанию. При создании компонента изначальное значение свойства Font (класс TFont) имеет параметры, приведенные в табл. П3.5.

Таблица П3.5

Свойство	Значение
Color	ClWindowText
Height	MulDiv(10, GetDeviceCaps(DC, LOGPIXELSY), 72)
Name	System
Pitch	FpDefault
Size	10
Style	[]

Свойство Height

Это свойство задает вертикальный размер компонента или формы.

Свойство HelpContext

Задает номер контекста справочной системы. Этот номер должен быть уникальным для каждого компонента. Если компонент активен (находится в фокусе), нажатие клавиши FI приводит к отображению экрана справочной системы (если такой существует для данного компонента).

Свойство Hint

Задает текст, который будет отображаться при обработке события OnHint, происходящего, если курсор находится в области компонента.

Свойство Left

Задает горизонтальную координату левого угла компонента относительно формы в пикселях. Для форм это значение указывается относительно экрана.

Свойство ParentColor

Это свойство позволяет указать, каким цветом будет отображаться компонент. Если значение этого свойства равно True, компонент использует цвет (значение свойства Color) родительского компонента. Если же значение свойства ParentColor равно False, компонент использует значение собственного свойства Color.

Свойство ParentCtl3D

Это свойство позволяет указать, каким образом компонент будет определять, является он трехмерным, или нет. Если значение этого свойства равно True, то вид компонента задается значением свойства Ctl3D его владельца, если же значение этого свойства равно False — то значением его собственного свойства Ctl3D.

Свойство ParentFont

Это свойство позволяет указать, каким образом компонент будет определять используемый им шрифт. Если значение этого свойства равно True, \ используется шрифт, заданный у владельца компонента, если же это значение равно False, то шрифт задается значением его собственного свойства Font.

Свойство PopupMenu

Это свойство задает название локального меню, которое будет отображаться при нажатии правой кнопки мыши. Локальное меню отображается только в случае, когда свойство AutoPopup имеет значение True или когда вызывается метод Popup.

Свойство TabOrder

Задает порядок получения компонентами фокуса при нажатии клавиши Tab. По умолчанию этот порядок определяется размещением компонентов в форме: первый компонент имеет значение этого свойства, равное 0, второй — 1 и т.д. Для изменения этого порядка необходимо изменить значение свойства TabOrder определенного компонента. TabOrder может использоваться только совместно со свойством TabStop.

Свойство TabStop

Это свойство позволяет указать, может компонент получать фокус или нет. Компонент получает фокус, если значение его свойства TabStop равно True.

Свойство Tag

С помощью этого свойства можно “привязать” к любому компоненту значение типа Longint.

Свойство Top

Это свойство задает вертикальную координату левого верхнего угла интерфейсного элемента относительно формы в пикселях. Для формы это значение указывается относительно экрана.

Свойство Visible

Это свойство позволяет определить, видим ли компонент на экране. Значением этого свойства управляют методы Show и Hide.

Свойство Width

Это свойство задает горизонтальный размер интерфейсного элемента или формы в пикселях.

2. Компоненты страницы STANDARD

2.1. *TMainMenu*

Компонент TMainMenu служит для создания главного меню формы установки компонента на форму необходимо создать его опции. Для этого следует путем двойного нажатия на левую клавишу "мыши" вызвать конструктор меню. Создание

опций меню - достаточно простой процесс. Надо выбрать, перейти в окно инспектора объектов, в строке Caption набрать необходимое и нажать клавишу Enter. Для создания новых опций необходимо выбирать справа, для создания подопций - снизу. Для определения символа быстрого доступа к опции перед ним ставится символ "&". Для вставки разделить черты очередной элемент называется "-". Для создания разветвленных меню, т.е. таких, у которых подопции вызывают новые списки подопций, нажмите Ctrl-*Вправо*, где *Вправо* - клавиша смещения курсора вправо.

Каждый элемент меню является объектом класса TMenuItem и обладает свойствами, приведенными в табл. П3.6.

Таблица П3.6

Свойство	Значение
Property Break: TMenuBreak;	Позволяет создать многоколончатый список подменю
Property Checked: Boolean;	Если True, рядом с опцией появляется галочка
Property Command: Word;	Используется при разработке приложений, обращающихся непосредственно к API-функциям Windows
Property Count: Integer;	Содержит количество опций в подчиненном меню, связанном с данным элементом (только для чтения)
Property Default: Boolean;	Определяет, является ли данная опция подменю умалчиваемой (умалчиваемая опция выделяется цветом и выбирается двойным щелчком мыши на родительской опции)
Property GroupIndex: Byte;	Определяет групповой индекс для зависимых опций
Property Items[Index: integer]: TMenuItem;	Позволяет обратиться к любой опции подчиненного меню по ее индексу
Property MenuIndex: Integer;	Определяет индекс опции в списке Items родительской опции
Property RadioItem: Boolean;	Определяет, зависит ли данная опция от выбора других опций в той же группе GroupIndex. Только одна опция группы может иметь True в свойстве Checked. Рядом с такой опцией вместо галочки изображается круг
Property Shortcut: TShortCut	Задает клавиши быстрого выбора данной опции

2.2. *TPopupMenu*

Данный компонент является локальным меню, которое становит доступным, когда пользователь нажимает правую кнопку мыши в рабочей области формы или компонента. Обычно локальное меню используется для динамического изменения свойств того или иного интерфейсного элемента.

Редактируется локальное меню так же, как и главное, с помощью конструктора меню. Чтобы связать нажатие правой кнопки мыши с раскрытием вспомогательного меню, в свойство PopupMenu необходимо поместить имя компонента меню.

Свойство Alignment задает местонахождение локального меню.

2.3. *TLabel*

Компоненты класса TLabel (метки) предназначены для размещения на форме различного рода текстовых надписей (табл. П3.7).

Таблица П3.7

Свойство	Значение
Property AutoSize: Boolean;	Указывает, будет ли метка изменять свои размеры в зависимости от помещенного в ее свойство Caption текста (True - будет)
Property FocusControl: TWinControl;	Содержит имя оконного компонента, который связан с меткой (выбор компонента Label приводит к перемещению фокуса на связанный с ним компонент)
TtextLayout = (tlTop, tlCenter, tlBottom); **Property** Layout: TTextLayout;	Определяет выравнивание текста по вертикали относительно границ метки: tlTop – текст располагается вверху; tlCenter – текст центрируется по вертикали; tlBottom – текст располагается внизу
Property ShowAccelChar: Boolean;	Если содержит True, символ & в тексте метки предшествует символу-акселератору
Property Transparent: Boolean;	Определяет прозрачность фона метки. Если False, фон закрашивается собственным цветом Color, в противном случае используется фон родительского компонента
Property WordWrap: Boolean;	Разрешает/запрещает разрыв строки на границе слова. Для вывода многострочных надписей задайте AutoSize=False, WordWrap=True и установите подходящие размеры метки

2.4. *TEdit*

Компонент класса TEdit представляет собой однострочный редактор текста (табл. П3.8). С его помощью можно вводить и/или отображать достаточно длинные текстовые строки. Следует помнить, что этот компонент не распознает символы конца строки (#13#10).

Таблица П3.8

Свойство	Значение
Property AutoSelect: Boolean;	Указывает, будет ли выделяться весь текст в момент получения компонентом фокуса ввода
Property AutoSize: Boolean;	Если True и BorderStyle = bsSingle, высота компонента автоматически меняется при изменении свойства Font.Size
TBorderStyle = bsNone..bsSingle; **Property** BorderStyle: TBorderStyle	Определяет стиль обрамления компонента: bsNone-нет обрамления; bsSingle-компонент обрамляется одной линией
TEditCharCase = (ecNormal, ecUpperCase, ecLowerCase); **Property** CharCase: TEditCharCase;	Определяет автоматическое преобразование высоты букв: ecNormal – нет преобразования; ecUpperCase – все буквы заглавные; ecLowerCase –все буквы строчные. Правильно работает с кириллицей
Property HideSelection: Boolean;	Если False, выделение текста сохраняется при потере фокуса ввода
Property MaxLength: Integer;	Определяет максимальную длину текстовой строки. Если имеет значение 0, длина строки не ограничена
Property Modified: Boolean;	Содержит True, если текст был изменен
Property OnChange: TNotifyEvent;	Определяет обработчик события OnChange, которое возникает после любого изменения текста
Property OEMConvert: Boolean;	Содержит True, если необходимо перекодировать текст из кодировки MS-DOS в кодировку Windows и обратно
Property PasswordChar: Char;	Если символ PasswordChar определен, он заменяет собой любой символ текста при отображении в окне. Используется для ввода паролей
Property ReadOnly: Boolean;	Если содержит True, текст не может изменяться
Property SelLength: Integer;	Содержит длину выделенной части текста
Property SelStart: Integer;	Содержит номер первого символа выделенной части текста
Property SelText: String:	Содержит выделенный текст

Методы компонента приведены в табл. П3.9.

