

Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра «Вычислительные методы и программирование»

ПРОГРАММИРОВАНИЕ АЛГОРИТМОВ В СРЕДЕ DELPHI

Лабораторный практикум по курсам
«Программирование» и «Основы алгоритмизации и программирование»
для студентов 1 – 2-го курсов всех специальностей БГУИР
В 2-х частях
часть 1

Под общей редакцией А. К. Сеницына
2-е издание, с изменениями

Минск 2004

УДК 519.6 (075.8)
ББК 22.193 я73
С 38

Синицын А.К.

С 38 Программирование алгоритмов в среде DELPHI: Лабораторный практикум по курсам «Программирование» и «Основы алгоритмизации и программирование» для студентов 1 – 2-го курсов всех специальностей БГУИР в 2-х частях часть 1 / А.В. Гуревич, В.Е. Карцев, С.В. Колосов, А.А. Лавренов, А.А. Навроцкий, А.К. Синицын, А.В. Щербаков; Под общ. ред. А.К. Синицына – Мн.: БГУИР, 2002. – 80 с.: ил.

ISBN 985-444-324-8

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

В практикум вошло 10 лабораторных работ.

УДК 519.6 (075.8)

ББК 22.193 я73

ISBN 985-444-324-8

© Коллектив авторов, 2002
© БГУИР, 2002

СОДЕРЖАНИЕ

ТЕМА 1. ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ АЛГОРИТМОВ	4
ТЕМА 2. ПРОГРАММИРОВАНИЕ РАЗВЕТВЛЯЮЩИХСЯ АЛГОРИТМОВ .	15
ТЕМА 3. ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ	21
ТЕМА 4. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ МАССИВОВ	29
ТЕМА 5. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ПОДПРОГРАММ И МОДУЛЕЙ	35
ТЕМА 6. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ МНОЖЕСТВ И СТРОК	40
ТЕМА 7. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ МЕХАНИЗМА ОБРАБОТКИ ИСКЛЮЧИТЕЛЬНЫХ СИТУАЦИЙ.	48
ТЕМА 8. ПРОГРАММИРОВАНИЕ С ОТОБРАЖЕНИЕМ ГРАФИЧЕСКОЙ ИНФОРМАЦИИ	56
ТЕМА 9. ИСПОЛЬЗОВАНИЕ ЗАПИСЕЙ ДЛЯ РАБОТЫ С КОМПЛЕКСНЫМИ ЧИСЛАМИ.	64
ТЕМА 10. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ФАЙЛОВ	69
ПРИЛОЖЕНИЕ 1. ПРОЦЕДУРЫ И ФУНКЦИИ ДЛЯ РАБОТЫ СО СТРОКАМИ	79
ПРИЛОЖЕНИЕ 2. МАТЕМАТИЧЕСКИЕ ФОРМУЛЫ	80
ЛИТЕРАТУРА	81

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.

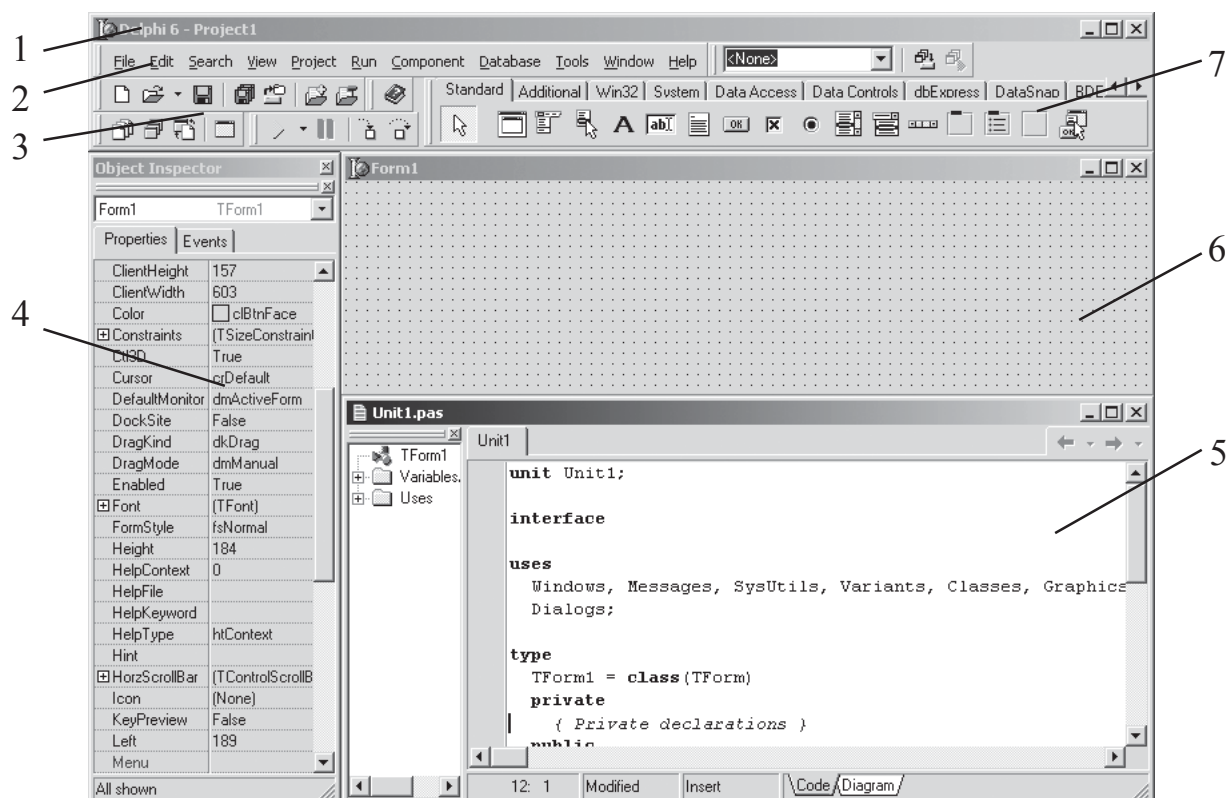


Рис.1.1.

- 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 содержит все необходимые средства для управления проектом. Пиктограммы облегчают доступ к наиболее часто применяемым командам основного меню. Через меню компонентов

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.

Модуль имеет следующую структуру:

```

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

```

В разделе объявлений описываются типы, переменные, заголовки процедур и функций, 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.

Приложение в среде DELPHI состоит из файлов с исходным текстом (расширение .pas), файлов форм (расширение .dfm) и файла проекта (расширение .dpr), который связывает вместе все файлы проекта. 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

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

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

1.3.1. Настройка формы.

Для создания нового проекта выберите в основном меню пункт File–New Application.

Пустая форма в правом верхнем углу имеет кнопки управления, которые предназначены: для свортывания 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.


При написании программы следует обратить внимание на то, что численные значения переменных x,y,z имеют действительный тип, поэтому для преобразования строковой записи числа, находящегося в переменной Edit1.Text, в действительное, надо использовать стандартную функцию $X:=StrToFloat(Edit1.Text)$.

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 целая и дробная часть действительного числа разделяется символом, заданным в разделе “Языки и стандарты” панели управления Windows (по умолчанию – 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. Прodelайте 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,ffixed,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 Standard 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).

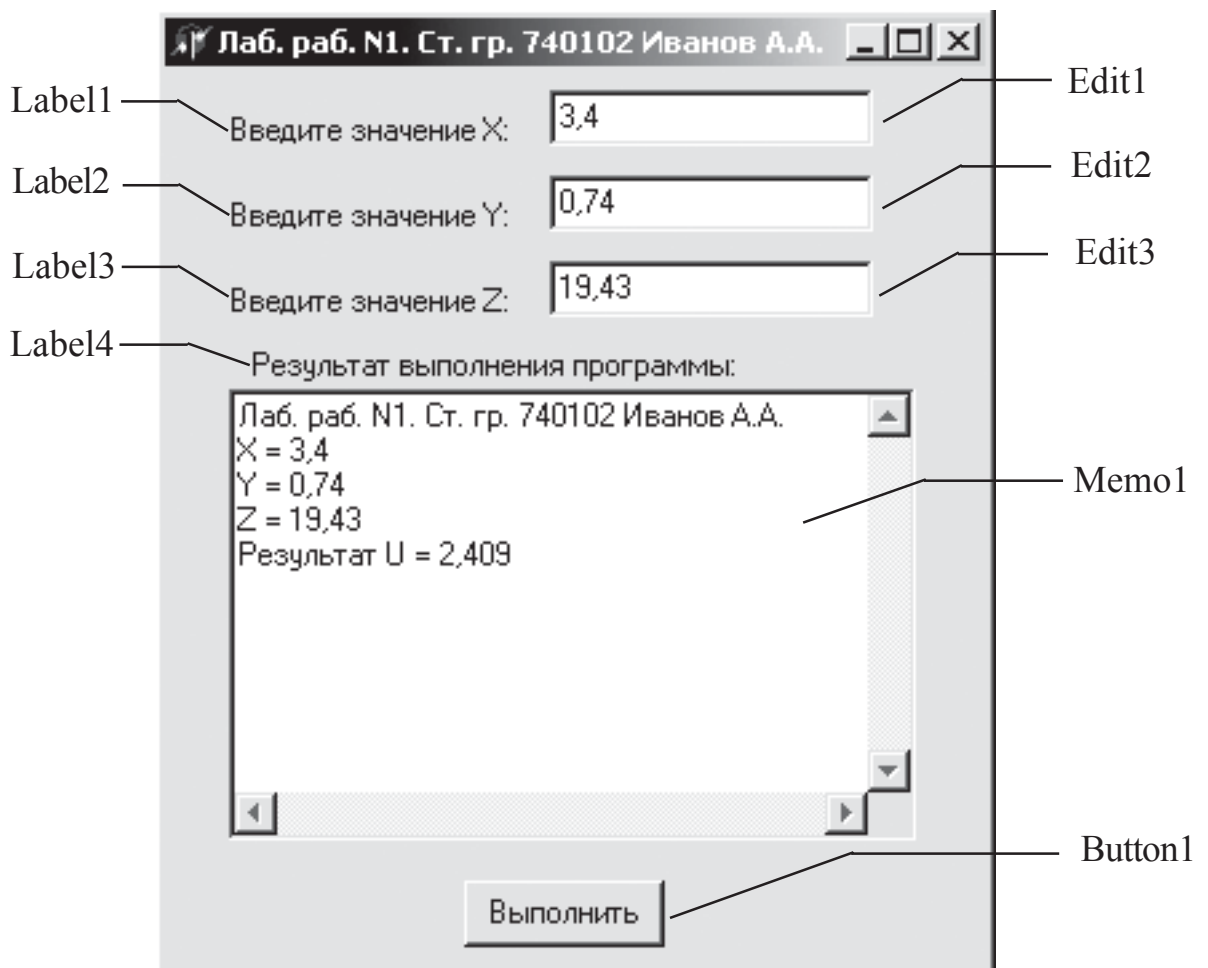



Рис. 1.2

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” - появятся новые результаты. Завершить работу программы можно или нажав кнопку  на форме или перейти в окно DELPHI и выбрать в главном меню пункт Run – ProgramReset. Последний способ выхода из программы обычно используют в случае ее заикливания.

