Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра «Вычислительные методы и программирование»

А.К. Синицын, А.А. Навроцкий

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ В СРЕДЕ DELPHI. БАЗОВЫЕ ТИПЫ И ПРОСТЕЙШИЕ АЛГОРИТМЫ

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

Минск 2005

УДК 681.3.06 (075.8)
ББК 32.973-018 я73
С 38

Р е ц е н з е н т:
заведующий кафедрой информатики БГУИР,
д-р физ.-мат. наук, проф. Л.И. Минченко

Синицын А.К.

С 38 Основы алгоритмизации и программирования в среде DELPHI. Базовые типы и простейшие алгоритмы: Лаб. практикум по курсу «Основы алгоритмизации и программирования» для студ. 1 - 2-го курсов всех спец. БГУИР / А.К. Синицын, А.А. Навроцкий. – Мн.: БГУИР, 2005. – 80 с.: ил.
ISBN 985-444-904-1

Практикум содержит 10 тем, в которых даны краткие теоретические сведения по основам программирования на языке Object Pascal в среде DELPHI, рассмотрены простейшие алгоритмы. После каждой темы приведен набор индивидуальных заданий.

УДК 681.3.06 (075.8)
ББК 32.973-018 я 73

ISBN 985-444-904-1

СОДЕРЖАНИЕ

ТЕМА 1. ОСНОВЫ РАБОТЫ В СРЕДЕ DELPHI. ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ АЛГОРИТМОВ

ТЕМА 2. ОБРАБОТКА СОБЫТИЙ В СРЕДЕ DELPHI. ПРОГРАММИРОВАНИЕ РАЗВЕТВЛЯЮЩИХСЯ АЛГОРИТМОВ

ТЕМА 3. СРЕДСТВА ОТЛАДКИ ПРОГРАММ В СРЕДЕ DELPHI. ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ

ТЕМА 4. ОБРАБОТКА ИСКЛЮЧИТЕЛЬНЫХ СИТУАЦИЙ. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ МАССИВОВ

ТЕМА 5. УКАЗАТЕЛИ И ИХ ИСПОЛЬЗОВАНИЕ ПРИ РАБОТЕ С ДИНАМИЧЕСКИМИ МАССИВАМИ

ТЕМА 6. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ПОДПРОГРАММ И МОДУЛЕЙ

ТЕМА 7. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ МНОЖЕСТВ И СТРОК. СИСТЕМЫ СЧИСЛЕНИЯ

ТЕМА 8. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ЗАПИСЕЙ И ФАЙЛОВ

ТЕМА 9. ПРОГРАММИРОВАНИЕ С ОТОБРАЖЕНИЕМ ГРАФИЧЕСКОЙ ИНФОРМАЦИИ

ТЕМА 10. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ОБЪЕКТОВ И КЛАССОВ

ПРИЛОЖЕНИЕ 1. ПРОЦЕДУРЫ И ФУНКЦИИ ДЛЯ ПРЕОБРАЗОВАНИЯ СТРОКОВОГО ПРЕДСТАВЛЕНИЯ ЧИСЕЛ

ПРИЛОЖЕНИЕ 2. МАТЕМАТИЧЕСКИЕ ФОРМУЛЫ

ЛИТЕРАТУРА

ТЕМА 1. ОСНОВЫ РАБОТЫ В СРЕДЕ DELPHI. 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 потребностей, что значительно повышает производительность работы. При запуске DELPHI вы можете увидеть на экране картинку, подобную представленной на рис. 1.1.

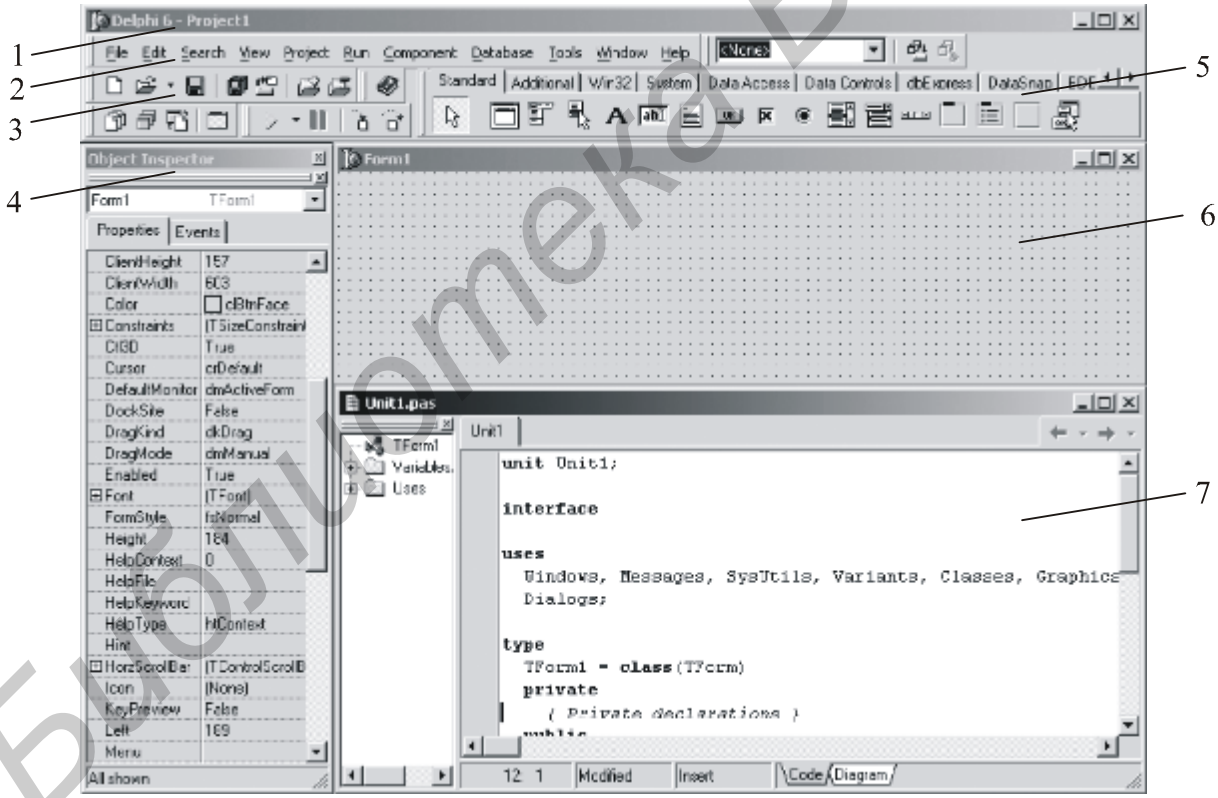

Рис.1.1:

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

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. Посредством 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 визуальные и невизуальные компоненты. При выполнении программы помещенные визуальные 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) (см. листинг 1.1).

Программа в среде DELPHI составляется как описание алгоритмов, которые будут выполняться при возникновении того или иного события (например щелчок мышью по кнопке – событие OnClick, создание формы – OnCreate). Для каждого обрабатываемого 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

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.

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 посредством 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.

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

1.3. Пример написания программы

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

$$u = tg^2(x+y) * \frac{|e^{3y} - x^2|}{\sqrt{arctg(z) + \ln(x)}}.$$

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 свертывания формы в пиктограмму, для разворачивания формы на весь экран и возвращения к исходному размеру и для закрытия формы. С помощью мыши, «захватывая» одну из кромок формы или выделенную строку заголовка, отрегулируйте нужные размеры формы и ее положение на экране.

1.3.2. Изменение заголовка формы

Новая форма имеет одинаковые имя (Name) и заголовок (Caption) – Form1. Для изменения заголовка вызовите окно инспектора объектов и щелкните кнопкой мыши по форме. На странице Properties инспектора объектов найдите свойство Caption и в правой ячейке наберите «Лаб. раб. N1. Ст. гр. 740102 Иванов А.А.».

1.3.3. Размещение строки ввода (TEdit)

Для ввода данных, а также вывода информации, которая вмещается в одну строку, используется однострочное окно редактирования (компонент TEdit). Доступ к отображаемой в окне информации в виде строки из символов (тип String) осуществляется с помощь свойства Text.

В составляемой ниже программе с помощью компонентов TEdit будут вводиться значения переменных x, y, z (см. рис. 1.2).

Выберите в меню компонентов Standard пиктограмму  и щелкните мышью в том месте формы, где вы хотите ее поставить. Поместите три компонента TEdit в форму, в тексте программы (см. листинг 1.1) появится три новые переменные – Edit1, Edit2, Edit3. Захватывая компоненты мышью, отрегулируйте размеры окон и их положение. С помощью инспектора объектов установите шрифт и размер символов, отражаемых в строке Edit (свойство Font).

На этапе написания программы следует обратить внимание на то, что численные значения переменных x, y, z имеют действительный тип, а компонент TEdit в переменной Text содержит отображаемую в окне строку символов. Для преобразования строковой записи числа, находящегося в переменной Edit.Text, в действительное его представление, надо использовать стандартную функцию x:=**StrToFloat**(Edit1.Text). Если исходные данные имеют целочисленный тип, например integer, то используется стандартная функция **StrToInt**. При этом в записи числа не должно быть пробелов, а целая и дробная часть действительного числа разделяются символом, заданным в разделе «Языки и стандарты» панели управления Windows (по умолчанию – запятой). Некоторые процедуры и функции для преобразования строк приведены в прил. 1 и теме 7.

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  и щелкните мышью в нужном месте формы (появится надпись Label1). Проделайте это для четырех надписей (в тексте программы автоматически появятся четыре новые переменные типа TLabel). Для каждой надписи, щелкнув на ней мышью, отрегулируйте размер и положение на форме. В свойство Caption введите строку, например «Введите значение X:», а также выберите размер символов (свойство Font).

1.3.5. Размещение многострочного окна вывода (TMemo)

Для вывода результатов работы программы в виде отчета, содержащего несколько строк текста, обычно используется текстовое окно (компонент TMemo).

Выберите в меню компонентов пиктограмму  и поместите компонент TMemo на форму. В тексте программы появилась переменная. С помощью мыши отрегулируйте размеры и местоположение Memo1. Для отображения вертикальной и горизонтальной полос прокрутки на странице Properties инспектора объектов установите свойство ScrollBars в положение SSBoth.

Информация, которая отображается построчно в окно типа TMemo, находится в свойстве Memo1.Lines. Новая строка добавляется методом Memo1.Lines.Add (переменная типа String). Для чистки окна во время выполнения программы используется метод Memo1.Clear.

Если выводятся данные, находящиеся в переменных действительного или целого типа, то их надо предварительно преобразовать к типу String. 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 действительного типа, то при использовании метода Memo1.Lines.Add(‘Значение u=’+FloatToStrF(u,fffixed,8,2)) будет выведена строка «Значение u= -256,39». При этом под все число отводится восемь позиций, из которых две позиции занимает его дробная часть.

1.3.6. Написание программы обработки события создания формы (FormCreate)

После запуска программы на некотором этапе ее выполнения происходит создание спроектированной формы (событие OnCreate). Создадим подпрограмму – обработчик этого события (TForm1.FormCreate). Она заносит начальные значения переменных x, y, z в соответствующие окна TEdit, а окно TMemo очищает и помещает в него строку с указанием номера группы и фамилию студента. Для этого дважды щелкнем мышью в 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 (см. пример, расположенный ниже).

1.3.7. Написание программы обработки события нажатия кнопки (ButtonClick)

Поместите на форму кнопку (компонент TButton), для чего необходимо выбрать в меню компонентов Standart пиктограмму  . С помощью инспектора объектов измените заголовок (Caption) – Button1 на слово «Выполнить» или другое по вашему желанию. Отрегулируйте положение и размер кнопки.

После этого два раза щелкните мышью по кнопке, появится текст подпрограммы с заголовком процедуры обработчика события «щелчок мышью по кнопке» (**Procedure TForm1.ButtonClick(Sender:TObject**);).

Наберите текст этой процедуры (см. листинг 1).

Внимание! Заголовки процедур ButtonClick и FormCreate создаются средой Delphi автоматически (если набрать их вручную – программа работать не будет). При запуске программы на выполнение все функции обработки событий, у которых между begin и end не было написано текста, удаляются автоматически по соответствующему запросу среды Delphi. Поэтому не следует вручную удалять ошибочно созданные обработчики.

1.3.8. Запуск и работа с программой

Запустить программу можно, выбрав в главном меню пункт Run – Run или нажав клавишу F9, или щелкнув мышью по пиктограмме ▶. При этом происходит трансляция и, если нет ошибок, компоновка программы и создание единого загружаемого файла с расширением *exe*. На экране появляется активная форма программы (см. рис.1.2).

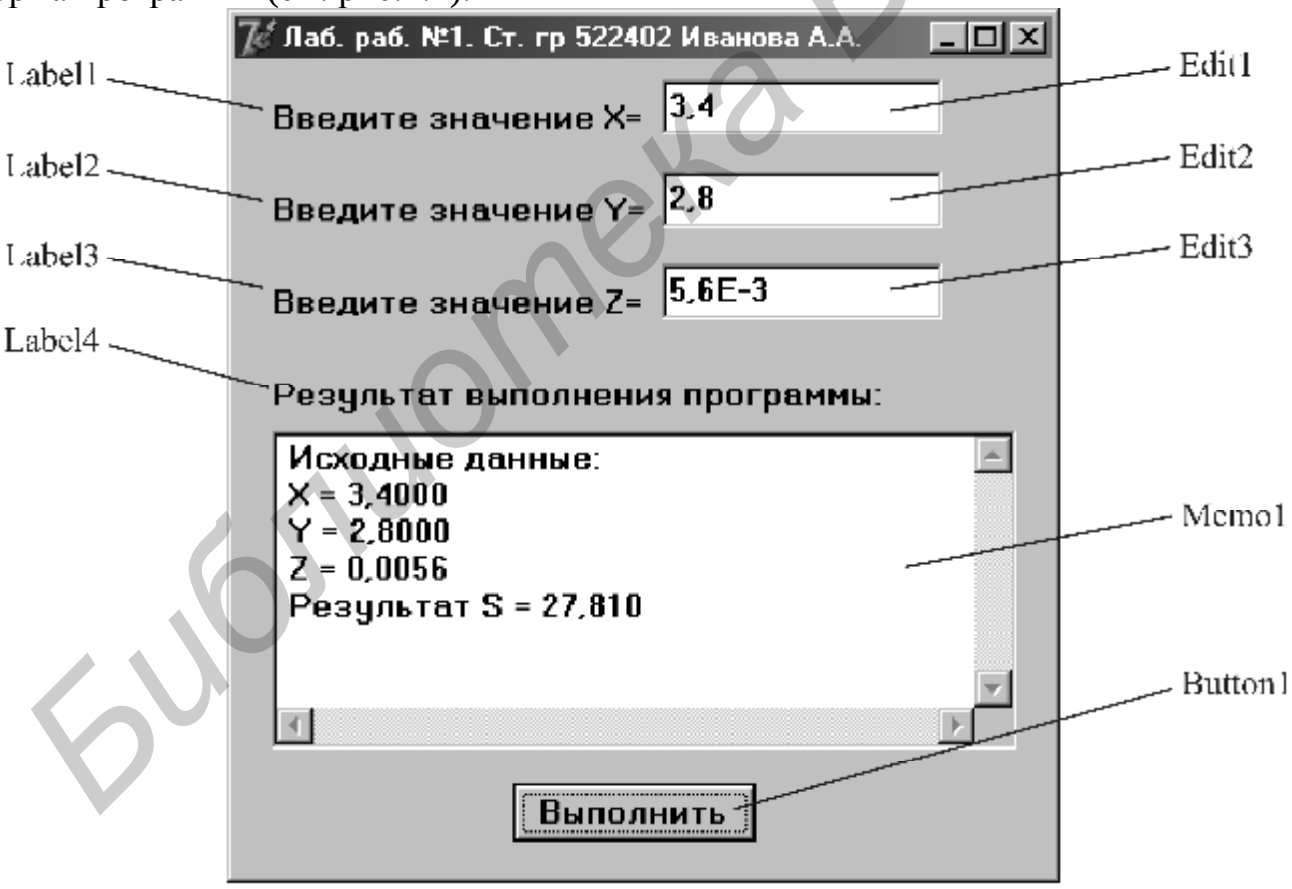

Рис.1.2

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 перейдя в окно DELPHI и выбрав в главном меню

пункт Run – ProgramReset. Последний способ выхода из программы обычно используют в случае ее зацикливания.

