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

С. В. Колосов, И. Н. Коренская, И. В. Лущицкая

ПРОГРАММИРОВАНИЕ В СРЕДЕ DELPHI. ПРАКТИКУМ ПО КУРСУ «ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ»

Рекомендовано УМО вузов Республики Беларусь по образованию в области информатики и радиоэлектроники в качестве учебно-методического пособия для студентов учреждений, обеспечивающих получение высшего образования по специальностям, закрепленным за УМО

Минск БГУИР 2012

УДК 004. 43(076.5)
ББК 32.973.26-018.1я73
К61

Р е ц е н з е н т ы:
заведующий кафедрой «Дискретная математика и алгоритмика»
учреждения образования «Белорусский государственный университет»,
доктор физико-математических наук, профессор В. М. Котов;

доцент кафедры «Прикладная информатика» учреждения образования
«Белорусский государственный аграрный технический университет»,
кандидат технических наук А. И. Шакирин

Колосов, С. В.

К61 Программирование в среде DELPHI. Практикум по курсу «Основы алгоритмизации и программирования» : учеб.-метод. пособие / С. В. Колосов, И. Н. Коренская, И. В. Лущицкая; под общ. ред. С. В. Колосова. – Минск : БГУИР, 2012. – 128 с.

ISBN 978-985-488-773-9.

В практикуме приведены краткие теоретические сведения по основам программирования в среде DELPHI. Рассмотрены приемы работы со средой визуального программирования. Приведены примеры решения тестовых задач по восьми тематическим разделам, а также после каждой темы приведено 30 индивидуальных заданий. Практикум предназначен для начального обучения студентов основам алгоритмизации и программирования в среде визуального программирования Delphi.

УДК 004. 43(076.5)
ББК 32.973.26-018.1я73

ISBN 978-985-488-773-9

Содержание

Лабораторная работа № 1. Программирование линейных алгоритмов

Цель работы: научиться составлять простейшие программы в среде DELPHI: написать и отладить программу линейного алгоритма.

1.1. Интегрированная среда разработчика DELPHI

Среда DELPHI визуально реализуется в виде нескольких одновременно раскрытых на экране монитора окон. Количество, расположение, размер и вид окон может меняться программистом в зависимости от его текущих нужд, что значительно повышает производительность работы. При запуске DELPHI на экране появляются пять окон (рис. 1.1).

- главное – **Delphi 7;**
-стартовая форма – **Form1**;
-редактор свойств объектов – **Object Inspector**;
-просмотр списка объектов – **Object TreeView**;
-редактор кода – **Unitl.pas**.

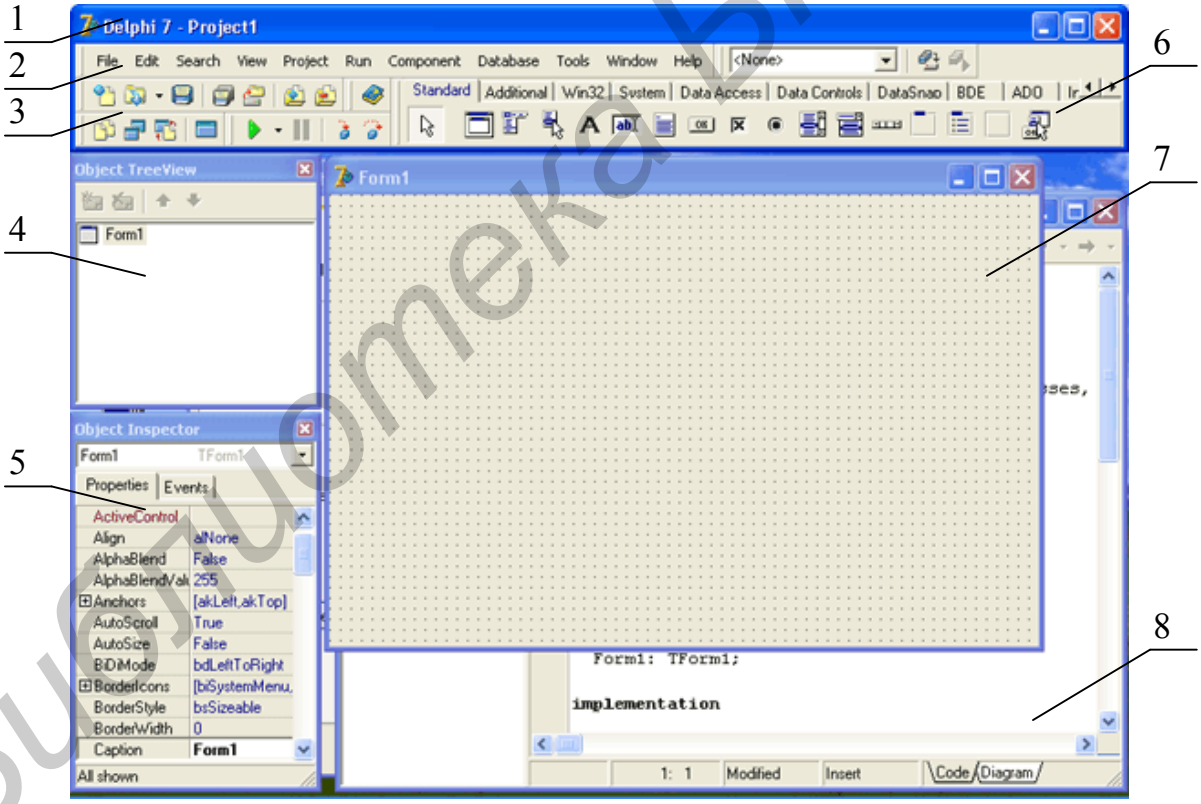


Рис. 1.1. Вид экрана после запуска Delphi:

1 – главное окно; 2 – основное меню; 3 – значки основного меню;
4 – окно просмотра дерева объектов; 5 – окно инспектора объектов;
6 – палитра компонентов; 7 – окно пустой формы; 8 – 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. Через меню компонентов осуществляется доступ к набору стандартных сервисных программ среды DELPHI, которые описывают некоторый визуальный элемент (компонент), помещенный программистом в окно формы. Каждый компонент имеет определенный набор свойств (параметров), которые можно задавать. *Например* цвет, заголовок окна, надпись на кнопке, размер и тип шрифта и др.

Окно инспектора объектов (вызывается с помощью клавиши **F11**) предназначено для изменения свойств выбранных компонентов и состоит из двух страниц. Страница **Properties** (Свойства) предназначена для изменения необходимых свойств компонента, страница **Events** (События) – для определения реакции компонента на то или иное событие (*например* нажатие определенной клавиши или щелчок «мышью» по кнопке).

Окно формы представляет собой проект Windows-окна программы. В это окно в процессе написания программы помещаются необходимые компоненты. Причем при выполнении программы помещенные компоненты будут иметь тот же вид, что и на этапе проектирования.

Окно программного кода предназначено для просмотра, написания и редактирования кода программы. В системе DELPHI используется язык программирования Object Pascal. При первоначальной загрузке в окне кода программы находится код, содержащий минимальный набор операторов для нормального функционирования пустой формы в качестве Windows-окна. При помещении некоторого компонента в окно формы код программы автоматически дополняется описанием необходимых для его работы библиотек стандартных программ (раздел **Uses**) и типов переменных (раздел **Type**).

Программа в среде DELPHI составляется как описание алгоритмов, которые необходимо выполнить, если возникает определенное событие, связанное с формой (*например* щелчок «мыши» на кнопке – событие **OnClick**, создание формы – **OnCreate**). Для каждого обрабатываемого на форме события с помощью страницы **Events** инспектора объектов в коде программы организуется процедура (**Procedure**), между ключевыми словами **Begin** и **End** которой программист записывает на языке Object Pascal требуемый программный код.

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

1.2. Структура программ DELPHI

Delphi для каждого приложения создает несколько файлов со следующими расширениями:

-*.dpr – файл описания проекта, где описываются все формы проекта (**Project1.dpr**) и запускаеся само приложение (**Application.run**). Для просмотра кода этого файла надо выполнить: **Project – View Source**;

-*.pas – файл модуля **Unit**, который является кодом программы для данной формы **Form1** (**Unit1.pas**);

-*.dfm – файл описания формы и ее компонент (**Unit1.dfm**). Он может храниться как в виде бинарного файла, так и в виде текстового файла;

-*.res – ресурсный файл, в котором хранятся значки, картинки, меню, константы, помещаемые на форму (**Project1.res**);

-*.dof – файл настроек проекта (**Project1.dof**);

-*.dcu – результат трансляции модуля с расширением ***.pas**, т.е. программный код модуля, представленный в машинных кодах;***.exe** – результат редактирования программы, т.е. объединение всех модулей ***.dcu** в одну готовую к выполнению программу.