Таблица П3.9

Метод	Действие
1	2
Procedure Clear:	Удаляет весь текст
procedure ClearSelection;	Удаляет выделенный текст
procedure CopyToClipboard;	Копирует выделенный текст в Clipboard
procedure CutToClipboard;	Копирует выделенный текст в Clipboard, после чего удаляет выделенный текст из компонента
function GetSelTextBuf(Buffer: PChar; ButSize: Integer): Integer;	Копирует не более ButSize символов выделенного текста в буфер Buffer

Окончание табл. ПЗ.9

1	2
procedure PasteFrornClipboard;	Заменяет выделенный текст содержимым Clipboard, а если нет выделенного текста, копирует содержимое Clipboard в позицию текстового курсора
procedure SelectAll;	Выделяет весь текст
Procedure SetSelTextBuf(Buffer: PChar);	Заменяет выделенный текст содержимым Buffer, а если нет выделенного текста, копирует содержимое Buffer в позицию текстового курсора

2.5. *TMemo*

Компоненты класса TMemo предназначены для ввода, редактирования и (или) отображения достаточно длинного текста, содержащего большое количество строк (табл. ПЗ.10).

Большинство свойств этого компонента аналогичны свойствам класса TEdit. Свойство Wordwrap аналогично свойству TLabel.WordWrap.

Таблица ПЗ.10

Свойство	Значение
Property Lines: TStrings;	Содержит редактируемый текст. Используется для построчного доступа. Методы Add, Delete, Insert используются для добавления, удаления и вставки строк
TScrollStyle = (ssNone, ssHorizontal, ssVertical, ssBoth); **Property** ScrollBars: TScrollStyle;	Определяет наличие в окне редактора полос прокрутки: ssNone – нет полос; ssHorizontal – есть горизонтальная полоса; ssVertical- есть вертикальная полоса; ssBoth - есть обе полосы
Property WantRetums: Boolean;	Если содержит True, нажатие Enter вызывает переход на новую строку, в противном случае – обрабатывается системой. Для перехода на новую строку в этом случае следует нажать Ctrl+Enter
Property WantTabs: Boolean;	Если содержит True, нажатие Tab вызывает ввод E текст символа табуляции, в противном случае – обрабатывается системой. Для ввода символа табуляции в этом случае следует нажать Ctrl-Tab

2.6. *TButton*

Компонент TButton представляет собой стандартную кнопку и широко используется для управления программами (табл. ПЗ.11). Кнопка может содержать текст, описывающий выполняемое ей действие.

Таблица П3.11

Свойство	Значение
Property Cancel: Boolean;	Если имеет значение True, событие OnClick кнопки возникает при нажатии клавиши Esc
Property Default: Boolean;	Если имеет значение True, событие OnClick кнопки возникает при нажатии клавиши Enter
Property Enabled: Boolean:	Если имеет значение False, то кнопка недоступна для нажатия
TModalResult = Low(lnteger) .. High (Integer); **Property** TModalResult;	Определяет результат, с которым было закрыто модальное окно

В терминологии Windows модальными окнами называются такие специальные окна, которые, раз появившись на экране, блокируют работу пользователя с другими окнами вплоть до своего закрытия. Если у кнопки определено свойство ModalResult, нажатие на нее приводит к закрытию модального окна и возвращает в программу значение ModalResult как результат диалога с пользователем. В Delphi определены стандартные значения ModalResult (табл. П3.12).

Таблица П3.12

Константа	Действие
mrNone	Модальное окно не закрывается
mrOk	Была нажата кнопка Ok
mrCancel	Была нажата кнопка Cancel
mrAbort	Была нажата кнопка Abort
mrRetry	Была нажата кнопка Retry
mrIgnore	Была нажата кнопка Ignore
mrYes	Была нажата кнопка Yes
mrNo	Была нажата кнопка No
MrAll	Была нажата кнопка All

2.7. *TCheckBox*

Кнопка с независимой фиксацией позволяет выбрать или отменить определенную функцию. Свойство State позволяет установить значение кнопки. Кнопка может находиться во включенном, выключенном и неактивном состоянии (табл. П3.13).

Таблица П3.13

Свойство	Значение
TLeftRight = (taLeftJustify, taRightJustify); **Property** Alignment: TLeftRight;	Определяет положение текста: taLeftJustify -с левой стороны компонента; taRightJustify - с правой стороны
Property AllowGrayed: Boolean;	Разрешает (запрещает) использование неактивного состояния cbGrayed
Property Checked: Boolean;	Содержит выбор пользователя типа Да/Нет. Состояния cbUnchecked и cbGrayed отражаются как False
TcheckBoxState = (cbUnchecked, cbChecked, cbGrayed); **Property** State: CheckBoxState;	Содержит состояние компонента: cbUnchecked - нет; cbChecked - да; cbGrayed –неактивен

2.8. *RadioButton*

Кнопки с зависимой фиксацией предназначены для выбора одной опции из нескольких взаимоисключающих, поэтому таких кнопок должно быть как минимум две. Для группировки кнопок с зависимой фиксацией внутри формы их необходимо разместить внутри компонента Panel, GroupBox или ScrollBox. Состояние кнопки содержится в свойстве Checked.

2.9. *TListBox*

Интерфейсный элемент этого типа содержит список элементов, которые могут быть выбраны при помощи клавиатуры или мыши. В компоненте предусмотрена возможность программной прорисовки элементов, поэтому список может содержать не только строки, но и произвольные изображения (табл. П3.14).

Таблица П3.14

Свойство	Значение
1	2
Property Canvas: TCanvas;	Канва для программной прорисовки элементов
Property Columns: Longint;	Определяет количество колонок элементов в
Property ExtendedSelect: Boolean;	Если ExtendedSelect=True и MultiSelect=True, выбор элемента без одновременного нажатия Crtl или Alt отменяет предыдущий выбор

Окончание табл. П3.14

1	2
Property IntegralHeight: Boolean;	Если IntegralHeight=True и Style<>lbOwnerDraw-Variable, в списке показывается целое число элементов
Property ItemIndex: Integer;	Содержит индекс сфокусированного элемента. Если MultiSelect=False, совпадает с индексом выделенного элемента
Property ItemHeight: Integer;	Определяет высоту элемента в пикселях для Style=lbOwnerDrawFixed
Property Items: TStrings;	Содержит набор строк, показываемых в компоненте
Property Multiselect: Boolean;	Разрешает/отменяет выбор нескольких элементов
Property SelCount: Integer:	Содержит количество выбранных элементов
Property Selected[X: Integer]: Boolean;	Содержит признак выбора для элемента с индексом X (первый элемент имеет индекс 0)
Property Sorted: Boolean;	Разрешает/отменяет сортировку строк в алфавитном порядке
TlistBoxStyle = (IbStandard, lbOwnerDrawFixed, lbOwnerDrawVariable); **Property** Style: TUstBoxStyle;	Определяет способ прорисовки элементов: IbStandard элементы рисует Windows, lbOwnerDrawFixed – рисует программа, все элементы имеют одинаковую высоту, определяемую свойством ItemHeight, lbOwnerDrawVariable -рисует программа, элементы
Property TopIndex: Integer;	Индекс первого видимого в окне элемента

2.10. *TComboBox*

Комбинированный список представляет собой комбинацию списка TListBox и редактора TEdit и поэтому большинство его свойств и методов заимствованы у этих компонентов.

Существуют пять модификаций компонента, определяемые его свойством Style: csSimple, csDropDown, csDropDownList, csOwnerDrawFixed и csOwnerDrawVariable. В первом случае список всегда раскрыт, в остальных раскрывается после нажатия кнопки справа от редактора. В модификации csDropDownList редактор работает в режиме отображения выбора и его нельзя использовать для ввода новой строки. Модификации csOwnerDrawFixed и csOwnerDrawVariable используются для программной прорисовки модификации csDropDown.

Свойство DropDownCount определяет количество элементов списка, появление которых еще не приводит к необходимости прокрутки списка. Свойство DroppedDown определяет, раскрыт ли список в данный момент.

2.11. *TScrollBar*

Компонент TScrollBar является полосой прокрутки и обычно используется для визуального управления значением какой-либо величины (табл. П3.15).