В листинге 1.1 представлен текст программы. Для наглядности операторы, которые следует набрать, выделены жирным шрифтом, остальные операторы вставляются средой Delphi автоматически.

Листинг 1.1

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, 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:='2,8';       // Начальное значение y
      Edit3.Text:='5,6E-3';  // Начальное значение z
      Memo1.Clear;          // Очистка окна редактора Memo1
end;

procedure TForm1.Button1Click(Sender: TObject);
   var x,y,z,a,b,c,s : extended;
begin
 Memo1.Clear;          // Очистка окна редактора Memo1
 Memo1.Lines.Add(' Исходные данные:'); // Вывод строки в  Memo1
    x:=StrToFloat(Edit1.Text);         // Считывается значение X
```

```
Memo1.Lines.Add(' X = '+FloatToStrF(x,fffixed,8,4));
   y:=StrToFloat(Edit2.Text);        // Считывается значение Y
Memo1.Lines.Add(' Y = '+FloatToStrF(y,fffixed,8,4));
   z:=StrToFloat(Edit3.Text);        // Считывается значение Z
 Memo1.Lines.Add(' Z = '+FloatToStrF(z,fffixed,8,4));
  // Вычисляем арифметическое выражение
   a:=Sqr(Sin(x+y)/Cos(x+y));
   b:=abs(exp(3*y)-sqr(x));
   c:=sqrt(arctan(z)+ln(x));
    s:=a*b/c;
   // Выводим результат в окно Memo1
  Memo1.Lines.Add(' Результат S = '+FloatToStrF(s,fffixed,8,3));
end;
end.
```

1.4. Индивидуальные задания

По указанию преподавателя выберите индивидуальное задание. Уточните условие задания, количество, наименование, типы исходных данных. Нарисуйте схему алгоритма, разбив выражение на части. Установите необходимое количество окон Edit, меток label. Выберите необходимые типы переменных и функции их преобразования при вводе и выводе данных. Используйте прил. 2.

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

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

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

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

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

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

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

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

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

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

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

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

При x=16.55×10^{-3}, y=-2.75, z=0.15 b =-40.63069.

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

При x=0.1722, y=6.33, z=3.25×10^{-4} g =-205.306.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

ТЕМА 2. ОБРАБОТКА СОБЫТИЙ В СРЕДЕ DELPHI. ПРОГРАММИРОВАНИЕ РАЗВЕТВЛЯЮЩИХСЯ АЛГОРИТМОВ

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

2.1. Обработка событий

Обо всех происходящих в системе событиях, таких, как создание формы, нажатие кнопки мыши или клавиатуры и т.д., ядро системы Windows информирует работающие программы путем посылки соответствующих сообщений. Среда DELPHI позволяет принимать и обрабатывать большинство таких сообщений. Каждый компонент содержит относящиеся к нему 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. 2.1.

Таблица 2.1

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, обычно – установку начальных значений в окнах формы
OnKeyPress	Возникает при нажатии кнопки на клавиатуре. Параметр Key имеет тип Char и содержит ASCII-код нажатой клавиши (клавиша Enter клавиатуры имеет код #13, клавиша Esc – #27 и т.д.)
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

2.2. 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, 5:    u:=x-y;
        2, 4, 6: u:=x*y;
        else  u=0;
end;
```

В соответствии со значением in вычисляется u. Если in=0, то u=x+y, если in=1 или 5, то u=x-y, если in=2 или 4 или 6, то u=x*y и, наконец, u=0 при любых значениях in отличных от 0, 1, 2, 4, 5, 6.

2.3. 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 логической 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.

2.4. Пример написания программы

Задание: ввести три числа - x, y, z. Вычислить по усмотрению u=sin(x) или $u=x^2$, или $u=e^x$. Найти максимальное из трех чисел: u, y ,z. Предусмотреть возможность округления результата до целого.

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

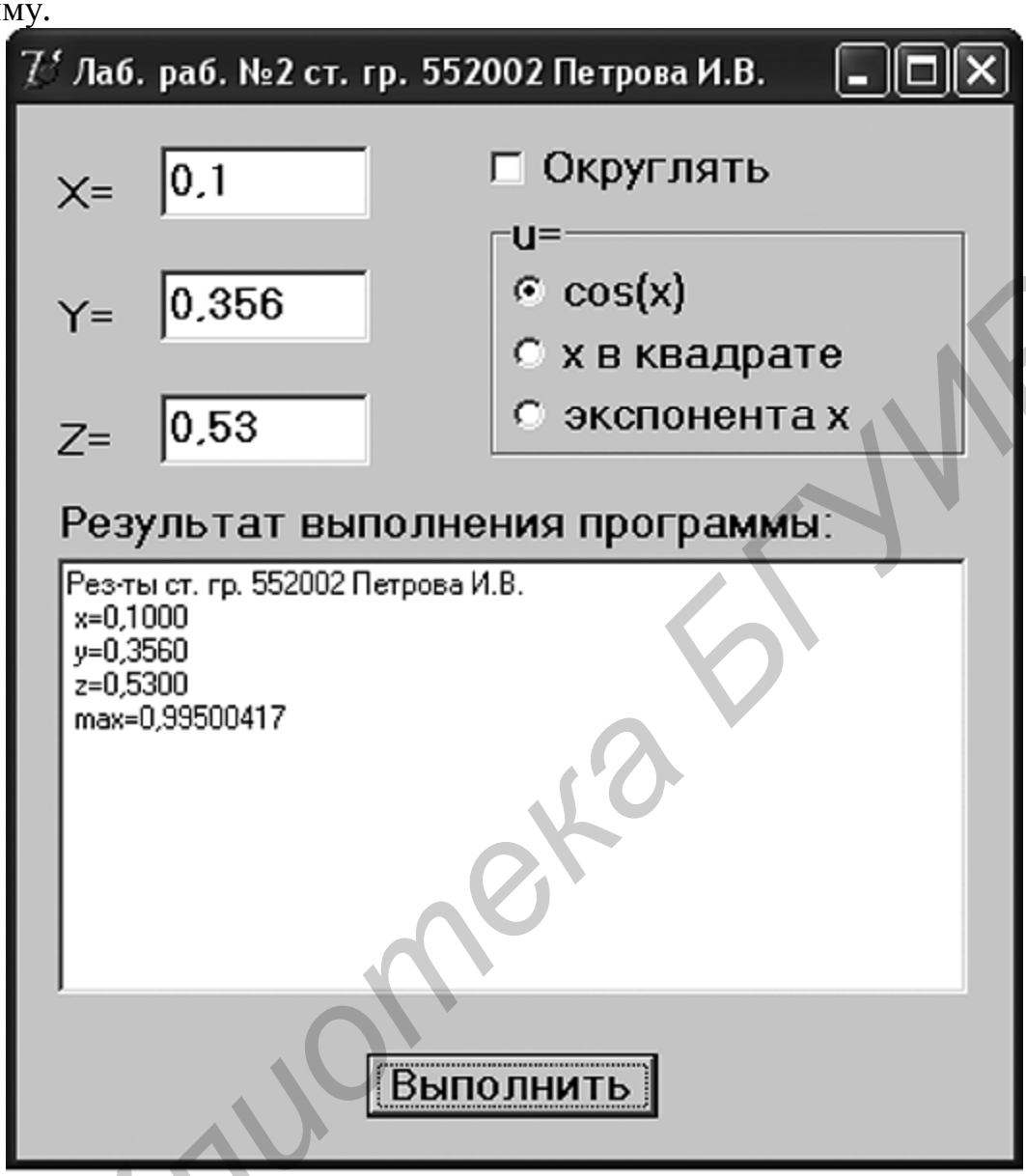

Рис. 2.1

2.4.1. Создание формы

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

2.4.2. 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 «Округлять». В тексте программы появилась переменная CheckBox1 типа TCheckBox. Теперь в зависимости от того, нажата или нет кнопка, логическая переменная **CheckBox1.Checked** будет принимать значения true или false.

2.4.3. 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=». 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 – «x в квадрате», в третьей – «экспонента x», нажмите ОК.

После этого на форме внутри окаймления появятся 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. Для того чтобы при запуске программы была выбрана первая кнопка, измените значение свойства ItemIndex на 0.

Форма приведена на рис. 2.1. Текст программы приведен на листинге 2.1.

Листинг 2.1

```
unit Unit2;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    CheckBox1: TCheckBox;
    RadioGroup1: TRadioGroup;
    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:='0,1';
  Edit2.text:='0,356';
  Edit3.text:='0,53';
  Memo1.Clear;
  Memo1.Lines.Add('Рез-ты ст. гр. 552002 Петрова И.В.');
  RadioGroup1.ItemIndex:=0;    // Активна первая кнопка RadioGroup1
end;

procedure TForm1.Button1Click(Sender: TObject);
 var x,y,z,u,ma:extended;
begin
      // Ввод исходных данных
     x:=StrToFloat(Edit1.Text);
     y:=StrToFloat(Edit2.Text);
     z:=StrToFloat(Edit3.Text);
       // Вывод введенных исходных данных
     Memo1.Lines.Add(' x='+FloatToStrF(x,ffFixed,8,4));
     Memo1.Lines.Add(' y='+FloatToStrF(y,ffFixed,8,4));
     Memo1.Lines.Add(' z='+FloatToStrF(z,ffFixed,8,4));
   // Проверка номера нажатой кнопки и выбор функции
     case RadioGroup1.ItemIndex of
        0: u:=cos(x);
        1: u:=sqr(x);
        2: u:=exp(x);
       end;
      // Нахождение максимального из трех чисел
    if u>y then ma:=u else ma:=y;
    if z>ma then ma:=z;
      // Вывод результата
 if CheckBox1.Checked then  // Проверка состояния кнопки CheckBox1
       Memo1.Lines.Add(' max='+IntToStr(Round(ma))
                          else
      Memo1.Lines.Add(' max='+FloatToStrF(ma,ffGeneral,8,2));
end;
end.
```

2.5. Выполнение индивидуального задания

По указанию преподавателя выберите индивидуальное задание. Нарисуйте схему алгоритма. В качестве f(x) использовать по выбору: sh(x) или x^2, или e^x. Отредактируйте вид формы и текст программы в соответствии с полученным

заданием. Используя теорию подразд. 2.1, создайте вместо обработчика Button1.Click обработчик Memo1Click или Label1DblClick

.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

ТЕМА 3. СРЕДСТВА ОТЛАДКИ ПРОГРАММ В СРЕДЕ DELPHI. ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ

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

3.1. 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 мышью на строке с ее описанием. Для получения более полной информации о характере ошибки необходимо обратиться к HELP, нажав клавишу F1. Следует обратить внимание на то, что одна ошибка может повлечь за собой другие, которые исчезнут при ее исправлении. Поэтому надо 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) или щелкнуть по серой полосе слева от оператора для обозначения точки прерывания (появится красная точка) и нажать клавишу F9. Выполнение программы будет остановлено на указанной строке. Для просмотра текущих значений можно поместить на нужную переменную курсор (на экране будет высвечено ее значение) либо нажать Ctrl-F7 (окно оценки и модификации) или Ctrl-F5 (окно наблюдения) и в появившимся диалоговом окне указать интере-

сующую переменную. 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.

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

Под циклом понимается многократное выполнение одних и тех же операторов при различных значениях промежуточных данных. Число повторений может быть задано в явной или неявной форме. Для организации повторений в языке Pascal предусмотрены три различных оператора цикла.

Оператор

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

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

Оператор

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

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

Оператор

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

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

Для прекращения выполнение цикла используется процедура **Break**, которая прерывает выполнение тела любого цикла и передает управление следующему за циклом оператору. Для прерывания текущей итерации цикла и передачи управления следующей используется процедура **Continue**.

3.3. Пример написания программы

Задание: написать и отладить программу, которая выводит таблицу значений функции $y(x)=e^{-x}$ и ее разложения в ряд $s(x)=\sum_{k=0}^{\infty}a^k=\sum_{k=0}^{\infty}(-1)^k\frac{x^k}{k!}$ для x

изменяющихся в интервале от x_N до x_K с шагом h. Функцию s(x) вычислять с точностью до 0,001. Вывести число итераций, необходимое для достижения заданной точности.

При составлении алгоритма вычисления удобно использовать рекуррентную последовательность (такую последовательность, каждое новое слагаемое которой зависит от одного или нескольких предыдущих). Для получения расчетной формулы рассмотрим значение слагаемого при различных значениях k: при $k=0;\ a_0=1\frac{1}{1};$ при $k=1;\ a_1=-1\frac{x}{1};$ при $k=2;\ a_2=1\frac{x\cdot x}{1\cdot 2};$ при $k=3;\ a_3=-1\frac{x\cdot x\cdot x}{1\cdot 2\cdot 3}$ и т.д. Видно, что на каждом шаге слагаемое дополнительно умножается на $-1\frac{x}{k}$. Исходя из этого формула рекуррентной последовательности будет иметь вид $a_k=-a_{k-1}\frac{x}{k}$. Полученная формула позволяет избавиться от многократного вычисления факториала и возведения в степень. Если в выражении имеется нерекуррентая часть, то ее следует рассчитывать отдельно.

Панель диалога представлена на рис. 3.1.