Текст программы имеет вид:

```
unit Unit1;

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 Мемо1
// Вывод строки в многострочный редактор Мемо1
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 Мемо1
  y:=StrToFloat(Edit2.Text); // Считывается значение Y
  Memo1.Lines.Add(' Y = '+Edit2.Text); // Вывод Y в окно Мемо1
  z:=StrToFloat(Edit3.Text); // Считывается значение Z
  Memo1.Lines.Add(' Z = '+Edit3.Text); // Вывод Z в окно Мемо1
  // 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 Мемо1
  Memo1.Lines.Add(' Результат U = '+FloatToStrF(u,ffixed,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. Нарисуйте схему алгоритма разбив выражение на части. 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.

С помощью инспектора объектов измените цвет формы, шрифт выводимых символов.

Индивидуальные задания

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

При $x=14.26$, $y=-1.22$, $z=3.5 \times 10^{-2}$ $t=0.564849$.

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

При $x=-4.5$, $y=0.75 \times 10^{-4}$, $z=0.845 \times 10^2$ $u=-55.6848$.

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

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

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

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

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

При $x=-15.246$, $y=4.642 \times 10^{-2}$, $z=20.001 \times 10^2$ $\alpha=-182.036$.

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

При $x=16.55 \times 10^{-3}$, $y=-2.75$, $z=0.15$ $\beta=-40.63069$.

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

При $x=0.1722$, $y=6.33$, $z=3.25 \times 10^{-4}$ $\gamma=-205.305$.

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

При $x=-2.235 \times 10^{-2}$, $y=2.23$, $z=15.221$ $\varphi=39.374$.

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

При $x=1.825 \times 10^2$, $y=18.225$, $z=-3.298 \times 10^{-2}$ $\psi=1.2131$.

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

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

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

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

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

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

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

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

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

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

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

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

16. Определить количество грузовиков, необходимое для перевозки N ящиков, если каждый грузовик перевозит по M ящиков.

17. Определить время окончания рабочего дня (в часах и минутах), если известны время его начала (в часах и минутах) и продолжительность (вместе с обедом) (в часах и минутах).

18. Перевести белорусское время (в часах) в московское. (Учесть, что 23 часа по белорусскому времени – это 0 часов по московскому).

19. Вывести на экран 0, если заданное число четное, или 1, если оно нечетное.

20. Найти сумму цифр заданного четырехзначного числа.

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

22. Присвоить целой переменной k третью от конца цифру в записи

положительного целого числа n .

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

24. Целой переменной S присвоить сумму цифр трехзначного целого числа k .

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

26. Определить f – угол (в градусах) между положением часовой стрелки в начале суток и ее положением в h часов, m минут и s секунд ($0 \leq h \leq 11$, $0 \leq m$, $s \leq 59$).

27. 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 ($0 \leq f < 360$, f – вещественное число).

28. 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.

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

30. По номеру n ($n > 0$) некоторого года определить s – номер его столетия (учесть, что, к примеру, началом XX столетия был 1901, а не 1900 год!).

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:=y-x  
      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 $u=\operatorname{tg}(x)$. 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 - $\operatorname{tg}(x)$, 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.

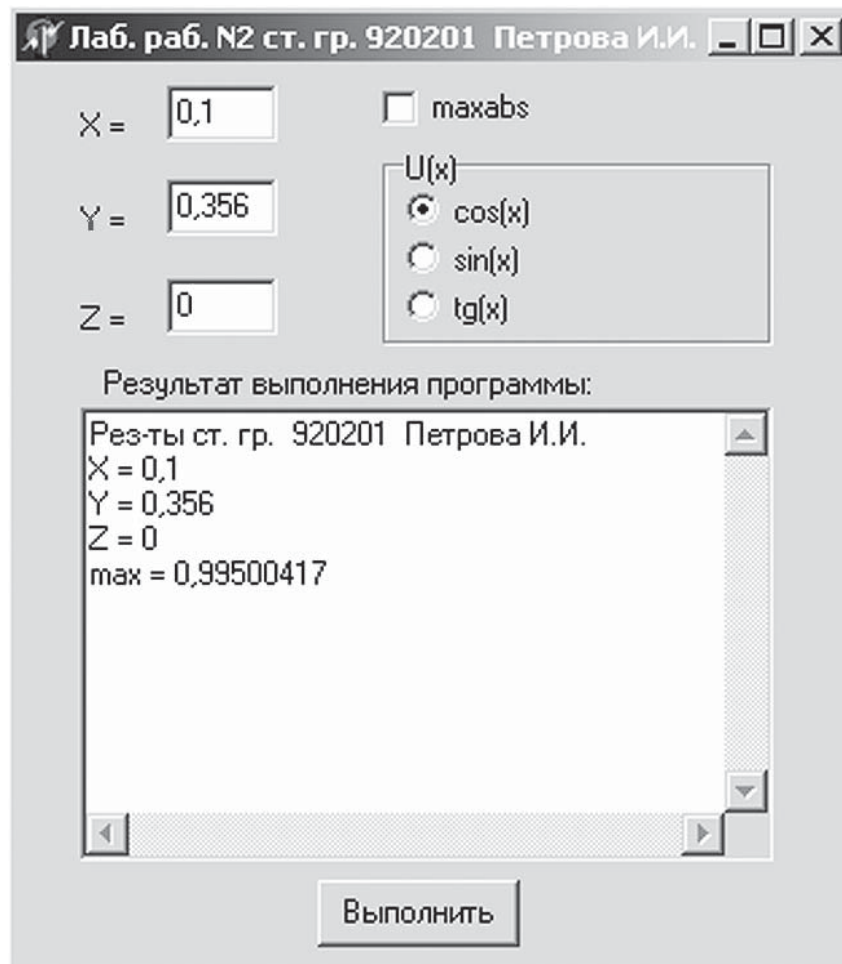


Рис. 2.1

Текст программы приведен ниже.

```
unit Unit2;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;
type
  TForm1 = class(TForm)
    CheckBox1: TCheckBox;
    RadioGroup1: TRadioGroup;
    Memo1: TMemo;
    Button1: TButton;
    Label1: TLabel;
    Label2: TLabel;
```

```

Label3: TLabel;
Label4: TLabel;
Edit1: TEdit;
Edit2: TEdit;
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
  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;
  if CheckBox1.Checked then // Проверка состояния кнопки CheckBox1
    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. Нарисуйте схему алгоритма. В качестве $f(x)$ использовать по выбору: $\sin(x)$, x^2 , e^x . Отредактируйте вид формы и текст программы, в соответствии с полученным заданием.

1.
$$a = \begin{cases} (f(x) + y)^2 - \sqrt{|f(y)x|}, & xy > 0 \\ (f(x) + y)^2 + \sqrt{|f(x)y|}, & xy < 0 \\ (f(x) + y)^2 + 1, & xy = 0. \end{cases}$$
2.
$$b = \begin{cases} \ln(|f(x)|) + (f(x)^2 + y)^3, & x/y > 0 \\ \ln|f(x)/y| + (f(x) + y)^3, & x/y < 0 \\ (f(x)^2 \cdot y)^3, & x = 0 \\ 0, & y = 0. \end{cases}$$
3.
$$c = \begin{cases} f(x)^2 + y^2 + \sin(y), & x - y = 0 \\ (f(x) - y)^2 + \cos(y), & x - y > 0 \\ (y - f(x))^2 + \operatorname{tg}(y), & x - y < 0. \end{cases}$$
4.
$$d = \begin{cases} (f(x) - y)^3 + \operatorname{arctg}(f(x)), & x > y \\ (y - f(x))^3 + \operatorname{arctg}(f(x)), & y > x \\ (y + f(x))^3 + 0.5, & y = x. \end{cases}$$
5.
$$e = \begin{cases} i\sqrt{f(x)}, & i - \text{нечетное}, \quad x > 0 \\ i/2\sqrt{|f(x)|}, & i - \text{четное}, \quad x < 0 \\ \sqrt{|if(x)|}, & \text{иначе.} \end{cases}$$
6.
$$g = \begin{cases} e^{f(x)-|b|}, & 0.5 < xb < 10 \\ \sqrt{|f(x) + b|}, & 0.1 < xb < 0.5 \\ 2f(x)^2, & \text{иначе.} \end{cases}$$
7.
$$s = \begin{cases} e^{f(x)}, & 1 < xb < 10 \\ \sqrt{|f(x) + 4 * b|}, & 12 < xb < 40 \\ bf(x)^2, & \text{иначе.} \end{cases}$$
8.
$$j = \begin{cases} \sin(5f(x) + 3m|f(x)|), & -1 < m < x \\ \cos(3f(x) + 5m|f(x)|), & x < m \\ (f(x) + m)^2, & x = m. \end{cases}$$
9.
$$l = \begin{cases} 2f(x)^3 + 3p^2, & x > |p| \\ |f(x) - p|, & 3 < x < |p| \\ (f(x) - p)^2, & x = |p|. \end{cases}$$
10.
$$k = \begin{cases} \ln(|f(x)| + |q|), & |xq| > 10 \\ e^{f(x)+q}, & |xq| < 10 \\ f(x) + q, & |xq| = 10 \end{cases}$$
11.
$$m = \frac{\max(f(x), y, z)}{\min(f(x), y)} + 5.$$
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 заданному номеру месяца определить сезон (весна, лето, осень или зима).
18. Определить, поместится ли прямоугольная коробка размером $a \times b$ в прямоугольный ящик размером $c \times d$ (учесть возможность поворота коробки).
19. Определить, является ли заданный год високосным.
20. 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.
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 фанты”, согласовав окончание слова “бутылка” с числом k .
25. Вывести на экран 1 или 0 в зависимости от того, имеют три заданных целых числа одинаковую четность или нет.
26. Вывести на экран 1 или 0 в зависимости от того, равна ли сумма двух первых цифр заданного четырехзначного числа сумме двух его последних цифр.
27. Вывести на экран 1 или 0 в зависимости от того, равен ли квадрат заданного трехзначного числа кубу суммы цифр этого числа.
28. Вывести на экран 1 или 0 в зависимости от того, есть ли среди первых трех цифр дробной части заданного положительного вещественного числа цифра ноль.
29. Вывести на экран 1 или 0 в зависимости от того, есть ли среди цифр заданного трехзначного числа одинаковые.
30. Значения переменных a, b и c поменять местами так, чтобы оказалось $a \leq b \leq c$.

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*, *while*, *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>=false. Заметим, что если <условие>=false при первом входе, то <операторы> не выполняются ни разу, в отличие от *repeat*, в котором хотя бы один раз они выполняются.

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 выполняются ни разу. Модификация оператора **for i:=i2 downto i1 do begin** <операторы> **end** организует повторения при убывающем изменении *i* на единицу.

Для прекращения выполнения цикла используется процедура **Break**, которая прерывает выполнение тела любого цикла и передает управление следующему за циклом оператору.

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 обратиться 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 появившемся диалоговом окне указать интересующую переменную (с помощью данного окна можно также изменить значение переменной во время выполнения программы). Нажимая клавишу F7 (пошаговое выполнение), можно построчно выполнять программу, контролируя изменение тех или иных переменных и правильность вычислений. Если курсор находится внутри цикла, то после нажатия F4 расчет останавливается после одного выполнения тела цикла. Для продолжения расчетов следует нажать <Run> меню Run или F9.

3.3. Порядок выполнения задания

Задание: написать и отладить программу, которая выводит таблицу значений функции $S(x)$ для x изменяющихся в интервале от X_1 до X_2 с шагом h .

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

Панель диалога представлена на рис.3.1.

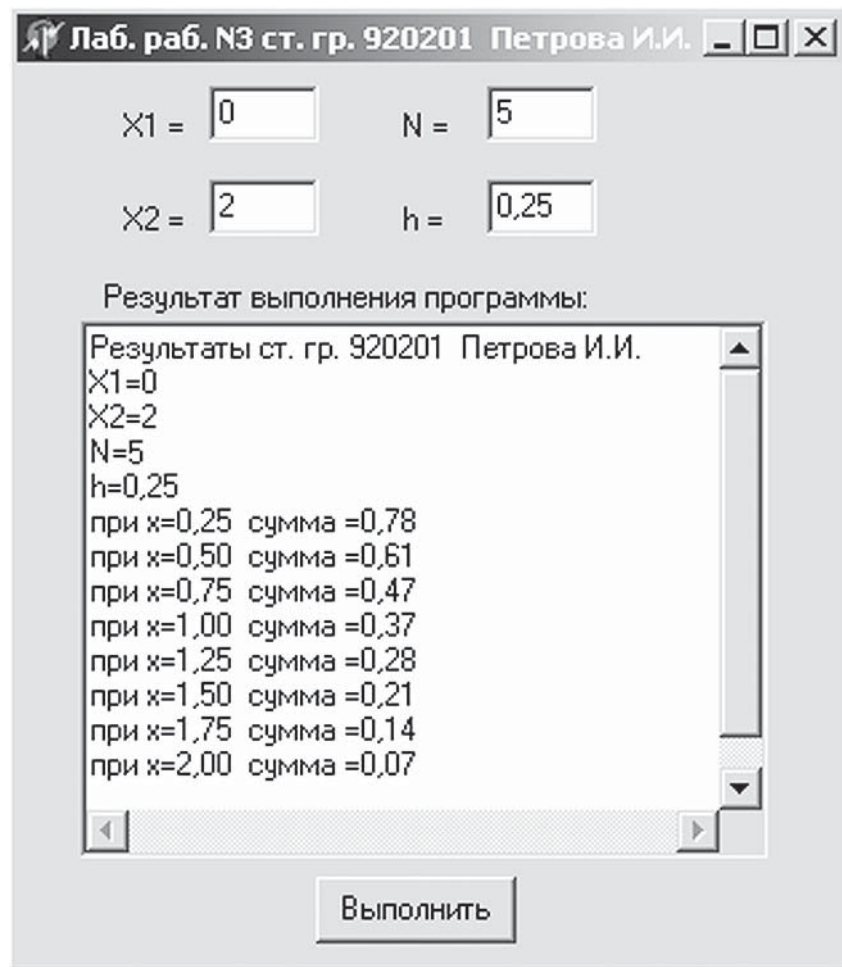


Рис. 3.1

Текст программы приведен ниже.

```

unit unit3;
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;
    Label5: TLabel;
    Edit1: TEdit;
  end;
  