Таблица П3.15

Свойство	Значение
SrollBarKind = (sbHorizontal, sbVertical); **Property** Kind: TScrollBarKind;	Определяет ориентацию компонента: sbHorizontal - бегунок перемещается по горизонтали; sbVertical - бегунок перемещается по вертикали
Property LargeChange: TScrollBarInc:	“Большой” сдвиг бегунка (при щелчке мышью рядом с концевой кнопкой)
Property Max: Integer;	Максимальное значение диапазона изменения числовой величины
Property Min: Integer;	Минимальное значение диапазона изменения числовой величины
Property Position: Integer;	Текущее значение числовой величины
Property SmallChange: TScrollBarInc;	“Малый” сдвиг бегунка (при щелчке мышью по концевой кнопке)

2.12. *TGroupBox*

Этот компонент служит контейнером для размещения дочерних компонентов и представляет собой прямоугольное окно с рамкой и текстом в разрыве рамки. Обычно с его помощью выделяется группа управляющих элементов, объединенных по функциональному назначению. После того как компоненты помещены в группу, она становится их родительским классом.

2.13. *TRadioGroup*

Компонент класса TRadioGroup представляет собой специальный контейнер, предназначенный для размещения зависимых переключателей класса TRadioButton. Каждый размещаемый в нем переключатель помещается специальный список Items и доступен по индексу, что упрощает обслуживание группы (табл. П3.16).

Таблица П3.16

Свойство	Значение
Property Columns: Integer;	Определяет количество столбцов-переключателей
Property ItemIndex: Integer;	Содержит индекс выбранного переключателя
Property Items: TStrings;	Содержит список строк с заголовками элементов. Добавление (удаление) элементов достигается добавлением (удалением) строк списка Items

2.14. *TPanel*

Панель используется в качестве контейнера для расположения других интерфейсных элементов (табл. П3.17).

Таблица П3.17

Свойство	Значение
Property BevelInner: TPanelBevel;	Определяет стиль внутренней кромки
Property BevelOuter: TPanelBevel;	Определяет стиль внешней кромки
TBevelWidth = 1..Maxlnt; **Property** BevelWidth: TBevelWidth;	Задает ширину кромок в пикселях
TBorderStyle = bsNone..bsSingle; **Property** BorderStyle: TBorderStyle;	Определяет стиль рамки: bsNone - нет рамки; bsSingle – компонент по периметру обводится линией толщиной в 1 пиксель
TBorderWidth: 0..Maxint; **Property** BorderWidth: TBorderWidth:	Определяет расстояние в пикселях от внешней кромки до внутренней
Property FullRepaint: Boolean;	Разрешает (запрещает) перерисовку панели и всех ее дочерних элементов при изменении ее размеров

3. Компоненты страницы ADDITIONAL

3.1. *TBitBtn*

Пиктографическая кнопка TBitBtn представляет собой разновидность стандартной кнопки TButton, которая помимо текста может содержать графическое изображение. Растровое изображение определяется с помощь свойства Glyph. В комплект поставки Delphi (поддиректория Images/Buttons входит около 160 различных вариантов растровых изображений для кнопок. Кроме того, пользователь может

самостоятельно создать растровое изображения помощью встроенного в Delphi графического редактора. Свойство Kind позволяет выбрать одну из 11 стандартных разновидностей кнопки.

Нажатие любой из кнопок, кроме bkCustom и bkHelp, закрывает модальное окно и возвращает в программу результат mrXXX: bkOk - mrOk, bkCancel - mrCancel и т.д. Кнопка bkClose для модального окна возвращает mrCancel, а для главного окна программы – закрывает его и завершает работу программы. Кнопка bkHelp автоматически вызывает раздел справочной службы, связанный с HeIpContext формы, на которую она помещена (табл. 3.18).

Таблица П3.18

Свойство	Значение
Property Glyph: TBitmap:	Определяет связанные с кнопкой растровые изображения (до 4)
TBitBtnKind = (bkCustom, bkOK, bkCancel, bkHelp, bkYes,bkNo, bkClose, bkAbort, bkRetry, bkIgnore,bkAII); **Property** Kind: TBitBtnKind;	Определяет разновидность кнопки
TButtonLayout = (blGlyphLeft, bIGlyphRight, bIGlyphTop, bIGlyphBottom); **Property** Layout: TButtonLayout;	Определяет край кнопки, к которому прижимается пиктограмма
Property Margin: Integer:	Определяет расстояние в пикселях от края кнопки до пиктограммы
TNumGlyphs: 1..4; **Property** NumGlyphs: TNumGlyphs;	Определяет количество растровых изображений. Таких состояний может быть четыре: нормальное, запрещенное, нажатое, и утопленное
Property Spacing: Integer;	Определяет расстояние в пикселях от пиктограммы до надписи на кнопке
TButtonStyle = (bsAutoDetect, bsWin31, bsNew); **Property** Style: TButtonStyle.;	Определяет стиль оформления кнопки, зависящий от операционной системы

3.2. *TSpeedButton*

Еще один вариант кнопки, который отличается от TBitBtn тремя обстоятельствами: во-первых, не предусмотрен вывод надписи, во-вторых, имеется возможность фиксации в утопленном состоянии и, в-третьих, она не может закрыть модальное окно.

3.3. *TMaskEdit*

Специализированный редактор TMaskEdit предназначен для ввода текста, соответствующего некоторому шаблону, задаваемому свойством EditMask: String. Если это свойство не задано, TMaskEdit работает как обычный редактор TEdit.

Шаблон состоит из трех частей, отделенных друг от друга символами “;”. Первая часть задает маску ввода, вторая - это символ “0” *или* “1”, определяющий, записывается ли в Text результат наложения маски или исходный текст (“0” -исходный текст). В третьей части указывается символ, который в окне редактора будет стоять в полях, предназначенных для ввода символов. Описатели полей ввода представлены в табл. П3.19.

Таблица П3.19

Символ	Значение
L	Должно содержать букву
1	Может содержать букву
A	Должно содержать букву или цифру
a	Может содержать букву или цифру
C	Должно содержать любой символ
c	Может содержать любой символ
0	Должно содержать цифру
9	Может содержать цифру
#	Может содержать цифру, “+”, “–”

Специальные символы приведены в табл. П3.20.

Таблица П3.20

Символ	Значение
\	Следующий символ - литерал. Позволяет вставить в маску литералы из символов описателей полей ввода и специальных символов
	На это место вставляется символ-разделитель Windows для часов, минут, секунд
/	На это место вставляется символ-разделитель Windows для полей даты.
/	Разделитель частей шаблона
!	Подавляет все ведущие пробелы
>	Все следующие за ним поля ввода преобразуют буквы к заглавным
<	Все следующие за ним поля ввода преобразуют буквы к строчным
o	Отменяет преобразование букв

3.4. *TDrawGrid*

Компонент TDrawGrid используется для отображения информации в виде таблицы (табл. П3.21). Таблица делится на две части - фиксированную и рабочую. Фиксированная часть служит для показа заголовков столбцов/рядов и для ручного управления их размерами. Рабочая часть содержит произвольное количество столбцов и рядов, содержащих как текстовую, так и графическую информацию, и может изменяться программно.

Таблица П3.21

Свойство	Значение
1	2
Property BorderStyle: TBorderStyle;	Определяет наличие или отсутствие внешней рамки таблицы
Property Col: Longint;	Содержит номер столбца сфокусированной ячейки
Property ColCount: Longint;	Содержит количество столбцов таблицы
Property ColWidths[lndex: Longint]: Integer;	Содержит ширину столбца с индексом Index
Property DefaultColWidth: Integer;	Содержит умалчиваемое значение ширины столбца
Property DefaultDrawing: boolean;	Разрешает/запрещает автоматическую прорисовку служебных элементов таблицы-фиксированной зоны, фона и прямоугольника сфокусированной ячейки и т.п.
Property DefaultRowHeight: Integer;	Содержит умалчиваемую высоту рядов
Property EditorMode: Boolean;	Разрешает/запрещает редактирование ячеек. Игнорируется, если свойство Options включает goAlwayseShowEditor или не включает soEditing
Property FixedColor: TColor:	Определяет цвет фиксированной зоны
Property FixedCols: Integer:	Определяет количество столбцов фиксированной зоны
Property FixedRows: Integer;	Определяет количество рядов фиксированной зоны
Property GridHeight: Integer;	Содержит высоту таблицы
Property GridLineWidth: Integer;	Определяет толщину линий, расчерчивающих таблицу
Property GridWidth: Integer;	Содержит ширину таблицы
Property LeftCol: Longint;	Содержит номер самого левого столбца, видимого в зоне прокрутки
Property Options: TGridOptions;	Содержит параметры таблицы (см. ниже)

Окончание табл. П3.21

1	2
Property Row: Longint;	Содержит номер ряда сфокусированной ячейки
Property RowCount: Lonqint:	Содержит количество рядов таблицы
Property RowHeights[lndex: Longint]: Integer;	Содержит высоту ряда с индексом Index
TGridRect = record case Integer of 0: (Left, Top, Right/ Bottom: Longint); 1: (TopLeft, BottomRight: TGridCoord), end; **Property** Selection: TGridRect;	Определяет группу выделенных ячеек в координатах: левая верхняя и правая нижняя ячейки (нумерация столбцов и рядов идет от нуля, включая столбцы и ряды фиксированной зоны). После выделения сфокусированной окажется правая нижняя ячейка
Property TabStops[Index: Longint]: Boolean;	Разрешает/запрещает выбирать столбец с индексом Index при обходе ячеек клавишей Tab. Игнорируется, если Options не содержит goTabs
Property TopRow: Longint;	Содержит номер самого верхнего ряда, видимого в прокручиваемой зоне ячеек
Property VisibleColCount: Integer;	Содержит количество столбцов, полностью видимых в зоне прокрутки
Property VisibleRowCount: Integer;	Содержит количество рядов, полностью видимых в зоне прокрутки

Смысл элементов множества TGridOptions приведен в табл. П3.22.