При выполнении лабораторных работ следует сохранять только файлы с расширениями ***.dpr**, ***.pas**, ***.dfm** и ***.res**. Остальные файлы являются рабочими и их можно не сохранять.

Модуль **Unit** является отдельной программной единицей, и результатом трансляции является машинный код, который записывается в файл с расширением ***.dcu**.

Структура модуля **Unit** может иметь следующий вид:

```
Unit Имя модуля;
Interface       // Интерфейсная часть модуля
Uses …;         // Имена подключаемых модулей
  // Объявления глобальных типов, констант, переменных,
  // заголовков процедур и функций, которые будут доступны в других
  // модулях, подключивших данный модуль
Implementation  // Секция реализации модуля
Uses …;         // Имена подключаемых модулей
  // Объявления внутренних констант, типов, переменных, процедур и функций,
  // доступных только внутри данного модуля. Здесь приводится реализация всех
  // процедур и функций, объявленных в интерфейсной секции модуля
Initialization  // Секция инициализации модуля (может отсутствовать)
  // В этой секции записываются операторы, которые будут выполнены
  // сразу после загрузки программы в память ПЭВМ. Секция инициализации
```

// будет выполняться в том порядке, в каком модули Unit описаны
// в основной программе в разделе Uses
Finalization // Секция **завершения** (может отсутствовать)
// Выполнение операторов этой секции происходит после окончания работы
// программы перед выгрузкой ее из оперативной памяти ЭВМ.
// Эта секция выполняется в обратном порядке по сравнению с порядком
// выполнения секций инициализации
End. // Конец модуля Unit.

1.3. Порядок выполнения задания

Составьте программу вычисления для заданных значений **x**, **y**, **z** *арифметического выражения* $u = tg^2(x + y) - e^{y-z}\sqrt{\cos x^2 + \sin z^2}$.

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

1.3.1. Настройка формы

Пустая форма в правом верхнем углу имеет кнопки управления. Они предназначенны:

– для свертывания формы в пиктограмму ;
– для разворачивания формы на весь экран ;
– возвращения к исходному размеру ;
– для закрытия формы .

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

1.3.2. Изменение заголовка формы

Новая форма имеет одинаковые имя (**Name**) и заголовок (**Caption**) – **Form1**. Имя формы менять не рекомендуется, так как оно входит в код программы.

Для изменения заголовка вызовите окно инспектора объектов (**F11**) и щелкните кнопкой мыши на форме. В **инспекторе объектов** найдите и щелкните мышью на вкладке **Properties** – свойство **Caption** . В выделенной строке наберите свои данные, *например* «*Лаб. раб. N1. Ст. гр. 740201 Иванов А.А.*».

1.3.3. Размещение строки ввода (TEdit)

Для ввода из формы в программу или вывода на форму информации, которая вмещается в одну строку, используют **строку ввода**, представляемую компонентом **TEdit**.

В данной программе с помощью компонента **TEdit** будут вводиться значения переменных **x, y**, **z** типа **Extended**.

Выберите в меню компонентов на вкладке Standard пиктограмму  . Щелкните мышью в том месте формы, где появится компонент. Вставьте три компонента **TEdit** на форму. Захватывая их «мышью», отрегулируйте их размеры и положение. Обратите внимание на то, что в коде программы появились три новые однотипные переменные **Edit1**, **Edit2**, **Edit3**. В каждой из этих переменных в свойстве **Text** будет содержаться строка символов (тип **String**) и отображаться в соответствующем окне **Edit**.

Так как численные значения переменных **x**, **y**, **z** имеют действительный тип, то для *преобразования строковой записи числа*, находящегося в **Edit1.Text**, *в действительное* используется стандартная функция **x:=StrToFloat(Edit1.Text)**;

Если исходные данные имеют целочисленный тип, *например* **Integer**, то применяется стандартная *функция преобразования из строки в целое число* – **y:=StrToInt(Edit1.Text)**.

При этом в записи числа не должно быть пробелов, а действительное число пишется с десятичной точкой (или запятой, что определяется настройкой ОС Windows).

С помощью инспектора объектов установите шрифт и размер символов, отображаемых в строке **TEdit** (свойство **Font**).

1.3.4. Размещение надписей (TLabel)

На форме (рис. 1.2) имеются четыре пояснительные надписи. Для нанесения таких надписей на форму используется компонент **TLabel**. Выберите в меню компонентов **Standard** пиктограмму  . Щелкните на ней мышью, а затем в том месте формы, где появится надпись **Label1**. Проделайте это для четырех надписей. Отрегулируйте их размер, предварительно выделив надпись на форме. В свойство **Caption** инспектора объектов введите строку поясняющего текста, *например* «*Введите значение X:*», а также выберите размер символов (свойство **Font**).

Обратите внимание на то, что в коде программы автоматически появились четыре новые переменные типа **TLabel**. В их свойствах **Caption** хранятся пояснительные строки.

1.3.5. Размещение многострочного окна вывода (TMemo)

Для вывода результатов работы программы обычно используется многострочный текстовый редактор в виде компонента с типом **TMemo**. Выберите в меню компонентов пиктограмму  и поместите компонент **TMemo** на форму. С помощью мыши отрегулируйте его размеры и местоположение. После установки с помощью инспектора объектов свойства **ScrollBars** – **ssBoth** в окне появятся вертикальная и горизонтальная полосы прокрутки.

Также в коде программы появится переменная **Memo1** типа **TMemo**. Информация, которая отображается построчно в окне типа **TMemo**, находится в массиве строк **Memo1.Lines**. Каждая строка имеет тип **String**.

Для очистки окна используется метод **Memo1.Clear**; Для добавления новой строки в окно применяется метод **Memo1.Lines.Add** (переменная типа **String**).

Для вывода числа, находящегося в переменной действительного или целого типа, его надо предварительно преобразовать к типу **String** и добавить в массив **Memo1.Lines**. *Например*, если переменная **u:=100;** – целого типа, то при использовании метода **Memo1.Lines.Add ('Значение u='+IntToStr(u));** в окне **Memo1** появится строка «*Значение u=100*». Если переменная **u:=256.38666;** – действительного типа, то с применением метода **Memo1.Lines.Add('Значение u='+FloatToStrF(u,ffFixed,8,2));** будет выведена строка: "*Значение u= 256,39*". В соответствии с заданным форматом **ffFixed,8,2** под все число отводится восемь позиций, из которых две позиции занимает его дробная часть.

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

1.3.6. Написание процедуры обработки события создания формы (FormCreate)

При запуске программы может возникать событие «создание формы» **OnCreate**. Создадим программу – обработчик этого события, которая заносит начальные значения переменных **x**, **y**, **z** в соответствующие строки **TEdit**, а в окне **TMemo** помещает строку с указанием номера группы и фамилии студента. Для этого дважды щелкнем мышью на любом свободном месте формы. На экране появится программный код, в котором автоматически прописывается заголовок процедуры – обработчика события создания формы: **Procedure TForm1.FormCreate(Sender:TObject)**. Между операторами **begin** и **end** можно вставлять операторы, необходимые для начальной инициализации формы (смотрите *пример*, приведенный ниже).

1.3.7. Написание процедуры обработки события нажатия кнопки (ButtonClick)

Поместите на форму кнопку, которая описывается компонентом **TButton**. Для этого выберите в меню компонентов вкладки **Standart** пиктограмму  и поместите компонент на форму. С помощью инспектора объектов измените заголовок **Caption** компонента **Button1** на «***Выполнить***» или какой-либо другой. Отрегулируйте положение и размер кнопки.

Дважды щелкните левой клавишей мышки на кнопке – появится код программы с заголовком процедуры обработчика события нажатия кнопки **Procedure TForm1.ButtonClick(Sender:TObject**);

В появившемся окне редактора кода программы наберите код этой процедуры, приведенный ниже в примере.

1.3.8. Запуск и работа с программой

Запустить программу можно тремя способами, нажав:

– **Run** в главном меню **Run** или

– клавишу **F9**, или

– пиктограмму  .

