Министерство образования Республики Беларусь
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра “Вычислительные методы и программирование”

А.Б. Закалюкин, С.В. Колосов,
А.А. Навроцкий, А.К. Синицын, А.И. Шакирин

ПРОГРАММИРОВАНИЕ В СРЕДЕ DELPHI

Лабораторный практикум для студентов всех специальностей

Минск 1998

УДК 621.3.6.

Программирование в среде DELPHI. Лабораторный практикум для студентов всех специальностей / А.Б. Закалюкин, С.В. Колосов, А.А. Навроцкий, А.К. Синицын, А.И. Шакирин; Под общ. ред. А.К. Синицына. – Мн.: БГУИР, 1998. – 94 с. ISBN 985-444–026–5.

В лабораторном практикуме приведены краткие теоретические сведения по основам программирования в среде DELPHI, а также по языку программирования Object Pascal. После каждой темы приведено 30 индивидуальных заданий.

В практикум вошло 8 лабораторных работ.

Ил. 10, табл.2, список лит.- 3 назв.

Рецензент: В.К. Полевиков,
доц. каф. “Вычислительная математика”, БГУ

ISBN 985–444–026-5

СОДЕРЖАНИЕ

ТЕМА 1. ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ АЛГОРИТМОВ

Цель лабораторной работы: научиться составлять каркас простейшей программы в среде DELPHI. Написать и отладить программу линейного алгоритма.

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 вы можете увидеть на экране картинку, подобную представленной на рис. 1.1.

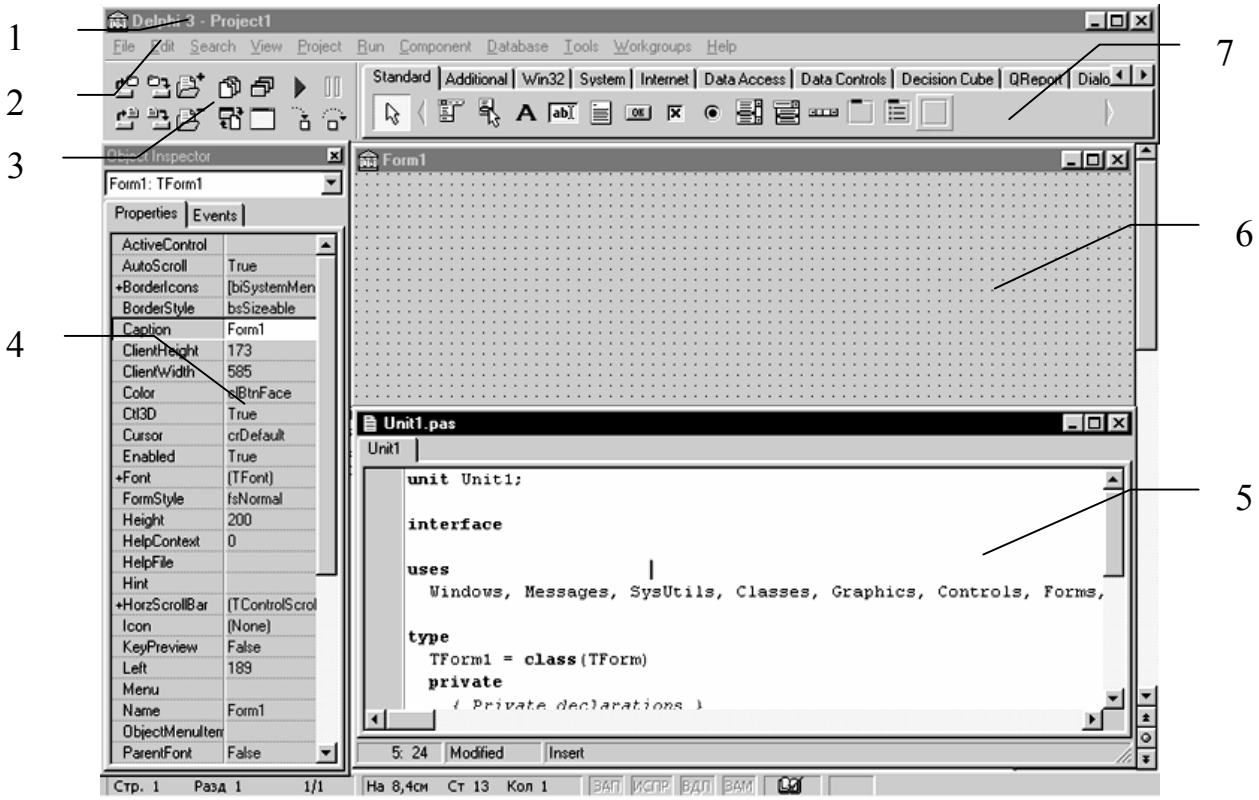

Рис.1.1.

1- главное окно; 2 – основное меню; 3 – пиктограммы основного меню;
4 - окно инспектора объектов; 5 – окно текста программы;
6- окно пустой формы; 7 – меню компонентов

Главное окно всегда присутствует на экране и предназначено для управления процессом создания программы. Основное меню (прил.1)содержит все необходимые средства для управления проектом. Пиктограммы облегчают доступ к наиболее часто применяемым командам основного меню. Через меню компонентов (прил. 2) осуществляется доступ к набору стандартных сервисных программ среды DELPHI, 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 объектов (вызывается с помощью клавиши F11) предназначено для изменения свойств выбранных компонентов и состоит из двух страниц. Страница *Properties* (Свойства) предназначена для изменения необходимых свойств компонента, страница *Events* (События) – для определения реакции компонента на то или иное событие (например, нажатие определенной клавиши или щелчок “мышью ” по кнопке).

Окно формы представляет собой проект Windows-окна программы. В это окно в процессе написания программы помещаются необходимые компоненты. Причем при выполнении программы помещенные компоненты 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-окна. При помещении некоторого компонента в окно формы текст программы автоматически дополняется описанием необходимых для его работы библиотек стандартных программ (раздел uses) и типов переменных (раздел type).

Программа в среде DELPHI составляется как описание алгоритмов, которые необходимо выполнить, если возникает определенное событие, связанное с формой (например щелчок “мыши” на кнопке – событие OnClick, создание формы – OnCreate). Для каждого обрабатываемого в форме события, с помощью страницы Events инспектора объектов в тексте программы организуется процедура (procedure), между ключевыми словами begin и end которой программист записывает на языке Object Pascal требуемый алгоритм.

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

1.2.Структура программ DELPHI

Программа в DELPHI состоит из файла проекта (файл с расширением *.dpr*), одного или нескольких файлов исходного текста (с расширением *.pas*), файлов с описанием окон формы (с расширением .dfm).

В ***файле проекта*** находится информация о модулях, составляющих данный проект. Файл проекта автоматически создается и редактируется средой DELPHI и не предназначен для редактирования.

Файл исходного текста – программный модуль (Unit) предназначен для размещения текстов программ. В этом файле программист размещает текст программы, написанный на языке PASCAL.

В разделе объявлений описываются типы, переменные, заголовки процедур и функции, которые могут быть использованы другими модулями, через операторы подключения библиотек (Uses). В разделе реализации располагаются тела процедур и функций, описанных в разделе объявлений, а также типы

переменных, процедуры и функции, которые будут функционировать только в пределах данного модуля. Раздел инициализации используется редко и его можно пропустить. Модуль имеет следующую структуру:

```
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: составить программу вычисления для заданных значений x, y, z арифметического выражения

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

Панель диалога программы организовать в виде, представленном на рис.1.2.

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  . С помощью мыши, «захватывая» одну из кромок формы или выделенную строку заголовка отрегулируйте нужные размеры формы и ее положение на экране.

1.3.2. Изменение заголовка формы

Новая форма имеет одинаковые имя (Name) и заголовок (Caption) - FORM1. Имя формы менять не рекомендуется, т.к. оно входит в текст программы.

Для изменения заголовка вызовите окно инспектора объектов (F11) и щелкните кнопкой мыши на форме. В форме инспектора объектов найдите и щелкните мышью на Properties – Caption . В выделенном окне наберите “Лаб. раб. N1. Ст. гр. 740102 Иванов А.А.”.

1.3.3. Размещение строки ввода (TEdit)

Если необходимо ввести из формы в программу или вывести на форму информацию, которая вмещается в одну строку, используют окно однострочного редактора текста, представляемого компонентом TEdit.

В данной программе с помощью однострочного редактора будут вводиться переменные x,y,z типа extended или integer.

Выберите в меню компонентов Standard пиктограмму  , щелкните мышью в том месте формы, где вы хотите ее поставить. Вставьте три компонента TEdit в форму. Захватывая их “мышью” отрегулируйте размеры окон 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(Edit1.Text).

При этом в записи числа не должно быть пробелов, а действительное число пишется с десятичной запятой.

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

1.3.4. Размещение надписей (TLabel)

На форме рис. 1.2 имеются четыре пояснительные надписи. Для нанесения таких надписей на форму используется компонент TLabel. Выберите в меню компонентов Standard пиктограмму  , щелкните на ней мышью. После этого в нужном месте формы щелкните мышью, появится надпись Label1. 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 (TMemo)

Для вывода результатов работы программы обычно используется текстовое окно, которое представлено компонентом (TMemo). Выберите в меню компонентов пиктограмму  и поместите компонент TMemo 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 окно типа TMemo, находится в массиве строк Memo1.Lines. Каждая строка имеет тип String.

Для чистки 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 метод 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 программу – обработчик этого события, которая заносит начальные значения переменных x, y, z в соответствующие окна TEdit, а в окне TMemo помещает 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 события создания формы: **Procedure TForm1.FormCreate(Sender:TObject)**. Между begin…end вставим текст программы (смотрите пример, расположенный ниже).

1.3.7. Написание программы обработки события нажатия кнопки (ButtonClick)

Поместите на форму кнопку, которая описывается компонентом TButton, для чего выберем в меню компонентов Standart пиктограмму 

. 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 программы, дополненной заголовком процедуры обработчика события - нажатия кнопки (**Procedure TForm1.ButtonClick(Sender:TObject**);).

Наберите текст этой процедуры, приведенный в примере.

1.3.8. Запуск и работа с программой

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 (рис.1.2).

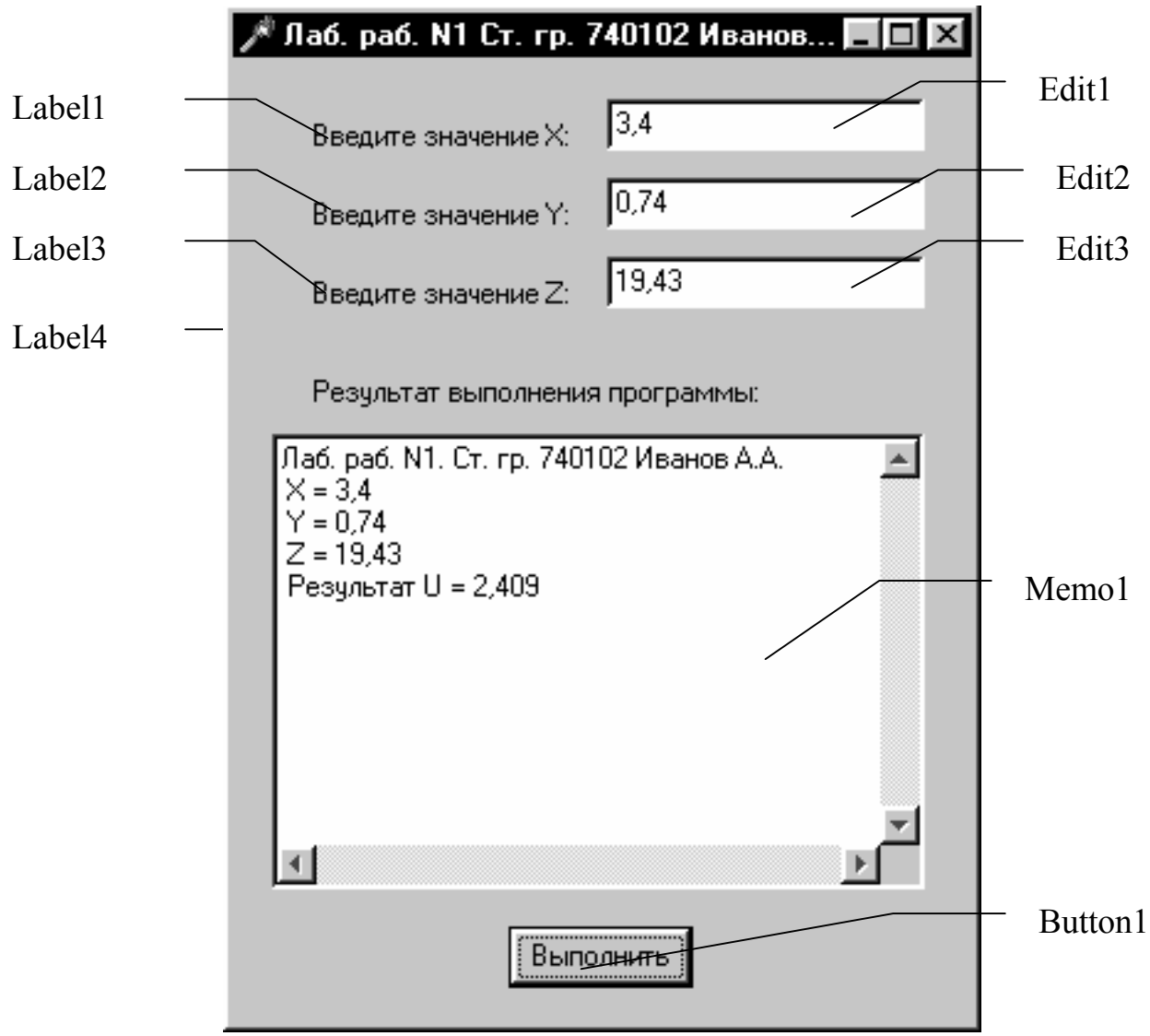

Рис.1. 2

Работа с программой происходит следующим образом. Нажмите (щелкните мышью) кнопку “Выполнить”. В окне Memo1 появляется результат. Измените исходные значения x, y, z в окнах Edit и снова нажмите кнопку ”Выполнить” - появится новые результаты. Завершить работу программы можно нажав или ProgramReset в главном меню Run, или кнопку  на форме.

Текст программы имеет вид:

```
unit tema1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

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

var
  Form1: TForm1;

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 Memo1
 Memo1.Lines.Add('Лаб. раб. N1. Ст. гр. 740102 Иванов А.А.');
end;

procedure TForm1.Button1Click(Sender: TObject);
 var
  x,y,z,a,b,c,u : extended;
begin
```

```
  x:=StrToFloat(Edit1.Text);                // Считывается значение X
 Memo1.Lines.Add(' X = '+Edit1.Text);   // Вывод X в окно Memo1
  y:=StrToFloat(Edit2.Text);                // Считывается значение Y
 Memo1.Lines.Add(' Y = '+Edit2.Text);   // Вывод Y в окно Memo1
  z:=StrToFloat(Edit3.Text);                // Считывается значение Z
 Memo1.Lines.Add(' Z = '+Edit3.Text);   // Вывод Z в окно Memo1

   // 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.
```

1.4. Выполнение индивидуального задания

Ниже приведено 30 вариантов задач. По указанию преподавателя выберите свое индивидуальное задание. Уточните условие задания, количество, наименование, типы исходных данных. В соответствии с этим установите необходимое количество окон Edit, тексты заголовков на форме, размеры шрифтов, а также типы переменных и функции преобразования при вводе и выводе результатов.

Прочтите в прил.1 описание меню File, Edit, Run, а в прил.2 описание компонентов TEdit, TMemo, TButton. С помощью инспектора объектов измените цвет формы, шрифт выводимых символов.

Индивидуальные задания

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

При x=14.26, y=-1.22, $z=3.5\times10^{-2}$ t=0.564849.

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

При x=-4.5, $y=0.75\times10^{-4}$,$z=0.845\times10^{2}$ u=-55.6848.

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

При x=3.74×10^{-2}, y=-0.825, z=0.16×10^{2}, v=1.0553.

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

При x=0.4×10^4 ,y=-0.875, z=-0.475×10^{-3} w=1.9873.

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

При x=-15.246, y=4.642×10^{-2}, z=20.001×10^2 α=-182.036.

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

При x=16.55×10^{-3}, y=-2.75, z=0.15 β=-38.902.

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

При x=0.1722, y=6.33, z=3.25×10^{-4} γ=-172.025.

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

При x=-2.235×10^{-2}, y=2.23, z=15.221 φ=39.374.

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

При x=1.825×10^2, y=18.225, z=-3.298×10^{-2} ψ=1.2131.

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

При x=3.981×10^{-2}, y=-1.625×10^3, z=0.512 a=1.26185.

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

При x=6.251, y=0.827, z=25.001 b=0.7121.

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

При x=3.251, y=0.325, z=0.466×10^{-4} c=4.025.

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

При x=17.421, y=10.365×10^{-3}, z=0.828×10^5 f=0.33056.

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

При x=12.3×10^{-1}, y=15.4, z=0.252×10^{3} g=82.8257.

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

При x=2.444, y=0.869×10^{-2}, z=-0.13×10^{3} h = -0.49871.

16. Вывести на экран *1* или *0* в зависимости от того, имеют 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 ноль.

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.

25. Целой переменной S присвоить сумму цифр трехзначного целого числа *k*.

26. Идет *k*-я секунда суток. Определить, сколько полных часов (*h*) и полных минут (*m*) прошло к этому моменту.

27. Определить *f* – угол (в градусах) между положением часовой стрелки в начале суток и ее положением в *h* часов, *m* минут и *s* секунд (*0<=h<=11, 0<=m, s<=59*).

28. Определить *h* – полное количество часов и *m* – полное количество минут, прошедших от начала суток до того момента(в первой половине дня), когда часовая стрелка повернулась на *f* градусов(*0<=f<360, f – вещественное число*).

29. Пусть *k* – целое от 1 до 365. Присвоить целой переменной *n* значение 1,2,…6 или 7 в зависимости от того, на какой день недели (понедельник, вторник,…, суббота или воскресенье) приходится *k*- й день невисокосного года, в котором 1 января - понедельник.

30. Поменять местами значения целых переменных *x* и *y*, не используя дополнительные переменные.

ТЕМА 2. ПРОГРАММИРОВАНИЕ РАЗВЕТВЛЯЮЩИХСЯ АЛГОРИТМОВ

***Цель лабораторной работы**:* научиться пользоваться простейшими компонентами организации переключений (TCheckBox, TRadioGroup). Написать и отладить программу разветвляющегося алгоритма.

2.1. Операторы **if** *и* **case** *языка Паскаль*

Для программирования разветвляющихся алгоритмов в языке Pascal используются специальные 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.

Например, если bl: boolean, x,y,u:integer, то фрагмент программы с оператором if может быть таким:

```
bl:=x>y;
  if  bl   then u:=x-y;
           else u:=x-y;
```

Оператор выбора **case** организует разветвления в зависимости от значения некоторой переменной перечисляемого типа.

Например, если In: integer, то после выполнения

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

В соответствии со значением in вычисляется u. Если in=0, то u=x+y, если in=1, то u=x-y, если in=2, то u=x*y и, наконец, u=0 при любых значениях in отличных от 0, 1 или 2.

2.2. 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 (рис.2.1) представлены кнопки-переключатели двух типов (TCheckBox, 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.

2.3. Пример написания программы

Задание: ввести три числа - x,y,z. Вычислить по усмотрению 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 рис. 2.1, и написать соответствующую программу.

2.3.1. Создание формы

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

2.3.2.Работа с компонентом TCheckBox

Выберите в меню компонентов Standard пиктограмму  и поместите ее в нужное место формы. С помощью инспектора объектов измените заголовок (Caption) на “maxabs”. В тексте программы появилась переменная CheckBox1 типа TCheckBox. Теперь в зависимости от того, нажата или нет кнопка, булевская переменная **CheckBox1.Checked** будет принимать значения true или false.

2.3.3. Работа с компонентом TRadioGroup

Выберите в меню 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 (кнопки размещаются в одном столбце). Дважды щелкните по правой части свойства Items мышью, появится строчный редактор списка заголовков кнопок. Наберите три строки с именами: в первой строке - cos(x), во второй - sin(x), в третьей - tg(x), нажмите ОК.

После этого на форме внутри окаймления появится три кнопки-переключателя с введенными надписями.

Обратите внимание на то, что в тексте программы появилась переменная RadioGroup1 типа TRadioGroup. Теперь при нажатии одной из кнопок группы в переменной целого типа **RadioGroup1.ItemIndex** будет находиться номер нажатой клавиши (отсчитывается от нуля), что используется в тексте приведенной программы.

2.3.4. Создание обработчиков событий FormCreate и Botton1Click

Процедуры - обработчики событий FormCreate и Botton1Click создаются аналогично тому, как и в первой теме. Текст процедур приведен ниже.

Запустите программу и убедитесь в том, что все ветви алгоритма выполняются правильно. Форма приведена на рис.2.1.

Текст программы приведен ниже.

Рис. 2.1

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;

type
  TForm1 = class(TForm)
    CheckBox1: TCheckBox;
    RadioGroup1: TRadioGroup;
    Memo1: TMemo;
    Button1: TButton;
    Edit1: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Edit2: TEdit;
    Label3: TLabel;
    Edit3: TEdit;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(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';
  Memo1.Clear;
  Memo1.Lines.Add('Рез-ты ст. гр.920201 Петрова И.И.');
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);
       Memo1.Lines.Add(' y='+Edit2.Text);
    z:=StrToFloat(Edit3.Text);
       Memo1.Lines.Add(' z='+Edit3.Text);
           // Проверка номера нажатой кнопки и выбор соответствующей ей функции
    case RadioGroup1.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,ffGeneral,8,2));

end;

end.
```

2.4. Выполнение индивидуального 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. $$a=\begin{cases}(f(x)+y)^2-\sqrt{f(x)y}, & xy\rangle 0\\ (f(x)+y)^2+\sqrt{|\mathrm{f(x)\,y}|}, & xy\langle 0\\ (f(x)+y)^2+1, & xy=0.\end{cases}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

16. Известно, что из четырех чисел a_1, a_2, a_3 и 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 *c* – 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<=b<=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) и 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 окончание слова “бутылка” с числом *k*.

25. Type курс=(С,В,Ю,З); {север, восток, юг, запад}
Приказ=(вперед, вправо, назад, влево);
Var K1,K2:курс; ПР:приказ;

Корабль сначала шел по курсу *K1*, а затем его курс был изменен согласно приказу ПР. Определить *K2* - новый курс корабля.

26. Type месяц=(январь,февраль,март,апрель,май,июнь,июль,август,
сентябрь,октябрь,ноябрь,декабрь);
день=1..31;
var d1,.d2:день;
m1,m2:месяц;
t:boolean;

Переменной *t* присвоить значение *1* если дата *d1, m1* предшествует (в рамках года) дате *d2, m2,* и значение *0* в других случаях.

27. *Type нота=(до,ре,ми,фа,соль,ля,си);*
интервал=(секунда,терция,кварта,квинта,секста,септима);
var n1,n2:нота;
i:интервал;

Определить *i*-й интервал, образованный нотами *n1* и *n2(n1≠n2)*: секунда - это интервал из двух соседних (по кругу) нот (например, ре и ми, си и до), терция - интервал через ноту (например, фа и ля, си и ре) и т.д.

28. *Type единица=(дециметр,километр,метр,миллиметр,сантиметр);*
длина=real;

Var x : длина;
P : единица;

Значение переменной *x*, означающее некоторую длину в единицах *p,*заменить на величину этой же длины в метрах.