Таблица П3.22

Элемент	Значение
1	2
goFixedVertLine	Столбцы фиксированной зоны разделяются вертикальными линиями
goFixedHorzLine	Ряды фиксированной зоны разделяются горизонтальными линиями
goVertLine	Столбцы рабочей зоны разделяются вертикальными линиями
goHorzLine	Ряды рабочей зоны разделяются горизонтальными линиями
goRangeSelect	Разрешено выделение нескольких ячеек. Игнорируется, если включен элемент goEdit
GoDrawFocus-Selected	Разрешено выделять сфокусированную ячейку так же, как выделенные
GoRowSizing	Разрешено ручное (мышью) изменение высоты строк
GoColSizing	Разрешено ручное изменение ширины рядов
GoRowMoving	Разрешено ручное перемещение рядов
goColMoving	Разрешено ручное перемещение столбца

Окончание табл. П3.22

1	2
goEditing	Разрешено редактирование ячейки. Игнорируется, если включен элемент goRowSelect. Редактирование начинается после щелчка мыши или нажатия клавиши F2 и завершается при щелчке по другой ячейке или нажатии Enter
goTabs	Разрешено выбирать ячейки клавишей Tab (Shifts-Tab)
goRowSelect	Обязывает выделять сразу все ячейки ряда
GoAlwaysShowEditor	Разрешено редактировать сфокусированную ячейку. Игнорируется, если не включен элемент goEditing
GoThumbTracking	Разрешено обновление при прокрутке. Если этот элемент отсутствует, обновление ячеек произойдет только после окончания прокрутки

3.5. *TStringGrid*

В отличие от компонента TStringGrid может отображать только текстовую информацию (табл. П3.23).

Таблица П3.23

Свойство	Значение
Property Cells[ACol, ARow: Integer]: string;	Определяет содержимое ячейки с табличными координатами (ACol, ARow)
Property Cols[Index: Integer]: TStrings;	Содержит все строки колонки с индексом Index
Property Objects [ACol, ARow: Integer]: TObject;	Обеспечивает доступ к объекту, связанному с ячейкой (ACol, ARow)
Property Rows[Index: Integer]: TStrings;	Содержит все строки ряда с индексом Index

3.6. *TImage*

Этот компонент служит для размещения на форме одного из трех поддерживаемых Delphi типов изображений: растровой картинки, пиктограмм или метафайла.

3.7. *TShape*

Компонент рисует одну из простейших геометрических фигур (прямоугольник, квадрат, скругленный прямоугольник, скругленный квадрат эллипс, окружность).

3.8. *TBevel*

Предназначен для выделения группы элементов или отделения их друг для друга и носит чисто оформительский характер.

3.9. *TScrollBox*

Компонент является контейнером для размещения других компонентов, имеет возможность прокрутки.

3.10. *TCheckListBox*

Группирует независимые переключатели, позволяя обратиться к любому из них по индексу (табл. П3.24).

Таблица П3.24

Свойство	Значение
Property AllowQrayed: Boolean;	Разрешает (запрещает) использовать в переключателях третье состояние cbGrayed
Property Checked[Index: Integer]: Boolean;	Содержит выбор пользователя типа Да/Нет для переключателя с индексом Index. Состояния cbUnchecked и cbGrayed отражаются как False
Property Sorted: Boolean:	Сортирует по алфавиту надписи на
Property State[Index: Integer]: TCheckBoxState;	Содержит состояние переключателя с индексом Index: cbUncheeked; cbChecked: cbGrayed

3.11. *TSplitter*

Предназначен для ручного (с помощью мыши) управления размерами контейнеров TPanel, TGroupBox или подобных им во время прогона программы (табл. П3.25).

Таблица П3.25

Свойство	Значение
Property Beveled: Boolean;	Управляет трехмерным изображением компонента. Если False, компонент виден как узкая полоска фона между разделяемыми им компонентами
NaturalNumber = 1..High(Integer); **Property** MinSize: NaturalNumber	Содержит минимальный размер любого из компонентов, которые разделяет TSplitter. Если выравнивание alLeft или alRight, минимальная ширина компонента - слева и справа от TSplitter, если alTop или alBottom, минимальная высота компонента - выше или ниже него

3.12. *TStaticText*

Подобен компоненту TLabel за исключением того, что, во-первых, он имеет Windows-окно и, во-вторых, в его свойстве BorderStyle: добавлено значение sbsSunken, которое создает иллюзию "вдавленности" компонента.

3.13. *TChart*

Облегчает создание специальных полей для графического представления данных.

4. Компоненты страницы DIALOGS

Использование диалоговых панелей

Работа со стандартными диалоговыми окнами осуществляется в три этапа:

1. На форму помещается соответствующий компонент и осуществляется настройка его свойств. Следует обратить внимание на то, что компонент-диалог не виден в момент работы программы, видно лишь создаваемое им стандартное окно.

2. Осуществляется вызов стандартного для диалогов метода Execute, который создает и показывает настроенное окно на экране. Вызов этого метода обычно располагается внутри обработчика какого-либо события. После обращения к Execute на экране появляется соответствующее диалоговое окно. Окно диалога является м5дальным окном, поэтому сразу после обращения к нему дальнейшее выполнение программы приостанавливается до тех пор, пока пользователь не закроет окно.

3. Использование введенных из диалогового окна данных (имя файла, застройки принтера и т.д.) для продолжения работы программы.

TOpenDialog и TSaveDialog

Эти компоненты имеют идентичные свойства и отличаются только внешним видом. Свойство FileName: (тип String) содержит маршрут поиска и имя выбранного файла при успешном завершении диалога программы. Для проверки наличия файла на диске -глобальная функция FileExists. Свойство Filter: String используется для фильтрации (отбора) файлов, показываемых в диалоговом окне. Это свойство можно устанавливать с помощью специального редактора или программно. Для доступа к редактору достаточно щелкнуть по кнопке в строке Filter окна инспектора объектов.

При программном вводе фильтры задаются одной длинной строкой, в которой символы “|” служат для разделения фильтров друг от друга, а также для разделения описания фильтруемых файлов от соответствующей маски выбора. С помощью свойства DefaultExt: String[3] формируется полное имя файла, если при ручном вводе пользователь не указал расширение. В этом случае к имени файла прибавляется разделительная точка.

Настройка диалога может варьироваться с помощью свойства
TOpenOption = (of ReadOnly, ofOverwritePrompt, ofHideReadOnly, ofNoChangeDir, ofShowHelp, ofNoValidate, ofAllowMultiSelect, ofExtensionDifferent, ofPathMustExist, ofFileMustExist, ofCreatePrompt, ofShareAware, ofNoReadOnlyRetuni.ofNoTestFileCreate, ofNoNetworkButton ofNoLongNames, ofOldStyleDialog, ofNoDereferenceLinks);

TOpenOptions = **set of** TOpenOption;

property Options: TOpenOptions;

Смысл значений этого свойства приведен в табл. П3.26.