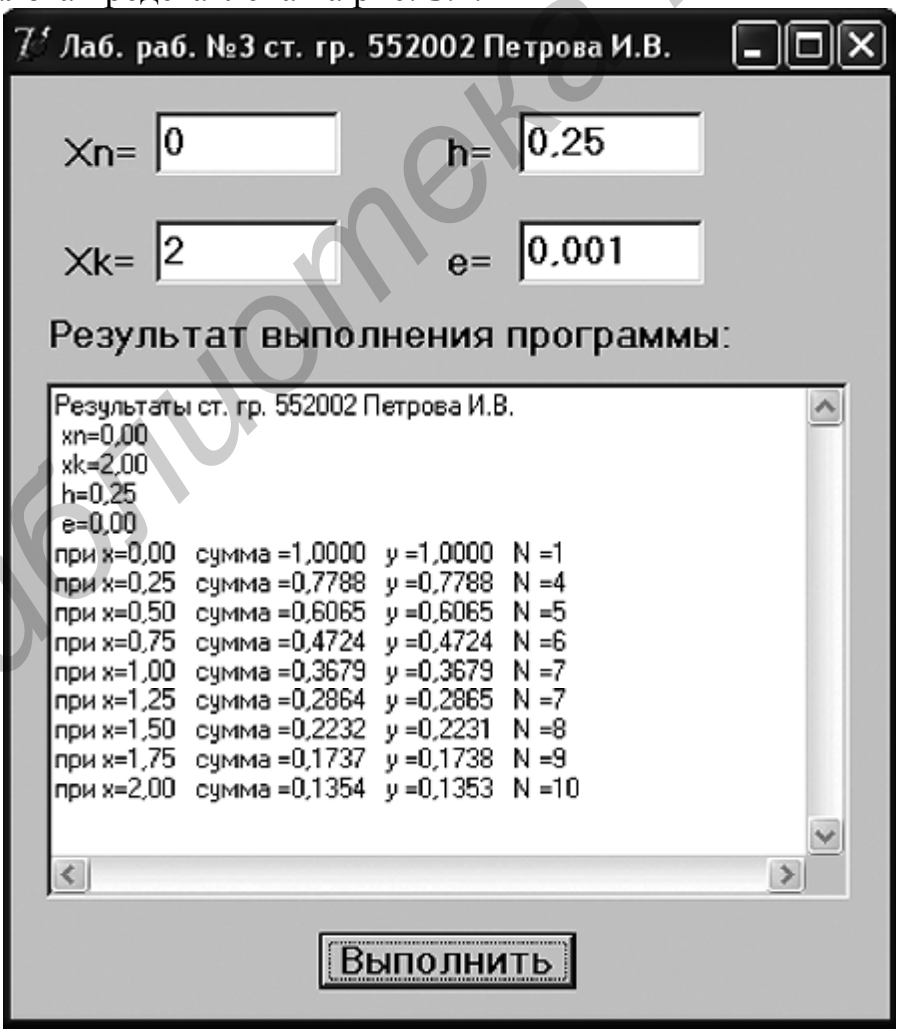

Рис. 3.1

Текст программы приведен на листинге 3.1.

Листинг 3.1

```
unit Unit3;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Label2: TLabel;
    Edit2: TEdit;
    Label3: TLabel;
    Edit3: TEdit;
    Label4: TLabel;
    Memo1: TMemo;
    Button1: TButton;
    Label5: TLabel;
    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:='0,25';
  Edit4.text:='0,001';
 end;

procedure TForm1.Button1Click(Sender: TObject);
 var xn,xk,x,h,e,a,s,y :extended;
     n,k:integer;
begin
   Memo1.Clear;
   Memo1.Lines.Add('Результаты ст. гр. 552002 Петрова И.В.');
```

```
    xn:=StrToFloat(Edit1.Text);
                    Memo1.Lines.Add(' xn='+FloatToStrF(xn,ffFixed,6,2));
    xk:=StrToFloat(Edit2.Text);
                     Memo1.Lines.Add(' xk='+FloatToStrF(xk,ffFixed,6,2));
    h:=StrToFloat(Edit3.Text);
                    Memo1.Lines.Add(' h='+FloatToStrF(h,ffFixed,8,3));
    e:=StrToFloat(Edit4.Text);
                    Memo1.Lines.Add(' e='+FloatToStrF(e,ffFixed,8,5));
       x:=xn;
  repeat
     a:=1;  S:=1;  n:=0;
       while (abs(a) > e) do begin
          n:=n+1;
           a:=-a*x/n;
           s:=s+a;
         end;
           y:=exp(-x);
    Memo1.Lines.Add('при x='+FloatToStrF(x,ffFixed,6,2)+'   сумма ='
                          + FloatToStrF(s,ffFixed,8,4)+' y ='
                          + FloatToStrF(y,ffFixed,8,4)+'   N ='+IntToStr(n));
       x:=x+h;
  until x>(xk+h/2)       // (xk+h/2) применяется для исключения
end;                     // потери последнего x

end.
```

3.4. Выполнение индивидуального задания

По указанию преподавателя выберите вариант задачи. Нарисуйте схему алгоритма. Спроектируйте панель диалога и напишите текст программы.

Вывести на экран таблицу значений функции *Y(x)* и ее разложения в ряд *S(x)* для *x*, изменяющихся от x_n до x_k с заданным количеством шагов *M* ($h = \frac{x_k - x_n}{M}$) и точностью *e*. Близость значений *S(x)* и *Y(x)* во всем диапазоне значений x указывает на правильность вычисления *S(x)* и *Y(x)*.

После написания программы и исправления ошибок трансляции изучите средства отладки программ, для чего установите курсор на первый оператор и нажмите клавишу F4. После этого, нажимая клавишу F7, выполните пошагово программу и проследите, как меняются все переменные в процессе выполнения.

Таблица 3.1

№	x_n	x_k	*S(x)*	*e*	*Y(x)*
1	2	3	4	5	6
1	0.1	1	$\sum_{n=0}^{\infty}(-1)^n \frac{x^{2n+1}}{(2n+1)!}$	0.001	$\sin x$

Окончание табл. 3.1

1	2	3	4	5	6
2	0.1	1	$\sum_{n=0}^{\infty}\frac{x^{2n}}{(2n)!}$	0.0001	$\frac{e^{x}+e^{-x}}{2}$
3	0.1	1	$\sum_{n=0}^{\infty}\frac{\cos n\frac{p}{4}}{n!}x^{n}$	0.001	$e^{x\cos\frac{p}{4}}\cos(x\sin\frac{p}{4})$
4	0.1	1	$\sum_{n=0}^{\infty}(-1)^{n}\frac{x^{2n}}{(2n)!}$	0.01	$\cos x$
5	0.1	0.7	$\sum_{n=0}^{\infty}\frac{2n+1}{n!}x^{2n}$	0.001	$(1+2x^{2})e^{x^{2}}$
6	0.1	1	$\sum_{n=0}^{\infty}\frac{x^{2n+1}}{(2n+1)!}$	0.0001	$\frac{e^{x}-e^{-x}}{2}$
7	0.2	1	$\sum_{n=0}^{\infty}\frac{(\ln 9)^{n}}{n!}x^{n}$	0.001	9^{x}
8	0.1	0.7	$\sum_{n=0}^{\infty}\frac{(2x)^{n}}{n!}$	0.0001	e^{2x}
9	0.3	1	$\sum_{n=0}^{\infty}\frac{n^{2}+1}{n!}\left(\frac{x}{2}\right)^{n}$	0.001	$\left(\frac{x^{2}}{4}+\frac{x}{2}+1\right)e^{\frac{x}{2}}$
10	0.1	0.5	$\sum_{n=0}^{\infty}(-1)^{n}\frac{x^{2n+1}}{2n+1}$	0.0001	$arctgx$
11	0.2	1	$\sum_{n=0}^{\infty}(-1)^{n}\frac{2n^{2}+1}{(2n)!}x^{2n}$	0.001	$\left(1-\frac{x^{2}}{2}\right)\cos x-\frac{x}{2}\sin x$
12	0.1	1	$\sum_{n=1}^{\infty}(-1)^{n}\frac{(2x)^{2n}}{(2n)!}$	0.0001	$2(\cos^{2}x-1)$
13	-2	-0.1	$\sum_{n=0}^{\infty}(-1)^{n}\frac{x^{2n}}{(2n+1)!}$	0.01	$\frac{\sin(x)}{x}$
14	0.2	0.8	$\sum_{n=1}^{\infty}\frac{n^{2}}{(2n+1)!}x^{n}$	0.0001	$\frac{1}{4}\left(\frac{x+1}{\sqrt{x}}sh\sqrt{x}-ch\sqrt{x}\right)$
15	0.1	0.8	$\sum_{n=1}^{\infty}(-1)^{n+1}\frac{x^{2n}}{2n(2n-1)}$	0.001	$xarctgx-\ln\sqrt{1+x^{2}}$

ТЕМА 4. ОБРАБОТКА ИСКЛЮЧИТЕЛЬНЫХ СИТУАЦИЙ. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ МАССИВОВ

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

4.1. Обработка 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 (метод **TApplication.HandleException**), вызываемый по умолчанию, информирует пользователя о возникновении ошибки и завершает выполнение программы. Для защиты от завершения в языке Object Pascal используется оператор **try**, который перехватывает исключительную ситуацию и дает возможность разработчику предусмотреть определенные действия при ее возникновении.

Конструкция блока **try… finally**:

```
try
        <операторы, выполнение которых может привести
        к возникновению исключительной ситуации>
finally
        <операторы, выполняемые всегда, вне зависимости
        от возникновения исключительной ситуации>
end;
```

При возникновении исключительной ситуации в одном из операторов управление сразу передается первому оператору блока **finally**.

Данная конструкция позволяет корректно завершить выполнение программы вне зависимости от возникающей исключительной ситуации. Обычно в блок **finally** помещают операторы, закрывающие открытые файлы, освобождающие выделенную динамическую память. Недостатком такой конструкции является то, что программа не информирует о том, возникала ли исключительная ситуация и, следовательно, не позволяет пользователю ее скорректировать.

Конструкция блока **try…except**:

```
try
        <операторы, выполнение которых может привести
        к возникновению исключительной ситуации>
except
        <операторы, выполняемые только в случае
        возникновения исключительной ситуации>
end;
```

При возникновении исключительной ситуации управление сразу передается в блок **except,** в противном случае блок **except** пропускается. Такая конструкция позволяет определить причину возникшей проблемы и рекомендовать пользователю определенные действия для ее исправления. В простейшем случае в разделе **except** пишутся операторы, выполняемые при возникновении любой исключительной ситуации. Для определения типа возникшей ошибки в разделе **except** используется конструкция, работающая по схеме оператора **case**:

on <тип исключительной ситуации 1> **do** <оператор 1>;
on <тип исключительной ситуации 2> **do** <оператор 2>;
…
else <операторы, выполняемые, если не определен
тип исключительной ситуации >;

Выполняется только оператор, стоящий после **do**, для реализуемой исключительной ситуации. Некоторые из возможных типов исключительных ситуаций представлены в табл. 4.1.

Таблица 4.1

Тип исключительной ситуации	Причина
EAbort	Намеренное прерывания программы, генерируемое процедурой **Abort**
EArrayError	Ошибка при операциях с массивами: использование ошибочного индекса массива, добавление слишком большого числа элементов в массив фиксированной длины (для использования требует подключения модуля mxarrays)
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+ })
EInvalidArgument	Недопустимое значение параметра при обращении к математической функции
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+ })

Для отладки программы, содержащей обработку исключительных ситуаций, надо отключить опцию **Stop on Delphi Exceptions**, находящуюся в **Tools – Debbuger Options …** , закладка **Language Exceptions (**для Delphi 3 надо отключить опцию **Break on exception,** находящуюся в **Tools – Environment Options,** закладка **Perferences)**.

Возникновение исключительной ситуации может быть инициировано преднамеренно. Для этого можно использовать процедуры **Abort**, **Assert (b : Boolean)**, а также с ключевое слово **raise**:

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

4.2. Использование функций ShowMessage и MessageDlg

Для вывода сообщений полезно использовать функции ShowMessage и MessageDlg. **Функция ShowMessage(Msg: string)** отображает диалоговое окно с заданным в Msg сообщением и кнопкой OK для закрытия окна. В заголовке окна отображается имя выполняемой программы. Функция **MessageDlg(const Msg: WideString; DlgType: TMsgDlgType; Buttons: TMsgDlgButtons; HelpCtx: Longint): Word** отображает диалоговое окно с заданными кнопками. Параметр Msg содержит текст сообщения. Параметр DlgType определяет вид отображаемого окна (табл. 4.2).

Таблица 4.2

mtWarning	Заголовок: «Warning». Знак: желтый треугольник с восклицательным знаком внутри
mtError	Заголовок: «Error». Знак: красный круг с перечеркиванием внутри
mtInformation	Заголовок: «Information». Знак: символ «i» на голубом поле
mtConfirmation	Заголовок: «Confirmation». Знак: символ «?» на зеленом поле
mtCustom	Заголовок соответствует имени выполняемого файла. Без знака

Параметр Buttons указывает, какие кнопки будут находиться в окне (табл. 4.3). Список необходимых кнопок заключается в квадратные скобки.

Таблица 4.3

mbYes	Кнопка «Yes»	mbRetry	Кнопка «Retry»
mbNo	Кнопка «No»	mbIgnore	Кнопка «Ignore»
mbOK	Кнопка «OK»	mbAll	Кнопка «All»
mbCancel	Кнопка «Cancel»	mbHelp	Кнопка «Help»
mbAbort	Кнопка «Abort»		

Параметр HelpCtx определяет номер контекстной справки для данного окна.

Результатом выполнения функции является значение, соответствующее нажатой кнопке. Возвращаемое значение имеет имя, состоящее из букв mr и имени кнопки, например: mrYes, mrOK, mr Help.

4.3. 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 определяется 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 word;  // 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 – двухмерный массив символьного типа.
```

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

```
F:=2*a[3]+a[ss[4]+1]*3;
a[n]:=1+sqrt(abs(a[n-1]));
```

4.4. 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 двухмерной таблицы, каждая ячейка которой представляет собой окно однострочного редактора (аналогично окну TEdit). Доступ к информации осуществляется с помощью свойства Cells[ACol : Integer; ARow : Integer] : String, где ACol, ARow - индексы элемента двухмерного массива. Свойства ColCount и RowCount устанавливают количество столбцов и строк в таблице, а свойства FixedCols и FixedRows задают количество столбцов и строк фиксированной зоны. Фиксированная зона выделена другим цветом, и в нее запрещен ввод информации с клавиатуры.

4.5. Пример написания программы

Задание: создать программу для определения вектора $\vec{Y} = A * \vec{B}$, где A – квадратная матрица размером NxN, а $\vec{Y}$, $\vec{B}$ – одномерные массивы размерностью N.

Элементы вектора $\vec{Y}$ определяются по формуле $Y_i = \sum_{j=1}^{N} A_{ij} * B_j$. Значения N вводить в компонент TEdit, A и B – в компонент TStringGrid. Результат – после нажатия кнопки типа TButton, вывести в компонент TStringGrid.

4.5.1. 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 4 (четыре строки и четыре столбца), а FixedCols и FixedRows установите 1 (один столбец и одна строка с фиксированной зоной). Так как компоненты StringGrid2 и StringGrid3 имеют только один столбец, то у них: ColCount= 1, RowCount=4, 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.

Панель диалога приведена на рис. 4.1.

Рис. 4.1

Текст программы приведен на листинге 4.1.

Листинг 4.1

```
unit Unit4;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms, Dialogs, Grids, StdCtrls, mxarrays;
type
  TForm1 = class(TForm)
```