```

```

    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(Edit1.Text);    Memo1.Lines.Add(' x1='+Edit1.Text);
    x2:=StrToFloat(Edit2.Text);    Memo1.Lines.Add(' x2='+Edit2.Text);
    N:=StrToInt(Edit3.Text);        Memo1.Lines.Add(' N='+Edit3.Text);
    h:=StrToFloat(Edit4.Text);      Memo1.Lines.Add(' h='+Edit4.Text);
    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 вариант задачи. Нарисуйте схему алгоритма. Спроектируйте 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 a до b с шагом $h=(b-a)/10$. Близость значений $S(x)$ и $Y(x)$ во всем диапазоне значений x указывает на правильность вычисления $S(x)$ и $Y(x)$.

Таблица 3.1

№	a	b	$S(x)$	n	$Y(x)$
1	2	3	4	5	6
1	0.1	1	$x - \frac{x^3}{3!} + \dots + (-1)^n \frac{x^{2n+1}}{(2n+1)!}$	160	$\sin x$
2	0.1	1	$1 + \frac{x^2}{2!} + \dots + \frac{x^{2n}}{(2n)!}$	100	$\frac{e^x + e^{-x}}{2}$
3	0.1	1	$1 + \frac{\cos \frac{\pi}{4}}{1!} x + \dots + \frac{\cos n \frac{\pi}{4}}{n!} x^n$	120	$e^{x \cos \frac{\pi}{4}} \cos(x \sin \frac{\pi}{4})$
4	0.1	1	$1 - \frac{x^2}{2!} + \dots + (-1)^n \frac{x^{2n}}{(2n)!}$	80	$\cos x$
5	0.1	1	$1 + 3x^2 + \dots + \frac{2n+1}{n!} x^{2n}$	140	$(1 + 2x^2)e^{x^2}$
6	0.1	1	$x + \frac{x^3}{3!} + \dots + \frac{x^{2n+1}}{(2n+1)!}$	80	$\frac{e^x - e^{-x}}{2}$
7	0.1	1	$\frac{x^3}{3} - \frac{x^5}{15} + \dots + (-1)^{n+1} \frac{x^{2n+1}}{4n^2 - 1}$	120	$\frac{1+x^2}{2} \operatorname{arctg} x - \frac{x}{2}$
8	0.1	1	$1 + \frac{2x}{1!} + \dots + \frac{(2x)^n}{n!}$	100	e^{2x}
9	0.1	1	$1 + 2\frac{x}{2} + \dots + \frac{n^2+1}{n!} \left(\frac{x}{2}\right)^n$	140	$\left(\frac{x^2}{4} + \frac{x}{2} + 1\right) e^{\frac{x}{2}}$

1	2	3	4	5	6
10	0.1	0.5	$x - \frac{x^3}{3} + \dots + (-1)^n \frac{x^{2n+1}}{2n+1}$	150	$\arctg x$
11	0.1	1	$1 - \frac{3}{2}x^2 + \dots + (-1)^n \frac{2n^2+1}{(2n)!} x^{2n}$	100	$\left(1 - \frac{x^2}{2}\right) \cos x - \frac{x}{2} \sin x$
12	0.1	1	$-\frac{(2x)^2}{2} + \frac{(2x)^4}{24} - \dots + (-1)^n \frac{(2x)^{2n}}{(2n)!}$	80	$2(\cos^2 x - 1)$
13	-2	-0.1	$-(1+x)^2 + \frac{(1+x)^4}{2} + \dots + (-1)^n \frac{(1+x)^{2n}}{n}$	160	$\ln \frac{1}{2+2x+x^2}$
14	0.2	0.8	$\frac{x}{3!} + \frac{4x^2}{5!} + \dots + \frac{n^2}{(2n+1)!} x^n$	120	$\frac{1}{4} \left(\frac{x+1}{\sqrt{x}} \operatorname{sh} \sqrt{x} - \operatorname{ch} \sqrt{x} \right)$
15	0.1	0.8	$\frac{x^2}{2} - \frac{x^4}{12} + \dots + (-1)^{n+1} \frac{x^{2n}}{2n(2n-1)}$	180	$x \arctg x - \ln \sqrt{1+x^2}$

В заданиях с № 16 по 30 (табл. 3.2.) необходимо вывести на экран таблицу значений функции $Y(x)$ и ее разложения в ряд $S(x)$ с точностью ε . Близость значений $S(x)$ и $Y(x)$ во всем диапазоне значений x указывает на правильность вычисления $S(x)$ и $Y(x)$. Вывести число итераций, необходимое, для достижения заданной точности.

Таблица 3.2

№	a	b	S(x)	ε	Y(x)
1	2	3	4	5	6
16	-0,9	0,9	$x + \frac{x^3}{3} + \dots + \frac{x^{2k+1}}{2k+1}$	10^{-4}	$\frac{1}{2} \ln \frac{1+x}{1-x}$
17	0,1	0,9	$1 - \frac{x^2}{3!} + \dots + (-1)^k \frac{x^{2k}}{(2k+1)!}$	10^{-5}	$\frac{\sin x}{x}$
18	-0,9	0,9	$1 + \frac{x}{3} + \sum_{k=2}^{\infty} (-1)^{k-1} \frac{1 \cdot 2 \cdot 5 \cdot 8 \cdot \dots \cdot (3k-4)}{3^k k!} x^k$	10^{-3}	$\sqrt[3]{1+x}$
19	-3	3	$1 + \frac{\ln 9}{1} x + \dots + \frac{(\ln 9)^k}{k!} x^k$	10^{-4}	9^x
20	-1	1	$\frac{x^3}{2 \cdot 3} + \frac{1 \cdot 3 \cdot x^5}{2 \cdot 4 \cdot 5} + \dots + \frac{(2k-1)!!}{(2k)!!} \cdot \frac{x^{2k+1}}{(2k+1)}$	10^{-3}	$-x + \arcsin x$
21	-0,9	0,9	$-\frac{x^2}{2 \cdot 4} + \frac{3 \cdot x^3}{2 \cdot 4 \cdot 6} + \dots + (-1)^{k-1} \frac{(2k-3)!!}{(2k)!!} x^k$	10^{-3}	$\sqrt{1+x} - 1 - \frac{x}{2}$

Окончание табл. 3.2

1	2	3	4	5	6
22	-0,5	0,5	$\frac{x^3}{3} + \frac{x^7}{7} + \dots + \frac{x^{4k-1}}{4k-1}$	10^{-5}	$\frac{1}{4} \ln \frac{1+x}{1-x} - \frac{1}{2} \arctg x$
23	-0,3	0,4	$\frac{1}{1+x} + \frac{2x}{1+x^2} + \dots + \frac{2^{k-1} x^{(2^{k-1}-1)}}{1+x^{(2^{k-1})}}$	10^{-4}	$\frac{1}{1-x}$
24	-2	2	$\frac{\cos x}{1} + \frac{\cos 3x}{9} + \dots + \frac{\cos(2k-1)x}{(2k-1)^2}$	10^{-4}	$\frac{\pi(\pi - 2 x)}{8}$
25	-0,5	0,5	$\sum_{k=1}^{\infty} \left(\frac{(-1)^{k+1}}{k} + \frac{(-1)^k \cdot 6}{k^3 \pi^2} \right) \sin k\pi x$	10^{-3}	$\frac{\pi x^3}{2}$
26	-0,85	0,95	$\sum_{k=2}^{\infty} \frac{(-1)^{k-1} (4k-5)!!}{(4k)!!} x^k$	10^{-4}	$\sqrt[4]{x+1} - \frac{4-x}{4}$
27	1	2,5	$\sum_{k=1}^{\infty} \frac{\cos kx}{k^2}$	10^{-5}	$\frac{3x^2 - 6\pi x + 2\pi^2}{12}$
28	-1,5	1,5	$\sum_{k=1}^{\infty} (-1)^{k-1} \frac{\cos kx}{k}$	10^{-4}	$\ln(2 \cos \frac{x}{2})$
29	-1	1,3	$\sum_{k=2}^{\infty} (-1)^k \frac{\cos kx}{k^2 - 1}$	10^{-5}	$\frac{2x \sin x - 2 + \cos x}{4}$
30	-2,5	1,3	$\sum_{k=1}^{\infty} \frac{\sin(2k-1)x}{2k-1}$	10^{-4}	$\frac{\pi \cdot \operatorname{sign} x}{4}$

31. Подсчитать k - количество цифр в десятичной записи целого неотрицательного числа n.

32. Переменной t присвоить значение 1 или 0 в зависимости от того, является ли натуральное число k степенью 3.

33. Дано n вещественных чисел. Вычислить разность между максимальным и минимальным из них.

34. Дана непустая последовательность различных натуральных чисел, за которой следует 0. Определить порядковый номер наименьшего из них.

35. 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.

36. Дано n вещественных чисел. Определить, образуют ли они возрастающую последовательность.

37. Дана последовательность из n целых чисел. Определить, со скольких отрицательных чисел она начинается.

38. Определить k – количество трехзначных натуральных чисел, сумма цифр

которых равна n ($1 \leq n \leq 27$). Операции деления ($/$, div и mod) не использовать.

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

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

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

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

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

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

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

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

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 следующей конструкции:

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, где A - квадратная матрица размерностью $N \times N$, а Y, B – одномерные массивы размерностью N . Элементы вектора Y определяются по формуле. 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.

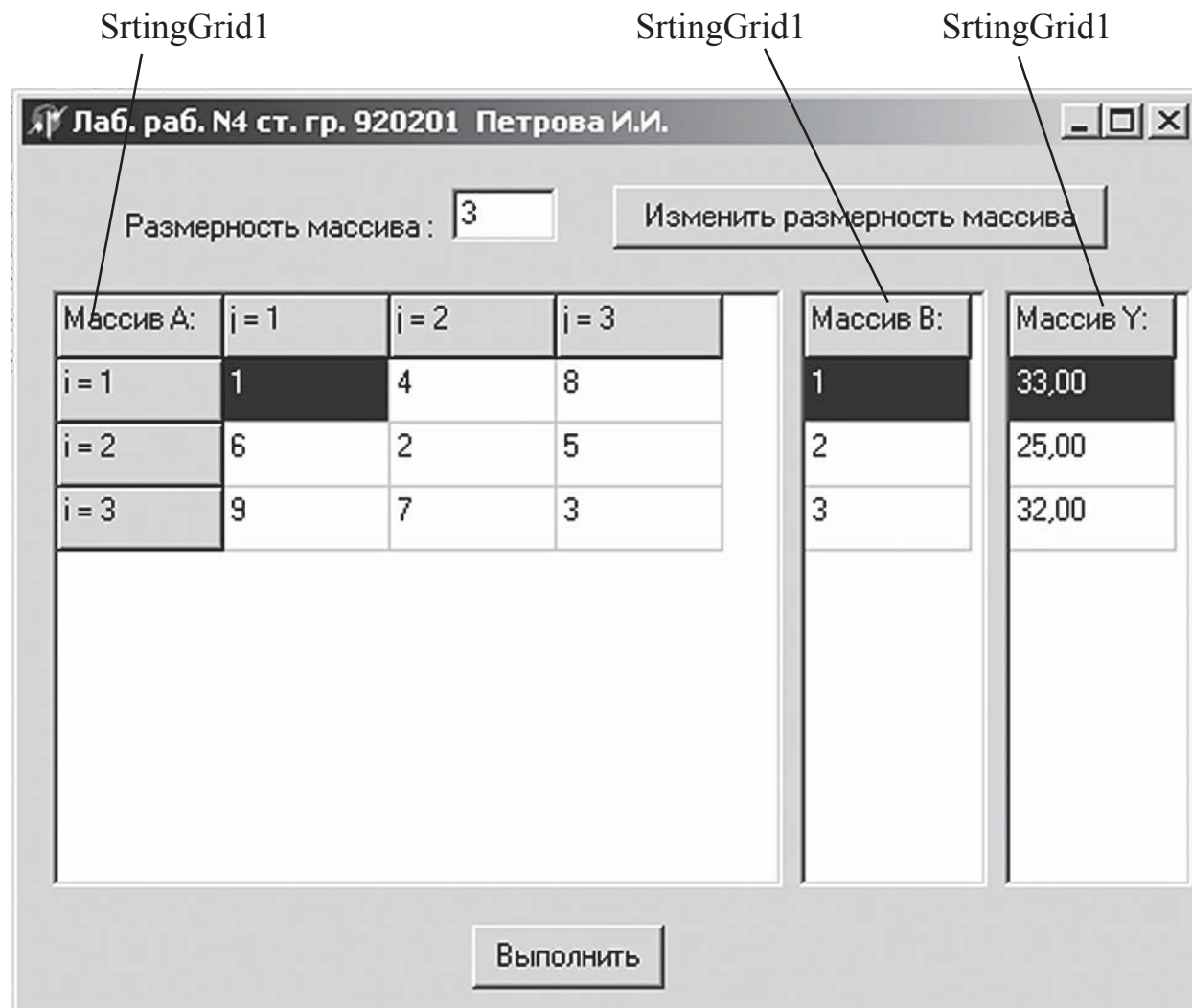


Рис. 4.1

4.3.1. Настройка компонента TStringGrid

Для установки компонента TStringGrid на форму необходимо на странице 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 для компонентов StringGrid1 и StringGrid2 необходимо в инспекторе объектов раскрыть раздел Options (нажав на знак “+”, стоящий слева от Options) и свойство goEditing установить в положение True.

Текст программы приведен ниже.