Таблица П3.26

Значение	Действие
1	2
ofReadOnly	Показывает только шрифты с набором символов Windows
ofOverwritePromt	Требует согласия пользователя при записи в существующий файл
ofHideReadOnly	Прячет переключатель. Только для чтения
ofNochangeDir	Запрещает смену каталога
ofShowHelp	Включает в окно кнопку HELP
ofNoValidate	Запрещает автоматическую проверку правильности набираемых в имени файла символов
ofAllowMultiSelec	Разрешает множественный выбор файлов
ofExtensiondiffer	При завершении диалога наличие этого значения в свойстве Options говорит о том, что пользователь ввел расширение, отличающееся от умалчиваемого
ofPathMustExist	Разрешает указывать файлы только из существующих каталогов
ofFileMustExist	Разрешает указывать только существующие файлы
ofCreatePromt	Требует подтверждения для создания несуществующего файла
ofShareAware	Разрешает выбирать файлы, используемые другими параллельно выполняемыми программами
ofNoreadOnlyRetur	Запрещает выбор файлов, имеющих атрибут “Только для чтения”
ofNotestfileCreate	Запрещает проверку доступности сетевого или локального диска

Окончание табл. П3.26

1	2
ofNoNetworkButton	Запрещает вставку кнопки для создания сетевого диска
ofNoLongNames	Запрещает использование длинных имен файлов
ofOldStyleDialog	Создает диалог в стиле Windows 3.x

TOpenPictureDialog uTSavePictureDialog

Специализированные диалоги для открытия и сохранения графических файлов являются расширенными вариантами компонентов TOpenDialog и TSaveDialog, в которых предусмотрено наличие стандартного фильтра для выбора графических файлов и панель предварительного просмотра.

TFontDialog

Компонент используется для вызова стандартной диалоговой панели выбора шрифтов и их характеристик. Свойство Device определяет тип устройства, для которого выбирается fdScreen - экран; fdPrinter - принтер; fdBoth - шрифты, поддерживаемые и экраном, и принтером. Диапазон возможных значений размеров шрифтов определяется свойствами MinFontSize и MaxFontSize. Значения этих свойств задаются в пунктах (1 пункт равен приблизительно 0.36 мм). Если свойства содержат 0, ограничения на размер шрифта отсутствуют. Свойство Options используется для настройки диалога. Смысл значений этого свойства приведен в табл. П3.27.

Таблица П3.27

Значение	Действие
1	2
fdAnsiOnly	Показывает только шрифты с набором символов Windows
fdTrueTypeOnly	Показывает только TrueType-шрифты
fdEffects	Включает в окно переключатели "Подчеркнутый" и Зачеркнутый, а также список выбора цвета шрифта
fdFixedPitchOnly	Включает только моноширинные шрифты
fdForceFontExist	Предупреждает о выборе несуществующего шрифта
fdNoFaceSel	Запрещает выделение имени шрифта в момент открытия окна
fdNoOEMFonts	Запрещает выбор MS-DOS-шрифтов
fdNoSimLlIations	Исключает шрифты, которые синтезируются графическим интерфейсом Windows
fdNoSizeSel	Запрещает выделение размера шрифта в момент открытия окна
fdNoStyleSel	Запрещает выделение стиля шрифта в момент открытия окна
fdNoVectorFonts	Исключает векторные шрифты
fdShowHelp	Включает в диалоговое окно кнопку Help

Окончание табл. П3.27

1	2
fdWysiwyg	Включает шрифты, которые поддерживаются и экраном, и принтером
fdLimitSize	Включает ограничения на размер шрифта, заданные свойствами MaxFontSize и MinFontSize
fdScalableOnly	Включает только масштабируемые шрифты (векторные и TrueType)
fdApplyButton	Включает в окно кнопку "Применить"

TColorDialog

Компонент используется для вызова и обслуживания стандарт диалогового окна выбора цвета.

TPrintDialog

Компонент служит для создания стандартного диалогового окна для выбора параметров печати (табл. П3.28).

Таблица П3.28

Свойство	Значение
property Collate: Boolean;	Если имеет значение True, то окно показывается с выбранным переключателем "Разобрать" (Collate). Если этот переключатель выбран, печать нескольких копий документа будет идти по копиям: сначала первая копия, затем вторая и т.д., в противном случае - по страницам: сначала все копии первой страницы, затем второй и т.д.
property Copies: Integer;	Определяет количество копий (0 - одна копия)
property FromPage: Integer;	Определяет начальную страницу печати
property MaxPage: Integer;	Определяет верхнюю границу диапазона страниц для свойств FromPage, ToPage
property MinPage: Integer;	Определяет нижнюю границу диапазона страниц для свойств FromPage, ToPage
property Options: TPrintDialogOptions;	Определяет настройку окна: poPrintToFile –печатать файл; poPageNums - разрешает выбор диапазона страниц; poSelection - разрешает печать выбранного текста; poWarning - предупреждать пользователя о неустановленном принтере; poHelp - вставить в окно кнопку Help; poDisablePrintToFile - запрещает печать файл
property PrintRange: TPrintRange;	Определяет диапазон печатаемых страниц: prAllPages - все страницы; prSelection - выделенный фрагмент текста; prPageNums - страницы по номерам
property PrintToFile: Boolean;	Содержит True, если пользователь выбрал печать в
property ToPage: Integer;	Определяет конечную страницу печати

TPrinterSetupDialog

Компонент создает окно настройки параметров принтера, вид которого зависит от типа принтера. Этот диалог взаимодействует с драйвером принтера и не возвращает в программу никакой информации, поэтому его метод Execute процедура, а не функция.

TFindDialog

Стандартное диалоговое окно компонента TFindDialog используется для поиска фрагмента текста (табл. П3.29).

Таблица П3.29

Свойство	Значение
property FindText: string;	Указывает образец для поиска
property Left: Integer;	Содержит горизонтальную позицию левого верхнего угла места появления окна
property Options: TFindOptions:	Определяет настройку диалога
property Position: TPoint;	Содержит горизонтальную и вертикальную позицию левого верхнего угла места появления окна
property Top: Integer;	Содержит вертикальную позицию левого верхнего угла места появления окна

Для компонента определен следующий тип, использующийся в свойстве Options: TFindOptions. Его значения приведены в табл. П3.30.

Таблица П3.30

Значение	Действие
1	2
FrDown	Устанавливает поиск вперед по тексту
frDown frFindNext	Сообщает программе, что пользователь нажал кнопку "Найти далее"
FrHideMatchCase	Убирает выбор в переключателе "С учетом регистра"
frHideWholeWord	Убирает выбор в переключателе "Только слово целиком"
frHideUpDown	Прячет кнопки выбора направления поиска
frMatchCase	Устанавливает выбор в переключателе "С учетом регистра”
frDisableMatchCase	Запрещает выбор "С учетом регистра"
frDisableUpDown	Запрещает выбор направления поиска
frDisableWholeWord	Запрещает выбор. Только слово целиком

Окончание табл. П3.30

1	2
frReplace	Используется в компоненте TReplaceDialog и указывает на необходимость замены текущего выбора
frReplaceAll	Используется в компоненте TReplaceDialog и указывает на необходимость замены всех вхождений образца поиска
frWholeWord	Устанавливает выбор в переключателе "Только слово целиком"
frShowHelp	Включает в окно кнопку Help

TReplaceDialog

Компонент создает и обслуживает окно поиска и замены текстового фрагмента. Класс TReplaceDialog наследует большинство свойств класса TFindDialog. Дополнительно в компоненте определено свойство ReplaceText (тип String), в котором содержится текст замены, и событие OnReplace, которое возникает при нажатии кнопки "Заменить" или "Заменить все".

Приложение 4

ТИПЫ ДАННЫХ ЯЗЫКА OBJECT PASCAL

Простые типы данных языка Object Pascal

1 .Целые типы

Диапазон возможных значений целых типов зависит от их внутреннего представления, которое может занимать 1, 2 или 4 байта (табл. П4.1).

Таблица П4.1

Название	Длина, байт	Диапазон значений
Byte	1	0...255
Shortint	1	-128...+127
Smallint	2	-32 768. ..+32 767
Word	2	0...65 535
Integer	4	-2 147 483 648. ..+2 147 483 647
Longint	4	-2 147 483 648...+2 147 483 647
Cardinal	4	0... 2147483647

К целочисленным типам применимы процедуры и функции, прведенные в табл. П4.2.

Таблица П4.2

Обращение	Тип результата	Действие
abs (x)	x	Возвращает модуль x
chr (Byte)	Char	Возвращает символ по его коду
dec(x[,i])	—	Уменьшает значение x на i, а при отсутствии 1 - на 1
inc(x[,i])	—	Увеличивает значение v на i, а при отсутствии i - на 1
Hi (word)	Byte	Возвращает старший байт аргумента
Hi(integer)	Byte	Возвращает третий по счету байт
Lo(integer)	Byte	Возвращает младший байт аргумента
Lo (word)	Byte	Возвращает младший байт аргумента
Odd(LongInt)	Boolean	Возвращает True, если аргумент - нечетное число
Random(word)	—	Возвращает псевдослучайное число, равномерно распределенное в диапазоне 0...(word)
sqr (x)	x	Возвращает квадрат аргумента
swap (integer)	Integer	Меняет местами байты в слове
swap(word)	Word	Меняет местами байты в слове

2. Логические типы

К логическим относятся типы Boolean, ByteBool, Bool, WordBool и LongBool. В стандартном Паскале определен только тип Boolean, остальные логические типы введены в Object Pascal для совместимости с Windows: типы Boolean и ByteBool занимают по 1 байту каждый, Bool и WordBool - по 2 байта, LongBool - 4 байта. Значениями логического типа может быть одна из предварительно объявленных констант: False (ложь) или True (истина). Для них справедливы правила:

0rd(False) = 0;

Ord(True) <> 0;

Succ(False) = True;

Pred(True) = False.