```
    Edit1: TEdit;
    Edit2: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Button1: TButton;
    StringGrid1: TStringGrid;
    StringGrid2: TStringGrid;
    Button2: TButton;
    StringGrid3: TStringGrid;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

const
  Nmax=10;                  // Максимальный размер массива
Type
Mas2 = array[1..Nmax,1..Nmax] of extended;      // Объявление типа
                                  //двухмерного массива размерностью Nmax
Mas1 = array[1..Nmax] of extended;        // Объявление типа
                                  //одномерного массива размерностью Nmax
var
  Form1: TForm1;
  A : Mas2;                       // Объявление двухмерного массива
  B, Y : Mas1;                    // Объявление одномерных массивов
  N, M, i, j : integer;

implementation
{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
  begin
  N:=3;       // Число строк в массиве
  M:=3;       // Число столбцов в массиве
      Edit1.Text:=FloatToStr(N);
      Edit2.Text:=FloatToStr(M);
 {Задание числа строк и столбцов в таблицах}
   StringGrid1.RowCount:=N+1;
   StringGrid1.ColCount:=M+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);   // Изменить размер
 begin                                            // таблицы
    N:=StrToInt(Edit1.Text);
    M:=StrToInt(Edit2.Text);
 {Задание числа строк и столбцов в таблицах}
   StringGrid1.RowCount:=N+1;
   StringGrid1.ColCount:=M+1;
   StringGrid2.RowCount:=N+1;
   StringGrid3.RowCount:=N+1;
 {Заполнение верхнего и левого столбцов поясняющими подписями}
   for i:=1 to N do StringGrid1.Cells[0,i]:=' i= '+IntToStr(i);
   for i:=1 to M do StringGrid1.Cells[i,0]:=' j= '+IntToStr(i);
end;

procedure TForm1.Button2Click(Sender: TObject);
 begin
 {$R+}
     try
  {Заполнение массива A элементами из таблицы StringGrid1}
  for i:=1 to N do
   for j:=1 to M 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]);
     except
      on ERangeError do begin ShowMessage('Выход за пределы
              массива. Уменьшите размер массива'); Exit; end;
      on EConvertError do begin ShowMessage('В ячейке отсутствует
                значение либо число введено неправильно'); Exit; end;
       else begin ShowMessage('Возникла неизвестная
              исключительная ситуация!'); Exit; end;
     end;

      try
    {Умножение массива A на массив B}
  for i:=1 to N do begin
      Y[i]:=0;
```

```
  for j:=1 to M do Y[i]:=Y[i]+A[i,j]*B[j];
              end;
     except
      on EInvalidOp do begin MessageDlg('Неправильная операция с плавающей точкой',mtError,[mbCancel],0);Exit; end;
      on EOverFlow do begin MessageDlg('Переполнение при выполнении операции с плавающей точкой',mtError,[mbCancel],0); Exit; end;
       else begin MessageDlg('Возникла неизвестная исключительная ситуация! ',mtError,[mbCancel],0); Exit; end;
     end;
    {Вывод результата в таблицу StringGrid3}
  for i:=1 to N do StringGrid3.Cells[0,i]:=FloatToStrF(y[i],fffixed,6,0);
end;
end.
```

4.6. Выполнение индивидуального задания

По указанию преподавателя выберите вариант задачи. Нарисуйте схему алгоритма. Во всех заданиях 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. В местах возможного возникновения ошибок использовать конструкции для обработки исключительных ситуаций.

1. Задана матрица размером *NxM*. Получить массив *B*, присвоив его *k*-му элементу значение 0, если все элементы *k*-го столбца матрицы нулевые, и значение 1 в противном случае.

2. Задана матрица размером *NxM*. Получить массив *B*, присвоив его *k*-му элементу значение 1, если элементы *k*–й строки матрицы упорядочены по убыванию, и значение 0 в противном случае.

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

4. Задана матрица размером *NxM*. Определить количество «особых» элементов матрицы, считая элемент «особым», если он больше суммы остальных элементов своего столбца.

5. Задана матрица размером *NxM*. Определить количество «особых» элементов матрицы, считая элемент «особым», если все элементы строки, находящиеся слева от него, меньше его, а справа – больше.

6. Дана матрица размером *NxM*. Упорядочить ее строки по возрастанию их первых элементов.

7. Дана матрица размером *NxM*. Упорядочить ее строки по возрастанию суммы их элементов.

8. Дана матрица размером *NxM*. Упорядочить ее строки по возрастанию их наибольших элементов.

9. Определить, является ли заданная квадратная матрица *n*-го порядка симметричной относительно побочной диагонали.

10. Задана матрица *A* размером *NxM*. Получить массив *B*, присвоив его *k*-му элементу значение максимального элемента в *k*–м столбце матрицы *A*.

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

12. В матрице размером *NxM* поменять местами строку, содержащую элемент с наибольшим значением, со строкой, содержащей элемент с наименьшим значением.

13. 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.

14. В матрице *n*-го порядка найти сумму элементов, лежащих выше побочной диагонали, и произведение элементов, лежащих ниже главной диагонали.

15. Дана матрица размером *NxM*. Поменять местами все четные и нечетные строки матрицы.

ТЕМА 5. УКАЗАТЕЛИ И ИХ ИСПОЛЬЗОВАНИЕ ПРИ РАБОТЕ С ДИНАМИЧЕСКИМИ МАССИВАМИ

Цель лабораторной работы: изучить способы работы с динамическими массивами данных.

5.1. Динамическое распределение памяти

В языке Паскаль наряду с обычным статическим возможна организация динамического распределения памяти, при которой оперативная память для размещения данных выделяется непосредственно во время выполнения программы по мере надобности. Если переменная, соответствующая этим данным, становится ненужной, то она удаляется, а выделенная под нее память освобождается.

Обеспечение работы с динамическими данными реализуется с помощью переменных типа ***указатель***, которые вводятся следующим образом:

```
Type
      Pukaz=^<тип переменной>;
      Pint = ^integer;
      TMas = array[1..4] of integer
      PMas = ^TMas;
Var
      Uk : Pukaz;
      a, b : PInt;          // Типизированные указатели
      p, q : pointer;       // Нетипизированные указатели
      U : Pmas;             // Указатель на одномерный массив
      C : array[1..2,1..2] of integer; // Обычный двухмерный массив
begin
...
      U:=Addr(C);   // Указателю U присваивается
                    // адрес статической переменной C
      a:=p;         // Адрес указателя p копируется в указатель a
      p:=Nil;       // Очистка адреса
...
```

Каждая переменная типа ***указатель*** (a, b, p, q, u) занимает 4 байта памяти, и содержит адрес первого байта некоторого участка оперативной памяти, данные на котором интерпретируются в соответствии с объявленным типом. С помощью указателя (адреса) реализуется доступ к данным, расположенным в ячейке с этим адресом, например u^[2]:=3. Более того, появляется возможность интерпретации данных разного типа, например, одномерного массива как двухмерного:

```
      ...
      c[1,2]:=5;  c[2,1]:=8;
u:=@c;   // Указателю и присваивается адрес C (@ эквивалентно Addr);
```

```
Write(u[2],u[3]);       // Распечатается 5 и 8
...
```

Организация динамического распределения памяти осуществляется в свободной от загруженных программ области оперативной памяти, называемой *кучей* (heap). Для этого используются процедуры **New, Dispose** или **GetMem, FreeMem**.

Процедура

New(var P: Pointer);

выделяет участок памяти, размер которого определяется типом указателя P. Указателю P присваивается адрес первого байта этого участка. После того как необходимость работы с этой переменной отпала, данный участок памяти освобождается с помощью процедуры

Dispose(var P: Pointer);

После освобождения адрес в указателе P остается, однако доступ к ячейке с этим адресом запрещен.

Например:

```
Var   a, b : PInt;
begin
      New(a);
            b:=a;
            a^:=9;
            b^:=5
            Write(a);  // Распечатается 5
      …
      Dispose(b);
```

Здесь выделена память под переменную типа integer. Указатель b стал указывать на ту же область памяти, что и указатель a . В выделенную область сначала занесено значение, равное 9, затем туда же – значение, равное 5. Для освобождения памяти можно использовать как Dispose(b), так и Dispose(a).

Процедура **GetMem** используется при работе как с типизированными, так и с нетипизированными указателями, поэтому задается не только имя указателя, но и объем необходимой памяти в байтах:

```
GetMem(P:pointer;size:Word);
```

Процедура

```
FreeMem(P:pointer;size:Word);
```

освобождает область памяти, связанную с указателем *P* и имеющую размер, равный *size*.

Для определения размера, который требуется выделить под переменную, обычно пользуются функцией **sizeof(<имя типа>)**.

5.2. Организация динамических массивов

Обычно динамическое выделение и освобождение памяти используется при работе с большими массивами данных.

В случае, **когда максимальные размеры массива заранее известны**, динамическое распределение памяти может быть организовано следующим образом:

```
type   Tmas = array[1..1000, 1..2000] of byte;
       Pmas = ^Tmas
var    B : Pmas            // Указатель на динамический массив
begin
       New(B);             .. Выделение памяти
            <работа с массивом>
       Dispose(B);  // Освобождение памяти
```

При такой организации память выделяется нерационально и заведомо с избытком, при этом всегда нужно помнить об ограничении на индексы.

С помощью процедур Getmem и Freemem можно создавать **массивы с изменяемым размером – динамические массивы**. Для этого определим тип указателя на массив с небольшим размером, а затем выделим памяти столько, сколько необходимо:

```
type
    Tmas=array[1..1] of extended;
    Pmas=^Tmas;                          // Указатель на массив
 Var a : Pmas;
     mt, n : word;
begin
    mt:=sizeof(extended); // Определение количества байт,
                          // которое занимает указанный тип данных
  GetMem(a,mt*n);         // Выделение памяти под n чисел
          <работа с массивом размерностью n>
  FreeMem(a, mt*n);       // Освобождение памяти
```

При работе с такой программой необходимо отключать проверку выхода индекса за пределы массива и внимательно следить за тем, чтобы индекс не вышел за пределы выделенной памяти.

Организация двухмерного динамического массива реализуется следующим образом:

```
    Type
          TMas = array[1..1] of integer;
           PMas = ^TMas;
          TMas2 = array[1..1] of  PMas;
            PMas2 = ^TMas2;
  Var
          a : PMas2;
           N,M,i,j : integer;
 begin
          …
  GetMem(a,4*M);
   for i:=1 to M do GetMem(a[i],N*sizeof(integer));
          …
  // Работа с массивом a[i,j], 1≤i≤M, 1≤j≤N
```

```
        …
for i:=1 to M do FreeMem(a[i],N*sizeof(integer));
FreeMem(a,4*M);
```

5.3. 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 Clyph. Кроме того, имеется свойство Kind, которое задает одну из 11 стандартных разновидностей кнопок. Кнопка bkClose закрывает главное окно и завершает работу программы.

5.4. Пример написания программы

Задание: дан массив, состоящий из семи цифр. Упорядочить элементы массива по возрастанию. Панель диалога приведена на рис. 5.1.

Рис. 5.1

Текст программы приведен на листинге 5.1.

Листинг 5.1

```
unit Unit5;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, Grids;
type
  TForm1 = class(TForm)
    StringGrid1: TStringGrid;
    StringGrid2: TStringGrid;
    Label1: TLabel;
    Label2: TLabel;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    procedure FormCreate(Sender: TObject);
```

```
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

const N=7;

Type
        TMas = array[1..1] of integer;
          PMas = ^TMas;
var
  Form1: TForm1;
        a : PMas;
         i,j,t  :  integer;
implementation

{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
begin
        Randomize;
    for i:=1 to N do
      StringGrid1.Cells[i-1,0]:=IntToStr(Random(100));
end;

procedure TForm1.BitBtn1Click(Sender: TObject);
begin
 GetMem(a,N*sizeof(integer));

  for i:=1 to N do
      a[i]:=StrToInt(StringGrid1.Cells[i-1,0]);

  for i:=1 to N-1 do
   for j:=i+1 to N do
    if a[i]>a[j] then begin
                t:=a[i];
                a[i]:=a[j];
                a[j]:=t;
                    end;
   for i:=1 to N do
      StringGrid2.Cells[i-1,0]:=IntToStr(a[i]);
   FreeMem(a,N*sizeof(integer));
end;
end.
```

5.4. Выполнение индивидуального задания

Выполните задачу из подразд. 4.6, используя двухмерный динамический массив.

По указанию преподавателя выберите вариант задачи. Нарисуйте схему алгоритма. Выделение памяти под массив организовать динамически. Организуйте ввод данных из StringGrid1. Вывод организовать в зависимости от варианта в StringGrid2 или в Edit1.

1. Дан массив, состоящий из символов. Расположить его элементы в обратном порядке.

2. Дан массив, состоящий из символов. Преобразовать его по следующему правилу: сначала должны находиться цифры, а затем все остальные символы, сохраняя при этом взаимное расположение символов в каждой из этих двух групп.

3. Дан массив, состоящий из символов. Вывести на экран цифру, наиболее часто встречающуюся в этом массиве.

4. Дан массив, состоящий из символов. Определить количество различных элементов массива (т.е. повторяющиеся элементы считать один раз).

5. Дан массив, состоящий из символов. Элементы массива циклически сдвинуть на k позиций влево.

6. Дан массив, состоящий из символов. Элементы массива циклически сдвинуть на n позиций вправо.

7. Дан массив, состоящий из чисел. Преобразовать массив по следующему правилу: все отрицательные числа массива перенести в начало, а все остальные – в конец, сохраняя исходное взаимное расположение как среди отрицательных, так и среди положительных элементов.

8. 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.

9. Дан массив, состоящий из символов. Определить, симметричен ли он, т.е. читается ли он одинаково слева направо и справа налево.

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

11. 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 $i < j$).

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

13. Дан массив, состоящий из символов. Удалить из него повторные вхождения каждого символа.

14. Дан массив, состоящий из цифр. Удалить из него все четные числа.

15. Дан массив, состоящий из цифр. Удалить из него все отрицательные числа.

ТЕМА 6. 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.

6.1. Использование подпрограмм

Подпрограмма – это 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 имен формальных параметров с
                                    указанием их типов]);
    Const  [описание используемых констант];
    Type   [описание используемых типов];
    Var    [описание используемых переменных];
     Begin
        ...          // Операторы
End;
```

Вызов процедуры: <имя процедуры> ([список имен формальных параметров без указания их типов]);

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

```
Function <имя функции> ([список имен формальных параметров
                                с указанием их типов]): <тип результата>;
  Const [описание используемых констант];
   Type [описание используемых типов];
   Var [описание используемых переменных];
    Begin
      ...              // Операторы
  Result:= … ;// Присвоение результата вычислений переменной Result
   // или <имя функции>:= … ;
   End;                // Конец  функции
```

Вызов функции: y:=<имя функции > ([список имен формальных параметров без указания их типов]);

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

Type <имя> = **function** ([список формальных параметров]):<тип рез-тата>;
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, что требует дополнительных затрат памяти. Поэтому рекомендуется использовать параметры-константы или параметры-переменные. При использовании параметров-переменных (в описании перед ними ставится **Var**) и параметров-констант (перед ними ставится **const**) в подпрограмму передаются адреса (указатели фактических параметров) и она работает непосредственно с фактическими параметрами. Благодаря этому экономится память, а также организуется передача результата работы подпрограммы вызывающей программе через параметры-переменные. Через параметры-константы этого делать нельзя, т.к. их нельзя менять внутри подпрограммы. В качестве фактических параметров могут использоваться арифметические выражения, если формальный параметр – константа или параметр-значение.

6.2. Использование модулей

Модуль – автономно компилируемая программная единица, включающая в себя процедуры, функции, а также различные разделы описаний. Структура модуля представлена в подразд.1.2 и содержит следующие основные части: заголовок, интерфейсная часть, исполняемая, инициирующая 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.

Инициирующая часть начинается ключевым словом initialization и содержит операторы, которые исполняются перед началом выполнения основной программы (может отсутствовать).

Завершающая часть начинается ключевым словом finalization и выполняется в момент окончания работы программы (может отсутствовать).

6.3. Пример написания программы

Задание: написать программу вывода на экран таблицы функции, которую оформить в виде процедуры. В качестве функции использовать по выбору $y(x)=e^{-x}$ или $s(x)=\sum_{k=0}^{\infty}(-1)^k\frac{x^k}{k!}$. Панель диалога должна иметь вид, как на рис. 6.1.