```

unit unit4;
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],ffixed,6,2);
  end;
end;
end.

```

4.4. Индивидуальные задания

Во всех заданиях по теме «Массивы» переменные вводить и выводить с помощью компонента TEdit, массивы – с помощью компонента TStringGrid, в котором 0-й столбец и 0-ю строку использовать для отображения индексов массивов. Вычисления выполнять, после нажатия кнопки типа TButton.

1. Задана матрица размером $N \times M$. Получить массив B , присвоив его k -му элементу значение 0, если все элементы k -го столбца матрицы нулевые, и значение 1 в противном случае.
2. Задана матрица размером $N \times M$. Получить массив B , присвоив его k -му элементу значение 1, если элементы k -й строки матрицы упорядочены по убыванию, и значение 0 в противном случае.
3. Задана матрица размером $N \times M$. Получить массив B , присвоив его k -му элементу значение 1, если k -я строка матрицы симметрична, и значение 0 в противном случае.
4. Задана матрица размером $N \times M$. Определить k – количество “особых” элементов матрицы, считая элемент “особым”, если он больше суммы остальных элементов своего столбца.
5. Задана матрица размером $N \times M$. Определить k – количество “особых” элементов матрицы, считая элемент “особым”, если в его строке слева от него находятся элементы, меньшие его, а справа – большие.
6. Задана символьная матрица размером $N \times M$. Определить k - количество различных элементов матрицы (т.е. повторяющиеся элементы считать один раз).
7. Дана матрица размером $N \times M$. Упорядочить ее строки по возрастанию их первых элементов.
8. Дана матрица размером $N \times M$. Упорядочить ее строки по возрастанию суммы их элементов.
9. Дана матрица размером $N \times M$. Упорядочить ее строки по возрастанию их наибольших элементов.
10. Определить, является ли заданная квадратная матрица n -го порядка симметричной относительно побочной диагонали.
11. Для матрицы размером $N \times M$ вывести на экран все ее седловые точки. Элемент матрицы называется седловой точкой, если он является наименьшим в своей строке и одновременно наибольшим в своем столбце или, наоборот.
12. В матрице n -го порядка переставить строки так, чтобы на главной диагонали

матрицы были расположены элементы, наибольшие по абсолютной величине.

13. В матрице n -го порядка найти максимальный среди элементов, лежащих ниже побочной диагонали, и минимальный среди элементов, лежащих выше главной диагонали.

14. В матрице размером $N \times M$ поменять местами строку, содержащую элемент с наибольшим значением со строкой, содержащей элемент с наименьшим 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 возрастанию.

20. Элементы массива X расположить в обратном порядке.

21. Элементы массива X циклически сдвинуть на k позиций влево.

22. Элементы массива X циклически сдвинуть на n позиций вправо.

23. Преобразовать массив X по следующему правилу: все отрицательные элементы массива перенести в начало, а все остальные – в конец, сохраняя исходное взаимное расположение, как среди отрицательных, так и среди остальных элементов.

24. Элементы каждого из массивов X и Y упорядочены по возрастанию. Объединить элементы этих двух массивов в один массив Z так, чтобы они снова оказались упорядоченными по возрастанию.

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 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.

5.1. Использование подпрограмм

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

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 формальн. 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.

5.2. Использование модулей

Модуль – автономно компилируемая программная единица, включающая в себя процедуры, функции, а также различные разделы описаний. Структура модуля представлена в п.1.2 и содержит следующие основные части: заголовок, интерфейсная часть, исполняемая, иницирующая и завершающая (последние две части могут отсутствовать).

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 ключевыми словами **interface** и **implementation** и содержит объявление тех конструкций и разделов описаний модуля, которые должны быть доступны другим программам.

Исполняемая часть начинается ключевым словом **implementation** и содержит описание процедур и функций, объявленных в интерфейсной части. Она может также содержать разделы описаний вспомогательных типов, констант, переменных, процедур и функций, которые будут использоваться только в исполняемой части и не будут доступны внешним программам.

Иницирующая часть начинается ключевым словом **initialization** и содержит операторы, которые исполняются перед началом выполнения основной программы.

Завершающая часть начинается ключевым словом **finalization** и выполняется в момент окончания работы программы.

5.3. 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 $\sin^2(x)$.

5.3.1. Создание модуля

Для создания нового модуля необходимо в меню **File** выбрать **New – Unit**. В результате будет создан файл с заголовком **Unit Unit2**. 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.

5.3.2. 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. 5.1).

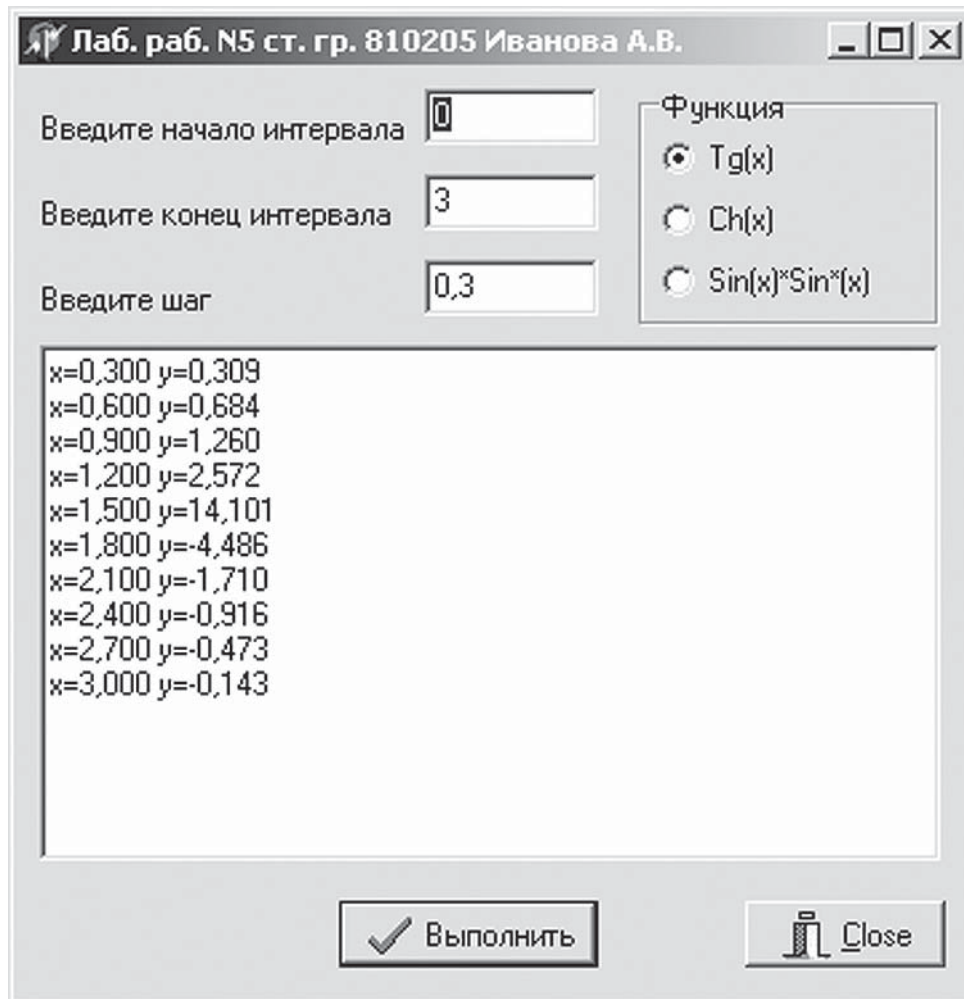


Рис. 5.1

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,ffixed,8,3)+
      ' y='+FloatToStrf(y,ffixed,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 значение 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.
```

5.4. 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 6. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ МНОЖЕСТВ И СТРОК

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

6.1 Тип множество и работа с ним

В математике под множеством понимается неупорядоченный набор различных однотипных элементов, определены также операции над множествами.

Для работы с множествами в Паскале введен специальный тип переменных :

type <имя типа>=set of <базовый тип>;

var A,B,C: имя типа;

здесь <базовый тип> - любой порядковый тип кроме Word, Integer, Longint, т.е. (перечисляемый, интервальный, char, byte, boolean).

6.1.1 Операции над множествами

Над переменными типа множеств допустимы операции присваивания сложения, вычитания, умножения. Эти операции дополняют две процедуры

Include (S, i); - добавление в множество *S* элемента *i*;

Exclude (S, i); - исключение из множества *S* элемента *i*.

Элемент *i* должен быть базового типа.

Эти операции выполняются значительно быстрее, чем их эквивалентные $s:=s+[i]; \quad s:=s-[i];$

Операции проверки условия

Результат $c=d$	будет true если множества одинаковы;
Результат $c<>d$	будет true если множества не одинаковы;
Результат $c>=d$	будет true если все элементы <i>d</i> принадлежат <i>c</i> ;
Результат $c<=d$	будет true если все элементы <i>c</i> принадлежат <i>d</i> ;
Результат $i \text{ in } c$	будет true если элемент <i>i</i> принадлежит <i>c</i> .

6.1.2 Примеры работы с множествами

1. Ввод *n* элементов множества:

```
.....
Var  A:set of char;
      s:char;
      n:Word;
begin
  A:=[];  // Очистка множества
  for i:=1 to n do begin
```

```

    read(s);
    A:=A+[s];
    end;

```

```

end.

```

2. Распечатать содержимое множества:

```

.....
Var  B:set of 1..100;
      k:byte;
begin
    for k:=1 to 100 do
        if k in B then Write(k);
    .....
end.

```

3. Сокращение проверок в операторе **if**:

Оператор **if (k=5) or (k=1) or (k=8) then ...** можно записать более изящно:
if k in [5,1,8] then ...

6.1. Типы данных для работы со строками

6.1.1. 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.

6.1.2. 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.

6.1.3. 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.

6.1.4. 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).

6.1.5. 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.

6.2. 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.

6.3. 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.

6.4. Компонент TBitBtn

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

6.5. 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 послыки соответствующих сообщений. Среда DELPHI позволяет принимать и обрабатывать большинство таких сообщений. Каждый компонент содержит обработчики сообщений на странице Events инспектора объектов.

Для создания обработчика события необходимо раскрыть список компонентов

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

Каждый компонент имеет свой набор обработчиков событий, однако некоторые из них присущи большинству компонентов. Наиболее часто применяемые события представлены в табл. 6.1.

Таблица 6.1

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

6.6. 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 Unit6;
interface
```

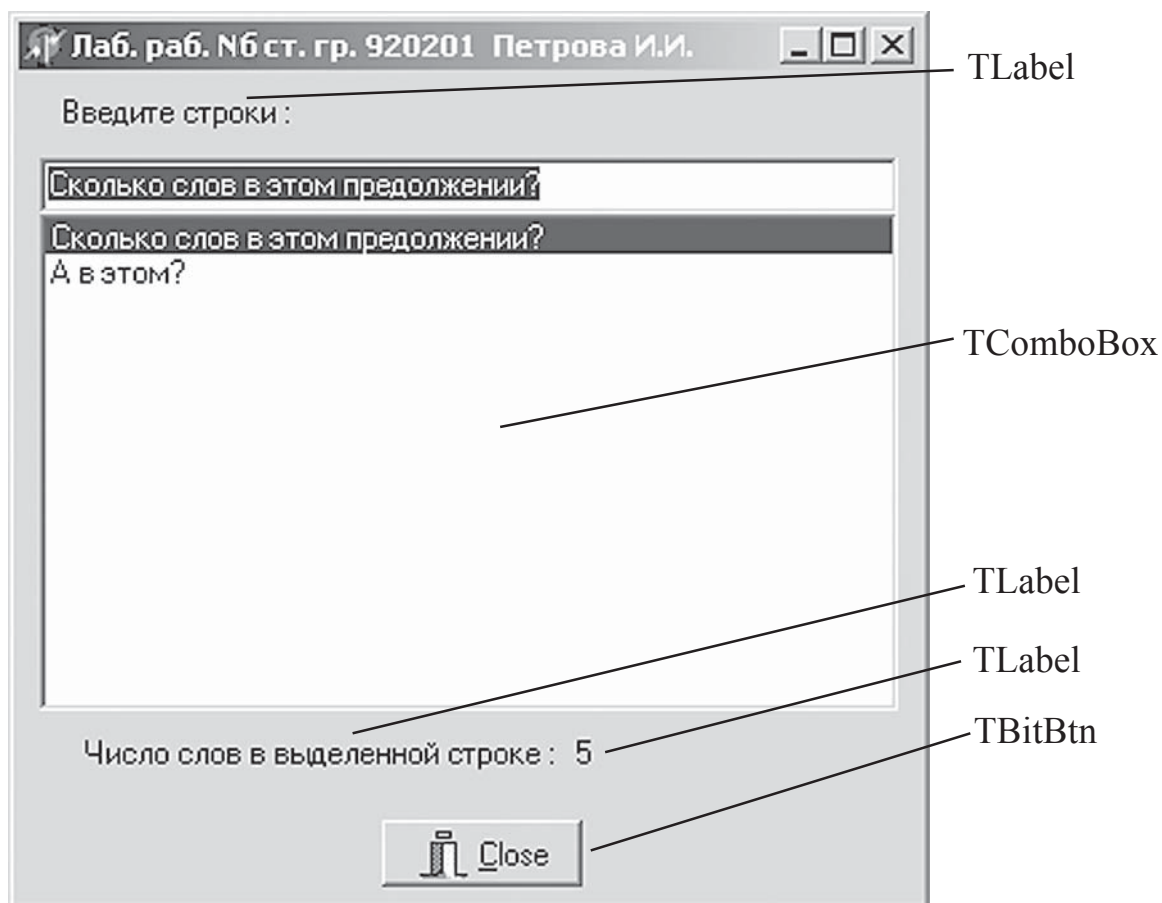



Рис. 6.1