3. Символьный тип

Значением символьного типа является множество всех символов. Каждому символу приписывается целое число в диапазоне 0...255. Это число служит кодом внутреннего представления символа, его возвращает функция ord.

Для кодировки в Windows используется код. Первая половина символов ПК с кодами 0...127 постоянна и содержит в себе служебные коды и латинский алфавит. Вторая половина символов с кодами 128...255 меняется для различных шрифтов. Символы с кодами 0...31 относятся к служебным кодам. Если эти коды используются в символьном тексте программы, они считаются пробелами.

К типу Char применимы операции отношения, а также встроенные функции

Chr (B) - функция типа Char, преобразует выражение B типа Byte в символ *i* возвращает его своим значением;

UpCase (СИ) - функция типа Char, возвращает прописную букву, если CH строчная латинская буква, в противном случае возвращает сам символ CH (для кириллицы возвращает исходный символ).

4. Перечисляемый тип

Перечисляемый тип задается перечислением тех значений, которые он может получать. Каждое значение именуется некоторым идентификатором и располагается в списке, обрамленном круглыми скобками.

Функции, поддерживающие работу с типами-диапазонами:

High (X) - возвращает максимальное значение типа-диапазона, к котором принадлежит переменная X;

Low (X) - возвращает минимальное значение типа-диапазона.

5. Вещественные типы

Значения вещественных типов определяют произвольное число лишь некоторой конечной точностью, зависящей от внутреннего формата вещественного числа (табл. П4.3).

Таблица П4.3

Название	Длина, байт	Кол-во значащих цифр	Диапазон значений	Примечание
Real	6	11...12	$2,9*10^{-39}$...$1,7*10^{39}$	При наличии сопроцессора использовать нежелательно, т.к. замедляет работу
Single	4	7. ..8	$1,5*10^{-45}$... $3,4*10^{38}$	-
Double	8	15...16	$5,0*10^{-324}$...$1,7*10^{308}$	-
Extended	10	19...20	$3,4*10^{-4951}$…$1,1*10^{4932}$	Применяется наиболее часто
Comp	8	19...20	-2^{63}…$+2^{63}-1$	Дробная часть отсутствует
Currency	8	19...20	±922337203685477,5807	Длина дробной части 4 десятичных разряда

Для работы с вещественными типами имеются стандартные функции (табл. П4.4).

Таблица П4.4

Обращение	Тип параметра	Тип результата	Примечание
abs(x)	Вещественный, целый	Тип аргумента	Модуль аргумента
ArcTan(x)	Вещественный	Вещественный	Арктангенс (в радианах)
Cos(x)	Вещественный	Вещественный	Косинус (в радианах)
Exp(x)	Вещественный	Вещественный	Экспонента
Frac(x)	Вещественный	Вещественный	Дробная часть числа
Int(x)	Вещественный	Вещественный	Целая часть числа
Ln(x)	Вещественный	Вещественный	Логарифм натуральный
Pi	—	Вещественный	π =3.141592653...
Random	—	Вещественный	Псевдослучайное число, равномерно распределенное в диапазоне 0...[1]
Random(x)	Целый	Целый	Псевдослучайное целое число, равномерно распределенное в диапазоне 0. ..x
Randomize	—	—	Инициация генератора псевдослучайных чисел
Sin (x)	Вещественный	Вещественный	Синус (в радианах)
Sqr(x)	Вещественный	Вещественный	Квадрат аргумента
Sqrt(x)	Вещественный	Вещественный	Корень квадратный

6. Тип дата-время

Тип дата - время определяется идентификатором TDateTime и предназначен для одновременного хранения и даты, и времени. Над данными типа TDateTime определены те же операции, что и над вещественными числами, а в выражениях этого типа могут участвовать константы и переменные целого и вещественного типов.

Приложение 5

ПРОЦЕДУРЫ И ФУНКЦИИ ДЛЯ РАБОТЫ СО СТРОКАМИ

Для работы со строками применяются следующие процедуры и функции (в квадратных скобках указываются необязательные параметры) (табл. П5.1).

Таблица П5.1

Метод	Действие
Процедуры и функции для работы со строками	
Function Concat(S1 [, S2, ..., SN]: String)-. String;	Возвращает строку, представляющую собой сцепление строк-параметров S1. S2,... , SN
Function Copy(St: String; Index, Count: Integer): String;	Копирует из строки St Count символов, начиная с символа с номером Index
Procedure Delete(St: String; Index, Count; Integers;	Удаляет Count символов из строки St, начиная с символа с номером Index
Procedure Insert(SubSt: String; St, Index: Integer);	Вставляет подстроку SubSt в строку St, начиная с символа с номером Index
Function Length(St: String): Integer;	Возвращает текущую длину строки St
Function Pos(SubSt, St: String): Integer;	Отыскивает в строке St первое вхождение подстроки SubSt и возвращает номер позиции, с которой она начинается. Если подстрока не найдена, возвращается нуль
Procedure SetLength(St: String; NewLength: Integer);	Устанавливает новую (меньшую) длину NewLength строки St, если NewLength больше текущей длины строки, обращение к SetLength итерируется
Подпрограммы преобразования строк в другие типы	
Function StrToCurr(St: String): Currency;	Преобразует символы строки St в целое число типа Currency. Строка не должна содержать ведущих или ведомых пробелов.
Function StrToDate(St: String): TDateTime;	Преобразует символы строки St в дату. Строка должна содержать два или три числа, разделенных правильным для Windows разделителем даты; (в русифицированной версии таким разделителем, является “.”) Первое число - день, второе - месяц, если указано третье число, оно задает год
Function StrToDateTime(St: String): TDateTime;	Преобразует символы строки St в дату и время. Строка должна содержать дату и разделенное пробелом время
Function StrToFloat(St: String): Extended;	Преобразует символы строки St в вещественное число. Строка не должна содержать ведущих или ведомых пробелов

Продолжение табл. П5.1

Function StrToInt(St: String): Integer;	Преобразует символы строки St в целое число. Строка не должна содержать ведущих или ведомых пробелов
I-unction StrToIntDef(St: String; Default: Integer): Integer;	Преобразует символы строки St в целое число. Если строка не содержит правильного представления целого числа, возвращается значение Default
I-unction StrToIntRange(St: String: Min, Max: Longint) : Longint;	Преобразует символы строки St в целое число и возбуждает исключение ERangeError, если число выходит из заданного диапазона Min Max
Function StrToTime(St: String): TDateTime;	Преобразует символы строки St во время
Procedure Val(St: String; var X; Code: Integer);	Преобразует строку символов St во внутреннее представление целой или вещественной переменной X, которое определяется типом этой переменной. Параметр Code содержит ноль, если преобразование прошло успешно, и тогда в X помещается результат преобразования; в противном случае он содержит номер позиции в строке St, где обнаружен ошибочный символ, и в этом случае содержимое X не меняется. В строке St могут быть ведущие и (или) ведомые пробелы
Подпрограммы обратного преобразования	
Function DateToStr(Value: TDateTime): String:	Преобразует дату из параметра Value в строку символов
Function DateTimeToStr(Value: TDateTime): String;	Преобразует дату и время из параметра Value в строку символов
Procedure DateTimeToString (var St: String: Format: String; Value: TDataTime);	Преобразует дату и время из параметра Value в строку St
Function FormatDateTime (Format: String; Value: TDateTime): String;	Преобразует дату и время из параметра Value в строку символов
Function FloatToStr(Value: Extended): String:	Преобразует вещественное значение Value в строку символов
Function FloatToStrF(Value: Extended; Format: TFloatFormat; Precision, Digits: Integer) : String;	Преобразует вещественное значение Value в строку символов с учетом параметров Precision и Digits (см. пояснения ниже)
Function FormatFloat(Format: String: Value: Extended): String;	Преобразует вещественное значение Value в строку
Function lntToStr(Value: Integer): String;	Преобразует целое значение Value в строку символов

Окончание табл. П5.1

Function TimeToStr(Value: TDateTime): String;	Преобразует время из параметра Value в строку символов
Procedure Str(X [.-width [: Decimals]]; var St: String);	Преобразует число X любого вещественного или целого типа в строку символов St; параметры Width и Decimals, если они присутствуют, задают формат преобразования: Width определяет общую ширину поля, выделенного под соответствующее символьное представление вещественного или целого числа X, а Decimals -количество символов в дробной части (этот параметр имеет смысл только в том случае, когда X – вещественное число)

Правила использования параметров функции FloatToStrF приведены в табл. П5.2.