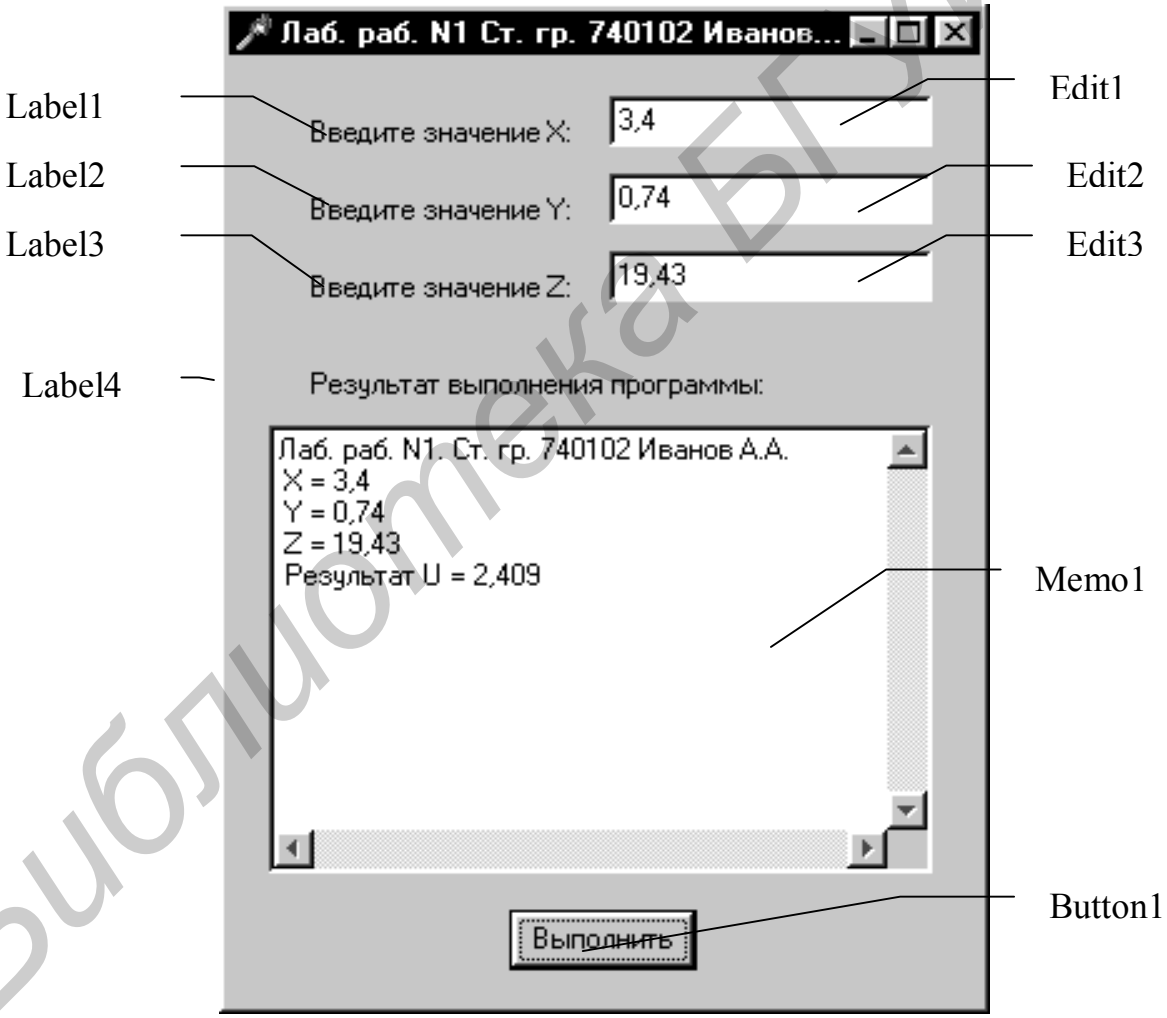


Рис. 1.2. Активная форма проекта

При этом происходит трансляция и, если нет ошибок, компоновка программы и создание единого загружаемого файла с расширением **.exe**. На экране появится активная форма программы (см. рис.1.2).

Для получения результатов вычисления с указанными на форме значениями после запуска программы надо нажать кнопку «***Выполнить***». В окне **Memo1** появится результат. При изменении исходных значений **x**, **y**, **z** в строках **TEdit** и нажатии кнопки «***Выполнить***» появятся новые результаты. Завершить работу программы можно выбрав меню **Run** – **Program Reset** или кнопку  на форме.

1.3.9. Код программы

```
unit Unit1;
interface
uses
      Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
      Dialogs, StdCtrls;
type
      TForm1 = class(TForm)
        Label1: TLabel;
        Edit1: TEdit;
        Label2: TLabel;
        Edit2: TEdit;
        Label3: TLabel;
        Edit3: TEdit;
        Label4: TLabel;
        Memo1: TMemo;
        Button1: TButton;
        procedure FormCreate(Sender: TObject);
        procedure Button1Click(Sender: TObject);
      private
        { Private declarations }
      public
        { Public declarations }
      end;
Var
  Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.FormCreate(Sender: TObject);
begin
   Edit1.Text:='3,4';          // Начальное значение x
   Edit2.Text:='0,74';       // Начальное значение  y
   Edit3.Text:='19,43';        // Начальное значение  z
   Memo1.Clear;                // Очистка окна редактора Memo1
```

```
  // Вывод строки в многострочный редактор Memo1
    Memo1.Lines.Add ('Лаб. раб. N1. Ст. гр. 740102 Иванов А.А.');
end;

procedure TForm1.Button1Click(Sender: TObject);
 var
   x,y,z,a,b,c,u : Extended;
begin
   x:=StrToFloat(Edit1.Text);                  // Считывание значения x
   Memo1.Lines.Add(' X = '+Edit1.Text);        // Вывод x в окно Memo1
   y:=StrToFloat(Edit2.Text);                  // Считывание значения y
   Memo1.Lines.Add(' Y = '+Edit2.Text);        // Вывод y в окно Memo1
   z:=StrToFloat(Edit3.Text);                  // Считывание значения z
   Memo1.Lines.Add(' Z = '+Edit3.Text);        // Вывод z в окно Memo1
   a:=Sqr(Sin(x+y)/Cos(x+y)); // Вычисление арифметического выражения
   b:=Exp(y-z);
   c:=Sqrt(Cos(Sqr(x))+Sin(Sqr(z)));
   u:=a-b*c;
  // Вывод результата в окно Memo1
   Memo1.Lines.Add('Результат U= '+FloatToStrF(u,ffFixed,8,3));
   end;
end.
```

1.4. Индивидуальные задания

1. Составьте программу вычисления выражения согласно указанному преподавателем варианту.

2. Уточните условие задания, количество, наименование, типы исходных данных. В соответствии с этим установить необходимое количество компонент **TEdit**, тексты заголовков на форме, размеры шрифтов, а также типы переменных и функции преобразования при вводе и выводе результатов. С помощью инспектора объектов изменить цвет формы, шрифт выводимых символов.

1. $t = \dfrac{2\cos\left(x - \dfrac{2}{3}\right)}{0{,}5 + \sin^2 y}\left(1 + \dfrac{z^2}{3 - z^2/5}\right)$

при **x** = 14,26, **y** = – 1,22, **z** = $3{,}5\cdot10^{-2}$. Ответ: t = 0,7492.

2. $u = \dfrac{\sqrt[3]{9 + |x - y|^2}}{x^2 + y^2 + 2} - e^{|x-y|}\mathrm{tg}^3 z$

при **x**= – 4,5, **y** = $0{,}75\cdot10^{-4}$, **z** = $0{,}845\cdot10^{2}$. Ответ: u = 3,51460.

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

при **x** = $3{,}74\cdot10^{-2}$, **y** = – 0,825, **z** = $0{,}16\cdot10^{2}$. Ответ: v = 1,0554.

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** = $0{,}4\cdot10^{4}$, **y** = – 0,875, **z** = $-0{,}475\cdot10^{-3}$. Ответ: w = 1,9873.

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

при **x** = – 15,246, **y** = $4{,}642\cdot10^{-2}$, **z** = 21. Ответ: α = – 182,038.

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

при **x** = $16{,}55\cdot10^{-3}$, **y** = – 2,75, **z** = 0,15. Ответ: β = – 40,6307.

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

при **x** = 0,1722, **y** = 6,33, **z** = $3{,}25\cdot10^{-4}$. Ответ: γ = – 205,306.