```

uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
    Forms, Dialogs, StdCtrls, ExtCtrls, Grids, Buttons;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    ComboBox1: TComboBox;
    BitBtn1: TBitBtn;
    Label2: TLabel;
    Label3: 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 клавиши
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: integer;
begin
    nst:=ComboBox1.ItemIndex;    // Определение номера выбранной строки
    st:=ComboBox1.Items[nst]; //Занесение выбранной строки в переменную st
    if st[1]<>' ' then n:=1    // Если строка начинается со слова, то n=1
    else n:=0;    // иначе n=0
    for i:=1 to length(st)-1 do // Перебор всех символов в строке
    if (st[i]=' ') and (st[i+1]<>' ') then Inc(n); // Если i-тый символ - пробел,
        // а i+1- не пробел , то увеличиваем число слов на единицу
    Label3.Caption:=IntToStr(n);    // Вывод числа слов в Label3
end;
end.

```

6.7. Индивидуальные задания

Во всех заданиях исходные данные вводить 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 чисел, разделенных пробелами. Вывести на экран числа этой строки в порядке возрастания их значений.

10. Дана строка символов, состоящая из произвольных десятичных чисел, 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, в которой могут встречаться цифры, пробелы, буква “Е” и знаки “+”, “-“. 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. Дана строка символов, в которой могут встречаться цифры, пробелы, буква “Е” и знаки “+”, “-“. Определить количество действительных чисел в строке.

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 7. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ МЕХАНИЗМА ОБРАБОТКИ ИСКЛЮЧИТЕЛЬНЫХ СИТУАЦИЙ.

Цель лабораторной работы: изучить средства обработки исключительных ситуаций. Написать программу с использованием механизма обработки исключительных ситуаций.

7.1. Обработка ошибок и исключительных ситуаций

Под исключительной ситуацией понимается некое ошибочное состояние, возникающее при выполнении программы и требующее выполнения определённых действий для продолжения работы или корректного ее завершения. Стандартный обработчик, вызываемый по умолчанию, информирует пользователя о возникновении ошибки и завершает выполнение программы. Для защиты от завершения можно использовать такую структуру, которая перехватывает исключительную ситуацию и дает возможность разработчику предусмотреть определенные действия при ее возникновении.

Среда Delphi имеет две конструкции для обработки исключительных ситуаций.

try		try
...		...
except		finally
on <тип искл.. ситуации1> do <оператор>;	и	<операторы>;
on <тип искл.. ситуации2> do <оператор>;		end;
...		
else		
<операторы>;		
end;		

Ключевое слово **try** предназначено для указания на начало блока обработки исключительных ситуаций. В этот блок помещают операторы, выполнение которых может привести к возникновению исключительной ситуации.

Если исключительная ситуация не возникает, то сначала выполняются операторы, расположенные внутри блока **try ... finally**, а затем – внутри блока **finally ... end**. При возникновении исключительной ситуации выполнение сразу передается в блок **finally**, который обычно служит для корректного завершения программы, т.е. закрытия файлов, очистки динамически выделенной памяти и т.д.

Когда возникает необходимость в самостоятельной обработке исключительной

ситуации, то применяется структура **tryexcept**. Если исключительная ситуация не возникает, то выполняются только операторы, расположенные внутри блока **tryexcept**, операторы, расположенные внутри блока **except ... end** не выполняются. При возникновении исключительной ситуации выполнение сразу передается в блок **except**, который содержит список директив, определяющих реакцию приложения на ту или иную исключительную ситуацию. Если программа не находит директивы, соответствующей возникшей исключительной ситуации, то выполняются операторы, стоящие после ключевого слова **else**.

Некоторые, наиболее распространенные типы исключительных ситуаций представлены в табл. 7.1.

Таблица 7.1.

Тип исключительной ситуации	Причина
Eabort	Намеренное прерывания программы
EArrayError	Ошибка при операциях с массивами: использование ошибочного индекса массива, добавление слишком большого числа элементов в массив фиксированной длины
EConvertError	Ошибка преобразования строки в другие типы данных
EDivByZero	Попытка целочисленного деления на ноль
ERangeError	Целочисленное значение или индекс вне допустимого диапазона (при включенной директиве проверки границ массива {\$R+})
EIntOverflow	Переполнение при операции с целыми числами (при включенной директиве {\$Q+ })
EZeroDivide	Деление на ноль числа с плавающей точкой
EOutOfMemory	Недостаточно памяти
EFileNotFound	Файл не найден
EInvalidFileName	Неверное имя файла
EInvalidOp	Неправильная операция с плавающей точкой
EOverflow	Переполнение при выполнении операции с плавающей точкой
EAssertionFailed	Возникает при намеренной генерации исключительной ситуации с помощью процедуры Assert (при включенной директиве {\$C+ })

Следует отметить, что среда Delphi сама отслеживает и обрабатывает возникающие исключительные ситуации. Для отладки программы, содержащей обработку исключительных ситуаций, надо отключить опцию **Stop on Delphi Exceptions** находящуюся в **Tools – Debugger Options ...**, закладка **Language Exceptions**.

Возникновение исключительной ситуации может быть инициировано преднамеренно. Для этого можно использовать процедуру **Abort** (класс **EAbort**). В отличие от **Break** и **Exit** она позволяет осуществить выход из глубоко вложенных циклов и процедур. Можно также использовать процедуру **Assert(expr : Boolean [; const msg: string])**, у которой первый параметр представляет собой логическое выражение, а второй - содержит текст сообщения.

Если значение первого параметра становится **false**, то происходит генерация исключительной ситуации **EAssertionFailed** и появляется диалоговое окно с соответствующим сообщением. Искусственно генерировать исключительную ситуацию можно также с использованием ключевого слова **raise**:

Raise <тип исключения>.Create(<текст сообщения>);

7.2. Системы счисления

Под позиционной системой счисления понимают способ записи чисел с помощью цифр, при котором значение цифры определяется ее порядком в записи числа. Число R в p -ичной системе счисления можно представить в развернутом виде

$$R = a_n a_{n-1} \dots a_1 a_0, a_{-1} \dots a_{-k} = a_n p^n + a_{n-1} p^{n-1} + \dots + a_1 p^1 + a_0 p^0 + a_{-1} p^{-1} + \dots + a_{-k} p^{-k},$$

где a_i – цифры, p – основание системы счисления. Количество цифр равно p . Для записи цифр в общем случае может быть использован любой набор p символов. Обычно при $p \leq 10$ используются символы $0 \div 9$, для $p \geq 10$ добавляются буквы латинского алфавита A, B, C, D, E, F которые в десятичной системе представляют числами 10, 11, 12, 13, 14, 15. Например $R = (2A, B)_{16} = 2 \cdot 16^1 + A \cdot 16^0 + B \cdot 16^{-1} = 2 \cdot 16 + 10 \cdot 1 + 11/16 = (42,6875)_{10}$

В компьютерной технике обычно используются системы с основанием равным степени двойки: двоичная, восьмеричная и шестнадцатеричная. Имеются процессоры, реализующие троичную систему счисления. Для удобства пользователей ввод – вывод и операции над числами в компьютере производят в десятичной системе счисления.

При переводе числа из десятичной в другую систему счисления, целая и дробная часть числа переводятся различным образом.

При переводе целой части, она делится на основание новой системы счисления, остаток представляет очередную цифру a_0, a_1, \dots, a_n , а частное снова делится на основание. Процесс повторяется до тех пор, пока частное не станет равным нулю. Заметим, что цифры получаются в порядке, обратном порядку их следования в записи числа.

При переводе дробной части она умножается на основание системы счисления. Целая часть полученного числа представляет очередную цифру $a_{-1}, a_{-2}, \dots, a_{-k}$, а дробная часть опять умножается на основание системы. Расчеты ведут до получения требуемого количества цифр.

7.3. Порядок выполнения индивидуального задания

Задание Написать обучающую программу, позволяющую освоить операции сложения, вычитания, умножения, целочисленного деления и нахождения остатка от целочисленного деления для любых двух целых чисел, записанных в семнадцатеричной системе счисления. Для обозначения чисел использовать цифры от 0 до 9 и буквы от a до g (прописные или строчные).

Панель диалога будет иметь вид (рис. 7.1).

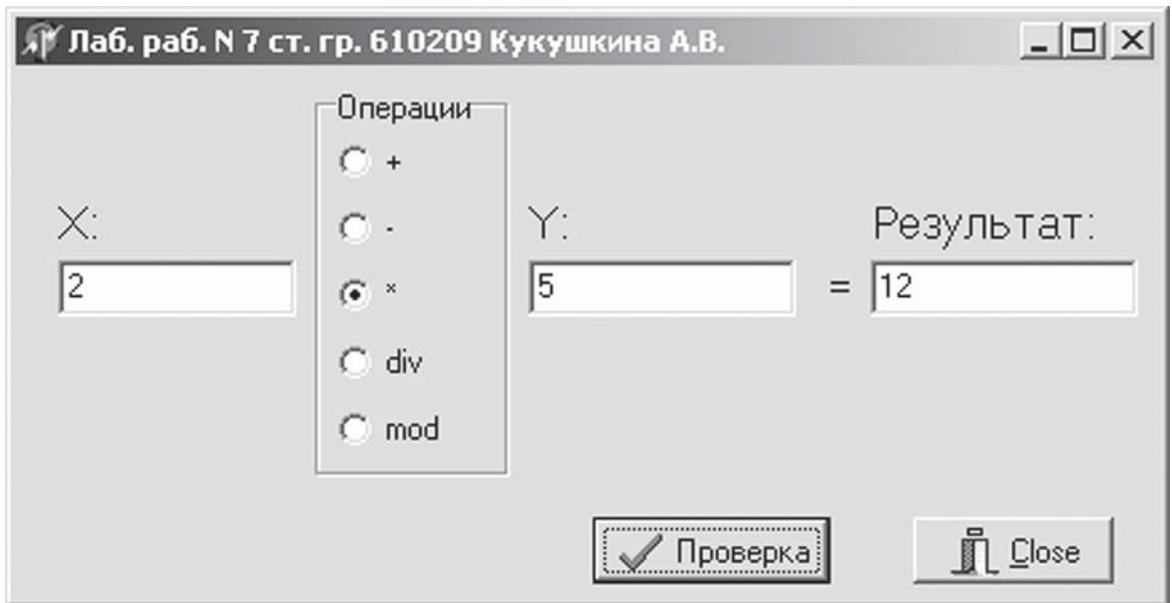


Рис. 7.1

Текст программы приведен ниже.

```
Unit 7;
interface
uses
    Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
    Dialogs, StdCtrls, ExtCtrls;
type
    TForm1 = class(TForm)
        Edit1: TEdit;
        Edit2: TEdit;
        Edit3: TEdit;
        Label1: TLabel;
        Label2: TLabel;
        Label3: TLabel;
        RadioGroup1: TRadioGroup;
        Label4: TLabel;
        Button1: TButton;
        BitBtn1: TButton;
        procedure Edit1Change(Sender: TObject);
        procedure Edit2Change(Sender: TObject);
        procedure Button1Click(Sender: TObject);
        procedure RadioGroup1Click(Sender: TObject);
    private
        { Private declarations }
    public
        { Public declarations }
    end;

var
    Form1: TForm1;
```

implementation

```
{ $R *.dfm }
{ $Q+ }
```

```
function f17_10(a:string):integer; // Преобразования числа из
var b:integer; // семнадцатеричной системы счисления в десятичную
    i,c:byte;
begin
    b:=1; Result:=0;
    i:=length(a);
    if a[1]<>'-' then c:=0 else c:=1;
    while i<>c do
        begin
            case a[i] of
                '0'..'9':Result:=Result+b*StrToInt(a[i]);
                'A'..'G':Result:=Result+b*(ord(a[i])-ord('A')+10);
                'a'..'g':Result:=Result+b*(ord(a[i])-ord('a')+10)
            else
                b:=StrToInt(a[i])
            end;
            b:=b*17;i:=i-1;
        end;
        if c=1 then Result:=-Result
    end;
```

```
function f10_17(a:integer):string; // Преобразование числа из десятичной
var                                     // системы счисления в семнадцатеричную
    b:boolean;
    c:char;
    i:shortint;
begin
    if a<0 then begin a:=abs(a);b:=true;end
        else b:=false;
    if a=0 then Result:='0'
        else Result:='';
    while a>0 do begin
        if a mod 17<10 then Result:=IntToStr(a mod 17)+Result
            else begin
                c:='A';
                for i:=1 to a mod 17 -10 do c:=succ(c);
                Result:=c+Result
            end;
        a:=a div 17
    end;
```



```

    if b then Result:='-'+Result
end;
```

```

procedure TForm1.Edit1Change(Sender: TObject);
begin
    if Edit1.Text<>'' then
    try
        Edit1.Color:=clYellow;
        if Edit2.Text<>'' then Edit3.Color:=clFuchsia;
        if (Edit1.Text[1]='-') and (Length(Edit1.Text)=1) then Exit;
        f17_10(Edit1.Text);
    except
        on EConvertError do if Edit1.Text<>'-' then begin
            ShowMessage('Возможны лишь цифры от 0 до 9 и буквы от А до G'+
                ' или от а до g!');
            Edit1.Text:=Copy(Edit1.Text,1,Length(Edit1.Text)-1);
            Edit1.Selstart:=Length(Edit1.Text)
                end;
        on EIntOverflow do begin
            ShowMessage('Слишком большое число!');
            Edit1.Text:=Copy(Edit1.Text,1,Length(Edit1.Text)-1);
            Edit1.Selstart:=Length(Edit1.Text)
                end
        else ShowMessage('Возникла неизвестная исключительная ситуация!')
        end
    end;
end;
```

```

procedure TForm1.Edit2Change(Sender: TObject);
var c:char;
begin
    if Edit2.Text<>'' then
    try
        Edit2.Color:=clYellow;
        if Edit1.Text<>'' then Edit3.Color:=clFuchsia;
        if (Edit2.Text[1]='-') and (Length(Edit2.Text)=1) then Exit;
        c:=Edit2.Text[Length(Edit2.Text)];
        if not(c in ['0'..'9','A'..'G','a'..'g'])
            then raise EAssertionFailed.Create('');
        f17_10(Edit2.Text);
    except
        on EAssertionFailed do if Edit2.Text<>'-' then begin
            ShowMessage('Возможны лишь цифры от 0 до 9 и буквы от А до G'+
                ' или от а до g!');
            Edit2.Text:=Copy(Edit2.Text,1,Length(Edit2.Text)-1);
            Edit2.Selstart:=Length(Edit2.Text)
                end;
        end;
    end;
end;
```

```

end;

on EIntOverflow do begin
    ShowMessage('Слишком большое число!');
    Edit2.Text:=Copy(Edit2.Text,1,Length(Edit2.Text)-1);
    Edit2.Selstart:=Length(Edit2.Text)
end
else ShowMessage('Возникла неизвестная исключительная ситуация!')
end
end;

procedure TForm1.Button1Click(Sender: TObject);
var p:string;
    x,y:integer;
begin
    try
        try
            x:=f17_10(Edit1.Text);
            y:=f17_10(Edit2.Text);
            case RadioGroup1.ItemIndex of
                0:p:=f10_17(x+y);
                1:p:=f10_17(x-y);
                2:p:=f10_17(x*y);
                3:p:=f10_17(x div y);
                4:p:=f10_17(x mod y);
            end;
            if (p<> Edit3.Text) and (p<>AnsiUpperCase(Edit3.Text)) then Abort
            else
                MessageDlg('Вы полностью правы!',mtInformation ,[mbOk],0);
        except
            on EIntOverflow do
                MessageDlg('Переполнение при выполнении операции!',
                mtError,[mbOk],0);
            on EDivByZero do MessageDlg('Делить на ноль нельзя!',
            mtError,[mbOk],0);
            on EConvertError do
                MessageDlg('Ошибка исходных данных!', mtError,[mbOk],0);
            on EAbort do begin
                MessageDlg('Вы неправильно вычислили!',
                mtWarning,[mbOk],0);
            end
            else MessageDlg('Неизвестная исключительная ситуация!',
            mtError,[mbOk],0);
        end
    finally
        Edit1.Color:=clWindow; Edit2.Color:=clWindow; Edit3.Color:=clWindow
    end
end

```