29. *Type сезон=(зима,весна,лето,осень);*
Var m:месяц; {определение «месяц» см. в 26}
S:сезон;

Определить *s*-сезон, на который приходится месяц *m*.

30. *Var k:1..9;* Вывести значение переменной *k* римскими цифрами.

ТЕМА 3. ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ

Цель лабораторной работы: изучить простейшие средства отладки программ в среде DELPHI. Составить и отладить программу циклического алгоритма.

3.1. Операторы организации циклов repeat, whyle, for *языка Pascal*

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 три различных оператора цикла.

Оператор

```
repeat
    <операторы>
until<условие>;
```

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

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

Оператор

```
for i:=i1 to i2 do begin
    <операторы>
          end;
```

организует повторение операторов при нарастающем изменении переменной цикла i от начального значения i1 до конечного i2 с шагом “единица”. Заметим, что если i2>i1, то <операторы> не выполнятся ни разу. Модификация оператора **for** i:=i2 **downto** i1 **do begin** <операторы> **end** организует повторения при убывающем изменении i на единицу.

3.2. Средства отладки программ в DELPHI

Практически в каждой вновь написанной программе после запуска обнаруживаются ошибки.

Ошибки первого уровня (ошибки компиляции) связаны с неправильной записью операторов (орфографические, синтаксические). 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 выполнения) 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 ноль и др. Поэтому перед использованием отлаженной программы ее надо 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 подозрительным участком и нажать клавишу F4 (выполнение до курсора). Выполнение программы будет остановлено на строке, содержащей курсор. Теперь можно увидеть, чему равно значение интересующих переменных. Для этого можно поместить на нужную переменную курсор (на экране будет высвечено ее значение) либо нажать Ctrl-F7 и в появившимся 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 <Run> меню Run.

3.3. Порядок выполнения задания

Задание: написать и отладить программу, которая выводит таблицу значений функции S(x) для x изменяющихся в интервале от X1 до X2 с шагом h.

$$S(x) = \sum_{k=0}^{N} (-1)^k \frac{x^k}{k!}$$

Панель диалога представлена на рис.3.1.
Текст программы приведен ниже.

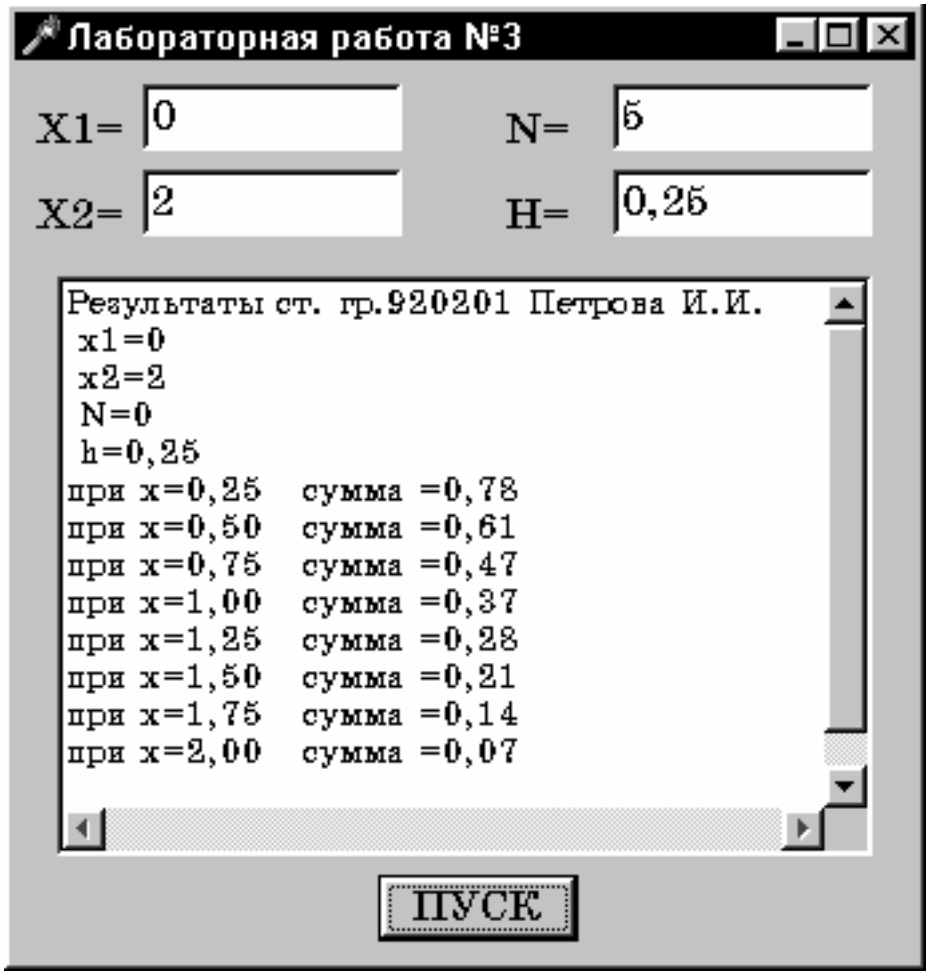

Рис. 3.1

```
unit tema3;

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
{$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.920201 Петрова И.И.');
end;

procedure TForm1.Button1Click(Sender: TObject);
 var x1,x2,x,h,a,s:extended;
     N,k,c:integer;
begin

     x1:=StrToFloat(Edit4.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='+Edit1.Text);
     h:=StrToFloat(Edit4.Text);
        Memo1.Lines.Add(' h='+Edit4.Text);
        c:=-1;    x:=x1;
  repeat
         a:=1;    S:=1;
      for k:=1 to N do
          begin
            a:=c*a*x/k;
            s:=s+a;
          end;
     Memo1.Lines.Add('при x='+FloatToStrF(x,ffFixed,6,2)+'   сумма ='
                             +FloatToStrF(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 программу и проследите, как меняются все переменные в процессе выполнения.

3.4. Выполнение индивидуального задания

По указанию преподавателя выберите нужный вариант задачи из нижеприведенного списка. Откорректируйте панель диалога и текст программы.

Индивидуальные 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 $h=(x_n-x_k)/10$. 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	$x-\frac{x^3}{3!}+\ldots+(-1)^n\frac{x^{2n+1}}{(2n+1)!}$	16	$\sin x$
2	0.1	1	$1+\frac{x^2}{2!}+\ldots+\frac{x^{2n}}{(2n)!}$	10	$\frac{e^x+e^{-x}}{2}$
3	0.1	1	$1+\frac{\cos\frac{\pi}{4}}{1!}x+\ldots+\frac{\cos n\frac{\pi}{4}}{n!}x^n$	12	$e^{x\cos\frac{\pi}{4}}\cos(x\sin\frac{\pi}{4})$
4	0.1	1	$1-\frac{x^2}{2!}+\ldots+(-1)^n\frac{x^{2n}}{(2n)!}$	8	$\cos x$
5	0.1	1	$1+3x^2+\ldots+\frac{2n+1}{n!}x^{2n}$	14	$(1+2x^2)e^{x^2}$
6	0.1	1	$x+\frac{x^3}{3!}+\ldots+\frac{x^{2n+1}}{(2n+1)!}$	8	$\frac{e^x-e^{-x}}{2}$
7	0.1	1	$\frac{x^3}{3}-\frac{x^5}{15}+\ldots+(-1)^{n+1}\frac{x^{2n+1}}{4n^2-1}$	12	$\frac{1+x^2}{2}arctgx-\frac{x}{2}$
8	0.1	1	$1+\frac{2x}{1!}+\ldots+\frac{(2x)^n}{n!}$	10	e^{2x}
9	0.1	1	$1+2\frac{x}{2}+\ldots+\frac{n^2+1}{n!}\left(\frac{x}{2}\right)^n$	14	$\left(\frac{x^2}{4}+\frac{x}{2}+1\right)e^{\frac{x}{2}}$

Окончание табл. 3.1

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

16. Подсчитать 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 0. 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 трехзначных натуральных чисел, сумма цифр которых равна n(1<=n<=27). Операции деления (/, div и mod) не использовать.

24. Вывести на экран в возрастающем порядке все трехзначные числа, в десятичной записи которых нет одинаковых цифр (операции деления не использовать).

25. Переменной t присвоить значение 1 или 0 в зависимости от того, можно или нет натуральное число n представить в виде трех полных квадратов.

26. Дано натуральное число n. Выяснить, входит ли цифра 3 в запись числа n^2.

27. Дано натуральное число n. Найти сумму его цифр.

28. Дано целое n>0, за которым следует n вещественных чисел. Определить, сколько среди них отрицательных.

29. Дано натуральное число n. Переставить местами первую и последнюю цифры числа n.

30. Дано натуральное число n. Заменить порядок следования цифр числа n на обратный.

ТЕМА 4. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ МАССИВОВ

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

4.1. 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[1,1,I*j-1].. 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 элемента 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[I]+1]*3;
A[n]:=1+sqrt(abs(a[n-1]));
```

4.2. Компонент TStringGrid

При работе с массивами ввод и вывод информации на экран удобно организовывать в виде таблиц. Компонент TStringGrid 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 - индекс элемента 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.

4.3. Порядок выполнения задания

Задание: создать программу для определения вектора $\vec{Y} = A * \vec{B}$, где A - квадратная матрица размерностью NxN, а Y, B – одномерные массивы размерностью N. Элементы вектора Y определяются по формуле $Y_i = \sum_{j=1}^{N} A_{ij} \cdot B_i$. Значения N вводить в компонент TEdit, A и B - в компонент TStringGrid. Результат, после нажатия кнопки типа TButton, вывести в компонент TStringGrid.

Панель диалога приведена на рис. 4.1.