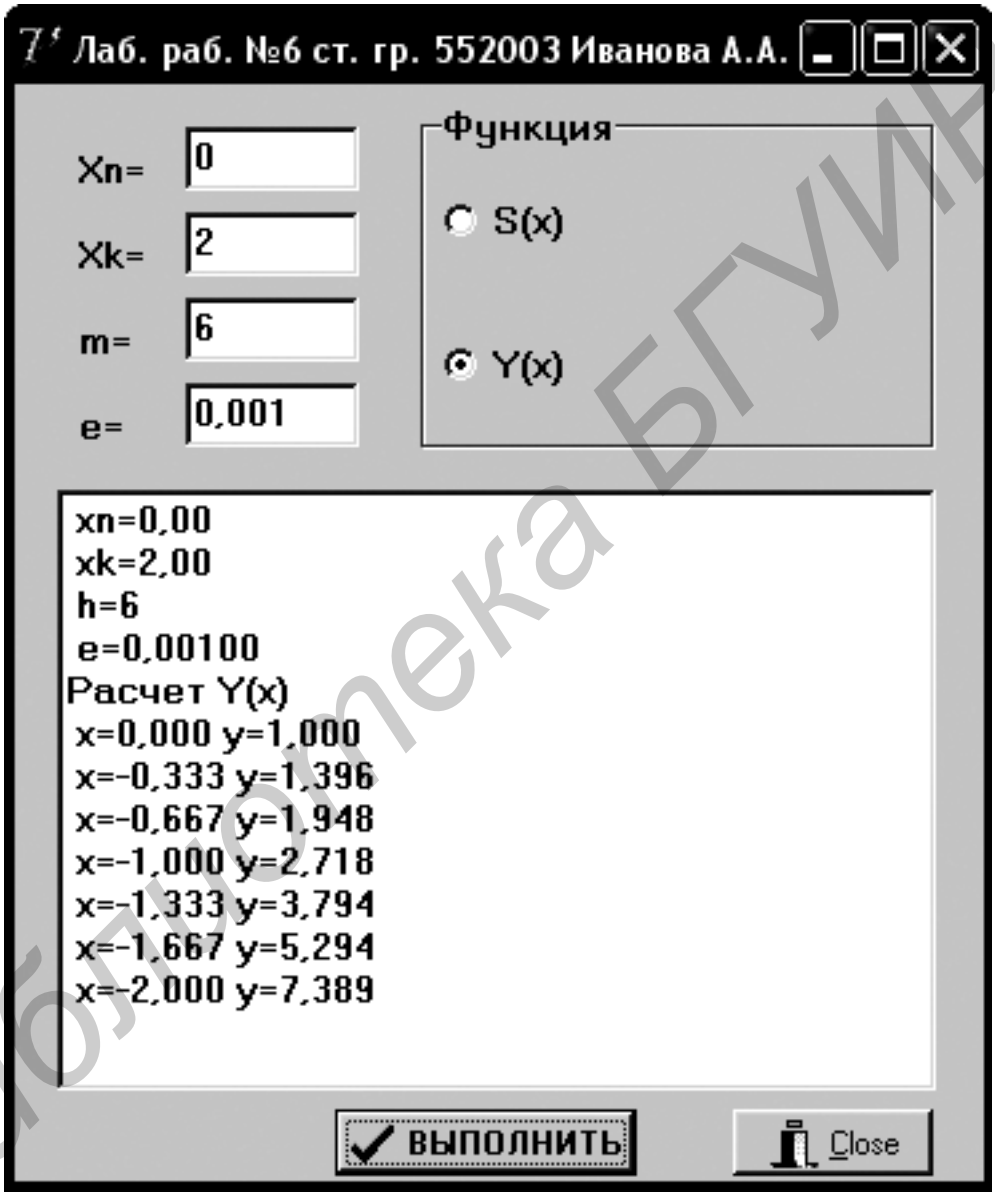

Рис. 6.1

6.3.1. Создание модуля

В среде Delphi модули могут создаваться как со своей формой, так и без нее. Для создания нового модуля без своей формы необходимо в меню File выбрать New – Unit. В результате будет создан файл с заголовком Unit Unit2. Имя модуля можно изменить на другое, отвечающее внутреннему содержанию модуля, например Unit Matfu. Для этого необходимо сохранить модуль с новым именем

(например Matfu.pas). Следует обратить внимание на то, что имя файла должно совпадать с именем модуля.

6.3.2. 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.

Тексты модуля (листинг 6.1) и вызывающей программы (листинг 6.2) приведены ниже (получение рекуррентной формулы см. подразд. 3.3).

Листинг 6.1

```
unit matfu;
interface
 uses StdCtrls, SysUtils;
 Type fun = function(x : extended):extended; // Объявление типа функция
          {Расчет таблицы функции f (табуляция функции)}
  procedure Tabl(f:fun;xn,xk:extended; m: word; Mem1 : TMemo);
implementation
  procedure Tabl;
     var x,y,h: extended;       i : integer;
    begin
    x:=xn;  h:=(xn-xk)/m;
 for i:=1 to m+1 do begin
     y:=f(x);
Mem1.Lines.Add('x='+FloatToStrf(x,fffixed,8,3)+
                        ' y='+FloatToStrf(y,fffixed,8,3));
       x:=x+h;
                          end;
     end;
end.  // Конец модуля matfu
```

Листинг 6.2

```
unit Unit6;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls, Matfu;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
```

```
    RadioGroup1: TRadioGroup;
    Memo1: TMemo;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    procedure FormCreate(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
  xn,xk,e : extended;
  m : word;
implementation
{$R *.dfm}

function sx(x:extended):extended;
 var a,n,s : extended;
     k : integer;
begin
    k:=-1;  a:=1;  n:=0; s:=1;
  try
        while (abs(a) > e) do begin
            n:=n+1;
            a:=-a*x/n;
            s:=s+a;
          end;
       Result:=s;
   except
on EInvalidOp do
            k:=MessageDlg('Неправильная операция с плавающей точкой. '+
                          ' Продолжить вычисления?',mtError,[mbYes,mbNo],0);
on EOverFlow do
            k:=MessageDlg('Переполнение при выполнении операции с'+
     ' плавающей точкой! Продолжить вычисления?',mtError,[mbYes,mbNo],0);
     else
             k:=MessageDlg('Возникла неизвестная исключительная ситуация!'+
                          ' Продолжить вычисления?',mtError,[mbYes,mbNo],0);
   end;
     case k of
      mrYes : Result:=0;
      mrNo  : Halt(1);
      end;
end;
```

```
function yx(x:extended):extended;
begin
  Result:=exp(-x);
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Memo1.Clear;
  RadioGroup1.ItemIndex:=0;
    Edit1.Text:='0';
    Edit2.Text:='2';
    Edit3.Text:='6';
    Edit4.Text:='0,001';
end;

procedure TForm1.BitBtn1Click(Sender: TObject);
begin
  Memo1.Clear;
       Memo1.Lines.Add('Лаб. раб. №6 ст. гр. 552003 Иванова А.А.');
     xn:=StrToFloat(Edit1.Text);
       Memo1.Lines.Add(' xn='+FloatToStrF(xn,ffFixed,6,2));
     xk:=StrToFloat(Edit2.Text);
       Memo1.Lines.Add(' xk='+FloatToStrF(xk,ffFixed,6,2));
     m:=StrToInt(Edit3.Text);
       Memo1.Lines.Add(' h='+IntToStr(m));
     e:=StrToFloat(Edit4.Text);
       Memo1.Lines.Add(' e='+FloatToStrF(e,ffFixed,8,5));
    case RadioGroup1.ItemIndex of
     0 : begin
          Memo1.Lines.Add('Расчет S(x)');
          Tabl(sx,xn,xk,m,Memo1);
         end;
     1 : begin
          Memo1.Lines.Add('Расчет Y(x)');
          Tabl(yx,xn,xk,m,Memo1);
        end;
    end;
end;
end.
```

6.4. Выполнение индивидуального задания

По указанию преподавателя выберите вариант задачи из заданий, приведенных в теме 3. В местах возможного возникновения ошибок использовать конструкции для обработки исключительных ситуаций.

ТЕМА 7. СИСТЕМЫ СЧИСЛЕНИЯ. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ МНОЖЕСТВ И СТРОК

Цель лабораторной работы: изучить правила перевода чисел из одной системы счисления в другую. Написать программу для работы со строками.

7.1. 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:

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

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 для $p \le 10$ используют символы 0…9, для $p \ge 10$ добавляют буквы латинского алфавита A, B, C, D, E, F, которые в десятичной системе представляют числами 10, 11, 12, 13, 14, 15. Например,

$$R = (D3,E)_{15} = D \cdot 15^1 + 3 \cdot 15^0 + E \cdot 15^{-1} = 13 \cdot 15^1 + 3 \cdot 15^0 + 14 \cdot 15^{-1} = (198,93)_{10}$$

$$R = (124,5)_8 = 1 \cdot 8^2 + 2 \cdot 8^1 + 4 \cdot 8^0 + 5 \cdot 8^{-1} = (84,625)_{10}.$$

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 системы счисления 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 $a_0, a_1, \mathbf{K}, a_n$, 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 $a_{-1}, a_{-2}, \mathbf{K}, a_{-k}$, а дробная часть опять умножается на основание системы. Расчеты ведут до получения требуемого количества цифр.

7.2. Тип множество

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

Для работы с множествами в Паскале введен тип переменных ***set of***:

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).

Множество-константа: [i_1, i_2, …, i_k], где i_k – элементы множества.

7.2.1. Операции над множествами

Над переменными типа «множество» допустимы операции присваивания сложения, вычитания, умножения. Эти операции дополняют две процедуры:

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

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

Эти операции выполняются значительно быстрее, чем эквивалентные им **s:=s+[i];** **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 *d* принадлежат *c*;
Результат *c<=d* будет ***true***, если все элементы *c* принадлежат *d*;
Результат *i* ***in*** *c* будет ***true***, если элемент *i* принадлежит *c*.

7.2.2. 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** ....

7.3. Типы данных для работы со строками и основные операции

Переменные строкового типа вводятся следующим образом:

```
Var
    s1 : String[N] ;      // Короткая строка,  N ≤ 255
    s2 : ShortString;     // Короткая строка, N = 255
    s3 : String;          // Длинная строка, N ≤ 2 Гбайт
    s4 : WideString;      // Широкая строка
    s5 : PChar;           // Нуль-терминальная строка
```

Короткие строки занимают фиксированное количество байт. Строка типа String[N] может содержать nt (0≤nt≤255) символов. Под переменную типа String[N] отводится N+1 байт. В s1[0] содержится текущая длина строки. Строка ShortString эквивалентна String[255].

Длинная строка типа String. При работе с этим типом данных, в отличие от String[N], память выделяется по мере необходимости (динамически) и может занимать практически всю доступную программе память. Фактически переменные этого типа являются типизированными указателями.

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

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

К каждому элементу строки можно обращаться с помощью индекса, аналогично обращению к элементу массива символов: s[i]:=s[j].

Основные операции над переменными строкового типа: присвоение, сцепление (s:=s1+’ fg’+s2), сравнение (>, ≥, =, <, ≤). Большей считается та строка, у которой больше первый (второй, третий, …) символ.

7.4. Некоторые процедуры и функции обработки строк

Процедуры:

Delete(St:String; pz,n:integer); - удаляет из строки n символов начиная с позиции pz.

Insert(St,St1;string;pz:integer); - вставляет строку St внутрь строки St1 начиная с позиции pz.

Функции:

St:=Copy(St1,pz,n); - выделяет из строки St1 подстроку St из n символов начиная с позиции pz, при этом St1 сохраняется.

N:=Length(St); - определяет текущую длину строки.

pz:=pos(St,St1); - определяет номер позиции, начиная с которой внутри строки St1 расположена строка St. Если внутри St1 нет St, тогда pz=0.

Описание функций преобразования чисел в строки и наоборот помещено в прил. 1.

7.5. Пример написания программы

Задание: написать программу подсчета количества неповторяющихся символов в каждом слове произвольной строки. Слова отделяются друг от друга одним или несколькими пробелами. Ввод строки заканчивать нажатием клавиши Enter.

Панель диалога будет иметь вид (рис. 7.1).

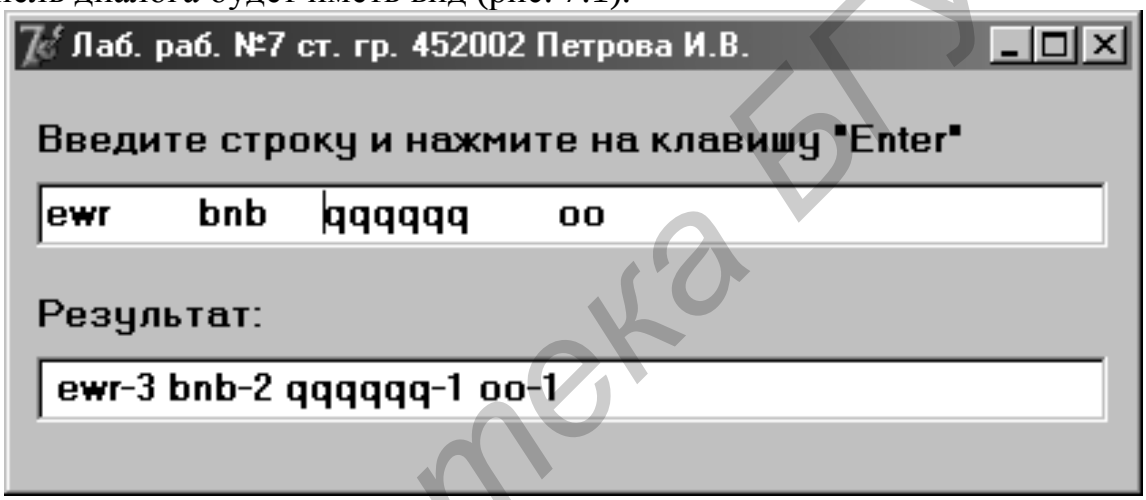

Рис. 7.1

Текст программы приведен на листинге 7.1.

Листинг 7.1

```
unit Unit7;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
   Forms, Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Label2: TLabel;
    Edit2: TEdit;
    Label1: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure Edit1KeyPress(Sender: TObject; var Key: Char);
  private     { Private declarations }
  public      { Public declarations }
  end;
var   Form1: TForm1;
```

```
implementation
{$R *.dfm}

Function Krz(St:string):Word; // Подсчет кол-ва различных символов
Var a : set of char;
        m,i : Word;
begin
        m:=0; a:=[];
        for i:=1 to Length(St) do
              if not(St[i] in a) then begin
               Include(a,St[i]);  Inc(m);
                                        end;
      Result:=m;
end;

Function FWrd(st : string) : string;
  var n,i,nst: integer;
     sl,srez : string;
 begin
         Result:='';  srez:='';  n:=Length(st);
         if n=0 then exit;
           st:=st+' '; n:=n+1; sl:='';
          for i:=1 to n do
           if st[i]<>' ' then sl:=sl+st[i]
                         else
             if  sl<>'' then begin
              srez:=srez+' '+sl+'-'+IntToStr(Krz(sl));   sl:='';
                             end;
     Result:=srez;
end;  // Конец функции FWrd

procedure TForm1.FormCreate(Sender: TObject);
begin
Edit1.Clear;  Edit2.Clear;
end;

procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
   if Key = #13 then Edit2.Text:=FWrd(Edit1.Text);
end;
end.
```

7.6. Выполнение индивидуального задания

По указанию преподавателя выберите вариант задачи. Для ввода строк и работы с ними использовать компонент TEdit. Ввод строки заканчивать нажатием клавиши Enter. Алгоритм оформить в виде подпрограммы.