```
        end;  
    end;  
  
    procedure TForm1.RadioGroup1Click(Sender: TObject);  
    begin  
        if Edit1.Text<>'' then Edit1.Color:=clYellow  
            else Edit1.Color:=clWindow;  
        if Edit2.Text<>'' then Edit2.Color:=clYellow  
            else Edit2.Color:=clWindow;  
        if (Edit1.Text<>') and (Edit2.Text<>') then Edit3.Color:=clFuchsia  
            else Edit3.Color:=clWindow;  
    end;  
  
end.
```

Индивидуальные задания

Составить программу калькулятора, работающего в указанной ниже системе счисления:

1. В троичной.
2. В четверичной.
3. В пятиричной.
4. В шестиричной.
5. В семиричной.
6. В восьмиричной.
7. В девятиричной.
8. В одиннадцатеричной.
9. В двенадцатеричной.
10. В тринадцатеричной.
11. В четырнадцатеричной.
12. В пятнадцатеричной.
13. В шестнадцатеричной.
14. В восемнадцатеричной.
15. В девятнадцатеричной.

ТЕМА 8. ПРОГРАММИРОВАНИЕ С ОТОБРАЖЕНИЕМ ГРАФИЧЕСКОЙ ИНФОРМАЦИИ

Цель лабораторной работы: изучить возможности построения графиков с помощью компонента TChart. Научится работать с графическими объектами. Написать и отладить программу с использованием функций отображения графической информации.

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.

Построение графика (диаграммы) производится после вычисления таблицы значений функции $y=f(x)$. Полученная таблица передается в специальный двумерный массив `Chart1.SeriesList[k]` (**k** – номер графика (0,1,2,...)) компонента TChart с помощью метода `AddXY`. Компонент TChart осуществляет всю работу по отображению графиков, переданных в объект `Chart1.SeriesList[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.

8.2 Использование класса TCanvas

Для рисования в Delphi используется класс TCanvas, который является не самостоятельным компонентом, а свойством многих компонентов, и представляет собой холст (контекст GDI в Windows) с набором инструментов для рисования.

Основные свойства класса TCanvas:

Property Pen:TPen; – карандаш,

Property Brush:TBrush; – кисть,

Property Font:TFont; – шрифт.

Некоторые методы класса TCanvas:

Procedure Ellipse(X1, Y1, X2, Y2: Integer) – чертит эллипс в охватывающем

прямоугольнике (X1, Y1), (X2, Y2). Заполняет внутреннее пространство эллипса текущей кистью.

Procedure LineTo (X, Y: Integer) – чертит линию от текущего положения пера до точки (X, Y).

Procedure MoveTo(X, Y: Integer) – перемещает карандаш в положение (X, Y) без вычерчивания линий.

Procedure Polygon (Points: array of TPoint) – вычерчивает карандашом многоугольник по точкам, заданным в массиве Points. Конечная точка соединяется с начальной и многоугольник заполняется кистью. Для вычерчивания без заполнения используйте метод Polyline.

Procedure Rectangle (X1, Y1, X2, Y2: Integer) – вычерчивает и заполняет прямоугольник (X1, Y1), (X2, Y2). Для вычерчивания без заполнения используйте FrameRect или PolyLine.

Procedure TextOut (X, Y: Integer; const Text: String) – выводит текстовую строку Text так, чтобы левый верхний угол прямоугольника, охватывающего текст, располагался в точке (X, Y).

8.3. Пример написания программы

Задание: составить программу, отображающую движение автомобиля с изменяющейся скоростью. Вывести с помощью компонента TChar график скорости.

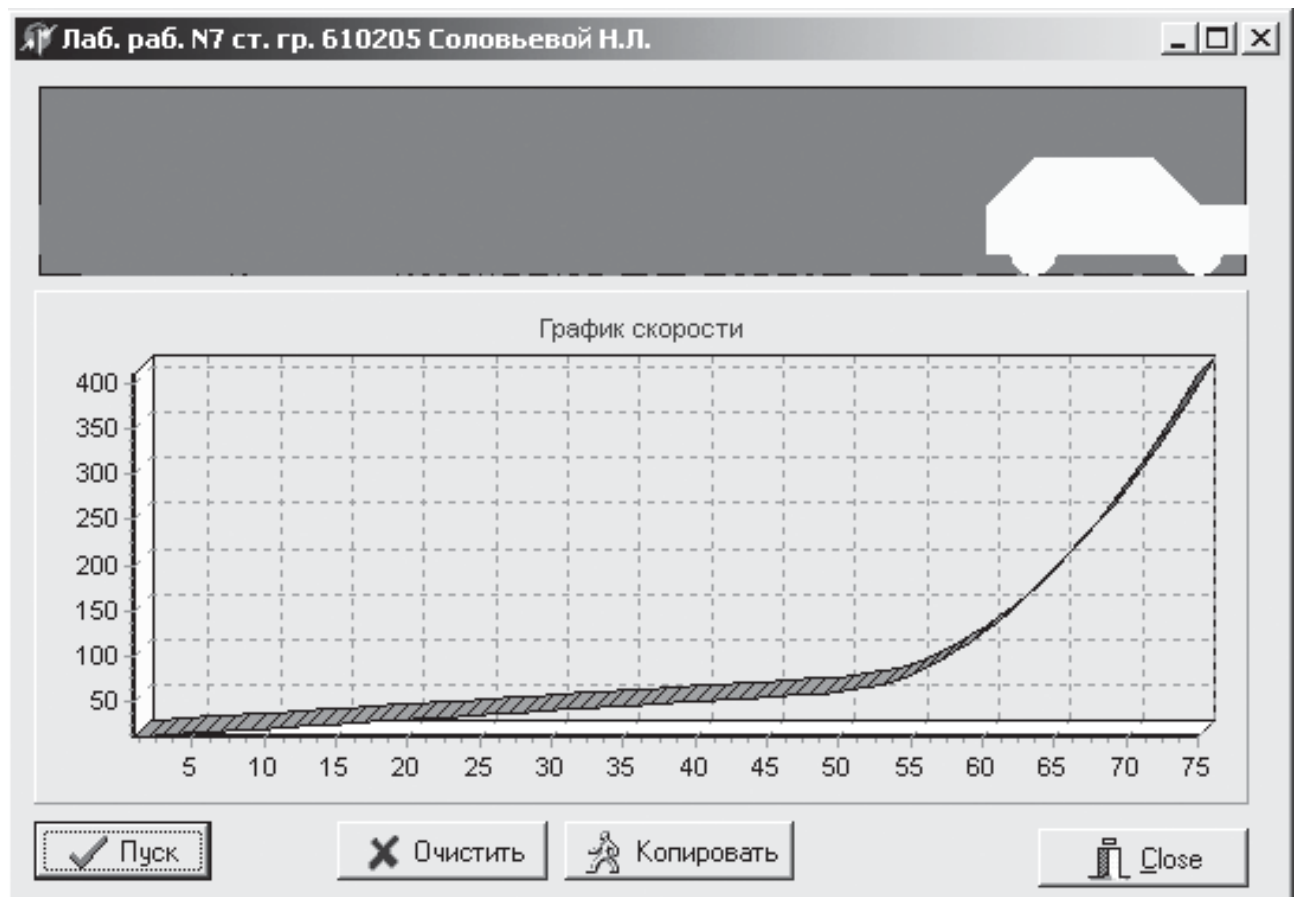



Рис. 8.1

8.3.1. Настройка формы

Панель диалога программы организуется в виде, представленном на рис. 8.1. Причем поле с автомобилем не видно на этапе настройки формы и появляется только как результат выполнения кода программы. Поэтому в том месте формы, где предполагается делать вывод графической информации следует оставить свободное место.

Компонент TChart вводится в форму путем нажатия пиктограммы  в меню компонентов.

8.3.2. Работа с компонентом TChart

Для изменения параметров компонента TChart необходимо дважды щелкнуть по нему мышью в окне формы. Появится окно редактирования EditingChart1 (рис. 8.2). Для создания нового объекта Series1 щелкнуть по кнопке Add на

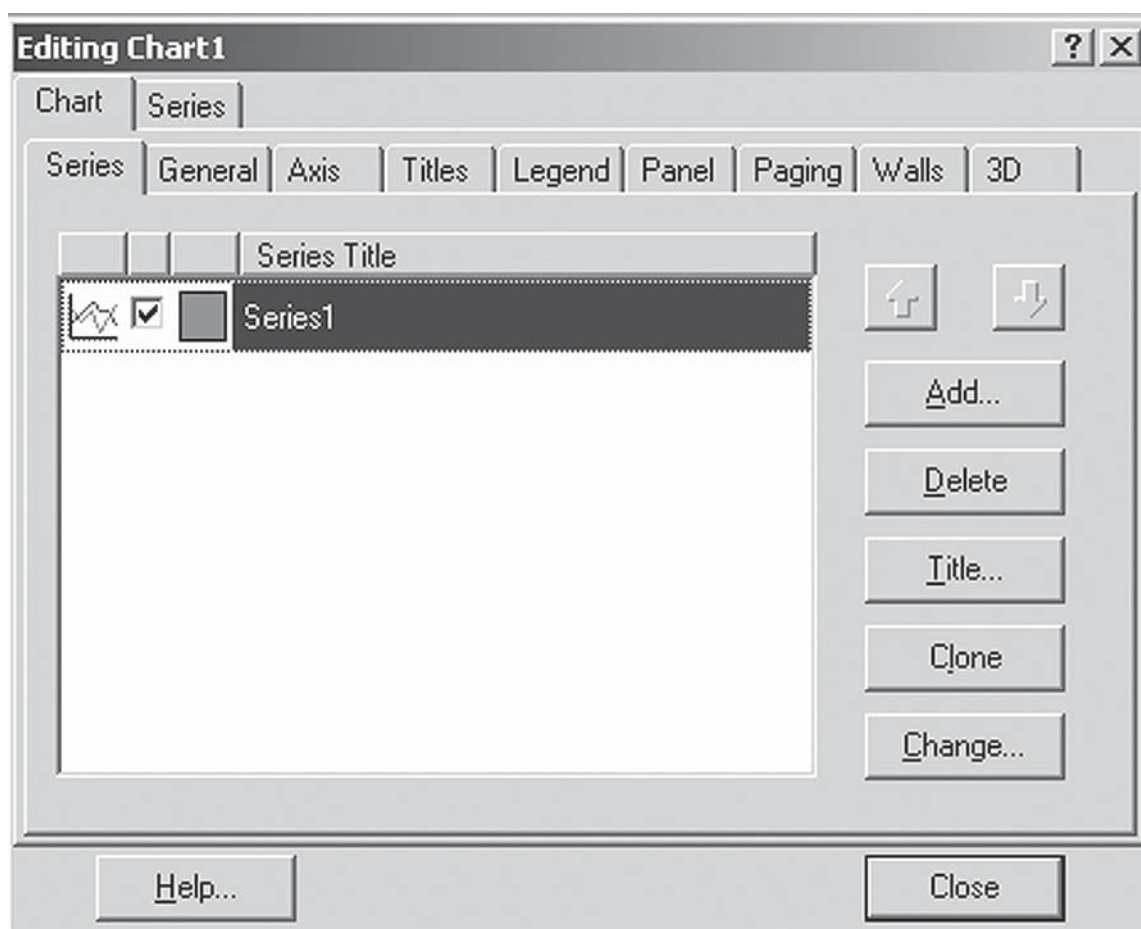


Рис. 8.2

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 ОК появится новая серия с названием Series1. Для изменения названия нажать кнопку Title. Название графика вводится на странице Titles. Разметка осей меняется на странице Axis.

Нажимая различные кнопки меню, познакомиться с другими возможностями EditingChat.

Текст программы имеет вид:

```

unit Unit7;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls, TeeProcs, TeEngine, Chart, Series;

type
  TForm1 = class(TForm)
    Chart1: TChart;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    BitBtn3: TBitBtn;
    Series1: TLineSeries;
    BitBtn4: TBitBtn;
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.BitBtn1Click(Sender: TObject);
var
  h : extended;
  hx,x,n : integer;
  Procedure Shar(xx,yy: integer;cc:TColor);//Вывод изображения машины
  begin
    Canvas.Pen.Color:=cc; // Установка цвета карандаша
    Canvas.Brush.Color:=cc; // Установка цвета кисти
    Canvas.Polygon([Point(xx+0,yy+20),Point(xx+0,yy+40), // Рисует корпус
      Point(xx+110,yy+40),Point(xx+110,yy+20), // машины
      Point(xx+90,yy+20),Point(xx+70,yy+0), Point(xx+20,yy+0)]);
    Canvas.Ellipse(xx+10,yy+30,xx+30,yy+50); // Рисует колесо
  end;
end;

```



```

Canvas.Ellipse(xx+80,yy+30,xx+100,yy+50); // Рисуем колесо
end; // Конец процедуры вывода изображения машины
begin // Начало процедуры BitBtn1Click
Canvas.Pen.Color := clBlack; // Установка цвета карандаша
Canvas.Brush.Color:=clGreen; // Установка цвета кисти
Canvas.Rectangle(10,10,520,90); // Рисуем прямоугольник
hx:=1; h:=0;
x:=10; n:=0;
while x<410 do begin
h:=h+0.01;
hx:=hx+Round(h); // Изменение шага для изменения скорости
Shar(x,40,clYellow); // Рисование машины
Sleep(10); // Задержка
Shar(x,40,clGreen); // Стирание нарисованной машины
Inc(x,hx);
Inc(n);
Chart1.SeriesList[0].AddXY(n,x); // Ввод данных для графика
end;
end;

procedure TForm1.BitBtn2Click(Sender: TObject);
begin
Chart1.SeriesList[0].Clear; // Очистка графика
end;