Настройка компонента TStringGrid

Для установки компонента TStringGrid на форму необходимо на странице

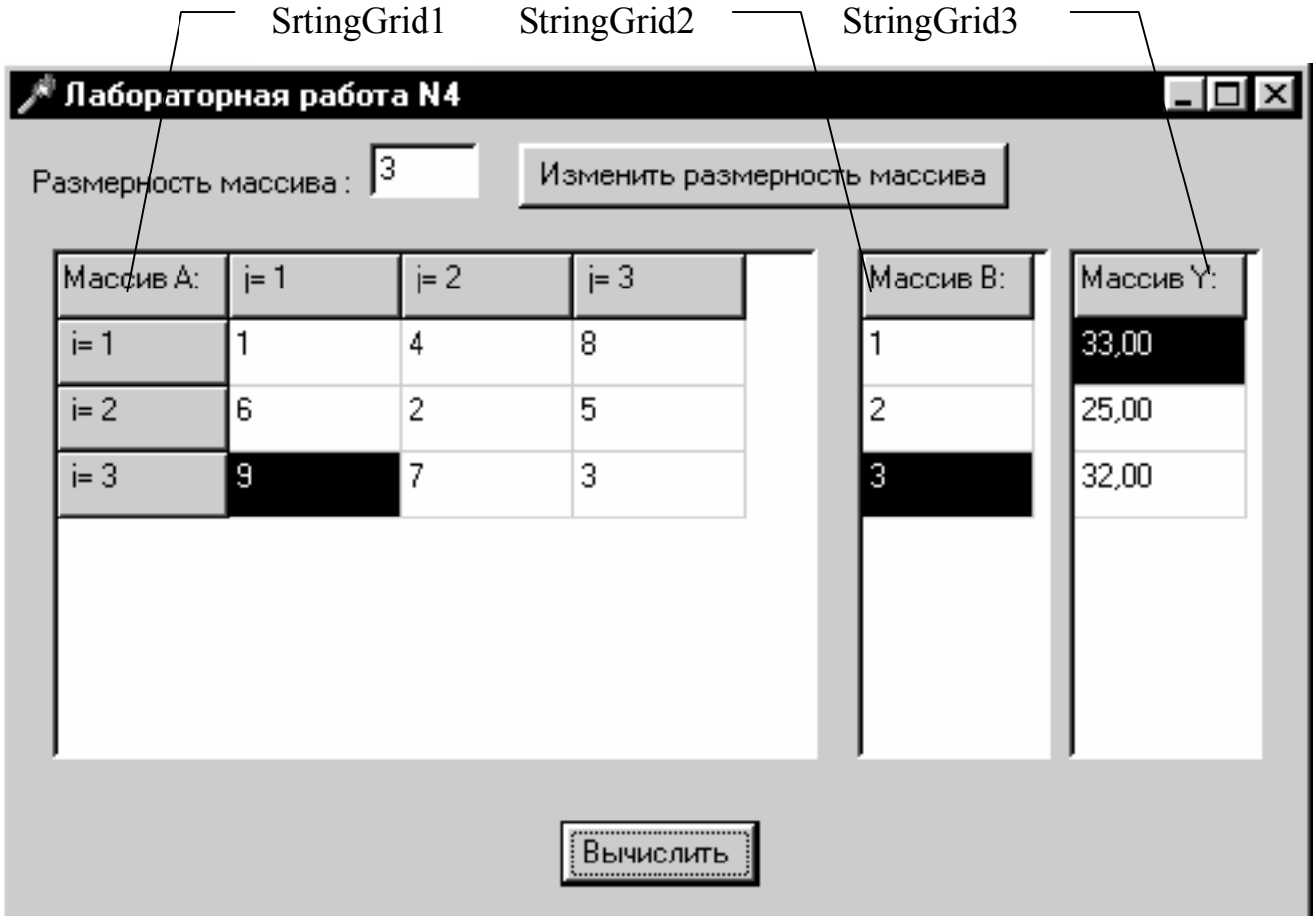

Рис. 4. 1

Additional меню 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 (две строки и два столбца), а FixedCols и FixedRows установите 1 (один столбец и одна строка с фиксированной зоной). Т.к.

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 приведен ниже.

```
unit tem4;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Grids;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Button1: TButton;
    Button2: TButton;
    StringGrid1: TStringGrid;
    StringGrid2: TStringGrid;
    StringGrid3: TStringGrid;
    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
  Mas2 = array[1..Nmax,1..Nmax] of extended;  // Объявление типа двумерного массива
размерностью Nmax
  Mas1 = array[1..Nmax] of extended;              // Объявление типа одномерного массива
размерностью Nmax
var
  Form1: TForm1;
  A : Mas2;              // Объявление двумерного массива
  B,Y : Mas1;           // Объявление одномерных массивов
  N,i,j : integer;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
```

```
begin
    N:=3;          // Размерность массива
      Edit1.Text:=FloatToStr(N);
 {Задание числа строк и столбцов в таблицах}
   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 TForm1.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 TForm1.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.
```

4.4. Индивидуальные задания

Во всех заданиях по теме «Массивы» скалярные переменные вводить с помощью компонента TEdit с соответствующим пояснением в виде компонента 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 *NxM*. 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 *NxM*. Получить массив *B*, присвоив его *k*-му элементу значение 1, если элементы *k*–й строки матрицы упорядочены по убыванию, и значение 0 в противном случае.

3. Задана матрица размером *NxM*. Получить массив *B*, присвоив его *k*-му элементу значение 1, если *k*-я строка матрицы симметрична, и значение 0 в противном случае.

4. Задана матрица размером *NxM*. Определить *k* – количество “особых” элементов матрицы, считая элемент “особым”, если он больше суммы остальных элементов своего столбца.

5. Задана матрица размером *NxM*. Определить *k* – количество “особых” элементов матрицы, считая элемент “особым”, если в его строке слева от него находятся элементы, меньшие его, а справа – б*о*льшие.

6. Задана символьная матрица размером *NxM*. Определить *k* - количество различных элементов матрицы (т.е. повторяющиеся элементы считать один раз).

7. Дана матрица размером *NxM*. Упорядочить ее строки по неубыванию их первых элементов.

8. Дана матрица размером *NxM*. Упорядочить ее строки по неубыванию суммы их элементов.

9. Дана матрица размером *NxM*. 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 *NxM* 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 *NxM* поменять местами строку, содержащую элемент с наибольшим значением со строкой, содержащей элемент с наименьшим значением.

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 возростанию.

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 упоряченными по неубыванию.

25. Дан массив из *k* символов. Определить, симметричен ли он, т.е. читается ли он одинаково слева направо и справа налево.

26. Дано два массива. Найти наименьшее среди тех элементов первого массива, которые не входят во второй массив.

27. Определить количество инверсий в этом массиве X (т.е. таких пар элементов, в которых большее число находится слева от меньшего: $x_i > x_j$ при $i<j$).

28. Дан массив из строчных латинских букв. Вывести на экран в алфавитном порядке все буквы, которые входят в этот текст по одному разу.

29. Вывести на экран заданный массив из *k* символов, удалив из него повторные вхождения каждого символа.

30. Определить сколько различных символов входит в заданный текст, содержащий не более *k* символов и оканчивающийся точкой (в сам текст точка не входит).

ТЕМА 5. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ СТРОК

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

5.1. Типы данных для работы со строками

5.1.1. 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.

5.1.2. 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 прил. 4.

5.1.3. 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.

5.1.4. 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).

5.1.5. 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.

5.2. 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.

5.3. 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.

5.4. Компонент TBitBtn

Компонент TBitBtn расположен на странице Additonal палитры компонентов и представляет собой разновидность стандартной кнопки TBotton. Его отличительная особенность – наличие растрового изображения на поверхности кнопки, которое определяется свойством Clyph. 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.

5.5. 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 присуши большинству компонентов. Наиболее часто применяемые события представлены в табл. 5.1.

Таблица 5.1

Событие	Описание события
OnActivate	Форма получает это событие при активации
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

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 Ctlr, 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

5.6. Порядок выполнения индивидуального 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 TComboBox. Ввод строки заканчивать нажатием клавиши Enter. Для выхода из программы использовать кнопку Close.

Панель диалога будет иметь вид (рис. 5.1).

Текст программы приведен ниже.

```
unit tema5;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons;

type
  TForm1 = class(TForm)
    Label2: TLabel;
    Label3: TLabel;
    BitBtn1: TBitBtn;
    ComboBox1: TComboBox;
    Label1: TLabel;
    procedure FormActivate(Sender: TObject);
    procedure ComboBox1KeyPress(Sender: TObject; var Key: Char);
    procedure ComboBox1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
```

```
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

    // Обработка события активизации формы
procedure TForm1.FormActivate(Sender: TObject);
begin
ComboBox1.SetFocus;                          // Передача фокуса ComboBox1
end;

   // Обработка события нажатия левой клавиши мыши
procedure TForm1.ComboBox1KeyPress(Sender: TObject; var Key: Char);
begin
if key=#13 then begin                          // Если нажата клавиша Enter, то...
ComboBox1.Items.Add(ComboBox1.Text);  // Строка из окна редактирования
                                                    // заносится
 в список выбора
ComboBox1.Text:='';                          // Очистка окна редактирования
               end;
end;

procedure TForm1.ComboBox1Click(Sender: TObject);
var st : string;
     n,i,nst,ind: integer;
begin
  n:=0;                                          // Содержит число слов
      ind:=0;
  nst:=ComboBox1.ItemIndex;                 // Определение номера выбранной строки
  st:=ComboBox1.Items[nst];            // Занесение выбранной строки в переменную st
    for i:=1 to Length(st) do begin            // Просмотр всех символов строки st
       case ind of
          0 : if st[i]<>' ' then begin             // Если встретился символ после пробела
                ind:=1;
                 n:=n+1;                           // Число слов увеличивается на единицу
                           end;
          1 : if st[i]=' ' then ind:=0;           // Если встретился пробел после символов
                         end;
        end;

   Label3.Caption:=IntToStr(n);              // Вывод числа слов в Label3

end;

end.
```

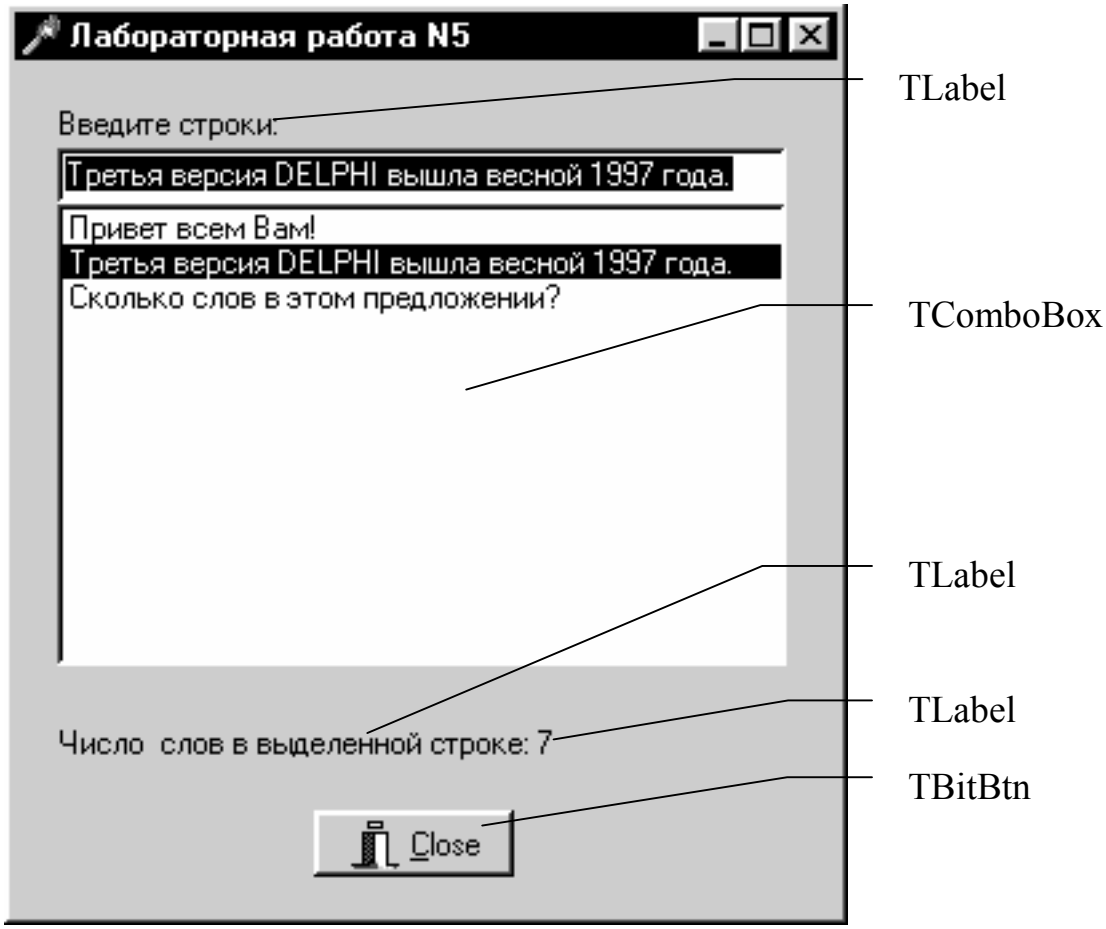

Рис. 5. 1.

5.7. Индивидуальные задания

Во всех заданиях исходные данные вводить с помощью компонента TEdit в компонент TListBox либо с помощью свойства Text в свойство Items компонента TComboBox. Скалярный результат выводить с помощью компонента TLabel. Ввод 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 “+” и “-“. 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 первые *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*- 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 $S_1 S_2 \ldots S_m$, в которой могут встречаться цифры, пробелы, буква “E” и знаки “+”, “-“. Известно, что первый символ S_1 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»).

25. Составить программу, которая читает построчно текст другой программы (ввести с клавиатуры) на языке *Pascal,* обнаруживает комментарии и выводит их на экран.