Таблица П5.2

Значение Format	Описание
ffExponent	Научная форма представления с множителем eXX (“умножить на 10 в степени XX”). Precision задает общее количество десятичных цифр мантиссы. Digits - количество цифр в десятичном порядке XX. Число округляется с учетом первой отбрасываемой цифры: 3.1416E+00
ffFixed	Формат с фиксированным положением разделителя целой и дробной частей. Precision задает общее количество десятичных цифр в представлении числа. Digits - количество цифр в дробной части. Число округляется с учетом первой отбрасываемой цифры: 3,14
ffGeneral	Универсальный формат, использующий наиболее удобную для чтения форму представления вещественного числа. Соответствует формату ffFixed, если количество цифр в целой части меньше или равно Precision, а само число - больше или равно 0,00001, в противном случае соответствует формату ffExponent: 3,1416
ffNumber	Отличается от ffFixed использованием символа - разделителя тысяч при выводе больших чисел (для русифицированной версии Windows таким разделителем является пробел). Для Value = 7t* 1000 получим 3 141,60
ffCurrency	Денежный формат. Соответствует ffNumber, но в конце строки ставится символ денежной единицы (для русифицированной версии Windows - символы “р.”). Для Value = я*1000 получим: 3141,60р

Приложение 6

МАТЕМАТИЧЕСКИЕ ФОРМУЛЫ

Язык Object Pascal имеет ограниченное количество встроенных математических функций. Поэтому при необходимости использовать другие функции следует применять известные соотношения. В табл. П6.1 приведены выражения наиболее часто встречающихся функций через встроенные функции языка Object Pascal.

Таблица П6.1

Функция	Соотношение	Соотношение на языке Object Pascal
$Log_a(x)$	$\frac{Ln(x)}{Ln(a)}$	Ln(x)/Ln(a)
x^a	$e^{a*Ln(x)}$	Exp(a*Ln(x))
Tg(x)	$\frac{Sin(x)}{Cos(x)}$	Sin(x)/Cos(x)
Ctg(x)	$\frac{Cos(x)}{Sin(x)}$	Cos(x)/Sin(x)
ArcSin(x)	ArcTg(√(x/(1-sqr(x))))	ArcTan(Sqrt(x/(1-sqr(x))))
ArcCos(x)	π/2-ArcSin(x)	Pi/2-ArcTan(Sqrt(x/(1-sqr(x))))
ArcCtg(x)	π/2-ArcTg(x)	Pi/2-ArcTan(x)
Sh(x)	$\frac{e^x - e^{-x}}{2}$	(Exp(x)-Exp(-x))/2
Ch(x)	$\frac{e^x + e^{-x}}{2}$	(Exp(x)+Exp(-x))/2
Csc(x)	1/Sin(x)	1/Sin(x)
Sc(x)	1/Cos(x)	1/Cos(x)

Приложение 7

МОДУЛЬ MATH

Модуль Math, который существенно расширяет набор встроенных математических функций. Особенностью реализации содержащихся в нём функций и процедур является их оптимизация для работы с арифметическим сопроцессором класса Pentium, так что все они производят необходимые вычисления за рекордно малое время.

В табл. П7.1 приводятся все процедуры и функции, используемые в модуле Math. Подпрограммы становятся доступными программе только после ссылки на модуль Math в предложении Uses.

Таблица П7.1

Метод	Действие
Тригонометрические функции	
Function ArcCos(X: Extended): Extended;	При обращении к функции необходимо указать параметр типа Extended, в результате чего функция вернёт арккосинус от этого параметра типа Extended.
Function ArcSin(X: Extended): Extended;	При обращении к функции необходимо указать параметр типа Extended, в результате чего функция вернёт арксинус от этого параметра типа Extended.
Function ArcTan2(X: Extended): Extended;	При обращении к функции необходимо указать два параметра X и Y типа Extended, в результате чего функция вычисляет арктангенс Y/X и вернет угол в правильном квадранте типа Extended.
Function Cotan(X: Extended): Extended;	При обращении к функции необходимо указать угол типа Extended, в результате чего функция вернёт котангенс от этого параметра типа Extended.
Function ArcHypot(X: Extended): Extended;	При обращении к функции необходимо указать два параметра X и Y типа Extended, в результате чего функция вернёт корень квадратный из (X^2 + Y^2) – гипотенуза прямоугольного треугольника по двум катетам типа Extended.
Procedure SinCos(Theta: Extended; var Sin, Cos:Extended);	При обращении к функции необходимо указать угол Theta типа Extended, в результате чего функция возвращает одновременно Sin и Cos угла Theta.
Function Tan(X: Extended): Extended;	При обращении к функции необходимо указать угол типа Extended, в результате чего функция вернёт тангенс от этого угла типа Extended.
Функции преобразования углов	
Function CycleToRad(Cycles: Extended): Extended;	При передаче параметра Cycles функция вернёт значение Radians:=Cycles*2PI.
Function DegToRad(Degrees: Extended): Extended;	При передаче параметра Cycles функция вернёт значение Radians:=Degrees*PI/180.

Продолжение табл. П7.1

Function GradToRad(Grads: Extended): Extended;	При передаче параметра Cycles функция вернёт значение Radians:=Grads*PI/200.
Function RadToDeg(Radians: Extended): Extended;	При передаче параметра Cycles функция вернёт значение Degrees:=Radians*180/PI.
Function RadToGrad(Radians: Extended): Extended;	При передаче параметра Cycles функция вернёт значение Grads:=Radians*200/PI.
Function RadToCycle(Radians: Extended): Extended;	При передаче параметра Cycles функция вернёт значение Cycles:=Radians/2PI.
Гиперболические функции	
Function ArcCosh(X: Extended): Extended;	При обращении к функции необходимо указать параметр типа Extended, в результате чего функция вернёт гиперболический арккосинус от этого параметра типа Extended.
Function ArcSinh(X: Extended): Extended;	При обращении к функции необходимо указать параметр типа Extended, в результате чего функция вернёт гиперболический арксинус от этого параметра типа Extended.
Function ArcCosh(X: Extended): Extended;	При обращении к функции необходимо указать параметр типа Extended, в результате чего функция вернёт гиперболический арктангенс от этого параметра типа Extended.
Function Cosh(X: Extended): Extended;	При обращении к функции необходимо указать угол типа Extended, в результате чего функция вернёт гиперболический косинус от этого угла типа Extended.
Function Sinh(X: Extended): Extended;	При обращении к функции необходимо указать угол типа Extended, в результате чего функция вернёт гиперболический синус от этого угла типа Extended.
Function Tanh(X: Extended): Extended;	При обращении к функции необходимо указать угол типа Extended, в результате чего функция вернёт гиперболический тангенс от этого угла типа Extended.
Логарифмические функции	
Function LnXP1(X: Extended): Extended;	При обращении к функции необходимо указать параметр X типа Extended, в результате чего функция вернёт натуральный логарифм от (X+1) типа Extended. Данная функция используется, когда значение X близко к нулю.
Function Log10(X: Extended): Extended;	При обращении к функции необходимо указать параметр X типа Extended, в результате чего функция вернёт десятичный логарифм от X типа Extended.
Function Log2(X: Extended): Extended;	При обращении к функции необходимо указать параметр X типа Extended, в результате чего функция вернёт двоичный логарифм от X типа Extended.