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

при **x** = $-2{,}235\cdot10^{-2}$, **y** = 2,23, **z** = 15,221. Ответ: φ = 39,3741.

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

при **x** = $1{,}825\cdot10^{2}$, **Y** = 18,225, **z** = $-3{,}298\cdot10^{-2}$. Ответ: ψ = 1,2131.

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

при **X** = $3{,}981\cdot10^{-2}$, **y** = $-1{,}625\cdot10^{3}$, **Z** = 0,512. Ответ: a = 1,26185.

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

при **x** = 6,251, **y** = 0,827, **z** = 25,001. Ответ: b = 0,7121.

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

при **x** = 3,251, **y** = 0,325, **z** = $0{,}466\cdot10^{-4}$. Ответ: c = 4,23655.

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

при **x** = 17,421, **y** = $10{,}365\cdot10^{-3}$, **z** = $0{,}828\cdot10^{5}$. Ответ: f = 0,33056.

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

при **x** = $12{,}3\cdot10^{-1}$, **y** = 15,4, **z** = $0{,}252\cdot10^{3}$. Ответ: g = 82,8257.

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

при **x** = 2,444, **y** = $0{,}869\cdot10^{-2}$, **z** = $-0{,}13\cdot10^{3}$. Ответ: h = – 0,4987.

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

при **x** = $3{,}999\cdot10^{-2}$, **y** = – 6,011, **z** = $0{,}245\cdot10^{3}$. Ответ: S = 1,000.

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

при **x** = 1,78, **y** = 3,83, **z** = $13{,}57\cdot10^{2}$. Ответ: u = 1,5525.

18. $r=e^{|y-x|}\mathrm{arctg}(x)-\dfrac{1}{4}\mathrm{arctg}(x)\ln(y-\sqrt{x})(x-\dfrac{y}{2})$

при **x** = 1,471, **y** = 4,62, **z** = $17{,}247\cdot10^{5}$. Ответ: r = 37,0234.

19. $t=\sin^2(\mathrm{arctg}(z)\cdot x^3)+\ln(y-\sqrt{|x|})(x+\dfrac{y}{2})$

при **x** = $2{,}126\cdot10^{-2}$, **y** = 4,62, **z** = $18{,}52\cdot10^{5}$. Ответ: t = 3,4930.

20. $c=(3x)^y-\dfrac{y(\mathrm{arctg}z-\dfrac{\pi}{6})}{\sin|x|+\dfrac{1}{y^2+1}}+3(x+z^2)$

при **x**= 7,5, **y** = $23{,}8\cdot10^{-4}$, **z** = 1,4. Ответ: c = 29,3871.

21. $u=\ln(y-\sqrt{|x|})(x+\dfrac{y}{2})^3+\dfrac{1-\sin z}{1+\cos z}$

при **x** = $3{,}466\cdot10^{-3}$, **y** = 2,743, **z**= 1,43. Ответ: u = 2,5752.

22. $w = \ln\dfrac{1+\sqrt{\sin z}}{1-\sqrt{\cos z}} + 2\operatorname{arctg}\sqrt[3]{x+y}\cdot\dfrac{1+2y}{x^4}$

при **x** = $1{,}499\cdot10^{2}$, **y** = 0,1174, **z** = 0,43. Ответ: w = 3,5642.

23. $v = 2^{\arcsin 3z} + (1-\arccos 3z)^2\cdot\dfrac{x}{e^{y}+1}$

при **x** = 2,85, **y** = $6{,}23\cdot10^{-3}$, **z** = 0,14. Ответ: v = 1,3773.

24. $m = \dfrac{\cos z - \dfrac{z}{y-x}}{1+(y-x)^2} + \cos\dfrac{x}{\sqrt{2}} + 5^{\sqrt{x}}\dfrac{x}{\sqrt{y}}$

при **x** = 0,49722, **y** = $1{,}3343\cdot10^{5}$, **z** = 0,234. Ответ: m = 0,9431.

25. $p = \sqrt[3]{x^2}\,\dfrac{1-x}{1+y^2}\sin^3 z\cos^3 z + \dfrac{x}{\sqrt{1-x^4}}$

при **x** = 0,1231, **y** = $2{,}14\cdot10^{-3}$, **z** = 1,784. Ответ: p = 0,1212.

26. $n = \dfrac{e^{arctgz} + y\ln(1+x^2)+1}{1+y^2} + \dfrac{y^2}{\sqrt[3]{x^2+1}}$

при **x** = $3{,}7911\cdot10^{4}$, **y** = $8{,}412\cdot10^{-2}$, **z** = 1,683. Ответ: n = 5,5487.

27. $q = \dfrac{\sin z\cos y}{\sqrt{2-\sin^4 z}} + \dfrac{y^2}{x^2-2}\cdot e^{x}\sqrt{x-y^{x}}$

при **x** = 8,634, **y** = $1{,}52\cdot10^{-1}$, **z** = 0,437. Ответ: q = 0,8679.

28. $k = \ln\dfrac{1+\sqrt{\sin z}}{1-\sqrt{x+y}} + 2\operatorname{arctg}z - \ln\dfrac{|x-2|}{(y+2)^3}$

при **x** = $6{,}2\cdot10^{4}$, **y** = $8{,}234\cdot10^{-3}$, **z** = 1,6211. Ответ: k = – 6,9070.

29. $d = \sqrt[3]{\sin^2 z} + \dfrac{1}{\cos^2 z}\cdot\dfrac{\sqrt{x^3+1}}{y^4+2x+|y-5|}$

при **x** = $31{,}14\cdot10^{-2}$, **y** = 2,673, **z** = 0,245. Ответ: d = 0,0261.

30. $b = \dfrac{1}{\sqrt{3}}\ln\dfrac{\operatorname{tg}\dfrac{z}{2}+2-y^3}{\operatorname{tg}\dfrac{y}{2}+2+x^3}\cdot\dfrac{x}{e^{x}+1} + \dfrac{y}{y+3}$

при **x** = 42,26, **y** = $17{,}78\cdot10^{3}$, **z** = 4,45. Ответ: b = 17,0373.

1.5. Задания повышенной сложности

1. Найти сумму цифр заданного четырехзначного числа.

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

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

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

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

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

7. Из заданного четырёхзначного числа получить двузначное число, удалив из исходного четырёхзначного числа цифры сотен и единиц (*например*: 2783 → 28).

Лабораторная работа № 2. Программирование разветвляющихся алгоритмов

***Цель работы**:* научиться пользоваться простейшими компонентами организации переключений (**TCheckBox**, **TRadioGroup**). Составить блок-схему, написать и отладить программу разветвляющегося алгоритма.

2.1. Операции сравнения и логические операции

Операции сравнения используются при работе с *двумя* операндами и возвращают значение **True** (**1**), если результат сравнения – истина, и **False** (**0**), если результат сравнения – ложь. В языке Pascal определены следующие *операции сравнения*: **<** (меньше**), <=** (меньше или равно), **>** (больше), **>=** (больше или равно), **=** (равно), **<>** (не равно).

Логические операции применяются только к данным логического типа (**Boolean**). Существуют следующие *логические операции*:

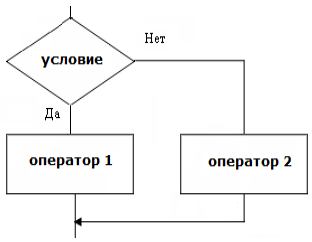
And – *логическое «и»* (пересечение),
Or – *логическое «или»* (объединение),
Xor – *исключающее «или»,*
Not – *отрицание* или *логическое «не».*

Примеры : $0 \le y \le 20$ => (0**<=**y) **And** (y**<=**20)

$x \ne 0$ или $y = 0$ => (x**<>**y) **Or** (y**=**0)
x – четное => **Not Odd**(x)
y кратно 3 => y **Mod** 3 **=** 0

2.2. Оператор условной передачи управления If

Оператор **If** проверяет результат логического выражения или значение переменной типа **Boolean** и организует разветвление вычислений.

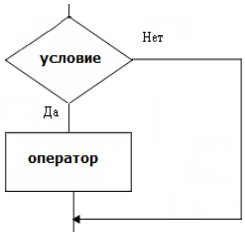


Форматы оператора If:

1. Полная форма:

If *условие* **Then** *оператор_1*
Else *оператор_2**;***

Если *условие* (*логическое выражение)* истинно, то выполняется *оператор_1*, иначе – *оператор_2*.