26. Составить программу, которая читает построчно текст другой программы (ввести с клавиатуры) на языке *Pascal*, подсчитывает количество ключевых слов «*begin*» и «*end*» и выводит на экран соответствующее сообщение.

27. Разработать программу, которая заданное целое число от 1 до 1999 выводит на экран римскими цифрами.

28. Дан текст из заглавных латинских букв, за которым следует пробел. Определить, является ли этот текст правильной записью римскими цифрами целого числа от 1 до 999, и, если является, вывести на экран это число арабскими цифрами(в десятичной системе).

29. Дан текст из *k* символов. Вывести на экран только строчные русские буквы, входящие в этот текст.

30. Дан текст из *k* символов. Вывести на экран в алфавитном порядке все различные прописные русские буквы, входящие в этот текст.

ТЕМА 6. ПРОГРАММИРОВАНИЕ 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.

6.1. Программирование с использованием переменных типа запись

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

```
Type                                    {Объявление типа запись}
        TStudent = record
                Fio: string[20];                {Поле ф.и.о.}
                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:=720603;
            . . .
```

Доступ к полям можно осуществлять также при помощи оператора with:

```
With Student do
        Begin
                Fio:= 'Иванов А.И.';
               Group:=720603;
        End;
```

6.2. 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 или открытие (Reset(F);)
                                 // уже существующего файла
        . . .
Read(F, Stud);                   // Чтение данных из файла или
                                 // запись (Write(F, Stud)) в файл
        . . .
CloseFile(F);                    // Закрытие файла
```

6.3. Подпрограммы 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 текстовой 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.

SeekEof(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.

6.4. Компоненты TOpenDialog и TSaveDialog

Компоненты TOpenDialog и TSaveDialog находятся на странице DIALOGS (см. прил. 2). 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 содержатся 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.

6.5. Порядок 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.

6.5.1. Настройка компонентов TOpenDialog и TSaveDialog

Для установки компонентов TOpenDialog и TSaveDialog на форму необходимо на странице Dialogs меню 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

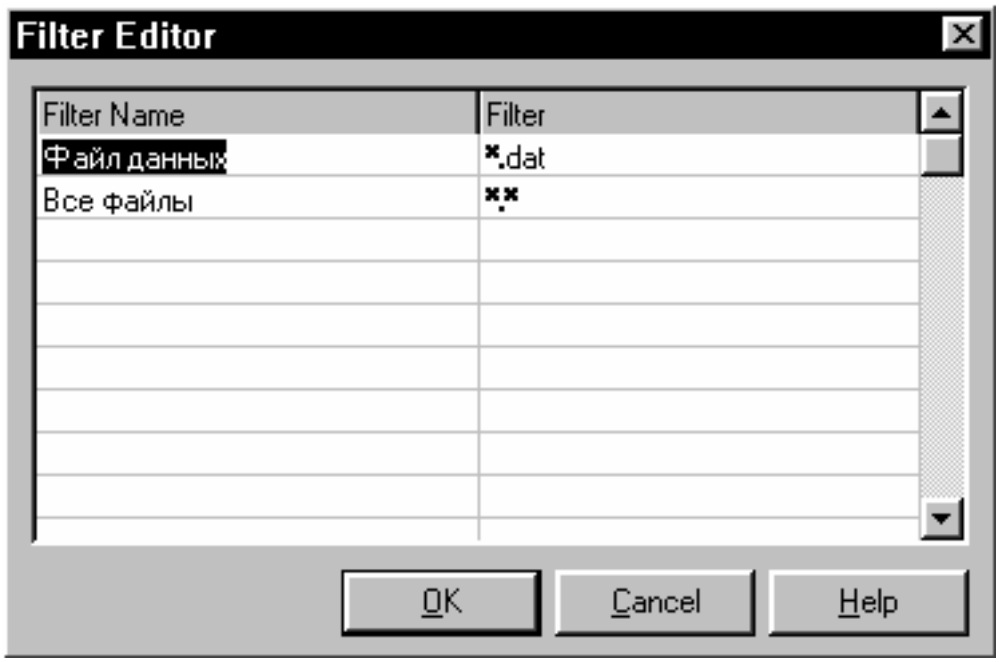

Рис. 6.1

– маску. Для OpenDialod1 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).

Для того, чтобы файл автоматически записывался с расширением .dat, в свойстве DefaultExt запишем требуемое расширение - .dat.

Аналогичным образом настроим SaveDialog1 для текстового файла (расширение .txt).

6.5.2. 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 “Сортировка” 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

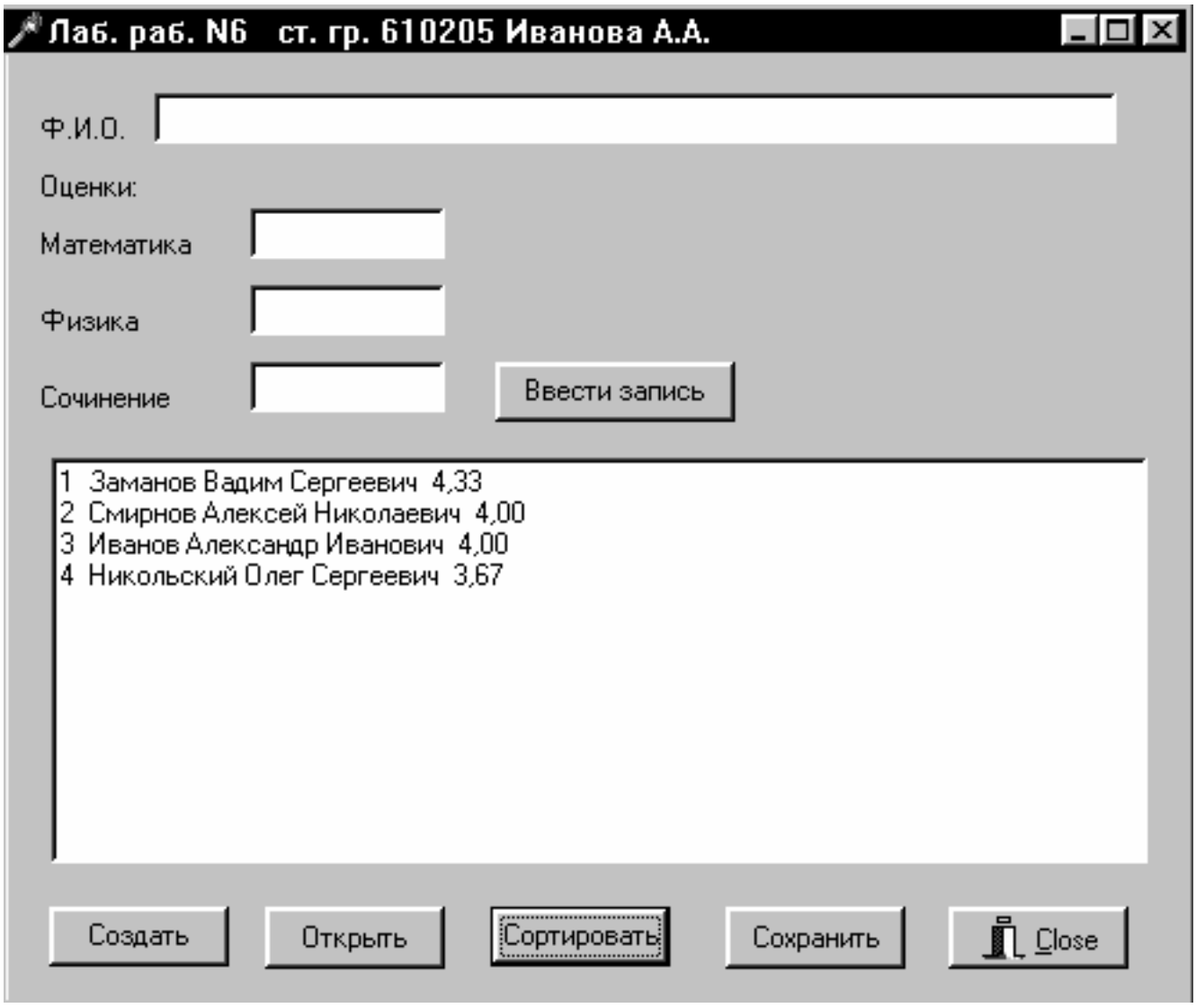

Рис. 6.2

текстовой файл, содержащий отсортированную ведомость. Файл записей закрывается одновременно с программой при нажатии на кнопку “Close” или  .

Текст программы приведен ниже.

```
unit tema6;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, ExtCtrls;

type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Memo1: TMemo;
    Button1: TButton;
    Button3: TButton;
    Splitter1: TSplitter;
    Button5: TButton;
    BitBtn1: TBitBtn;
    SaveDialog1: TSaveDialog;
    Button2: TButton;
    OpenDialog1: TOpenDialog;
    Button4: TButton;
    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);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  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;        // Имя файла

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);      // Ввести новую запись
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); // Создание нового файла записей
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
   end;
    Button1.Show;                    // Сделать видимой кнопку "Ввести запись"
end;

procedure TForm1.Button3Click(Sender: TObject); // Открыть существующий файл
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                // Сортировка массива записей
  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); // Сохранение 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.
```

6.6.Выполнение индивидуального 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, почасовой 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 цехи 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 t 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.),марку изделия, дату 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, квалификацию (разряд или образование), стаж работы по специальности, заработную плату, 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. Поля шахматной доски характеризуются записью

```
Type
 Pole=record
        Ver:(a,b,c,d,e,f,g,h);  {вертикальные координаты}
        Hor:1..8;               {горизонтальные координаты}
       end;
```

Вывести шахматную доску, пометив крестиками все поля, которые «бьет» ферзь, стоящий на поле с координатами Ver_i и Hor_i, и ноликами - остальные поля.

24. Поля шахматной доски характеризуются записью (см. задание 23)

```
Var  Figura:Pole;
```

Вывести сообщение, может ли конь за один ход перейти с поля $Figura_i$ на поле $Figura_j$.

25. *Type*

```
Karta=record
        m: (piki,trefi,bubni,chervi);                    {масть}
        d:(shest,sem,vosem,devjat,desjat,valet,dama,korol,tuz);   {достоинство}
      end;
Var   k1,k2:Karta;
```

Вывести сообщение, «бьет» ли карта *k1* карту *k2,* с учетом того что масть m_i является козырной.

26. Для участия в 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.

27. При постановке на учет в ГАИ автолюбители указывают следующие данные: марка автомобиля, год выпуска, номер двигателя, номер кузова, цвет, номерной знак, Ф.И.О и адрес владельца. Вывести список автомобилей, проходящих техосмотр в текущем году, сгруппированных по маркам автомобилей. Учесть, что если текущий год четный, техосмотр проходят автомобили с четными номерами двигателей, иначе – с нечетными номерами.

28. Для участия в конкурсе исполнителей необходимо заполнить следующую анкету: Ф.И.О., год рождения, название страны, класс музыкального инструмента(гитара, фортепиано, скрипка, виолончель). Вывести список самых молодых лауреатов конкурса по классам инструментов в порядке занятых мест.

29. Список группы студентов содержит следующую информацию: Ф.И.О., рост и вес. Вывести Ф.И.О. студентов, рост и вес которых чаще всего встречаются в списке.

30. 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.

ТЕМА 7. ПРОГРАММИРОВАНИЕ 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.

7.1. Использование подпрограмм

Подпрограмма – это именованная, определенным образом оформленная группа операторов, которая может быть вызвана любое количество раз из любой точки основной программы.

Подпрограммы используются 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;
```