procedure TForm1.BitBtn3Click(Sender: TObject);
begin
Chart1.CopyToClipboardMetafile(True); // Передача графика в буфер обмена
end;
end.

```

8.4. Выполнение индивидуального задания

Решить задачу в соответствии с заданным вариантом и используя функции класса TCanvas нарисовать соответствующие геометрические фигуры.

1. Даны три числа a , b , c . Необходимо определить, существует ли треугольник с такими длинами сторон.

2. Даны четыре числа a , b , c , d . Необходимо определить, существует ли четырехугольник с такими длинами сторон.

3. Найти взаимное расположение двух окружностей радиусов R_1 и R_2 с центрами в точках (x_1, y_1) , (x_2, y_2) соответственно.

4. Найти взаимное расположение окружности радиуса R с центром в точке (x_0, y_0) и прямой, проходящей через точки с координатами (x_1, y_1) и (x_2, y_2) (пересекаются, касаются, не пересекаются).

5. Определить количество точек с целочисленными координатами, лежащих внутри окружности радиуса R с центром в точке (x_0, y_0) .
6. Найти координаты точек пересечения двух окружностей радиуса R_1 и R_2 с центрами в точках (x_1, y_1) и (x_2, y_2) соответственно.
7. Найти координаты точки, симметричной данной точке M с координатами (x_1, y_1) относительно прямой $Ax+By+C=0$.
8. Даны две точки $M_1(x_1, y_1)$, $M_2(x_2, y_2)$ и прямая $Ax+By+C=0$. Необходимо найти на этой прямой такую точку $M_0(x_0, y_0)$, чтобы суммарное расстояние от нее до двух данных точек было минимально.
9. Даны три точки с координатами (x_1, y_1) , (x_2, y_2) , (x_3, y_3) , которые являются вершинами некоторого прямоугольника со сторонами, параллельными осям координат. Найти координаты четвертой точки.
10. Даны координаты четырех точек (x_1, y_1) , (x_2, y_2) , (x_3, y_3) , (x_4, y_4) . Необходимо определить, образуют ли они выпуклый четырехугольник.
11. Даны координаты четырех точек (x_1, y_1) , (x_2, y_2) , (x_3, y_3) , (x_4, y_4) . Необходимо определить, образуют ли они: а) ромб; б) квадрат; в) трапецию.
12. Даны координаты двух вершин (x_1, y_1) и (x_2, y_2) некоторого квадрата. Необходимо найти возможные координаты других его вершин.
13. Даны координаты двух вершин (x_1, y_1) и (x_2, y_2) некоторого квадрата, которые расположены по диагонали, и точка (x_3, y_3) . Необходимо определить, лежит или не лежит точка внутри квадрата.
14. Даны координаты трех вершин (x_1, y_1) , (x_2, y_2) , (x_3, y_3) треугольника. Необходимо найти координаты точки пересечения его медиан.
15. Даны координаты трех вершин (x_1, y_1) , (x_2, y_2) , (x_3, y_3) треугольника. Необходимо найти длины его высот.

16-30. 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.

31. Разработать программу реализующую игру «Бега лошадей по кругу ипподрома». Предусмотреть возможность устанавливать ставки на лошадей и расчета выигрыша. Скорость движения лошадей должна задаваться случайным образом функцией Random.

32. Разработать программу реализующую игру «Бега лошадей по прямой». Предусмотреть возможность устанавливать ставки на лошадей и расчета выигрыша. Скорость движения лошадей должна задаваться случайным образом функцией Random.

33. Разработать программу игры в крестики – нолики. В основу положить компонент DrawGrid.

34. Разработать программу игры «Минер» по подобию такой же игры в системе

Windows. Начальная расстановка мин должна выполняться случайным образом. В основу положить компонент DrawGrid.

35. Разработать программу игры стрельбы из подводной лодки по кораблю используя вид из перископа. На заднем плане должен периодически проплывать корабль с постоянной поперечной скоростью. С помощью клавиш влево-вправо следует менять вид в перископе. Клавиша «Ввод» должна запускать торпеду. В перископе должна отображаться траектория движения торпеды с уменьшением скорости движения при приближении к кораблю. Попадание должно сопровождаться видимым взрывом и исчезновением корабля.

36. Разработать программы игры «Бомбометание с самолета по наземной цели». С летящего с постоянной скоростью самолета клавишей «Ввод» производить бомбометание. Траектория движения бомбы должна соответствовать физическим законам падения тел на землю. Попадание в цель должно сопровождаться видимым взрывом и исчезновением цели. Самолет должен периодически вылетать из-за края канвы компонента рисования.

37. Разработать программу игры «Морской бой». Программа должна случайным образом на сетке 10x10 расставлять корабли: один четырех клеточный, два трех клеточных, три двух клеточных и четыре одноклеточных. Они не могут изгибаться и соприкасаться друг с другом. Игрок выбирает определенный квадрат и как бы стреляет в него. Программа должна сообщать, попал ли игрок в корабль или нет. Она также должна отображать все старые выстрелы и показывать ячейки, куда уже не имеет смысла стрелять. Аналогично строится и вторая таблица, где игрок располагает свои корабли по которым уже случайным образом стреляет программа. Выигрывает тот, кто быстрее потопит корабли неприятеля. Предусмотреть в конце игры показ расположения кораблей программы. Для таблиц использовать компоненты TdrawGrid.

38. Разработать программы игры «Стрельба из пушки». Пушка должна стрелять через гору по какой-то цели. Траектория полета снаряда должна подчиняться законам физики. Игрок может управлять углом подъема ствола относительно горизонта и начальной скоростью снаряда в дискретных величинах (определяется типом снаряда). При попадании должен происходить видимый взрыв и исчезновение цели.

39. Разработать программу показа в форме текущего времени в виде обычных стрелочных часов со стрелками часов, минут и секунд.

40. Разработать программу простейшего графического редактора (аналога программы Paint системы Windows). Он должен рисовать в канве компонента TpaintBox произвольные кривые с помощью мышки. Предусмотреть возможность:

- а) изменения толщины кривых,
- б) изменение цвета кривых,
- в) сохранение рисунка в графическом файле.

41. Разработать программу простейшего графического редактора (аналога программы Paint системы Windows). Он должен рисовать в канве компонента

ТраintBox ломанные линии с помощью нажатия на клавиши мышки. Предусмотреть возможность:

- а) изменения толщины линий,
- б) изменение цвета линий,
- в) сохранение рисунка в графическом файле.

42. Разработать программу простейшего графического редактора (аналога программы Paint системы Windows). Он должен рисовать в канве компонента ТраintBox с помощью мышки прямоугольники. Предусмотреть возможность:

- а) изменения толщины линий,
- б) изменение цвета линий,
- в) заливку областей текущей кистью,
- г) изменение цвета кисти,
- д) сохранение рисунка в графическом файле.

43. Разработать программу простейшего графического редактора (аналога программы Paint системы Windows). Он должен рисовать в канве компонента ТраintBox с помощью мышки эллипсов. Предусмотреть возможность:

- а) изменения толщины линий,
- б) изменение цвета линий,
- в) заливку областей текущей кистью,
- г) изменение цвета кисти,
- д) сохранение рисунка в графическом файле.

44. Разработать программу простейшего графического редактора (аналога программы Paint системы Windows). Он должен рисовать в канве компонента ТраintBox любой текст в указанном мышкой месте. Предусмотреть возможность:

- а) изменения типа, размера и цвета шрифта,
- б) сохранение рисунка в графическом файле.

45. Разработать программу простейшего графического редактора (аналога программы Paint системы Windows). Он должен помещать в канву компонента ТраintBox из графического файла произвольный рисунок и обеспечивать возможность:

- а) стирания произвольной области рисунка,
- б) изменение размеров стирки,
- в) сохранение рисунка в графическом файле.

ТЕМА 9. ИСПОЛЬЗОВАНИЕ ЗАПИСЕЙ ДЛЯ РАБОТЫ С КОМПЛЕКСНЫМИ ЧИСЛАМИ.

Цель лабораторной работы: изучить правила работы с записями и их использование для работы с комплексными числами.

9.1. Определение 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      TStudent = record           {Объявление типа}
           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 а 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;
```

9.2. Использование типа запись для работы с комплексными числами.

В языке Паскаль отсутствуют стандартные операции с комплексными числами, поэтому можно создать модуль, в котором все операции под комплексными числами будут оформлены в виде набора функций и процедур. Пример модуля приведен ниже.

```
Unit Cmplx;
Interface
Type Complex=record
    re:extended; // Действительная часть числа
    im:extended; // Мнимая часть числа
end;
function Addc(x,y:Complex):Complex; // (x+y)
function Mulc(x,y:Complex):Complex; // (x*y)
function Divc(x,y:Complex):Complex; // (x/y)
```

```

Implementation
function Addc;
begin
  Addc.re:=x.re+y.re;
  Addc.im:=x.im+y.im;
end;
function Mulc;
begin
  Mulc.re:=x.re*y.re-x.im*y.im;
  Mulc.im:=x.re*y.im+x.im*y.re;
end;
function Divc;
var z;extended;
begin
  z:=sqr(y.re)+sqr(y.im);
  Divc.re:=(x.re*y.re+x.im*y.im)/z;
  Divc.im:=(x.im*y.re-x.re*y.im)/z;
end;
end.

```

Вычисление значения выражения $u=(a+b)/(a \cdot b)$ (где, a и b – комплексные числа) выглядит следующим образом: $u:=\text{Divc}(\text{Addc}(a,b),\text{Mulc}(a,b))$.

9.3. Порядок выполнения задания.

Задание: написать программу для расчета комплексной функции, представленной в виде ряда: $\dot{S}(X) = \sum_{k=1}^N \dot{a}_k e^{jkx}$, $\dot{a}_k = \frac{j+k}{jk+1}$ (точка над переменной означает комплексное число). Вывести графики действительной и мнимой частей $S_{re}(X)$, $S_{im}(X)$. Для выполнения этой задачи модуль Cmplx нужно дополнить двумя функциями:

```

Function Expj(a:extended):Complex; //Expj=exp(j*a)
begin
  Expj.re:=cos(a);
  Expj.im:=sin(a);
end;
Function Cplx(a,b:Extended):Complex; //Cplx=a+j*b
begin
  Cplx.re:=a;
  Cplx.im:=b;
end;

```

9.3.1. Работа с программой

После запуска программы на выполнение появится диалоговое окно программы. Необходимо ввести исходные данные для расчета: N , X_{\min} , X_{\max} , h . Затем нажать кнопку “Расчет”, после чего будут выведены графики.

Панель диалога будет иметь вид (рис. 9.1).

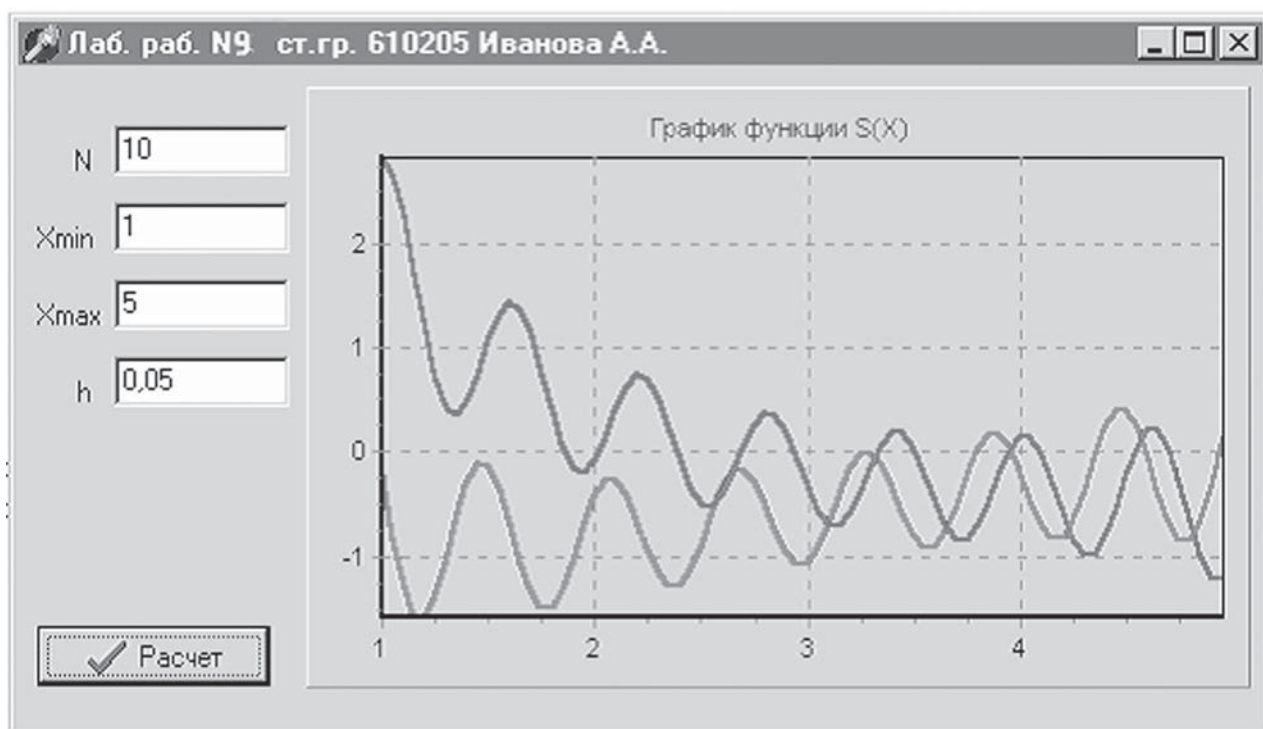


Рис. 9.1

Текст программы приведен ниже.

```
unit unit9;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Buttons, TeEngine, Series, ExtCtrls, TeeProcs, Chart,
  Cmplx;
type
  TForm1 = class(TForm)
    Chart1: TChart;
    Edit1: TEdit;
    Label1: TLabel;
    Edit2: TEdit;
    Label2: TLabel;
    BitBtn1: TBitBtn;
    Edit3: TEdit;
    Label3: TLabel;
    Edit4: TEdit;
    Label4: TLabel;
    Series1: TFastLineSeries;
    Series2: TFastLineSeries;
  procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
```



```

    { Public declarations }
end;
var
    Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.BitBtn1Click(Sender: TObject);    //Расчет
var
    n,k:integer;
    xmin,xmax,h,x:extended;
    s,a,b,c,d:complex;
begin
    series1.Clear;
    series2.Clear;
    n:=strtoint(edit1.text);
    xmin:=strtofloat(edit2.text);
    xmax:=strtofloat(edit3.text);
    h:=strtofloat(edit4.text);
    x:=xmin;
    repeat
        s.re:=0;
        s.im:=0;
        for k:=1 to n do
            begin
                b:=Cplx(k,1);
                c:=Cplx(1,k);
                a:=Divc(b,c);
                d:=mulc(a,Expj(k*x));
                s:=addc(s,d);
            end;
            series1.AddXY(x,s.re,"",clteecolor);    //Вывод реальной части
            series2.AddXY(x,s.im,"",clteecolor);    //Вывод мнимой части
            x:=x+h;
        until x>xmax;
    end;
end.