1. Дана строка символов, состоящая из групп нулей и единиц, разделенных пробелами. Найти количество групп с пятью символами.

2. Дана строка, представляющая собой запись числа в четырнадцатеричной системе счисления. Преобразовать ее в строку, представляющую собой запись числа в десятеричной системе счисления.

3. Дана строка, представляющая собой запись числа в десятеричной системе счисления. Преобразовать ее в строку, представляющую собой запись числа в восьмеричной системе счисления.

4. Дана строка, представляющая собой запись числа в десятеричной системе счисления. Преобразовать ее в строку, представляющую собой запись числа в двоичной системе счисления.

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

6. Дана строка, состоящая из групп нулей и единиц, разделенных пробелами. Найти и вывести на экран самую короткую группу.

7. Дана строка, состоящая из групп нулей и единиц. Каждая группа отделяется от другой одним или несколькими пробелами. Подсчитать количество символов в самой длинной группе.

8. Дана строка, состоящая из групп нулей и единиц. Каждая группа отделяется от другой одним или несколькими пробелами. Найти и вывести на экран группы с четным количеством символов.

9. Дана строка, состоящая из групп нулей и единиц. Каждая группа отделяется от другой одним или несколькими пробелами. Подсчитать количество повторяющихся символов в группах с нечетным количеством символов.

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

11. Дана строка, представляющая собой запись числа в двоичной системе счисления. Преобразовать ее в строку, представляющую собой запись числа в десятеричной системе счисления

12. Дана строка символов, состоящая из произвольного текста, слова разделены пробелами. Вывести на экран порядковый номер слова, накрывающего k-ю позицию (если на k-ю позицию попадает пробел, то номер предыдущего слова), и найти в нем количество повторяющихся символов.

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

ТЕМА 8. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ЗАПИСЕЙ И ФАЙЛОВ

Цель лабораторной работы: изучить правила работы с компонентами TOpenDialog и TSaveDialog. Изучить правила работы с типом «запись». Написать программу с использованием файлов.

8.1. 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[30];                  // Поле Ф.И.О.
            Group:integer;                   // Поле номера студ. группы
            Ocn:array[1..3] of integer;      // Поле массива оценок
                       end;
Var  Student: TStudent;        // Объявление переменной типа запись
```

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

```
Student.Fio:= 'Иванов А.И.';  // Внесение данных в поля записи
Student.Group:=720603;
```

Для сокращения текста доступ к полям осуществляется также при помощи оператора with:

```
With Student do begin
        Fio:= 'Иванов А.И.';
        Group:=720603;
                  end;
```

8.2. Работа с файлами

Файл – это именованная область данных на внешнем физическом носителе. В Object Pascal различают три вида файлов в зависимости от способа их организации и доступа к элементам: текстовые, типизированные и нетипизированные.

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

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

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

Порядок работы с файлами следующий:

```
            . . .
AssignFile(Fl, 'Filenam1.txt'); // Связывание файловой переменной
AssignFile(Fw, 'Filenam2.txt');       // с именем файла на диске
Rewrite(Fl);                  // Создание нового файла Fl
Reset(Fw);)                   // Открытие уже существующего файла Fw
            . . .
Write(Fl, Stud);              // Запись данных в файл Fl
Read(Fw, Stud);               // Чтение данных из файла Fw
            . . .
CloseFile(Fl);                // Закрытие файла Fl
CloseFile(Fw);                // Закрытие файла Fw
```

8.3. Подпрограммы работы с файлами

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

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

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

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

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

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

CloseFile(F) - закрывает ранее открытый файл.

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

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

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

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

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

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

IoResult: 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.

8.4. Компоненты TOpenDialog и TSaveDialog

Компоненты TOpenDialog и TSaveDialog находятся на странице DIALOGS. Все компоненты этой страницы являются невизуальными, т.е. не видны в момент работы программы. Поэтому их можно разместить в любом удобном месте формы. Оба рассматриваемых компонента имеют идентичные свойства и отличаются только внешним видом. После вызова компонента появляется диалоговое окно, с помощью которого выбирается имя программы и путь к ней. В случае успешного завершения диалога имя выбранного файла и маршрут поиска 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.

8.5. Пример написания программы

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 в окно TMemo, и записать эту информацию в текстовой файл.

8.5.1. Настройка компонентов TOpenDialog и TSaveDialog

Для установки компонентов TOpenDialog и TSaveDialog на форму необходимо на странице Dialogs меню компонентов щелкнуть мышью соответственно по пиктограммам  или  и поставить их в любое свободное место формы. Настройка 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 OpenDialod1 установим значения маски, как показано на рис. 8.1. 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).

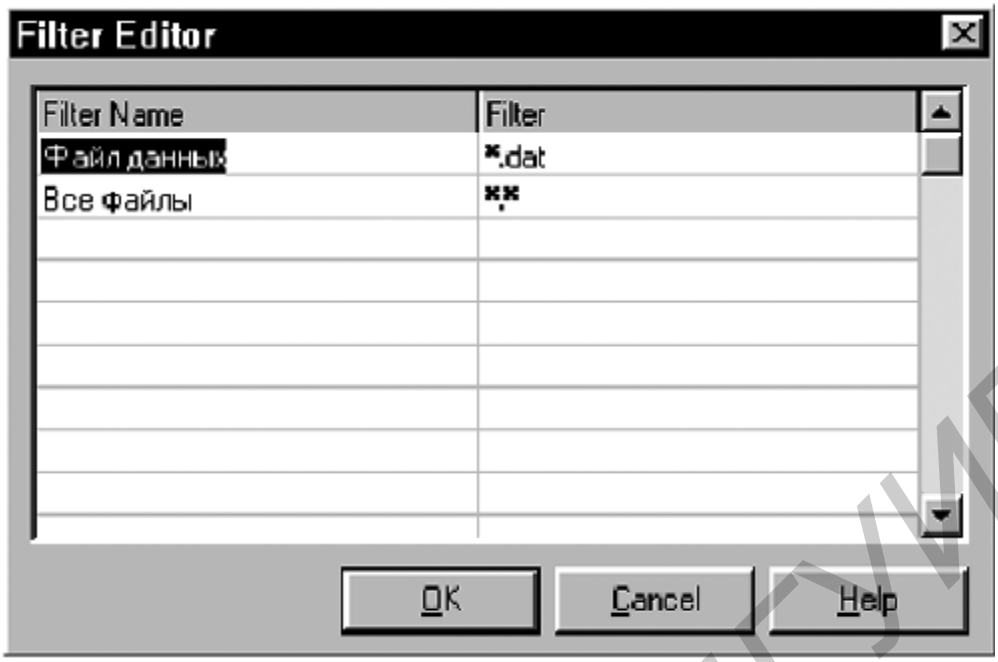

Рис. 8.1

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

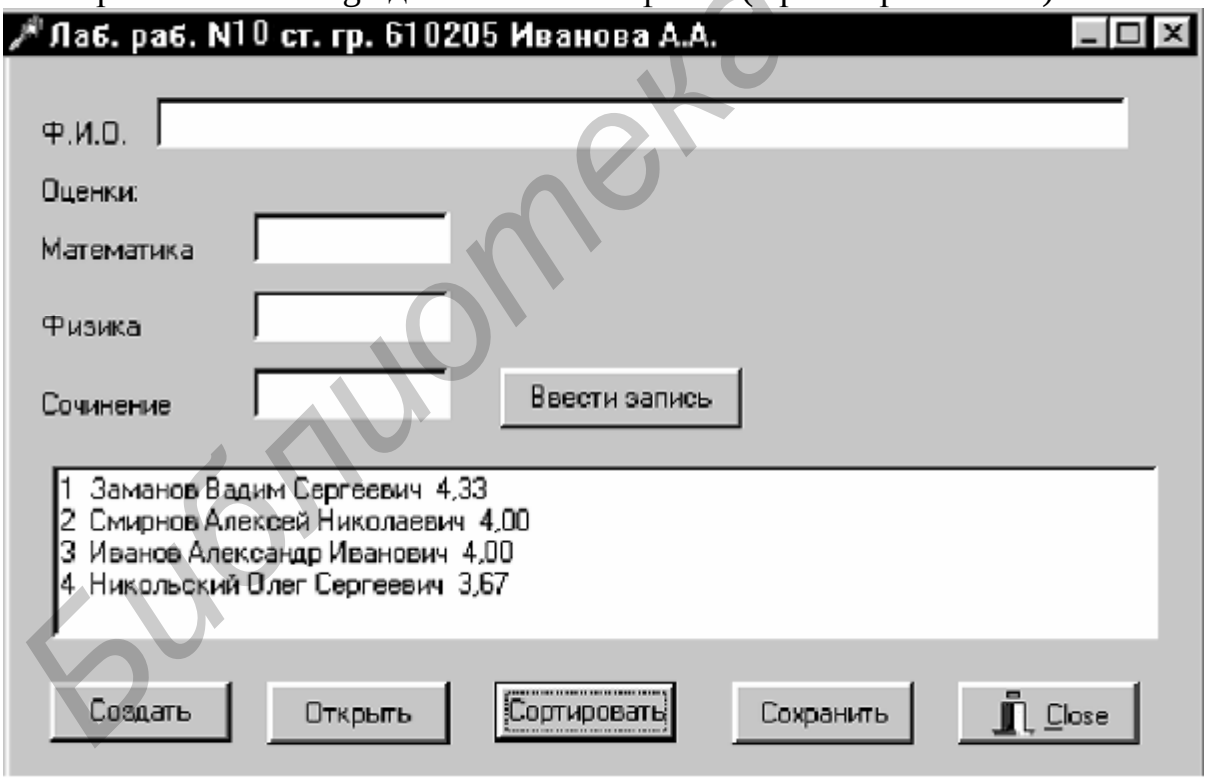

Рис. 8.2

8.5.2. Работа с программой

После запуска программы на выполнение появится диалоговое окно программы. Кнопка «Ввести запись» видна не будет. Необходимо создать новый файл записей, нажав на кнопку «Создать» или открыть ранее созданный, нажав кнопку «Открыть». После этого станет видна кнопка «Ввести запись» и можно

будет вводить записи. При нажатии на кнопку «Сортировка» будет проведена сортировка ведомости по убыванию среднего балла и диалоговое окно примет вид, как на рис. 8.2. Затем при нажатии на кнопку «Сохранить» будет создан текстовой файл, содержащий отсортированную ведомость. Файл записей закрывается одновременно с программой при нажатии на кнопку «Close» или .

Текст программы приведен на листинге 8.1.

Листинг 8.1

```
unit unit8;
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;
   Button5: TButton;
   BitBtn1: TBitBtn;
   SaveDialog1: TSaveDialog;
   Button2: TButton;
   OpenDialog1: TOpenDialog;
   Button4: TButton;
   procedure FormCreate(Sender: TObject);
   procedure Button1Click(Sender: TObject);
   procedure Button2Click(Sender: TObject);
   procedure Button3Click(Sender: TObject);
   procedure Button4Click(Sender: TObject);
   procedure Button5Click(Sender: TObject);
   procedure BitBtn1Click(Sender: TObject);
 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.Clear;   Edit2.Clear;   Edit3.Clear;   Edit4.Clear;
   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.Clear;   Edit2.Clear;   Edit3.Clear;   Edit4.Clear;
end;
procedure TForm1.Button2Click(Sender: TObject); //Создание нового
begin                                                    // файла записей
   OpenDialog1.Title :='Создать новый файл';   // Изменение заголовка
                                                         //окна диалога
  if OpenDialog1.Execute then  // Выполнение стандартного диалога
    begin                                  // выбора имени файла
     FileNameZ:= OpenDialog1.FileName;   // Возвращение имени
                                                    // дискового файла
     AssignFile(Fz, FileNameZ);     // Связывание файловой переменной Fz
                                   // с именем файла
      Rewrite(Fz);                 // Создание нового файла
    end;
     Button1.Show;              // Сделать видимой кнопку  «Ввести запись»
```

```
end;

procedure TForm1.Button3Click(Sender: TObject); // Открыть
begin                                          // существующий  файл
 if OpenDialog1.Execute then        // Выполнение стандартного диалога
  begin                             // выбора имени файла
   FileNameZ:= OpenDialog1.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;
   Button1.Show;                // Сделать видимой кнопку  «Ввести запись»
end;

procedure TForm1.Button4Click(Sender: TObject);  // Сортировка записей
 var i,j :  word;
      st : TStudent;
begin
 for i:=1 to nzap-1 do              // Сортировка массива записей
  for j:=i+1 to nzap do
   if Stud[i].sball < Stud[j].sball then begin
        st:=Stud[i];
       Stud[i]:=Stud[j];
       Stud[j]:=st;
                                    end;
    Memo1.Clear;
  for i:=1 to nzap do       // Вывод в окно Memo1 отсортированных записей
       with stud[i] do
    Memo1.Lines.Add(IntToStr(i)+' '+fio+' '+FloatToStrf(sball,fffixed,4,2));
end;

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

8.6. Выполнение индивидуального задания

По указанию преподавателя выберите вариант задачи. 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. 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 р.

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. 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.

6. 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.

7. 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.

8. 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.

9. Информация о сотрудниках предприятия содержит Ф.И.О., номер отдела, должность, дату начала работы. Вывести списки сотрудников по отделам в порядке убывания стажа.

10. 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.

11. 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.

12. 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.

13. 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.

14. В радиоателье хранятся квитанции о сданной в ремонт радиоаппаратуре. Каждая квитанция содержит следующую информацию: наименование группы изделий(телевизор, радиоприемник и т. п.), марка изделия, дата приемки в ремонт, состояние готовности заказа (выполнен, не выполнен). Вывести информацию о состоянии заказов на текущие сутки по группам изделий.

15. 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.

ТЕМА 9. ПРОГРАММИРОВАНИЕ С ОТОБРАЖЕНИЕМ ГРАФИЧЕСКОЙ ИНФОРМАЦИИ

Цель лабораторной работы: изучить возможности построения изображений с использованием класса TCanvas и графиков с помощью компонента TChart.

9.1. Как рисуются изображения

Нарисовать картинку в среде Delphi можно на многих компонентах (например на форме, на TPaintBox), однако наиболее удобно использовать компонент TImage (страница Additional). Нарисованную в Image1 картинку можно перенести в отчет, используя процедуру **ClipBoard.Assign(Image1.Picture)** (модуль Clipbrd). Для рисования используют класс TCanvas, который является свойством многих компонентов, и представляет собой прямоугольный холст в виде матрицы из пикселей и набор инструментов для рисования на нем. Каждый пиксел имеет координату (x, y), где x – порядковый номер пикселя, начиная от левой границы холста, а y – порядковый номер пикселя, начиная от верхней границы холста. Левый верхний угол холста имеет координату (0, 0), а правый (Image1.Width, Image1.Height).

Основные *свойства* класса TCanvas:

Property Pen : TPen; – карандаш (имеет цвет и толщину),

Property Brush : TBrush; – кисть (имеет цвет),

Property Font : TFont; – шрифт.

Некоторые *методы* класса TCanvas:

Procedure Ellipse(X1, Y1, X2, Y2: Integer) – чертит эллипс в охватывающем прямоугольнике (X1, Y1), (X2, Y2) и заполняет внутреннее пространство эллипса текущей кистью.