В отличие от процедур функции могут использоваться в выражениях в качестве операнда, поэтому они имеют следующую структуру:

```
Function <имя функции> ([список формальных параметров]): <тип результата>;
      Const [описание используемых констант];
      Type [описание используемых типов];
      Var [описание используемых переменных];
Begin
…                    // Операторы
      Result:= … ; // Занесение результата вычислений в Result
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 осуществляется 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 функция по окончании своей работы.

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

7.2. Использование модулей

Модуль – автономно компилируемая программная единица, включающая в себя процедуры, функции, а также различные компоненты раздела описаний. Структура модуля представлена в п.1.2 и 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.

Инициирующая и завершающая части модуля используются крайне редко.

7.3. Порядок 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 $\sin^2(x)$.

7.3.1. 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. Т.к. 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 Matfu;. Затем необходимо сохранить файл с именем, совпадающим

с именем заголовка модуля: Matfu.pas. Следует обратить внимание на то, что имя файла должно совпадать с именем модуля, иначе DELPHI не сможет подключить его к другой программе.

7.3.2. 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 – MatFu. Теперь в проекте можно использовать функции, содержащиеся в модуле.

Панель диалога будет иметь следующий вид (рис. 7.1).

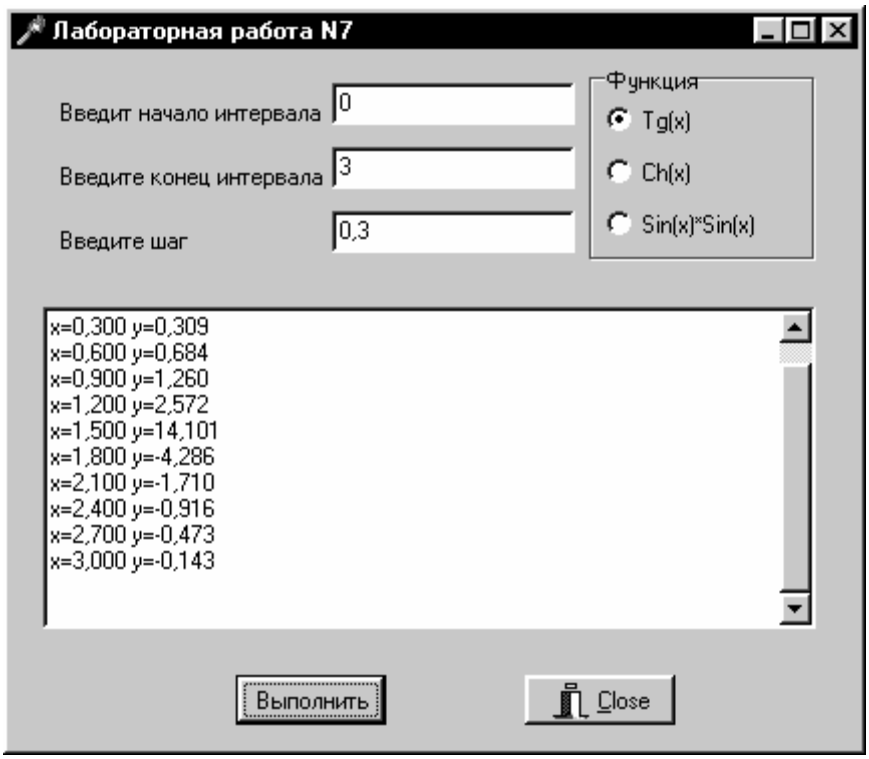

Рис. 7.1

Тексты модуля и вызывающей программы приведены ниже.

Текст модуля:

```
unit Matfu;

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;
  begin
    Result:=Sin(x)/Cos(x);
 end;
```

```
 Function Ch;
  begin
    Result:=(exp(x)-exp(-x))/2;
 end;
 Function Sin2;
  begin
    Result:=sqr(sin(x));
 end;

end.
```

Текст вызывающей программы:

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, ExtCtrls,MatFu;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Memo1: TMemo;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    RadioGroup1: TRadioGroup;
    procedure FormCreate(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
 Type
   fun = function(x:extended):extended; // Объявление типа функция
var
  Form1: TForm1;

implementation

{$R *.DFM}

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

```
 Edit1.Text:='0';
 Edit2.Text:='3';
 Edit3.Text:='0,3';
 Memo1.Clear;
  RadioGroup1.ItemIndex:=0;
end;

procedure Tabl(f:fun;xn,xk,h:extended);  // Расчет таблицы
   var x,y: extended;
  begin
    x:=xn;
   repeat
     y:=f(x);
    Form1.Memo1.Lines.Add('x='+FloatToStrf(x,fffixed,8,3)+
                        ' y='+FloatToStrf(y,fffixed,8,3));
       x:=x+h;
   until (x>xk);
end;

procedure TForm1.BitBtn1Click(Sender: TObject);
var xn,xk,h : extended;
begin
  xn:=StrToFloat(Edit1.Text); //  Начальное занчение интервала
  xk:=StrToFloat(Edit2.Text); //  Конечное значение интервала
  h:=StrToFloat(Edit3.Text);  //  Шаг расчета
   case RadioGroup1.ItemIndex of   // Выбор функции
    0 : Tabl(tg,xn,xk,h);
    1 : Tabl(ch,xn,xk,h);
    2 : Tabl(sin2,xn,xk,h);
     end;
end;
end.
```

7.4. Выполнение индивидуального задания

По указанию преподавателя выберите вариант задачи из заданий, приведенных в теме 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.

ТЕМА 8. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ СРЕДСТВ ДЛЯ ОТОБРАЖЕНИЯ ГРАФИЧЕСКОЙ ИНФОРМАЦИИ

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

8.1. Как строится график 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).

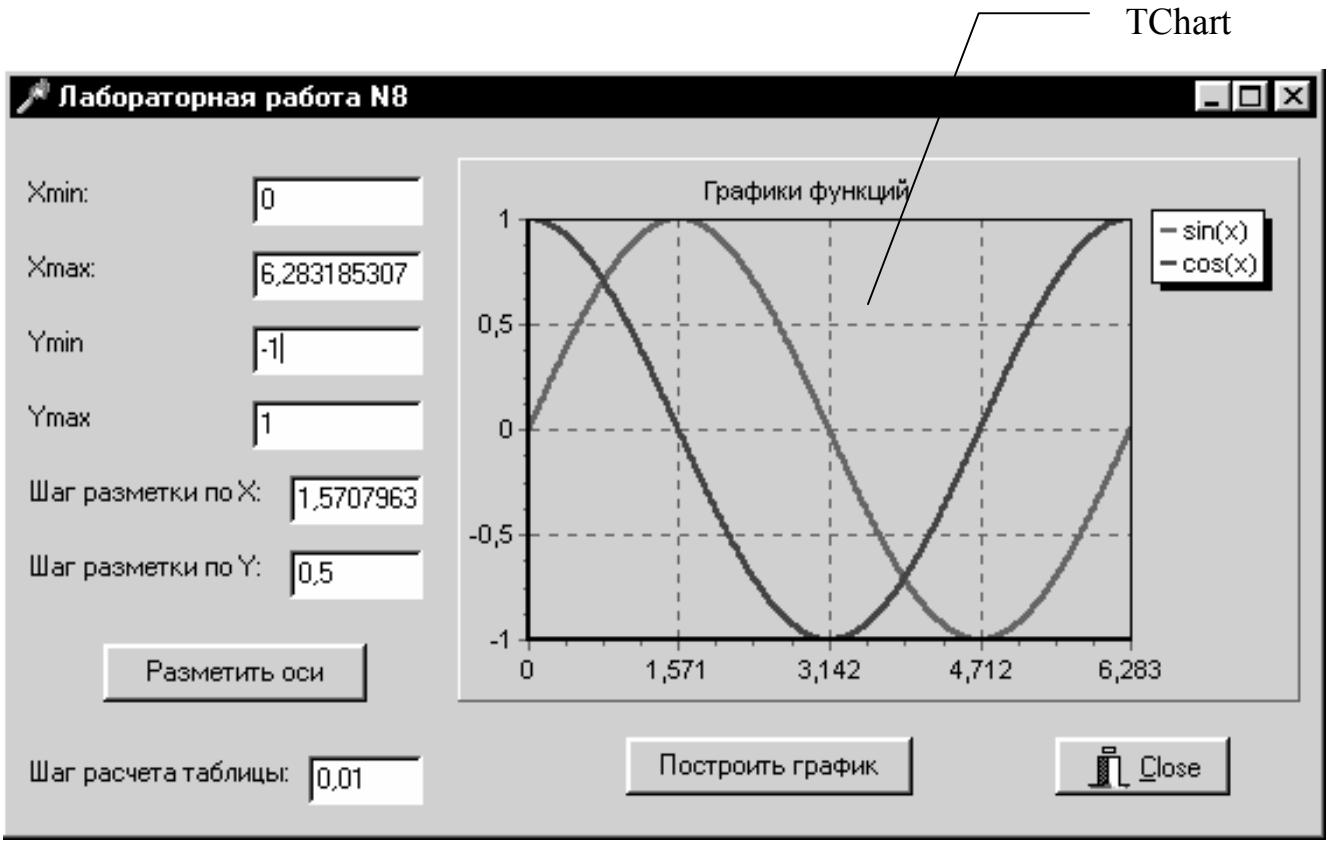

Рис.8.1

Построение графика (диаграммы) производится после вычисления таблицы значений функции y=f(x) на интервале [Xmin, Xmax]с заданным шагом. Полученная таблица передается в специальный двумерный массив Series**k** (**k** – номер графика) компонента TChart с помощью метода Add. Компонент PChart осуществляет всю работу по отображению графиков, переданных в объект Series**k:** 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 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 Chart1.BottomAxis содержит значение максимального предела нижней оси графика и при его изменении во время работы программы автоматически изменяется изображение графика (см. нижеприведенную программу).

8.2. Пример написания программы

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.

8.2.1. Настройка формы

Панель диалога программы организуется в виде, представленном на рис.8.1.

Для ввода исходных данных используются окна TEdit. Компонент TChart вводится в форму путем нажатия пиктограммы  в меню компонентов Standard.

8.2.2. Работа с компонентом TChart

Для изменения параметров компонента TChart необходимо дважды щелкнуть по нему мышью в окне формы. Появится окно редактирования EditingChat1 (рис. 8.2). Для создания нового объекта Series1 щелкнуть по кнопке

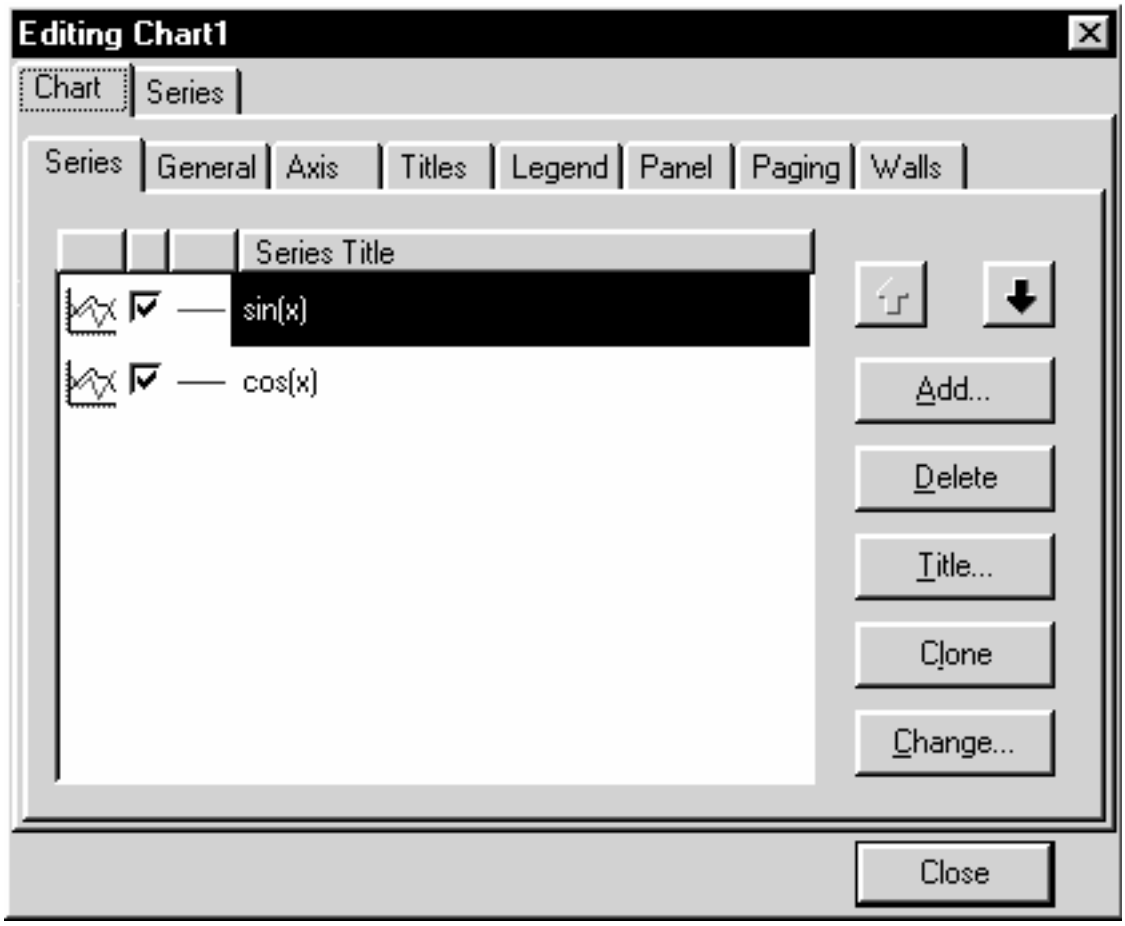

Рис.8.2

Add на странице Series. В появившемся диалоговом окне TeeChart Gallery 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 Series1. Для изменения названия нажать кнопку Title… В

появившемся 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 EditingChat.

8.2.3. Написание программы обработки события создания формы

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

8.2.4. Написание программ 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 Series**k** 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 имеет вид:

```
unit tem8;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, TeeProcs, TeEngine, Chart, Buttons, StdCtrls, Series;

type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Button1: TButton;
    Button2: TButton;
    BitBtn1: TBitBtn;
    Chart1: TChart;
```

```
    Series2: TLineSeries;
    Label6: TLabel;
    Edit6: TEdit;
    Label7: TLabel;
    Edit7: TEdit;
    Series1: 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
   {Установка начальных параметров координатных осей}
 Xmin:=0;
 Xmax:=2*pi;
 Ymin:=-1;
 Ymax:=1;
 Hx:=pi/2;
 Hy:=0.5;
 h:=0.01; // Установка шага расчета таблицы
   {Вывод данных в окна однострочных редакторов}
  Edit1.Text:=FloatToStr(Xmin);
  Edit2.Text:=FloatToStr(Xmax);
  Edit3.Text:=FloatToStr(Ymin);
  Edit4.Text:=FloatToStr(Ymax);
  Edit5.Text:=FloatToStr(Hx);
  Edit6.Text:=FloatToStr(Hy);
  Edit7.Text:=FloatToStr(h);
  Chart1.BottomAxis.Automatic:=False;// Отключение автоматического определения
                                     // параметров нижней оси
  Chart1.BottomAxis.Minimum:=Xmin;    // Установка левой границы нижней оси
  Chart1.BottomAxis.Maximum:=Xmax;    // Установка правой границы нижней оси
  Chart1.LeftAxis.Automatic:=False;       // Отключение автоматического определения
                                     // параметров левой оси
  Chart1.LeftAxis.Minimum:=Ymin;        // Установка нижней границы левой оси
  Chart1.LeftAxis.Maximum:=Ymax;       // Установка верхней границы левой оси
  Chart1.BottomAxis.Increment:=Hx;      // Установка шага разметки по нижней оси
  Chart1.LeftAxis.Increment:=Hy;          // Установка шага разметки по левой оси
```

```
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
{Чтение даннных из окон однострочных редакторов}
  Xmin:=StrToFloat(Edit1.Text);
  Xmax:=StrToFloat(Edit2.Text);
  Ymin:=StrToFloat(Edit3.Text);
  Ymax:=StrToFloat(Edit4.Text);
  Hx:=StrToFloat(Edit5.Text);
  Hy:=StrToFloat(Edit6.Text);
  Chart1.BottomAxis.Minimum:=Xmin;   // Установка левой границы нижней оси
  Chart1.BottomAxis.Maximum:=Xmax; // Установка правой границы нижней оси
  Chart1.LeftAxis.Minimum:=Ymin;       // Установка нижней границы левой оси
  Chart1.LeftAxis.Maximum:=Ymax;      // Установка верхней границы левой оси
  Chart1.BottomAxis.Increment:=Hx;     // Установка шага разметки по нижней оси
  Chart1.LeftAxis.Increment:=Hy;         // Установка шага разметки по левой оси
end;

procedure TForm1.Button2Click(Sender: TObject);
 var x,y1,y2: extended;
begin
  {Очистка графиков}
   Series1.Clear;
   Series2.Clear;
      Xmin:=StrToFloat(Edit1.Text);
      Xmax:=StrToFloat(Edit2.Text);
        h:=StrToFloat(Edit7.Text);  // Шаг расчета таблицы для графика
    x:=Xmin;     // Начальное значение по оси X
     repeat
      y1:=sin(x);                                  // Расчет функции
       Series1.AddXY(x,y1,'',clTeeColor);  // Вывод точки на график
      y2:=cos(x);                                 // Расчет функции
       Series2.AddXY(x,y2,'',clTeeColor);  // Вывод точки на график
        x:=x+h;                                   // Увеличение значения X на величину шага
      Until (x>Xmax);
end;