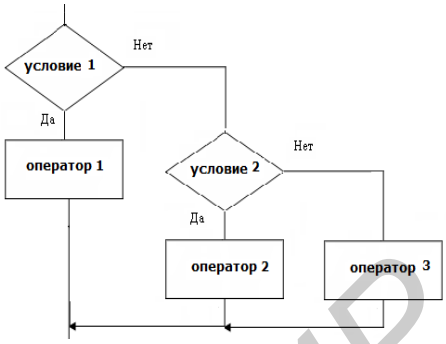


Пример: найти наибольшее из значений a и b, т.е. Max (a, b):

```
If  a > b Then Max := a
          Else  Max := b;
```

2. Сокращенная форма

If *условие* **Then** *оператор;*



Если *условие* (*логическое выражение)* истинно, то выполняется *оператор.*

Пример: найти |a|

```
If  a < 0 Then a:= -a;
```

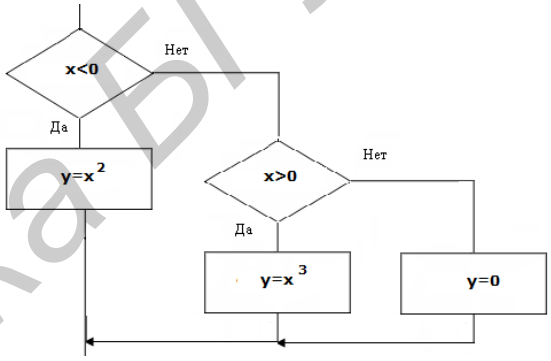
3. Вложенная форма:

If *условие_1* **Then** *оператор_1*
Else If *условие_2* **Then** *оператор_2*
Else *оператор_3;*

Если *условие_1* истинно, то выполняется *оператор_1*, иначе, если *условие_2* истинно, выполняется *оператор_2*, иначе выполняется *оператор_3.*

Пример:

$$y=\begin{cases} x^2, & x<0 \\ x^3, & x>0 \\ 0 & x=0 \end{cases}$$




```
If  x < 0 Then y := Sqr (x)
        Else If x>0 Then y:=Power (x, 3)
                    Else y := 0;
```

2.3. Оператор выбора Case

Оператор **Case** применяется для выбора одного из возможных вариантов.

Формат оператора:

Case *переменная_выбора* **Of**
значение 1: оператор 1;
значение 2: оператор 2;
...
значение n: оператор n;
Else *оператор n+1*
End;

переменная_выбора, значение 1,..., значение n – константа, переменная или выражение **порядкового** типа (целого, символьного, логического, интервального или перечисляемого типов). Значения *переменной выбора* и *значение 1,..., значение n* должны быть **одинакового типа**.

При выполнении оператора **Case** сначала вычисляется значение *переменной выбора,* которое последовательно сравнивается со *значением 1,..., значени-*

ем n и при совпадении значений выполняется соответствующий *оператор,* иначе выполняется *оператор n+1*. Конструкция *Else* может отсутствовать.

Пример: выбрать функцию f(x): x^2, |x|, $\sqrt{x}$

```
Case k Of
    1: f := Sqr (x);
    2: f := Abs (x);
    3: f := Sqrt(x);
    Else Memo1.Lines.Add ('F(x) не задана!')
End;
```

2.4. Оператор безусловной передачи управления GoTo

Оператор безусловной передачи управления имеет следующий вид:

GoTo <метка>;

Метка – это идентификатор, описанный в разделе меток **Label**.

Например:

```
Label M1;
……
GoTo M1;
……..
M1: y:=Sin(x);
……
```

Оператор **GoTo** передает управление оператору с указанной меткой, в данном случае – **M1**. Оператор, следующий за **GoTo**, обязательно должен иметь метку, иначе он никогда не получит управление. По возможности следует избегать использования оператора **GoTo**, так как это приводит к созданию неэффективных программ. Перескок внутри программы приводит к тому, что нужно заново обновлять очередь команд, готовых к выполнению в процессоре, и перенастраивать его управляющие регистры. Чем меньше в программе операторов **Goto**, тем выше квалификация программиста.

2.5. Кнопки-переключатели в Delphi

При написании программ 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** и **TRadioGroup**.

Компонент **TCheckBox** организует кнопку *независимого* переключателя, с помощью которой можно задать свой вариант решения (типа *да/нет*). В программе состояние кнопки связано со значением булевской переменной, которая проверяется с помощью оператора **If**.

Компонент **TRadiogroup** организует группу кнопок – *зависимых* переключателей. При нажатии одной из кнопок группы все остальные кнопки от-

ключаются. В программу передается номер включенной кнопки (0,1,2,..), который анализируется с помощью оператора **Case**.

2.6. Порядок выполнения задания

Вычислить значение выражения

$$s=\begin{cases}\left|f(x)\cos(x)\right|+\ln(y), & \left|x\cdot y\right|>10\\ e^{2f(x)+y}, & 3<\left|x\cdot y\right|\le 10\\ \sqrt{\left|f(x)\right|}+2\mathrm{tg}(y), & \text{иначе}\end{cases}$$

При выполнении задания предусмотреть:

- *выбор вида функции* f(x): sh(x), x^2 или e^x,
- *вывод информации о выбранной ветви вычислений,*
- *возможность округления полученного результата.*

Составить блок-схему алгоритма.

2.6.1. Создание формы проекта

Разместите на форме необходимые компоненты в соответствии с рис. 2.1.

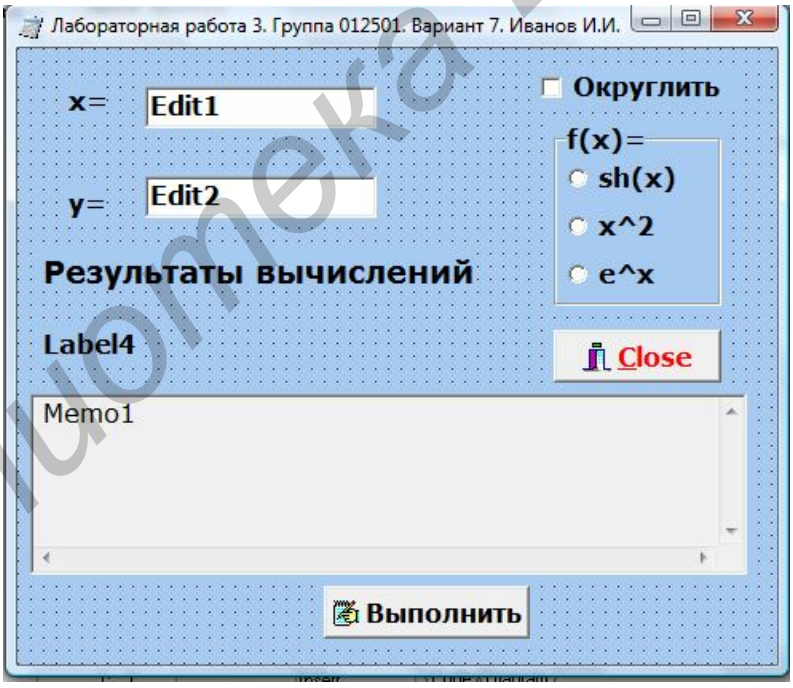


Рис. 2.1. Форма проекта

Для этого:

1) подпишите форму (свойство **Caption**);

2) установите на форму компоненты **TLabel** для вывода пояснений и информации о том, по какой ветви происходило вычисление;

3) установите на форму компоненты **TEdit** для ввода значений переменных **x** и **y**;

4) установите на форму компонент **TMemo** для вывода результата, добавьте в него обе линейки прокрутки (свойство **ScrollBars** – **ssBoth**);
5) установите на форму 2 кнопки **TBitBtn**: для одной из них задайте свойство **Kind** – **bkClose** (стандартная кнопка, закрывающая проект), другую кнопку подпишите **Выполнить** (свойство **Caption**), установите на нее значок (свойство **Glyph** и загрузите файл с расширением ***.bmp**);
6) измените для компонент **TEdit**, **TBitBtn**, **TMemo** вид курсора на **crHandPoint** (свойство **DragCursor**);
7) измените размеры и цвет формы (свойство **Color**), задайте шрифт для компонент (свойство **Font**).