```

Варианты индивидуальных заданий

Написать программу для расчета комплексной функции $\dot{S}(x)$. Вывести графики действительной и мнимой частей $\dot{S}_{re}(x)$, $\dot{S}_{im}(x)$. $x \in \overline{(1,3)}$, $N = 10$.

$$1. \dot{S}(x) = \sum_{k=0}^N \frac{j + e^{j+x}}{jk + x} e^{jk+x}, \quad \dot{S}_{re}(2) = -0.446, \quad \dot{S}_{im}(2) = 3.499.$$

$$2. \dot{S}(x) = \prod_{k=0}^N \frac{j+x \frac{2j-k-5}{jx-k}}{jxk+1}, \dot{S}_{re}(2) = -5.905 * 10^{-4}, \dot{S}_{im}(2) = -0.002.$$

$$3. \dot{S}(x) = \sum_{k=0}^N \frac{k}{2^{jx} + kx} e^{jkx+x}, \dot{S}_{re}(2) = 1.103, \dot{S}_{im}(2) = 2.408.$$

$$4. \dot{S}(x) = \prod_{k=0}^N \frac{\ln(kx-j)}{2j-x}, \dot{S}_{re}(2) = -0.031, \dot{S}_{im}(2) = 0.007.$$

$$5. \dot{S}(x) = \sum_{k=0}^N \frac{x^{2j-k} - \ln\left(\frac{x}{j-k}\right)}{\ln(j-x) + jkx} e^{jxk+x}, \dot{S}_{re}(2) = 9.605, \dot{S}_{im}(2) = 4.281.$$

$$6. \dot{S}(x) = \prod_{k=0}^N \frac{\ln(kx-j)}{2j-e^{jx}}, \dot{S}_{re}(2) = -524.624, \dot{S}_{im}(2) = 73.265.$$

$$7. \dot{S}(x) = \sum_{k=0}^N \frac{x^{2j-k} - 3j}{2^{x-j} + jkx} e^{\frac{x}{3j-k}-j}, \dot{S}_{re}(2) = -3.019, \dot{S}_{im}(2) = 1.085.$$

$$8. \dot{S}(x) = \prod_{k=0}^N \frac{j+x \frac{j-2k-3}{j-kx}}{\ln(jx)}, \dot{S}_{re}(2) = 466,381, \dot{S}_{im}(2) = -962.124.$$

$$9. \dot{S}(x) = \sum_{k=0}^N \frac{\sqrt{j-x}}{\ln(jx-kx)} e^{jx}, \dot{S}_{re}(2) = -3.548, \dot{S}_{im}(2) = 3.497.$$

$$10. \dot{S}(x) = \prod_{k=0}^N \frac{4\ln(\sqrt{jk-5})}{jk-x}, \dot{S}_{re}(2) = 7.92, \dot{S}_{im}(2) = -15.136.$$

$$11. \dot{S}(x) = \sum_{k=0}^N \frac{\sqrt{jk-x}}{\ln((jx-2k)^2)} e^{jx-k}, \dot{S}_{re}(2) = -0.556, \dot{S}_{im}(2) = 0.09.$$

$$12. \dot{S}(x) = \prod_{k=0}^N \frac{\ln(jk-x)}{e^{2j-x}}, \dot{S}_{re}(2) = 103.695, \dot{S}_{im}(2) = -189.537.$$

$$13. \dot{S}(x) = \sum_{k=0}^N \frac{\ln(\sqrt{jk-x})^2}{\ln((jx-2k)^2)} e^{jxk}, \dot{S}_{re}(2) = 0.55, \dot{S}_{im}(2) = 0.625.$$

$$14. \dot{S}(x) = \prod_{k=0}^N \frac{\ln(kx - \frac{j}{2j-x})}{\sqrt{jk-x}}, \dot{S}_{re}(2) = -0.399, \dot{S}_{im}(2) = -0.028.$$

$$15. \dot{S}(x) = \prod_{k=0}^N \frac{10\ln((kx-j)^2)}{(jx-5)^2}, \dot{S}_{re}(2) = -43.068, \dot{S}_{im}(2) = 22.94$$

ТЕМА 10. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ФАЙЛОВ

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

10.1. 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.

Текстовый 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);                 // Заккрытие файла

```

10.2. 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 текстовый файл для дописывания текста в конец файла.

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

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

CloseFile(F) - закрывает ранее открытый файл.

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

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

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

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

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

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

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

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

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

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

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

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

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.

10.3. 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`.

10.4. 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 текстовый файл.

10.4.1. 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 – маску. Для `OpenDialog1` установим значения маски как показано на рис. 10.1. Формат `*.dat` означает что, будут видны все файлы с расширением `dat`, а формат `*.*` - что будут видны все файлы (с любым именем и с любым расширением).

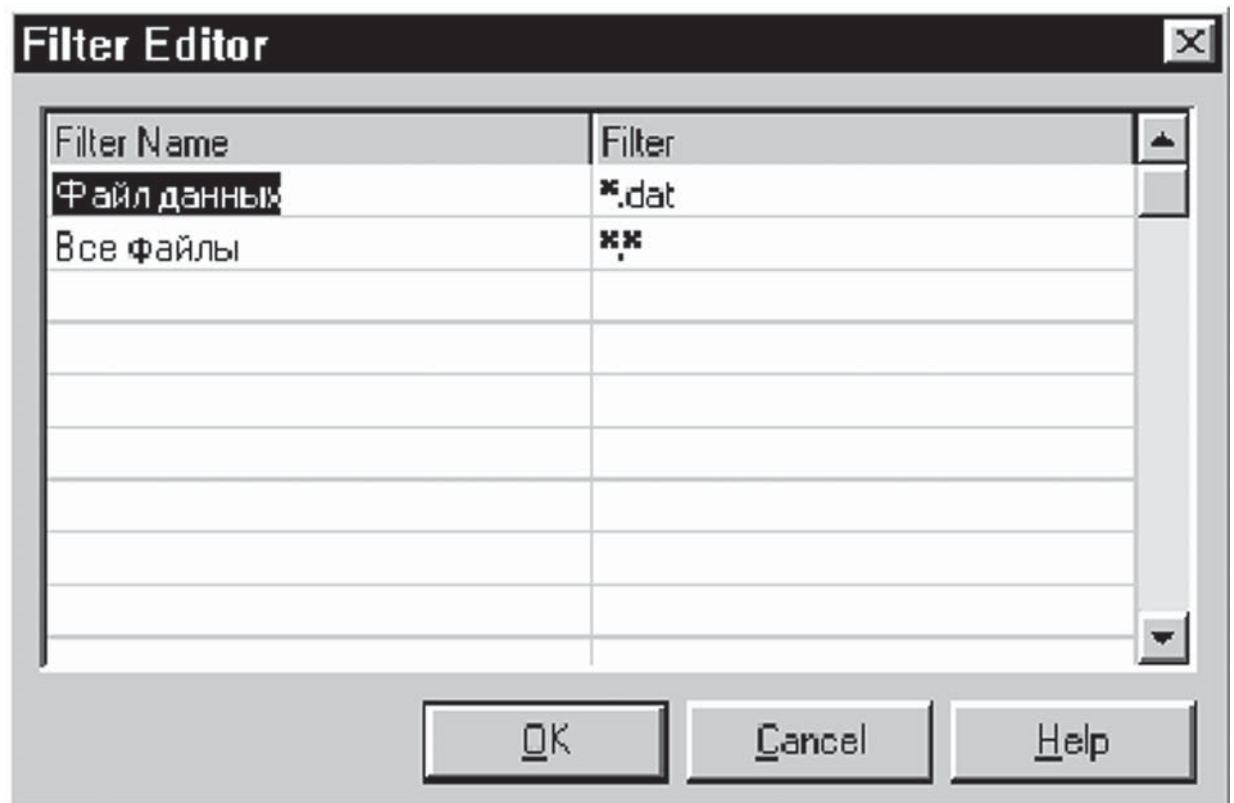


Рис. 10.1

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).

10.4.2. 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. 10.2. Затем при нажатии на кнопку “Сохранить” будет создан текстовый файл, содержащий отсортированную ведомость. Файл записей закрывается одновременно с программой при нажатии на кнопку “Close” или .

The screenshot shows a Windows application window with the title bar text "Лаб. раб. №10 ст. гр. 610205 Иванова А.А.". The window contains a form with the following elements:

- A text input field labeled "Ф.И.О." (Surname, Name, Patronymic).
- A section labeled "Оценки:" (Grades) containing three input fields for "Математика" (Mathematics), "Физика" (Physics), and "Сочинение" (Essay).
- A button labeled "Ввести запись" (Enter record) located to the right of the "Сочинение" input field.
- A list box displaying a table of student records, sorted by average score in descending order:

№	Ф.И.О.	Средний балл
1	Заманов Вадим Сергеевич	4,33
2	Смирнов Алексей Николаевич	4,00
3	Иванов Александр Иванович	4,00
4	Никольский Олег Сергеевич	3,67

At the bottom of the window, there is a row of five buttons: "Создать" (Create), "Открыть" (Open), "Сортировать" (Sort), "Сохранить" (Save), and "Close".

Рис. 10.2

Текст программы приведен ниже.

```

unit unit6;
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;
    OpenFileDialog1: 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
begin
    // записей
    OpenFileDialog1.Title := 'Создать новый файл'; // Изменение заголовка
    // окна диалога
    if OpenFileDialog1.Execute then // Выполнение стандартного диалога выбора
    begin
        // имени файла
        FileNameZ:= OpenFileDialog1.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
begin
    // файл
    if OpenFileDialog1.Execute then // Выполнение стандартного диалога
    begin
        // выбора имени файла
        FileNameZ:= OpenFileDialog1.FileName; // Возвращение имени
        // дискового файла
        AssignFile(Fz, FileNameZ); // Связывание файловой переменной Fz
        // с именем файла
        Reset(Fz); // Открытие существующего файла
    end;
    nzap:=0;
    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;
    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,ffixed,4,2));
end;

procedure TForm1.Button5Click(Sender: TObject); // Сохранение результатов
var i:word;                                     // сортировки в текстовом файле
begin
if SaveDialog1.Execute then // Выполнение стандартного диалога выбора
begin                                     // имени файла
    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.

```

Индивидуальные 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 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.

ПРИЛОЖЕНИЕ 1. ПРОЦЕДУРЫ И ФУНКЦИИ ДЛЯ РАБОТЫ СО СТРОКАМИ

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

<i>Подпрограммы преобразования строк в другие типы</i>	
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
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.
<i>Подпрограммы обратного преобразования</i>	
Function FloatToStr(Value: Extended): String;	Преобразует вещественное значение Value в строку символов
Function FloatToStrF(Value: Extended; Format: TFloatFormat; Precision, Digits: Integer) : String;	Преобразует вещественное значение Value в строку символов с учетом параметров Precision и Digits (см. пояснения ниже)
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.
<i>Правила использования параметров функции FloatToStrF</i>	
Значение Format	Описание
ffExponent	Научная форма представления с множителем eXX. Precision задает общее количество десятичных цифр мантиссы. Digits - количество цифр в десятичном порядке XX.
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).

ПРИЛОЖЕНИЕ 2. МАТЕМАТИЧЕСКИЕ ФОРМУЛЫ

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

Функция	Соотношение	Модуль Math
$\text{Log}_a(x)$	$\frac{\text{Ln}(x)}{\text{Ln}(a)}$	LogN(a, x)
x^a	$e^{a \cdot \text{Ln}(x)}$	Power(x,a)
$\text{Tg}(x)$	$\frac{\text{Sin}(x)}{\text{Cos}(x)}$	Tan(x)
$\text{Ctg}(x)$	$\frac{\text{Cos}(x)}{\text{Sin}(x)}$	Cotan(x)
$\text{ArcSin}(x)$	$\text{ArcTg}\left(\frac{x}{\sqrt{1-x^2}}\right)$	ArcSin(x)
$\text{ArcCos}(x)$	$\frac{\pi}{2} - \text{ArcSin}(x)$	ArcCos(x)
$\text{ArcCtg}(x)$	$\frac{\pi}{2} - \text{ArcTg}(x)$	
$\text{Sh}(x)$	$\frac{e^x - e^{-x}}{2}$	Sinh(x)
$\text{Ch}(x)$	$\frac{e^x + e^{-x}}{2}$	Cosh(x)
$\text{Csc}(x)$	$\frac{1}{\sin(x)}$	Cosecant(x)
$\text{Sc}(x)$	$\frac{1}{\cos(x)}$	Secant(x)

ЛИТЕРАТУРА

1. Фаронов В.В. DELPHI 3. Учебный курс. –М.: Нолидж, 1998. –400 с.
2. Дарахвелидзе П.Г., Марков Е.П. Delphi – среда визуального программирования: - СПб.: BHV –Санкт-Петербург, 1996. – 352 с.
3. Федеоров А.Г. Delphi 3.0. для всех: - М.: КомпьютерПресс, 1998. – 544 с.
Св. план 1999, поз. 15 (вед.)
4. Дарахвелидзе П.Г., Марков Е.П. Delphi 4 – СПб.: BHV –Санкт-Петербург, 1999. – 780 с.
5. Дж.Гленн Брукшир. Введение в компьютерные науки. «Вильямс» М, С-П, Киев. 2001.
6. Архангельский А.Я. Все о Delphi. М: Бином. 1999.
7. Фаронов В.В. Delphi 4. Учебный курс, М. 1999.

Св. план 2002, поз. _____

Учебное издание

А.К. Синицын, С.В. Колосов, А.А. Навроцкий,
А.В. Гуревич, В.Е. Карцев, А.А. Лавренов, А.В. Щербаков

**ПРОГРАММИРОВАНИЕ АЛГОРИТМОВ
В СРЕДЕ DELPHI**

Лабораторный практикум по курсу
«Основы алгоритмизации и программирование»
для студентов 1 – 2-го курсов всех специальностей БГУИР
часть 1

Под общей редакцией А.К.Синицына

Редактор ***Е.Н. Батурчик***

Подписано в печать

Формат 68х84 1/16

Бумага офсетная.

Печать ризографическая

Усл. печ. л.

Уч. изд. л. .

Тираж 500 экз.

Заказ

Издатель и полиграфическое исполнение:

Учреждение образования

“Белорусский государственный университет информатики и радиоэлектроники”

Лицензия ЛП №156 от 05.02.2001.

Лицензия ЛП №509 от 03.08.2001.

220013, Минск, П. Бровки, 6