end.
```

8.3. Выполнение индивидуального задания

Постройте графики функций для соответствующих вариантов из темы №1. Таблицу данных получить изменяя параметр X с шагом h. Ввод исходных данных организовать через окна TEdit. Самостоятельно выбрать удобные параметры настройки.

ПРИЛОЖЕНИПЕ 1. КОМАНДЫ ОСНОВНОГО МЕНЮ

В меню **File** находятся команды для выполнения операций с проектами, модулями и файлами.

Команда	Описание
New	Позволяет выбрать тип элемента из репозитория (архива, в котором хранятся заготовки для новых программ) и создать его
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 его 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 From Project	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.

Команда	Описание
Undo	Отменяет ранее выполненные действия
Redo	Восстанавливает отмененные действия
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.

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 текстом
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, броузера объектов и других информационных утилит.

Команда	Описание
Project Manager	Менеджер проектов (Project Manager)
Project Source	Отображает исходный текст файла проекта
Object Inspector	Инспектор объектов (Object Inspector)
Alignment Palette	Палитра выравнивания компонентов
Browser	Броузер объектов (Object Browser)
Breakpoints	Список точек останова (Breakpoints List)
Call Stack	Стек вызовов (Call Stack)
Watches	Список точек слежения за переменными (Watch List)
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.

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 репозиторий 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 фирму 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.

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.

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 компонентов

Меню **Database** содержит средства для работы с базами данных.

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 репозиторий объектов.

Команда	Описание
Environment Options	Вызывает диалоговую панель настройки среды
Repository	Вызывает репозиторий
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.

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».

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 Applications»
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 «О программе»

ПРИЛОЖЕНИЕ 2. СВОЙСТВА КОМПОНЕНТОВ

П2.1. Общие свойства компонентов

Многие стандартные визуальные компоненты имеют одинаковые свойства. Поэтому имеет смысл рассмотреть их отдельно, чтобы впоследствии больше не возвращаться к этому.

Свойство Align

Задает способ выравнивания компонента внутри формы. Имеет одно из следующих значений:

Значение	Описание
alNone	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 следующих значений:

Значение	Цвет
clBlack	Черный (Black)
clMaroon	Темно-красный (Maroon)
clGreen	Зеленый (Green)
clOlive	Оливковый (Olive green)
clNavy	Темно-синий (Navy blue)
clPurple	Фиолетовый (Purple)
clTeal	Сине-зеленый (Teal)
clGray	Серый (Gray)
clSilver	Серебряный (Silver)
clRed	Красный (Red)
clLime	Ярко-зеленый (Lime green)
clBlue	Голубой (Blue)
clFuchsia	Сиреневый (Fuchsia)
clAqua	Ярко-голубой (Aqua)
dWhite	Белый (White)

Цвета, приведенные в следующей таблице, являются системными цветами Windows и зависят от используемой цветовой схемы.

Значение	Цвет
clBackground	Текущий цвет фона окна
clActiveCaption	Текущий цвет заголовка активного окна
cllnactiveCaption	Текущий цвет заголовка неактивного окна
cIMenu	Текущий цвет фона меню
clWindow	Текущий цвет фона Windows
clWIndowFrame	Текущий цвет рамки окна
cIMenuText	Текущий цвет текста элемента меню
clWindowText	Текущий цвет текста внутри окна
cICaptionText	Текущий цвет заголовка активного окна
ciActiveBorder	Текущий цвет рамки активного окна
cllnactiveBorder	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
cIGrayText	Текущий цвет недоступного элемента меню
clBtnText	Текущий цвет текста кнопки

Помимо перечисленных в таблице цветов 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 DELPHY 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 следующие значения:

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 следующие параметры:

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 F1 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 пикселах. 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 Tab Stop.

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 Longlnt.

Свойство Top

Это свойство задает вертикальную координату левого верхнего угла интерфейсного элемента относительно формы в пикселах. 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 пикселах.

П2.2. Компоненты страницы STANDARD

П2.2.1. TMainMenu

Компонент TmainMenu служит для создания главного меню формы. После 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 опцию, перейти в окно инспектора объектов, в строке Caption набрать необходимое и нажать клавишу Enter. Для создания новых опций необходимо выбирать строку справа, для создания подопций – снизу. Для определения символа быстрого доступа к опции перед ним ставится символ “&”. Для вставки разделительной 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 следующими свойствами:

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: [nteger]: 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

П2.2.2. TPoupMenu

Данный компонент является локальным меню, которое становится 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 локально меню так же, как и главное, с помощью конструктора меню.

Чтобы связать нажатие правой кнопки мыши с раскрытием вспомогательного меню, в свойство PoupMenu необходимо поместить имя компонента-меню.

Свойство Alignment задает местонахождение локального меню.

П2.2.3. TLabel

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

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 AuroSize=False, WordWrap=True и установите подходящие размеры метки

П2.2.4. TEdit

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

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; ее UpperCase - 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:

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; BufSize: Integer) : Integer;	Копирует не более BufSize символов выделенного текста в буфер Buffer
procedure PasteFromClipboard;	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

П2.2.5. TMemo

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

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

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 WantReturns: 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 в текст символа табуляции, в противном случае – обрабатывается системой. Для ввода символа табуляции в этом случае следует нажать Ctrl-Tab

П2.2.6. TButton

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

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
TModalResulr = Low(Integer) .. 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 следующие стандартные значения ModalResult:

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

П2.2.7. 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.

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

П2.2.8. 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.

П2.2.9. 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.

Property Canvas: TCanvas;	Канва для программной прорисовки элементов
Property Columns: Longint;	Определяет количество колонок элементов в списке
Property ExtendedSelect: Boolean;	Если ExtendedSelect=True и MultiSelect=True, выбор элемента без одновременного нажатия Crti или Alt отменяет предыдущий выбор
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, IbOwnerDrawFixed, IbOwnerDrawVariable); **Property** Style: TListBoxStyle;	Определяет способ прорисовки элементов: IbStandard - элементы рисует Windows, IbOwnerDrawFixed - рисует программа, все элементы имеют одинаковую высоту, определяемую свойством ItemHeight, IbOwnerDrawVariable -рисует программа, элементы имеют разную высоту
Property Toplndex: Integer;	Индекс первого видимого в окне элемента

П2.2.10. 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 он 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 сыOwnerDrawFixed 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.

П2.2.11. TScrollBar

Компонент TScrollBar является полосой прокрутки и обычно он используется для визуального управления значением какой-либо величины.

TSrollBarKind = (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)

П2.2.12. 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.

П2.2.13. TRadioGroup

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

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

П2.2.14. TPanel

Панель используется в качестве контейнера для расположения других интерфейсных элементов.

Property BevelInner: TpanelBevel;	Определяет стиль внутренней кромки
Property BevelOuter: TpanelBevel;	Определяет стиль внешней кромки
TBevelWidth = 1..MaxInt; **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

П2.3. Компоненты страницы ADDITIONAL

П2.3.1. TBitBtn

Пиктографическая кнопка TBitBtn представляет собой разновидность стандартной кнопки TButton, которая помимо текста может содержать графическое изображение. Растровое изображение определяется с помощью свойства Clyph. В комплект поставки DELPHI (поддиректория Images/Buttons) входит около 160 различных вариантов растровых изображений для кнопок.

Кроме того, пользователь может самостоятельно создать растровое изображения с помощью встроенного в DELPHY графического редактора.

Свойство Kind позволяет выбрать одну из 11 стандартных разновидностей кнопки (рис.П2.1.)

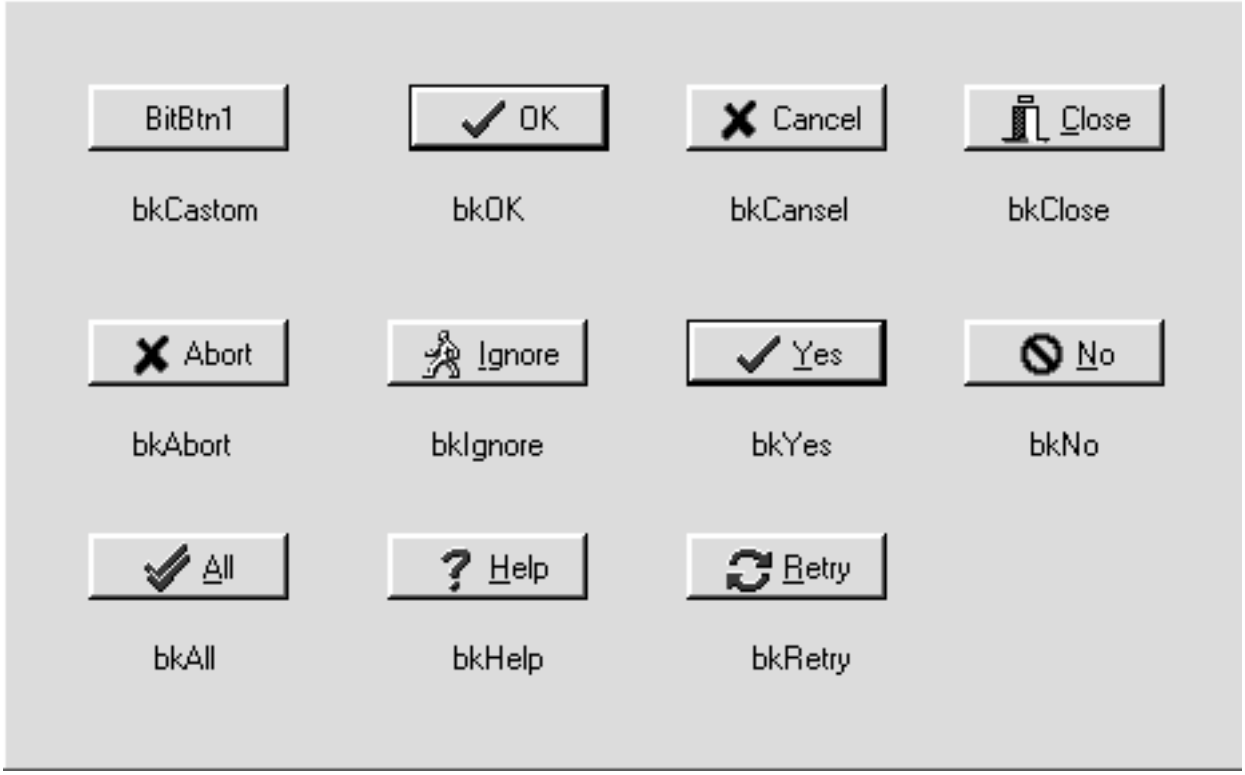

Рис. П2.1

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 HelpContext формы, на которую она помещена.

Property Glyph: TBitmap;	Определяет связанные с кнопкой растровые изображения (до 4)
TBitBtnKind = (bkCustom, bkOK, bkCancel, bkHelp, bkYes, bkNo, bkClose, bkAbort, bkRetry, bklgnore, bkAll); Property Kind: TBitBtnKind; .	Определяет разновидность кнопки
TButtonLayout = (blGlyphLeft, blGlyphRight, blGlyphTop, ,blGlyphBottom) ; 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

П2.3.2. 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.

П2.3.3. 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 «O» или «I», определяющий, записывается ли в Text результат наложения маски или исходный текст («O» - исходный текст). В третьей части указывается символ, который в окне редактора будет стоять в полях, предназначенных для ввода символов.

Описатели полей ввода представлены в следующей таблице:

Символ	Поле
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
O	Должно содержать цифру
9	Может содержать цифру
#	Может содержать цифру, «+», «-»

Специальные символы:

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

П2.3.4. TDrawGrid

Компонент TDrawGrid используется для отображения информации в виде таблицы. 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.

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[Index: 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)
Property Row: Longint;	Содержит номер ряда сфокусированной ячейки

Property RowCount: Longint;	Содержит количество рядов таблицы
Property RowHeights[Index: 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

Элементы множества TGridOptions имеют следующий смысл:

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
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

П2.3.5. *TStringGrid*

В отличие от компонента TStringGrid может отображать только текстовую информацию.

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

П2.3.6. TImage

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

П2.3.7. TShape

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

П2.3.8. TBevel

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

П2.3.9. TScrollBox

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

П2.3.10. TCheckListBox

Группирует независимые переключатели, позволяя обратиться к любому из них по индексу.

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 переключателях
Property State[Index: Integer]: TCheckBoxState;	Содержит состояние переключателя с индексом Index: cbUncheeked; cbChecked; cbGrayed

П2.3.11. TSplitter

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

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 alLefi или alRight, минимальная ширина компонента - слева и справа от TSplitter, если alTop или alBottom, минимальная высота компонента - выше или ниже него

П2.3.12. 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.

П2.3.13. TChart

Облегчает создание специальных полей для графического представления данных.

П2.4. Компоненты страницы DIALOGS

П2.4.1.Правила использования 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 модальным 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, настройки принтера и т.д.) для продолжения работы программы.

П2.4.2. TOpenDialog и TSaveDialog

Эти компоненты имеют идентичные свойства и различаются 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 отделения фильтров друг от друга, а также для отделения 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 и содержимое этого свойства.

Настройка диалога может варьироваться с помощью свойства

```
TOpenOption = (of Readonly, ofOverwritePrompt, ofHideReadOnly,
        ofNoChangeDir, ofShowHelp, ofNoValidate, ofAllowMuItiSelect,
        ofExtensionDifferent, ofPathMustExist, ofFileMustExist, ofCreatePrompt,
    ofShareAware, ofNoReadOnlyReturn,ofNoTestFileCreate,  ofNoNetworkButton,
    ofNoLongNames,  ofOldStyleDialog, ofNoDereferenceLinks);
TOpenOptions = set of TOpenOption;
property Options: TOpenOptions;
```

Значения этого свойства имеют следующий смысл:

ofReadOnly	Устанавливает переключатель “Только для чтения”
ofOverwritePrompt	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.
ofCreatePrompt	Требует подтверждения для создания несуществующего файла
ofShareAware	Разрешает выбирать файлы, используемые другими параллельно выполняемыми программами
ofNoReadOnlyRetur	Запрещает выбор файлов, имеющих атрибут “Только для чтения“
ofNoTestFileCreat	Запрещает проверку доступности сетевого или локального диска
ofNoNetworkButton	Запрещает вставку кнопки для создания сетевого диска
ofNoLongNames	Запрещает использование длинных имен файлов
ofOldStyleDialog	Создает диалог в стиле Windows 3.x

П2.4.3. TOpenPictureDialog и TSavePictureDialog

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

П2.4.4. 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. Значения этого свойства имеют следующий смысл:

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
fdNoSimulations	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
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“

П2.4.5. TColorDialog

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

П2.4.6. TPrintDialog

Компонент служит для создания стандартного диалогового окна для выбора параметров печати.

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: po PrintToFile -печатать в 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 в файл
property PrintRange: TPrintRange;	Определяет диапазон печатаемых страниц: prAll Pages - все страницы; prSelection -выделенный фрагмент текста; prPageNums -страницы по номерам
property PrintToFile: Boolean;	Содержит True, если пользователь выбрал печать в файл
property ToPage: Integer;	Определяет конечную страницу печати

П2.4.7. 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.

П2.4.8. TFindDialog

Стандартное диалоговое окно компонента TFindDialog используется для поиска фрагмента текста.

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 имеют такой смысл:

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

П2.4.9. TReplaceDialog

Компонент создает и обслуживает окно поиска и замены текстового фрагмента. Класс TReplaceDialog наследует большинство свойств классаTFind-Dialog. 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 3. ПРОСТЫЕ ТИПЫ ДАННЫХ ЯЗЫКА OBJECT PASCAL

П3.1.Целые типы

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

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... 2 147 483 647

К целочисленным типам применимы следующие процедуры и функции:

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 i - 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

П3.2.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:

Ord(False) == 0;
Ord(True) <> 0;

Succ(False) = True;
Pred(True) = False.

П3.3.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 О... 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 и возвращает его своим значением;

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

П3.4. 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 которому принадлежит переменная X;

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

П3.5. Вещественные типы

Значения вещественных типов определяют произвольное число лишь с некоторой конечной точностью, зависящей от внутреннего формата вещественного числа.

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:

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

ПЗ.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 4. ПРОЦЕДУРЫ И ФУНКЦИИ ДЛЯ РАБОТЫ СО СТРОКАМИ

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

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 ноль
Procedure SetLength(St: String; NewLength: Integer);	Устанавливает новую (меньшую) длину NewLength строки St, если NewLength больше текущей длины строки, обращение к SetLength игнорируется
Подпрограммы преобразования строк в другие типы т	
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 время, разделенные пробелом
Function StrToFloat(St: String): Extended;	Преобразует символы строки St в вещественное число. Строка не должна содержать ведущих или ведомых пробелов
Function StrToInt(St: String): Integer;	Преобразует символы строки St в целое число. Строка не должна содержать ведущих или ведомых пробелов

Function StrToIntDef(St: String; Default: Integer): Integer;	Преобразует символы строки St в целое число. Если строка не содержит правильного представления целого числа, возвращается значение Default
Function StrToIntRange(St: String; Min, Max: Longint) : Longint;	Преобразует символы строки St в целое число и возбуждает исключение ERangeError, если число выходит из заданного диапазона Mm 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: TFloatFor-mat; 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 IntToStr(Value: Integer) : String;	Преобразует целое значение Value в строку символов
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 показаны ниже:

Значение Format	Описание
fFfExponent	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 = π* 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 = π*1000 получим: 3 141,60р

ПРИЛОЖЕНИЕ 5. МАТЕМАТИЧЕСКИЕ ФОРМУЛЫ

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

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)
$ArcSin(x)$	$ArcTg\left(\sqrt{\frac{x}{1-x^2}}\right)$	ArcTan(Sqrt(x/(1-sqr(x))))
$ArcCos(x)$	$\frac{\pi}{2} - ArcSin(x)$	Pi/2- ArcTan(Sqrt(x/(1-sqr(x))))
$ArcCtg(x)$	$\frac{\pi}{2} - ArcTg(x)$	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
$Csc(x)$	$\frac{1}{\sin(x)}$	1/Sin(x)
$Sc(x)$	$\frac{1}{\cos(x)}$	1/Cos(x)