2.6.2. 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 **Округлить**. В коде программы появилась переменная **CheckBox1** типа **TCheckBox**. Теперь в зависимости от того, нажата кнопка или нет, булевская переменная **CheckBox1.Checked** будет принимать значения **True** или **False**.

2.6.3. 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 **F(x)**. Для того чтобы разместить на компоненте кнопки, необходимо свойство **Columns** установить равным единице (кнопки размещаются в одном столбце). Дважды щелкните по правой части свойства **Items** мышью, появится строчный редактор списка заголовков кнопок. Наберите три строки с именами: в первой строке – sh(*x*), во второй – x^2, в третьей – e^x, нажмите **ОК**.

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

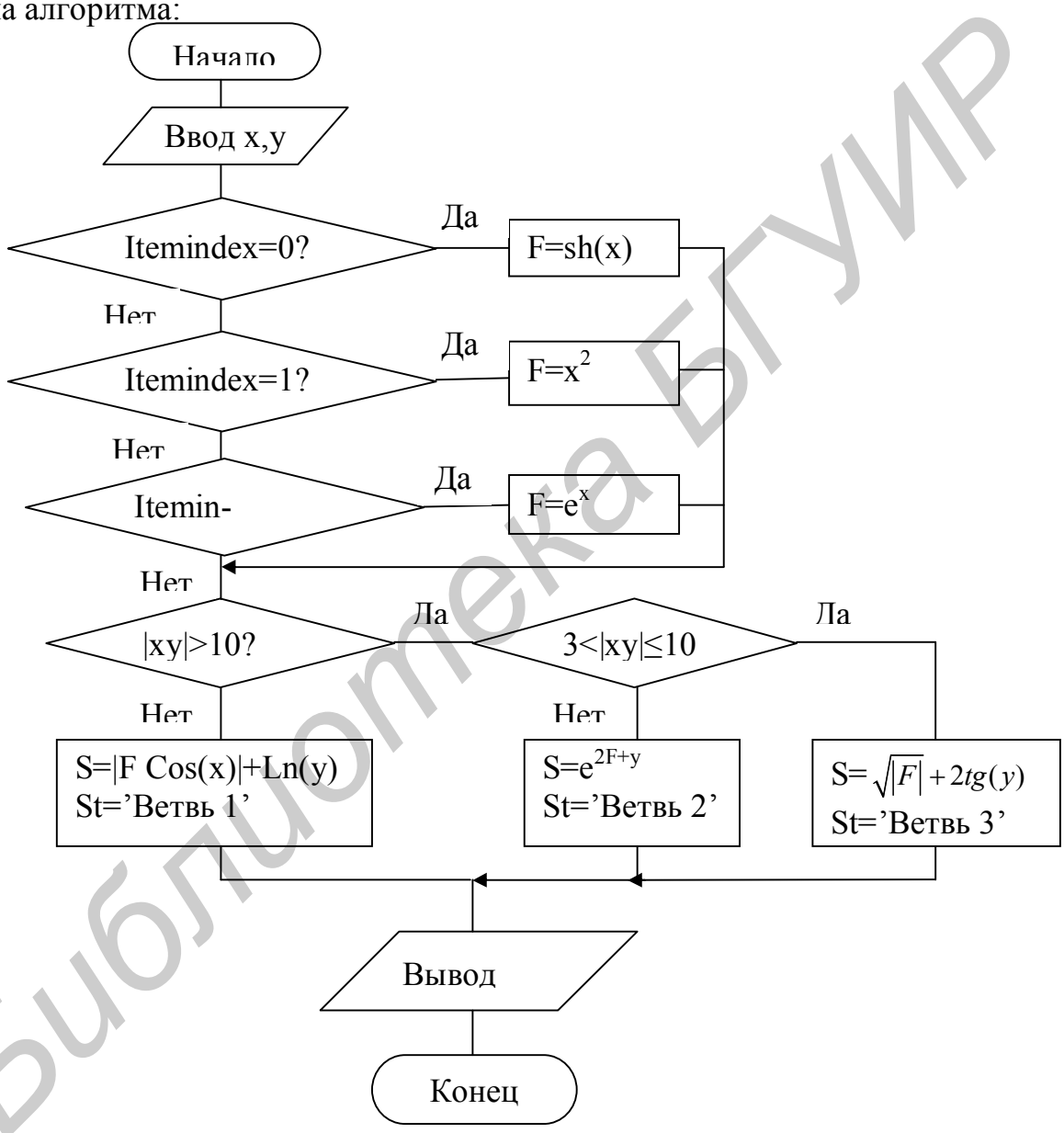
Обратите внимание на то, что в коде программы появилась переменная **RadioGroup1** типа **TRadioGroup**. Теперь при нажатии одной из кнопок группы в переменной целого типа **RadioGroup1.ItemIndex** будет находиться номер нажатой клавиши (отсчитывается от нуля), что используется в коде приведенной ниже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

2.6.4. Создание обработчиков событий FormCreate и ButtonClick

Создайте обработчик события **FormCreate**. Для этого дважды щелкните левой клавишей мыши в свободном месте формы. На экране появится код, в котором автоматически создастся заголовок процедуры – обработчика события создания формы: **Procedure TForm1.FormCreate(Sender:TObject)**.

Между **begin**…**end** вставьте код программы, соответствующий: вводу начальных значений переменных **x** и **y** из соответствующих строк ввода **TEdit**, очистке от надписи компонента **Label4,** выводу в окно **TMemo** строки с указанием темы лабораторной работы, номера группы и фамилии студента.

Обработчик события нажатия кнопки ButtonClick создается аналогично (**Procedure TForm1.ButtonClick(Sender:TObject);**). Ниже представлена блок-схема алгоритма:



2.6.5. Код программы

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;

type
  TForm1 = class(TForm)
    CheckBox1: TCheckBox;
    RadioGroup1: TRadioGroup;
    Memo1: TMemo;
    Button1: TButton;
    Edit1: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Edit2: TEdit;
    Label3: TLabel;
    Edit3: TEdit;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}
procedure TForm1.FormCreate(Sender: TObject);
begin
    Edit1.text:='0,1';          // Начальное значение X
    Edit2.text:='0,793';        // Начальное значение Y
    Label4.Caption:=' ';        // Очистка компонента
    Memo1.Clear;                // Очистка окна редактора Memo1
// Вывод строк текста в многострочный редактор Memo1
    Memo1.Lines.Add('Программирование разветвляющихся
алгоритмов');
    Memo1.Lines.Add('Выполнил студент группы 012501 Иванов И.И.');
    RadioGroup1.ItemIndex:=1;       // Выбор функции f(x): x2
    end;

procedure TForm1.Button1Click(Sender: TObject);
Var                 // Описание переменных
```