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

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

Procedure Polygon (Points: array of TPoint) – вычерчивает карандашом многоугольник по точкам, заданным в массиве Points. Например: Canvas.Polygon([Point(x1, y1), Point(x2, y2), Point(x3, y3]);. Конечная точка соединяется с начальной и многоугольник заполняется кистью. Для вычерчивания без заполнения используйте метод Polyline.

Procedure Rectangle (X1, У1, X2, У2: Integer) – вычерчивает и заполняет прямоугольник (X1, У1), (X2, У2). Для вычерчивания без заполнения используйте FrameRect или PolyLine.

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

9.2. 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 двухмерный массив Chart**I**.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 Chart**I**.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. Перенести график в отчет можно через буфер обмена, используя процедуру Chart**I**.CopyToClipboardMetafile(True).

9.3. Пример написания программы

Задание: составить программу, выводящую график функции sin(x) двумя способами. В первом случае использовать компонент Timage, во втором – компонент TChar. Для размещения исходных данных и двух графиков использовать три панели компонента TPageControl.

9.3.1. Работа с компонентом TPageControl

Если на одной форме необходимо разместить большое количество информации или разделить эту информацию, удобно использовать компонент TPageControl (страница Win32). Этот компонент может содержать несколько перекрывающих друг друга панелей, доступ к которым осуществляется с помощью закладок. Для добавления новой панели необходимо щелкнуть по компоненту правой кнопкой мыши и выбрать пункт NewPage. Размещение элементов на нужной панели производится точно так же, как и на форме.

Расположите на форме компонент PageControl1, создайте три панели и расположите на них компоненты в соответствии с рис. 9.1. – 9.3.

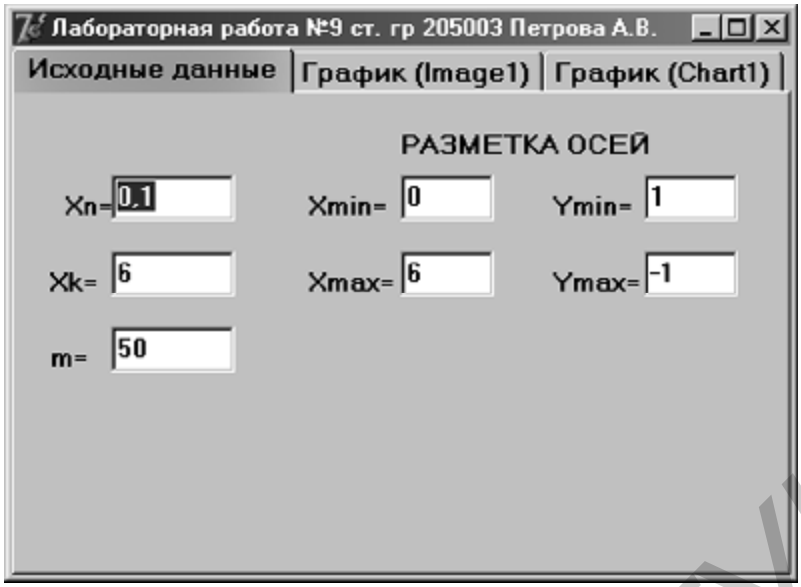

Рис. 9.1

Рис. 9.2

Рис. 9.3

9.3.2. Работа с компонентом TChart

Для изменения параметров компонента TChart необходимо дважды щелкнуть по нему мышью в окне формы. Появится окно редактирования EditingChat1 (рис. 9.4). Для создания нового объекта Series1 щелкнуть по кнопке Add на странице Series. В появившемся диалоговом окне TeeChart Gallery выбрать пиктограмму с надписью Line (график выводится в виде линий). Если нет необходимости представления графика в трехмерном виде, отключить независимый переключатель 3D. После нажатия на кнопку OK появится новая серия с названием Series1. Для изменения названия нажать кнопку Title. Закладка Legend задает список обозначений даиграммы (можно убирать с экрана). Название графика вводится на странице Titles. Разметка осей меняется на странице Axis. Страница Series задает характеристики (цвет, толщина линий) для определенного графика.

Данные по оси X автоматически сортируются, поэтому, если необходимо нарисовать, например окружность, сортировку отключают функцией Order: Chart1.Series[0].XValues.Order=loNone.

Нажимая различные кнопки меню, познакомьтесь с другими возможностями EditingChat.

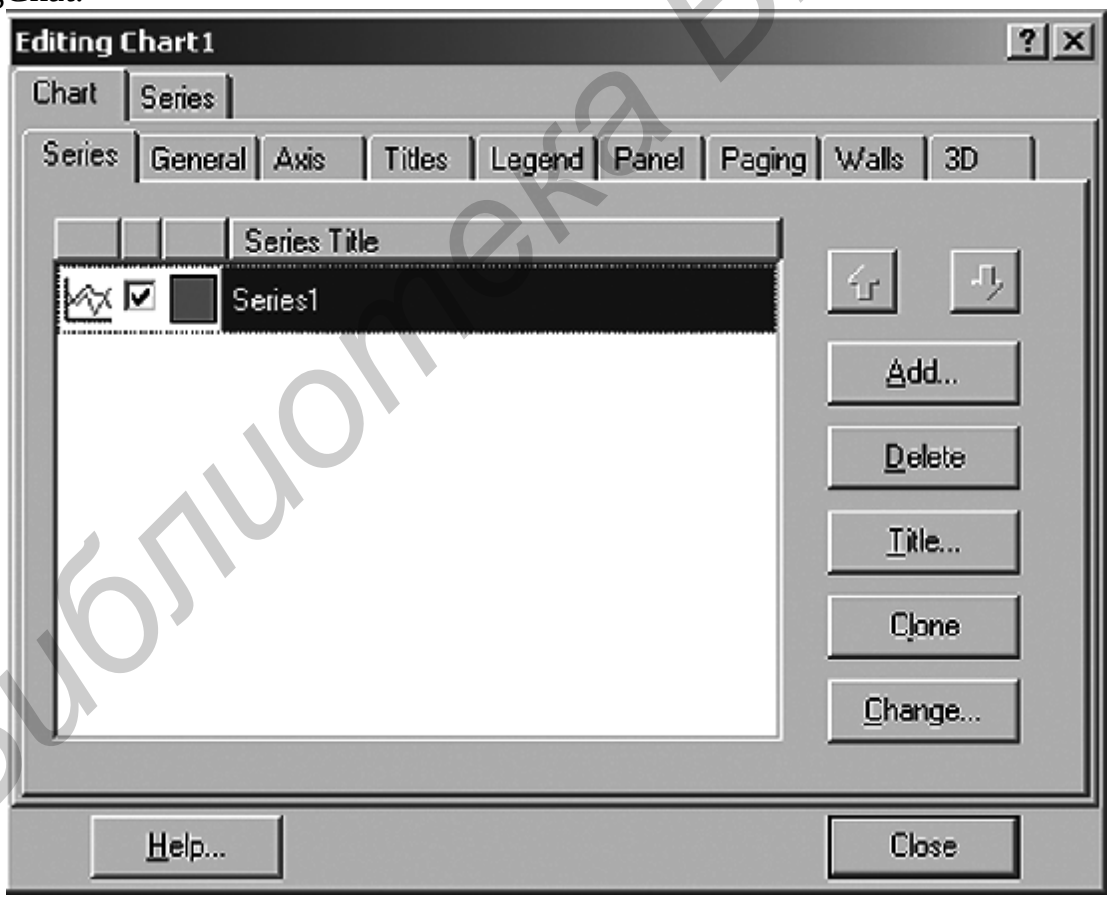

Рис. 9.4

Текст программы приведен на листинге 9.1.

Листинг 9.1

```
unit Unit9;
interface
```

```
uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms, Dialogs, ExtCtrls, TeeProcs, TeEngine, Chart, StdCtrls, ComCtrls,
Series, Clipbrd;
type
  TForm1 = class(TForm)
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    TabSheet2: TTabSheet;
    TabSheet3: TTabSheet;
    Label1: TLabel;
    Label2: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label3: TLabel;
    Label8: TLabel;
    Edit4: TEdit;
    Edit5: TEdit;
    Edit6: TEdit;
    Edit7: TEdit;
    Button1: TButton;
    Button2: TButton;
    Image1: TImage;
    Button3: TButton;
    Button4: TButton;
    Chart1: TChart;
    Series1: TLineSeries;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
  xn, xk, x, y, h, xomin, xomax, yomin, yomax : extended;
  m, i, delt : integer;
implementation
{$R *.dfm}
```

```
Function f(x: extended):extended;
 begin
  Result:=sin(x); // Функция
 end;

procedure TForm1.FormCreate(Sender: TObject);
begin
 Edit1.Text:='0,1';   Edit2.Text:='6';    Edit3.Text:='50';    Edit4.Text:='0';
 Edit5.Text:='6';     Edit6.Text:='-1';   Edit7.Text:='1';
end;

procedure TForm1.Button1Click(Sender: TObject);  // Рисование в Image1
 var xmax, ymax, xt,  yt, yo : integer;
    hx, hy : extended;
begin
  xn:=StrToFloat(Edit1.Text);
  xk:=StrToFloat(Edit2.Text);
  m:=StrToInt(Edit3.Text);
  xomin:=StrToFloat(Edit4.Text);
  xomax:=StrToFloat(Edit5.Text);
  yomin:=StrToFloat(Edit6.Text);
  yomax:=StrToFloat(Edit7.Text);
    with Image1.Canvas do begin
  Pen.Color:=clBlack;   // Установка цвета карандаша
  Brush.Color:=clGreen;  // Установка цвета кисти
  xmax:=Image1.Width;    // Чтение максимальной координаты по x
  ymax:=Image1.Height;   // Чтение максимальной координаты по y
   {Строим оси координат}
   yo:=ymax div 2;
  MoveTo(0,yo);       LineTo(xmax,yo);
  MoveTo(0,0);        LineTo(0,ymax);
  Pen.Color:=clRed;   // Установка цвета карандаша
  Pen.Width:=2;       // Установка толщины карандаша
    hx:=(xomax-xomin)/xmax;  // Масштабные коэффициенты, устанавли-
    hy:=(yomax-yomin)/ymax;  // вающие шаг (в пикселях) по X и по Y
{Вывод графика}
    h:=(xk-xn)/(m-1);
    x:=xn;
    y:=f(x);     // Первая точка
      MoveTo(Round(x/hx),Round(yo-y/hy));
 for i:=1 to m do begin
     x:=x+h;
     y:=f(x);
      LineTo(Round(x/hx),Round(yo-y/hy));
                end;
                    end;
```

```
end;


procedure TForm1.Button2Click(Sender: TObject);
begin
ClipBoard.Assign(Image1.Picture);  // Копировать в буфер обмена
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
  xn:=StrToFloat(Edit1.Text);
  xk:=StrToFloat(Edit2.Text);
  m:=StrToInt(Edit3.Text);
  xomin:=StrToFloat(Edit4.Text);
  xomax:=StrToFloat(Edit5.Text);
  yomin:=StrToFloat(Edit6.Text);
  yomax:=StrToFloat(Edit7.Text);
        {Установка осей}
    with Chart1 do begin
      LeftAxis.Automatic:=False;
      LeftAxis.Minimum:=yomin;
      LeftAxis.Maximum:=yomax;
      BottomAxis.Automatic:=False;
      BottomAxis.Minimum:=xomin;
      BottomAxis.Maximum:=xomax;
        SeriesList[0].Clear;
      h:=(xk-xn)/(m-1);     x:=xn;
  for i:=1 to m do begin
      y:=f(x);
        SeriesList[0].AddXY(x,y);
      x:=x+h;
                end;
              end;
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
Chart1.CopyToClipboardMetafile(True);  // Копировать в буфер обмена
end;
end.
```

9.4. Выполнение индивидуального задания

Постройте график соответствующей функции от x для вариантов из темы 3. Таблицу данных получить, изменяя параметр x с шагом h. Ввод исходных данных организовать через окна TEdit. Самостоятельно выбрать удобные параметры настройки.

По указанию преподавателя выберите вариант задачи. Решите задачу и, используя функции класса TCanvas, нарисуйте соответствующие геометрические фигуры. Расположите все рисунки в центре TImage, так чтобы они занимали 2/3 области окна. Все исходные данные имеют действительный тип. Используйте масштабирование.

Все задания оформить в одном проекте (использовать компонент TPageControl).

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

4. Отобразить взаимное расположение окружности радиусом 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 радиусом R с центром в точке (x_0, y_0).

6. Найти координаты точек пересечения двух окружностей радиусами 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 на 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.

ТЕМА 10. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ОБЪЕКТОВ И КЛАССОВ

Цель лабораторной работы: изучить особенности объектно-ориентированного программирования (ООП). Написать и отладить программу с использованием классов.

10.1. Понятие объекта и класса

Основным понятием ООП является ***объект***, который в Delphi представляет собой переменную структурированного типа, описываемую с помощью ключевого слова ***Class***. Подобно обыкновенной записи типа ***Record***, переменная типа ***Class*** под одним именем объединяет как данные различных типов (поля), так и процедуры и функции обработки этих данных (методы). Такое объединение данных и процедур их обработки называется ***инкапсуляция***.

В Delphi типы объектов называются **классами**, а сами объекты являются динамическими переменными (символ « ^ » не используется).

10.2. Наследственность и полиморфизм

Свойство ***наследственности*** заключается в том, что любой класс может быть порожден от другого класса с наследованием всех свойств. Если класс *B* порожден от класса *A*, то класс *A* называют – «класс-родитель», а *B* – «класс-потомок». Порожденный класс автоматически наследует поля и методы своего родителя. Прародителем всех классов в Delphi является класс ***TObject***.

Свойство ***полиморфизма*** позволяет использовать одинаковое название метода для решения сходных, но несколько отличающихся у разных родственных классов задач. В результате в объекте-родителе и объекте-потомке возможно существование двух одноименных методов, реализующих различные алгоритмы.

10.3. Создание, уничтожение и операция присваивания объектов

Как всякая динамическая переменная, объект перед началом работы с ним должен быть создан. Нужно выделить под него динамическую область памяти и **инициализировать** (т.е. подготовить к работе) созданный объект. Для этого в Delphi служит **конструктор *Create***, который является методом класса TObject. Применяется конструктор следующим образом:

<Имя-переменной-типа-класс> := <тип-класса> . Create;

После окончания работы с объектом выделенную под него память необходимо освободить, для чего служат **деструкторы** класса TObject ***Destroy*** или ***Free.*** Метод ***Free*** удобнее использовать, так как он в отличие от метода ***Destroy*** перед освобождением памяти проверяет, не была ли она уже освобождена ранее, т.е. работает более корректно:

<Имя-переменной-типа-класс> . Free;

При необходимости в состав любого пользовательского класса могут быть введены свои методы **Constructor** и **Destructor,** которые дополняют ***Create*** и ***Free*** новыми функциями.

Переменной родительского типа можно присвоить значение переменной типа потомка. Обратное же присваивание запрещено, т.к. при этом некоторые поля или методы могли бы оказаться несуществующими, т.е. у потомка их больше. Однако, если переменная родительского типа указывает на объект, фактически соответствующий переменной типа потомка, то при необходимости ее можно преобразовать к типу потомка с помощью оператора ***as*** и присвоить потомку.