Продолжение табл. П7.1

Function LogN(Base, X: Extended): Extended;	При обращении к функции необходимо указать параметры X и Base типа Extended, в результате чего функция вернёт логарифм по основанию Base от X типа Extended.
Экспоненциальные функции	
Function IntPower(Base: Extended; Exponent: Integer): Extended;	При обращении к функции необходимо ввести в качестве параметров основание Base типа Extended и степень Exponent целочисленного типа Integer, в результате чего функция вернёт значение Base в степени Exponent типа Extended.
Function Power(Base, Exponent: Extended): Extended;	При обращении к функции необходимо ввести в качестве параметров основание Base типа Extended и степень Exponent вещественного типа Integer, в результате чего функция вернёт значение Base в степени Exponent типа Extended.
Подпрограммы разного назначения	
Function Ceil(X: Extended): Integer;	При обращении к функции необходимо ввести параметр X вещественного типа, в результате чего функция вернёт ближайшее меньшее целое число, отбросив часть после запятой.
Function Floor(X: Extended): Integer;	При обращении к функции необходимо ввести параметр X вещественного типа, в результате чего функция вернёт ближайшее большее целое число, округляя X в большую сторону.
Procedure Frexp(X: Extended; var Mantissa: Extended; var Exponent: Integer);	При обращении к функции необходимо указать число вещественного типа Extended, в результате чего функция вернёт мантиссу и степень этого числа.
Function Ldexp(X: Extended; P: Integer): Extended;	При обращении к функции необходимо указать два числа: X вещественного Extended и P целого Integer типа. В результате функция вернёт результат формулы $X*P^2$ типа Extended.
Function Poly(X: Extended; const Coefficients: array of Double): Extended;	Функция принимает параметр X вещественного типа Extended и массив коэффициентов вещественного типа Double, в результате чего функция вернёт значение полинома $A*X^N+B*X^{N-1}+\ldots+Z$. Коэффициенты должны задаваться в порядке возрастания степени.
Статистические подпрограммы	
Function Max(A, B: Integer): Integer; overload; Function Max(A, B: Int64): Int64; overload; Function Max(A, B: Single): Single; overload; Function Max(A, B: Double):Double; overload; Function Max(A, B: Extended): Extended; overload;	При обращении к функции необходимо указать две переменные какого либо типа и функция вернёт наибольшее из них. Параметр overload указывает на то, что функция может быть переопределена.

Продолжение табл. П7.1

Function MaxIntValue(const Data: array of Integer): Integer;	Функция принимает массив целых чисел и возвращает наибольшее из них.
Function MaxValue(const Data: array of Double): Double;	Функция принимает массив вещественных Double чисел и возвращает наибольшее из них.
Function Mean(const Data: array of Double): Extended;	Функция принимает массив чисел типа Double и возвращает арифметическое среднее массива чисел.
Function MeanAndStdDev(const Data: array of Double; var Mean, StdDev: Extended);	Приняв массив вещественных чисел Double, функция возвращает среднее арифметическое всех чисел и стандартное отклонение для этого набора чисел.
Function Min(A, B: Integer): Integer; overload;	При обращении к функции необходимо указать две переменные какого либо типа и функция вернёт наименьшее из них. Параметр overload указывает на то, что функция может быть переопределена.
Function Min(A, B: Integer): Integer; overload;	
Function Min(A, B: Integer): Integer; overload;	
Function Min(A, B: Integer): Integer; overload;	
Function Min(A, B: Integer): Integer; overload;	
Function MinIntValue(const Data: array of Integer): Integer;	Функция принимает массив целых чисел и возвращает наименьшее из них.
Function MinValue(const Data: array of Double): Double;	Функция принимает массив вещественных Double чисел и возвращает наименьшее из них.
Procedure MomentSkewKurtosis(const Data: array of Double; var M1, M2, M3, M4, Skew, Kurtosis: Extended);	Функция принимает массив чисел вещественного типа Double и возвращает статистические моменты порядков с первого по четвёртый, а также асимметрию Skew и эксцесс Kurtosis для набора чисел.
Function Norm(const Data: array of Double): Extended;	Принимая массив вещественных чисел типа Double, функция возвращает квадратный корень из суммы квадратов этих чисел.
Function PopnStdDev(const Data: array of Double): Extended;	Принимая массив вещественных чисел типа Double, функция возвращает выборочное стандартное отклонение типа Extended.
Function PopnVariance(const Data: array of Double): Extended;	Принимая массив вещественных чисел типа Double, функция возвращает выборочную дисперсию типа Extended.

Продолжение табл. П7.1

Function RandG(Mean, StdDev: Extended): Extended;	Функция принимает две переменные вещественного типа и генерирует нормальную псевдораспределённую последовательность чисел с заданным средним значением Mean и стандартным отклонением StdDev.
Function StdDev(const Data: array of Double): Extended;	Принимая массив вещественных чисел типа Double, функция возвращает среднеквадратическое отклонение.
Function Sum(const Data: array of Double): Extended register;	Принимая массив вещественных чисел типа Double, функция возвращает сумму чисел из массива.
Procedure SumAndSquares(const Data: array of Double;	Функция принимает массив чисел вещественного типа и возвращает одновременно сумму и сумму квадратов этих чисел.
Function SumInt(const Data: array of Integer): Extended register;	Функция принимает массив чисел целочисленного типа и вычисляет их сумму.
Function SumOfSquares(const Data: array of Double): Extended;	Функция принимает массив чисел вещественного типа и вычисляет сумму их квадратов.
Function TotalVariance(const Data: array of Double): Extended;	Функция принимает массив чисел вещественного типа и вычисляет сумму квадратов всех величин от их среднего арифметического.
Function Variance(const Data: array of Double): Extended;	Функция принимает массив чисел вещественного типа и возвращает выборочную дисперсию для этих чисел, используя «несмещённую» формулу TotalVariance/(N-1).
Финансовые функции	
Type TpaymentTime = (ptEndOfPeriod, ptStartOfPeriod);	Специальный перечисляемый тип, используемый в финансовых функциях.
Function DoubleDecliningBalance (Cost, Salvage: Extended; Life, Period: Integer): Extended;	Функция принимает два вещественных и два целочисленных параметра и в результате возвращает значение амортизации методом двойного баланса.
Function FutureValue(Rate: Extended; NPeriods: Integer; Payment, PresentValue: Extended; PaymentTime: TPaymentTime): Extended;	Для этой функции необходимо передавать три вещественных, один целочисленный параметр и параметр собственного перечислительного типа, после чего функция вернет будущее значение вложения.

Продолжение табл. П7.1

Function InterestPayment(Rate: Extended; Period, NPeriods: Integer; PresentValue, FutureValue: Extended; PaymentTime: TPaymentTime): Extended;	Для этой функции необходимо передавать три вещественных, два целочисленных параметра и параметр собственного перечислительного типа, после чего функция вернет количество процентов по ссуде.
Function InterestRate(NPeriods: Integer; Payment, PresentValue, FutureValue: Extended; PaymentTime: TPaymentTime): Extended;	Для этой функции необходимо передавать три вещественных, один целочисленный параметр и параметр собственного перечислительного типа, после чего функция вернет норму прибыли, необходимую для получения заданной суммы.
Function InternalRateOfReturn(Guess: Extended; const CashFlows: array of Double): Extended;	Функция принимает массив чисел вещественного типа и вещественный параметр, после чего возвращает внутреннюю скорость оборота вложения для ряда последовательных выплат.
Function NetPresentValue(Rate: Extended; const CashFlows: array of Double; PaymentTime: TPaymentTime): Extended;	Функция принимает массив чисел вещественного типа, вещественный параметр и параметр собственного перечислительного типа, после чего возвращает чистую текущую стоимость вложения для ряда последовательных выплат.
Function NumberOfPeriods(Rate, Payment, PresentValue, FutureValue: Extended; PaymentTime: TPaymentTime): Extended;	Функция принимает четыре вещественных параметра и параметр собственного перечислительного типа, после чего функция вернет количество периодов, за которые вложение достигнет заданной величины.
Function Payment(Rate: Extended; NPeriods: Integer; PresentValue, FutureValue: Extended; PaymentTime: TPaymentTime): Extended;	Для этой функции необходимо передавать три вещественных, один целочисленный параметр и параметр собственного перечислительного типа и функция вернет размер периодической зарплаты для погашения ссуды при заданном числе периодов, процентной ставке, а также текущем и будущем значениях ссуды.
Function PeriodPayment(Rate: Extended; Period, NPeriods: Integer; PresentValue, FutureValue: Extended; PaymentTime: TPaymentTime): Extended;	Для этой функции необходимо передавать три вещественных, два целочисленных параметра и параметр собственного перечислительного типа, после чего функция вернет платежи по процентам за заданный период.

Окончание табл. П7.1

Function PresentValue(Rate: Extended; NPeriods: Integer; Payment, FutureValue: Extended; PaymentTime: TPaymentTime): Extended;	Для этой функции необходимо передавать три вещественных, один целочисленный параметр и параметр собственного перечислительного типа, после чего функция вернет текущее значение вложения.
Function SLNDepreciation(Cost, Salvage: Extended; Life: Integer): Extended;	Для вычисления функции необходимо сообщить в качестве параметров две переменные вещественного и одну целочисленного типа, в результате чего функцией будет возвращено значение амортизации методом постоянной нормы.
Function SYDDepreciation(Cost, Salvage: Extended; Life, Period: Integer): Extended;	Для вычисления функции необходимо сообщить в качестве параметров две переменные вещественного и две целочисленного типа, в результате чего функцией будет возвращено значение амортизации методом весовых коэффициентов.