```
   x, y, xy, s, f: Extended;
   st: String;      // Строка для вывода сообщения номера ветви вычисления
begin
   x:=StrToFloat(Edit1.Text);                    // Ввод исходных данных и
   Memo1.Lines.Add(' x='+Edit1.Text);            // их вывод в окно Memo1
   y:=StrToFloat(Edit2.Text);
   Memo1.Lines.Add(' y='+Edit2.Text);
   Case RadioGroup1.ItemIndex Of      // Проверка номера нажатой кнопки и
      0: f:= sinh (x);                // выбор соответствующей ей функции
      1: f:=sqr(x);
      2: f:=exp(x);
      End;
   xy:=x*y;
   If Abs(xy)>10 Then Begin
      s:=Abs(f*Cos(x))+Ln(y);
      st:= 'Ветвь 1';
      End Else If (3>xy) And (xy<=10) Then Begin
      s:=Exp(2*f+y);
      st:= 'Ветвь 2';
      End Else Begin
      s:=Sqrt(Abs(f))+2*Tan(y);
      st:= 'Ветвь 3';
      End;
   Label4.Caption:= st;       // Вывод сообщения о ветви вычисления в Label4
   // Вывод результата в окно Memo1
   If CheckBox1.Checked Then
   Memo1.Lines.Add('Rezult s='+IntToStr(Round(s)))
   Else Memo1.Lines.Add('Rezult s= '+FloatToStrF(s,ffGeneral,8,2));
   end;
end.
```

2.7. Индивидуальные задания

1. Составить программу вычисления выражения согласно указанному варианту.

2. При выполнении задания предусмотреть:

– выбор вида функции f(x): x^2, e^x, sh(x);

– вывод информации о выбранной ветви вычислений в компонент TLabel;

– возможность округления результата.

Составить блок-схему алгоритма.

1. $s=\begin{cases}\left(f(x)+y\right)^2-\sqrt[3]{\left|f(x)\right|}, & x\cdot y>0\\ \left(f(x)+y\right)^2+\sin(x), & x\cdot y<0\\ \left(f(x)+y\right)^2+y^3, & x\cdot y=0\end{cases}$

2. $s=\begin{cases}\ln\left(f(x)\right)+\sqrt[3]{\left|f(x)\right|}, & x/y>0\\ \ln\left|f(x)/y\right|\cdot\left(x+y\right)^3, & x/y<0\\ \left(f(x)^2+y\right)^3, & \text{иначе}\end{cases}$

3. $s=\begin{cases} f(x)^2+\sqrt[3]{y}+\sin(y), & x-y=0 \\ (f(x)-y)^2+\ln(|x|), & x-y>0 \\ (y-f(x))^2+\operatorname{tg}(y), & \text{x-y<0} \end{cases}$

4. $s=\begin{cases} \sqrt[3]{|f(x)-y|}+\operatorname{tg}(f(x)), & x>y \\ (y-f(x))^3+\cos(f(x)), & x<y \\ (y+f(x))^2+x^3, & \text{иначе} \end{cases}$

5. $s=\begin{cases} y\sqrt{|f(x)|}+3\sin(x), & x>y \\ x\sqrt{|f(x)|}, & x<y \\ \sqrt[3]{|f(x)|}+x^3/y, & \text{иначе} \end{cases}$

6. $s=\begin{cases} e^{f(x)-|y|}, & 0{,}5<x\cdot y<10 \\ \sqrt[3]{|f(x)+y|}, & 0{,}1<x\cdot y<0{,}5 \\ 2f(x)^2, & \text{иначе} \end{cases}$

7. $s=\begin{cases} e^{f(x)}, & 1<x\cdot b<10 \\ \sqrt[3]{|f(x)+4y|}, & 12<x\cdot b<40 \\ y\cdot f(x)^2, & \text{иначе} \end{cases}$

8. $s=\begin{cases} \left(f(x)^2+y\right)^3, & x/y<0 \\ \ln|f(x)/y|+x/y, & x/y>0 \\ \sqrt[3]{|\sin(y)|}, & \text{иначе} \end{cases}$

9. $s=\begin{cases} 2f(x)^3+3y^2, & x>|y| \\ |f(x)-y|, & 3<x<|y| \\ \sqrt[3]{|f(x)-y|}, & \text{иначе} \end{cases}$

10. $s=\begin{cases} \ln\left(|f(x)|+|y|\right), & |x\cdot y|>10 \\ e^{f(x)+y}, & |x\cdot y|<10 \\ \sqrt[3]{|f(x)|}+y, & \text{иначе} \end{cases}$

11. $s=\begin{cases} \operatorname{tg}(f(x))+\dfrac{x}{\sqrt[3]{y}}, & x\cdot y>0 \\ \ln|f(x)^2\cdot y|, & x\cdot y<0 \\ f(x)^3+\sin^2(y), & \text{иначе} \end{cases}$

12. $s=\begin{cases} \operatorname{tg}(x)+f(x)^2, & y>2\cdot x \\ |f(x)+y|^3, & y<2\cdot x \\ \sqrt[3]{x}\cdot\sin(x), & \text{иначе} \end{cases}$

13. $s=\begin{cases} (f(x)+\ln(|y|))^3, & x/y>0 \\ 2/3+\ln(|\sin(y)|), & x/y<0 \\ \sqrt[3]{f(x)^2}+y, & \text{иначе} \end{cases}$

14. $s=\begin{cases} \ln\left(f(x)\right)^3, & x^3>0 \\ \operatorname{tg}(x^3)+f(x), & x^3<0 \\ \sqrt[3]{|y^3-x^2|}, & \text{иначе} \end{cases}$

15. $s=\begin{cases} (f(x)^2+y^3)/x, & f(x)>0 \\ \ln|f(x)^3|+\cos(y), & f(x)<0 \\ \sqrt[3]{\sin^2(y)}, & \text{иначе} \end{cases}$

16. $s=\begin{cases} y\sqrt{f(x)}, & y-\textit{нечетное},\ x>0 \\ y/2\sqrt{|f(x)|}, & y-\textit{четное},\ x<0 \\ \sqrt{|yf(x)|}, & \text{иначе} \end{cases}$

17. $$s=\begin{cases}\ln\left(f(x)\right)+\left(f(x)^2+y\right)^3, x/y>0\\ \ln\left|f(x)/y\right|+\left(f(x)+y\right)^3, x/y<0\\ \left(f(x)^2+y\right)^3, \quad x=0\\ 0, \quad y=0.\end{cases}$$

18. $$s=\begin{cases}\sin\left(5f(x)+3y\left|f(x)\right|\right), -1<y<x\\ \cos\left(3f(x)+5y\left|f(x)\right|\right), \quad x>y\\ \left(f(x)+y\right)^2, \quad x=y.\end{cases}$$

19. $$s=\begin{cases}\left(\frac{5}{6}f^2(x)+2y^2\sqrt{\left|f(x)\right|}\right), y\cdot x<0\\ \cos\left(\frac{3f(x)}{2y}+5\sqrt[3]{y^2}\right)^3, \quad x\cdot y>0\\ \left(f^4(x)+3y\right)^2, \quad \text{иначе}\end{cases}$$

20. $$s=\begin{cases}\sqrt{\left|\sqrt[3]{f(x)}-\frac{2y+1}{y^3+2}\right|^3}-1, y>x\\ \ln^3\left|y^2+f(x)\right|, \quad x>y\\ \left(f^3(x)-tgy\right)^2, \quad \text{иначе}\end{cases}$$

21. $$s=\begin{cases}\left(f(x)-2\cos^2 y^3\right)e^{f(x)+y}, -1<y<1\\ \sqrt{\left(\left|\frac{3-f(x)}{y^2f^4(x)}\right|+5\right)^3}-e, \quad x\cdot y>0\\ \ln^3 f^2(x)+1, \quad \text{иначе}\end{cases}$$

22. $$s=\begin{cases}\frac{1}{2}e^{f^2(x)}\frac{\sqrt{\left|f(x)\right|}}{3y^3}, y\cdot x>0\\ ctg^2(y^3+1), \quad x\cdot y=0\\ \ln\left(f(x)+3\right)^2, \quad \text{иначе}\end{cases}$$

23. $$s=\begin{cases}\frac{f(x)^2+1}{2}+\frac{\left|y^3\right|}{x+1}, \quad x>0\\ \ln\left|f(x)^3+\frac{2}{3}\right|+e^{2\cos y}, \quad x<0\\ \sqrt[3]{f^2(x^3)}-\left(\frac{y}{2}\right)^2, \quad \text{иначе}\end{cases}$$

24. $$s=\begin{cases}\sin^3\left(y^2\left|f(x)\right|\right), -2<y<x\\ e^{\cos\left(2y\left|f(x)\right|\right)}-\ln x^2y^2, \quad x>y\\ \sqrt{\left|f(x)+y\right|}+2, \quad \text{иначе}\end{cases}$$

25. $$s=\begin{cases}\sqrt{\ln(f(x)^2+y^3)}-\frac{1+x}{2y}, \quad 2\cdot y>0\\ e^{\left|f(x)^3+1\right|}+\cos\left|x-y\right|, \quad 2\cdot y<0\\ \sqrt[3]{f^2(x)-2\left|x-y\right|}, \quad \text{иначе}\end{cases}$$

26. $$s=\begin{cases}\sin\left(\frac{2f(x)}{3y\left|f^3(x)\right|}+1\right), y\cdot f(x)<0\\ 2\ln\left(\left|f(x)\right|+3y^2\right), \quad y\cdot f(x)>0\\ e^{\left(f(x)+y\right)^2}, \quad \text{иначе}\end{cases}$$

27. $$s=\begin{cases}\dfrac{1-(\sqrt{f(x)}+y^3)^2}{2f(x)}, & f(x)>0\\ \ln\left|f(x)^3+\cos^2 y\right|, & f(x)<0\\ \sqrt[3]{x^2y^2-1}+e^{-2y}, & \text{иначе}\end{cases}$$

28. $$s=\begin{cases}\ln\left(3\left|yf^2(x)\right|+2\right), & -2<y<2\\ e^{\left(\frac{3}{2}f(x)-y\right)^3}+3x^3y^3, & x>y\\ \sqrt{f^2(x)+2|y|}, & \text{иначе}\end{cases}$$

29. $$s=\begin{cases}\dfrac{x^2-e^{y^4}}{\sqrt{\left|f(x)+\sqrt[3]{y}\right|}}, & 1<y\cdot f(x)<10\\ \ln^2\left|1+arc\cos(y)\right|, & y>0\\ tgf(x)^3, & \text{иначе}\end{cases}$$

30. $$s=\begin{cases}\dfrac{1}{3}f^2(x)-\sqrt{2y}, & 10<y<2x\\ e^{xy^2}-\ln\left|x^3+\sqrt[3]{3yf(x)}\right|, & x>y\\ \cos\left(f(x)+\sqrt[3]{y^2}\right), & \text{иначе}\end{cases}$$

2.8. Задания повышенной сложности

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

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

3. Значения переменных a, b и c поменять местами так, чтобы оказалось a<=b<=c.

4. Для заданного номера года определить: является он високосным или нет (високосным является год, если его номер делится на 4, за исключением тех, которые делятся на 100 и не делятся на 400).

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

6. Дано натуральное k. Определить k-ю цифру в последовательности 11010010001000100000…, в которой выписаны подряд степени 10.

7. Для целого числа k от 1 до 99 вывести фразу «*мне* k *лет*», учитывая при этом, что при некоторых значениях k слово *«лет»* надо заменить на слово «*год*» или «*года*».

Лабораторная работа № 3. Программирование циклических алгоритмов

Цель работы: изучить простейшие средства отладки программ в среде DELPHI. Составить блок-схему, написать и отладить программу циклического алгоритма.

3.1. Операторы организации циклов

Под циклом понимается многократное выполнение одних и тех же операторов при различных значениях промежуточных данных. Число повторений может быть задано в явной или неявной формах.

Для организации повторений в Delphi предусмотрены три различных оператора цикла.

3.1.1. Оператор цикла -- For

Оператор применяется, когда переменная цикла имеет порядковый тип.

Существует два варианта *оператора* **For***:*

I. С шагом **1**:

For <переменная> := <нач. значение> **To** <кон. значение> **Do**
<оператор>;

<переменная> – переменная цикла, должна быть **порядкового** типа;

<нач. значение> и **<кон. значение>** – начальное и конечное значения переменной цикла, **по типу совместимые** с ней, иначе при неявном приведении типов возможны ошибки; могут быть константами, переменными, выражениями.

Для выполнения цикла необходимо соблюдать условие:

<нач. значение> ≤ <кон. значение>

Выполнение цикла **For**:

1. Вычисляется начальное значение **<нач. значение>** и присваивается **переменной** цикла;
2. Проверяется условие выполнения цикла:

<нач. значение> ≤ <кон. значение>

3. Если условие истинно (выполняется), то выполняется **<оператор>** и пункты 4–5, иначе (условие ложно) цикл завершается и выполняется оператор, стоящий после цикла;
4. Значение **переменной** увеличивается на единицу;
5. Происходит переход к п. 2.

Из этой последовательности действий можно понять, какое количество раз отработает цикл **For** в каждом из трех случаев:

-<нач. значение> ≤ <кон. значение>: цикл будет работать

<кон. значение> – <нач. значение> + 1 раз;

-<нач. значение> = <кон. значение>: цикл выполнится ровно один раз;

-<нач. значение> ≤ <кон. значение>: цикл ни разу не выполнится.

II. С шагом **-1**: переменная цикла изменяется не от меньшего к большему, а в противоположном направлении

For <переменная>:= <нач. значение> **DownTo** <кон. значение> **Do** <оператор>;

DownTo – «вниз к»;

<переменная>– переменная цикла, должна быть **порядкового** типа;

<нач. значение>, <кон. значение> – начальное и конечное значения переменной цикла, **по типу совместимые** с ней; могут быть константами, переменными, выражениями.

Для выполнения цикла должно соблюдаться условие:

<нач. значение> ≥ <кон. значение>

Цикл **For–DownTo** выполняется следующим образом:

1. Вычисляется начальное значение **<нач. значение>** и присваивается **переменной** цикла;
2. Проверяется условие выполнения цикла: **<нач. значение> ≥ <кон. значение>**;
3. Если условие истинно (выполняется), то выполняется **<оператор>** и пункты 4–5; иначе (условие ложно) цикл завершается и выполняется оператор, стоящий после цикла;
4. Значение переменной цикла уменьшается на единицу;
5. Происходит переход к п. 2.

После окончания работы цикла **<переменная>** может потерять свое значение. Поэтому *попытки использовать переменную цикла сразу после завершения цикла (без присваивания ей какого-либо нового значения) могут привести к непредсказуемому поведению программы при отладке*.

Пример: вычислить значение факториала n!=1·2·3·…· n