10.4. Статический, виртуальный и динамический способы реализации полиморфизма

При объявлении в разделе **Var** и последующей работе с несколькими объектами каждый объект располагается по некоторому адресу. Причем все обычные поля копируются, а методы хранятся в одном экземпляре. Каждый раз, когда вызывается метод, ему через параметр-указатель с именем ***Self*** передается адрес **того экземпляра объекта**, который обращается к методу.

Полиморфизм можно организовать по-разному: используя **раннее связывание** метода с полями объекта, которое происходит на этапе компиляции, и **позднее связывание,** которое осуществляется непосредственно в нужный момент при выполнении программы.

Статические методы характеризуются тем, что связывание метода с полями осуществляется во время компиляции (раннее связывание).

Виртуальные и динамические методы связываются во время выполнения программы (позднее связывание). Если метод объявлен виртуальным или динамическим, то нельзя менять типы и число параметров.

Для реализации позднего связывания поступают следующим образом. В потомке замещающий метод объявляется директивой ***override***. Замещаемый одноименный метод родителя объявляется как динамический или виртуальный с помощью ключевых слов (***dynamic***) или (***virtual)***. Вызов перекрытого метода родительского класса в одноименном методе потомка достигается с помощью зарезервированного слова ***Inherited*** (унаследованный).

Встретив объявления ***dynamic*** или ***virtual***, компилятор создает таблицы соответствия DMT и VMT. В этих таблицах помещаются адреса точек входа методов. Адрес VMT «своего» класса хранится в каждом экземпляре объекта в особом, скрытом от программиста поле. Адрес DMT хранится в VMT. При каждом обращении к методу компилятор вставляет в соответствующую таблицу код, позволяющий извлечь затем из нее адрес точки входа в подпрограмму.

Отличие таблиц DMT и VMT в том, что DMT содержит адреса только тех методов, которые объявлены как ***dynamic*** в данном классе, а VMT содержит адреса **всех виртуальных методов** данного класса: как нововведенных, так и унаследованных от родителей.

При реализации позднего связывания в родительском классе часто используют *абстрактные* методы (*abstract*), т.е. такие виртуальные методы, тело которых не прописано. Классы, имеющие хотя бы один абстрактный метод, сами называются абстрактными. Объекты абстрактных методов не создаются.

10.5. Свойства

Правила ООП требуют, чтобы обращение к полям осуществлялось посредством методов, однако это не всегда удобно. Поэтому класс имеет **свойства** – специальный механизм, **регулирующий доступ к полям**. Свойства объявляются с помощью специальной конструкции ***property…, read…, write…*** .

Обычно свойство связано с некоторым полем и указывает те методы класса, которые должны использоваться при записи в это поле и при чтении из него.

Это делается, например, следующим образом:

```
Property x:TPole read GetPole write SetPole;
```

Метод ***Getpole*** служит для чтения, а ***SetPole*** – для записи. Если необходимо сделать доступ к полю только для чтения, то следует опустить ***write***.

10.6. Пример написания программы

Задание: составить родительский класс, который позволяет с помощью методов **Show** и **Hide** показывать и стирать объекты на экране. Написать классы-потомоки, которые рисуют круг, квадрат, комбинацию круга и квадрата (человечек). Предусмотреть возможность перемещения объектов и изменения их размера.

Панель диалога представлена на рис. 10.1.

Рис. 10.1

Текст модуля приведен на листинге 10.1, а текст программы – на листинге 10.2.

Листинг 10.1

```
Unit Unit2;
 Interface
```

```
uses  Graphics;
var   ColrBack:TColor;
 Type
  Tviz=class(Tobject)   // Абстрактный родительский класс
    ColrLine : Tcolor;
    Canvas : Tcanvas;
    x, y, r : word;
    Procedure Ris;virtual;abstract;   // Перекрываемый метод для рисования
    Procedure Draw(bl:boolean);
    procedure Show;  // Показать изображение
    procedure Hide;    // Стереть изображение
    procedure MovTo(dx,dy,dr:integer);  // Сдвинуть и изменить размер
      end;
  TKrug=class(Tviz)   // Класс рисования круга
    x1,y1,x2,y2:word;
    Constructor Create(x0,y0,r0:word; colrLine0:Tcolor;canvas0:Tcanvas);
    Procedure Ris; override;
      end;
  TKvad=class(Tkrug)   // Класс рисования квадрата
    Procedure Ris; override;
      end;
  TKrPr=class(Tkrug)   // Класс рисования круга на квадрате
    dy1:word;
    Constructor Create(x0,y0,r0,dy0:word; colrLine0:Tcolor;canvas0:Tcanvas);
    Procedure Ris; override;
      end;
Implementation   //Ниже описываются тела всех методов
  Procedure Tviz.Draw;      // В зависимости от значения булевой
    begin                   // переменной этот метод рисует картинки Ris
    with Canvas do begin    // либо цветом линии, либо цветом фона
    if bl then begin        // В последнем случае происходит стирание
          pen.color:=colrLine;     brush.color:=colrLine
            end
         else begin
          pen.color:=colrBack;     brush.color:=colrBack
           end;
          Ris;                  // Процедура ris что-то рисует
    end;    end;

  Procedure Tviz.Show;
    begin
      Draw(true);
  end;

Procedure Tviz.Hide;
begin
  Draw(false);
```

```
end;
procedure Tviz.MovTo;
begin
  Hide;
  x:=x+dx;  y:=y+dy;  r:=r+dr;   // Переход к новым координатам
  Show;
end;

   Constructor TKrug.Create;       // Начальные данные для рисования круга
    begin                          // Они такие же, как и для рисования квадрата,
    colrLine:=colrLine0;           // поэтому класс Tkvad наследует его
    canvas:=canvas0;
     x:=x0; y:=y0; r:=r0;
    end;

  Procedure Tkrug.Ris;             // Рисование круга
  Begin
     x1:=x-r;  x2:=x+r;  y1:=y-r;  y2:=y+r;
  Canvas.Ellipse(x1,y1,x2,y2);
  end;

  Procedure Tkvad.ris;             // Рисование квадрата
  Begin
     x1:=x-r;  x2:=x+r;  y1:=y-r;  y2:=y+r;
  Canvas.Rectangle(x1,y1,x2,y2);
  end;

  Constructor TKrpr.Create;        // Начальные данные для рисования круга
    Begin                          // на квадрате
  dy1:=dy0;
  Inherited Create(x0,y0,r0,colrLine0,canvas0); // Используется метод TKrug
  end;

  Procedure TkrPr.Ris;        // Рисование квадрата под кругом
     begin
     Inherited ris // Используется родительский метод рисования круга
     Canvas.Rectangle(x1,y2,x2,y2+dy1);
     end;
end.
```

Листинг 10.2

```
unit Unit10;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls;
type
  TForm1 = class(TForm)
```

```
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    Button5: TButton;
    Button6: TButton;
    Button7: TButton;
    Button8: TButton;
    BitBtn1: TBitBtn;
    Image1: TImage;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Button7Click(Sender: TObject);
    procedure Button8Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
  private     { Private declarations }
  public     { Public declarations }
  end;
var    Form1: TForm1;
implementation
{$R *.dfm}
uses unit2, Clipbrd;
var krug:Tkrug;
    kvad:Tkvad;
    krpr:Tkrpr;
    okno1:Timage; // Введена переменная для сохранения записи
    pxm1,pym1, xo,yo,ro:word;

procedure TForm1.Button1Click(Sender: TObject);  // Нарисовать
begin
okno1:=Form1.Image1;
colrBack:=clWhite;                    // Цвет фона – белый
pxm1:=okno1.ClientWidth;          // Считывание размеров окна
pym1:=okno1.ClientHeight;
with okno1.canvas do begin
pen.color:=colrBack;
brush.color:=colrBack;
Rectangle(0,0,Pxm1,Pym1);          // Заливка всего окна цветом фона
                              end;
xo:=pxm1 div 2;   yo:=pym1 div 2; // Вычисление координат центра окна
ro:=10;                              // Начальный размер круга и прямоугольника
 Krug:=Tkrug.Create(xo,yo,ro,clBlack,okno1.canvas);         // Цвет – черный
 Kvad:=Tkvad.Create(xo+80,yo,ro,clBlack,okno1.canvas);
```

```
 Krpr:=Tkrpr.Create(xo-80,yo,ro,2*ro,clBlack,okno1.canvas);
krug.Show;      // Рисование круга
kvad.Show;      // Рисование квадрата
Krpr.show;      // Рисование комбинации круга и квадрата
end;

procedure TForm1.Button2Click(Sender: TObject);
begin                   // Увеличить круг
Krug.MovTo(0,0,3);
end;

procedure TForm1.Button3Click(Sender: TObject);
begin                   // Двигать квадрат вправо вниз
Kvad.MovTo(3,3,0);
end;

procedure TForm1.Button4Click(Sender: TObject);
begin                   // Ход конем вправо вниз
krpr.MovTo(10,0,0);
okno1.Update;           // Прорисовка окна
sleep(200);             // Задержка
krpr.MovTo(0,5,0);
end;

procedure TForm1.Button5Click(Sender: TObject);
begin                   // Уменьшить
Krug.MovTo(0,0,-3);
end;

procedure TForm1.Button6Click(Sender: TObject);
begin                   // Двигать влево вверх
Kvad.MovTo(-3,-3,0);
end;

procedure TForm1.Button7Click(Sender: TObject);
begin                   // Ход конем влево вверх
krpr.MovTo(-10,0,0);
okno1.Update;
  sleep(200);
krpr.MovTo(0,-5,0);
end;

procedure TForm1.Button8Click(Sender: TObject);
begin           // Только для TImage !!! сохранить картинку
Clipboard.Assign(Image1.Picture);
end;
```

```
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
krug.Free;   kvad.Free;   Krpr.Free;
end;
end.
```

10.7. Выполнение индивидуального задания

По указанию преподавателя выберите вариант задачи.

Описать класс-родитель и класс-потомок, имеющие методы, указанные в соответствующем варианте задания (потомок наследует или переопределяет методы родителя и приобретает новые). Предусмотреть необходимое количество кнопок для демонстрации каждого из методов объектов.

1. Нарисовать перемещающееся и вращающееся колесо. Родительский класс – перемещающийся круг.

2. Нарисовать перемещающуюся повозку (прямоугольник на 2 колесах). Родительский класс – перемещающийся прямоугольник.

3. Нарисовать перемещающуюся ракету с пламенем из сопла. Родительский класс – перемещающийся отрезок.

4. Нарисовать перемещающуюся рожицу, двигающую глазами и открывающую рот. Родительский класс – перемещающийся эллипс.

5. Нарисовать солдатика, перемещающегося и отдающего честь. Родительский класс – перемещающийся прямоугольник.

6. Нарисовать перемещающийся кораблик, который может поднимать флаг. Родительский класс – перемещающийся прямоугольник.

7. Нарисовать перемещающийся автомобиль с открывающимися дверями и включающимися фарами. Родительский класс – перемещающийся прямоугольник.

8. Нарисовать перемещающегося сигнальщика, подающего различные сигналы. Родительский класс – перемещающийся прямоугольник.

9. Нарисовать перемещающийся самосвал, который может поднимать кузов. Родительский класс – перемещающийся прямоугольник.

10. Нарисовать перемещающийся самолет, который может при посадке выпускать шасси. Родительский класс – перемещающийся прямоугольник.

11. Нарисовать перемещающийся вагон, в котором открываются двери и окна. Родительский класс – перемещающийся прямоугольник.

12. Нарисовать перемещающийся паровоз, который выпускает дым. Родительский класс – перемещающийся прямоугольник.

13. Нарисовать перемещающийся воздушный шарик, который может лопнуть. Родительский класс – перемещающийся эллипс.

14. Нарисовать лифт, который доставляет людей на нужный этаж. Родительский класс – перемещающийся прямоугольник.

15. Нарисовать перемещающуюся тележку, на которой перевозят различные грузы. Родительский класс – перемещающийся прямоугольник.

Приложение 1

ПРОЦЕДУРЫ И ФУНКЦИИ ДЛЯ ПРЕОБРАЗОВАНИЯ СТРОКОВОГО ПРЕДСТАВЛЕНИЯ ЧИСЕЛ

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

Подпрограммы преобразования строк в другие типы	
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
Подпрограммы обратного преобразования	
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

Правила использования параметров функции FloatToStrF

Значение Format	Описание
fFfExponent	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)
ffCurrency	Денежный формат. Соответствует ffNumber, но в конце строки ставится символ денежной единицы (для русифицированной версии Windows - символы «р.»). Для Value = π*1000 получим: 3 141,60р

Приложение 2

МАТЕМАТИЧЕСКИЕ ФОРМУЛЫ

Язык Object Pascal имеет ограниченное количество встроенных математических функций (|x| → abs (x), Arctg(x) → ArcTan(x), e^x → Exp(x), π → Pi, $\sqrt{x}$ → Sqrt(x), x^2 → Sqr(x), Ln(x), Cos(x), Sin (x) и др.). 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
$Log_a(x)$	$\frac{Ln(x)}{Ln(a)}$	$LogN(a, x)$
x^a	$e^{a*Ln(x)}$	$Power(x,a)$
$Tg(x)$	$\frac{Sin(x)}{Cos(x)}$	$Tan(x)$
$Ctg(x)$	$\frac{Cos(x)}{Sin(x)}$	$CoTan(x)$
$ArcSin(x)$	$ArcTg\left(\frac{x}{\sqrt{1-x^2}}\right)$	$ArcSin(x)$
$ArcCos(x)$	$\frac{p}{2} - ArcSin(x)$	$ArcCos(x)$
$ArcCtg(x)$	$\frac{p}{2} - ArcTg(x)$	
$Sh(x)$	$\frac{e^x - e^{-x}}{2}$	$Sinh(\mathbf{x})$
$Ch(x)$	$\frac{e^x + e^{-x}}{2}$	$Cosh(x)$
$Sign(x)$	1, если $x>0$; 0, если $x=0$; -1, если $x<0$	$Sign(x)$

ЛИТЕРАТУРА

1. Архангельский А.Я. Программирование в Delphi 7. – М.: ЗАО «Издательство БИНОМ», 2003.

2. Фаронов В.В. Delphi 6: Учебный курс. –М.: Издатель Молгачева С.В., 2001.

3. Дж. Гленн Брукшир. Введение в компьютерные науки. – М., СПб, Киев: Вильямс, 2001.

4. Синицын А. К., Колосов С. В., Навроцкий А. А. и др. Программирование алгоритмов в среде Delphi: Лаб. практикум. В 2 ч. Ч. 1. – Мн.: БГУИР, 2004.

5. Колосов С. В. Программирование в Delphi. Учеб. пособие. – Мн.: БГУИР, 2005.

Св. план 2006, поз. 46

Учебное издание

Синицын Анатолий Константинович,
Навроцкий Анатолий Александрович

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ В СРЕДЕ DELPHI. БАЗОВЫЕ ТИПЫ И ПРОСТЕЙШИЕ АЛГОРИТМЫ

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

Редактор Е.Н. Батурчик

Подписано в печать 8.12.2005.	Формат 60x84 1/16.	Бумага офсетная.
Гарнитура «Таймс».	Печать ризографическая.	Усл. печ. л. 4,77.
Уч.-изд. л. 4,3.	Тираж 1000 экз.	Заказ 604.

Издатель и полиграфическое исполнение: Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
Лицензия на осуществление издательской деятельности №02330/0056964 от 01.04.2004.
Лицензия на осуществление полиграфической деятельности №02330/0131518 от 30.04.2004.
220013, Минск, П. Бровки, 6