```
F:=1;                          F:=1;
For i := 1 To n Do             For i := n DownTo 1 Do
F:= F*i;                       F:= F*i;
```

Фрагменты блок-схемы вычисления факториала приведены на рис. 3.1. и рис. 3.2:

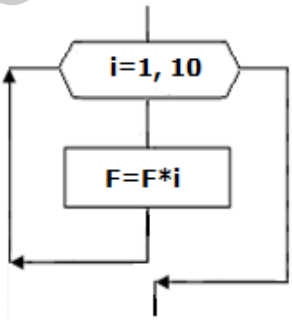


Рис. 3.1. Цикл **For** с шагом **1**

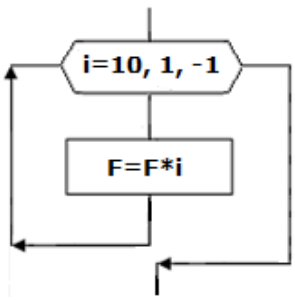


Рис. 3.2. Цикл **For** с шагом **-1**

Правила использования цикла For:

1. Переменная цикла должна быть внутренней переменной подпрограммы.
2. Внутри цикла нельзя изменять значения переменной цикла, а также переменных, входящих в расчет начального и конечного значения цикла.
3. Циклы можно вкладывать друг в друга, но нельзя их пересекать.
4. Можно передавать управление из цикла вовне его, но извне передавать управление внутрь цикла нельзя.
5. Можно передавать управление из цикла вовне его с последующим возвратом в тот же цикл.

Если заранее неизвестно, сколько раз необходимо выполнить операторы цикла, то удобнее всего применять ***цикл с предусловием*** (**While**) или ***цикл с постусловием*** (**Repeat**–**Until**).

С помощью таких циклов можно запрограммировать любые повторяющиеся фрагменты алгоритмов. На практике данные циклы чаще всего используют в двух случаях:

1. Число повторений заранее неизвестно (*например* цикл до достижения требуемой точности результата и т. п.);
2. Число повторений заранее известно, но шаг переменной цикла не равен 1 или переменная цикла не является переменной порядкового типа.

3.1.2. Оператор цикла с предусловием While

Формат оператора:

While <логическое выражение> **Do**
<оператор>;

<логическое выражение> – условие выполнения цикла, которое может быть переменной, константой или выражением, имеющим логический тип.
While – «пока, в то время как».

Выполнение цикла **While**:

1. Вычисляется значение условия выполнения цикла.
2. Если оно истинно (**True**), то выполняется оператор цикла, иначе (**False**) цикл завершается.

Цикл **While** обеспечивает выполнение оператора цикла до тех пор, пока условие имеет значение **True** (истина).

Обычно в операторе цикла изменяется значение переменной, которая входит в логическое выражение.

Условие проверяется перед началом каждого выполнения цикла. Поэтому, если до первого выполнения цикла условие имеет значение **False** (ложь), оператор цикла не выполнится ни одного раза.

Если в цикле содержится не один, а несколько операторов, то они заключаются в «операторные скобки» **Begin End**.

Примеры:

1. Вычислить значение суммы $S = \sum_{i=1}^{n} i^2 = 1 + 2^2 + 3^2 + \ldots + 9^2 + 10^2$.

```
S:=0;
i := 0;
While i<n Do
Begin
   Inc(i);
   S:= S+ i*i;
End;
```

2. Вычислить значение факториала $n! = \prod_{i=1}^{n} i = 1 \cdot 2 \cdot 3 \cdot 4 \cdots n$

```
F:=1;
 i := 0;
 While i<n Do
 Begin
    Inc(i);
    F:= F*i;
 End;
```

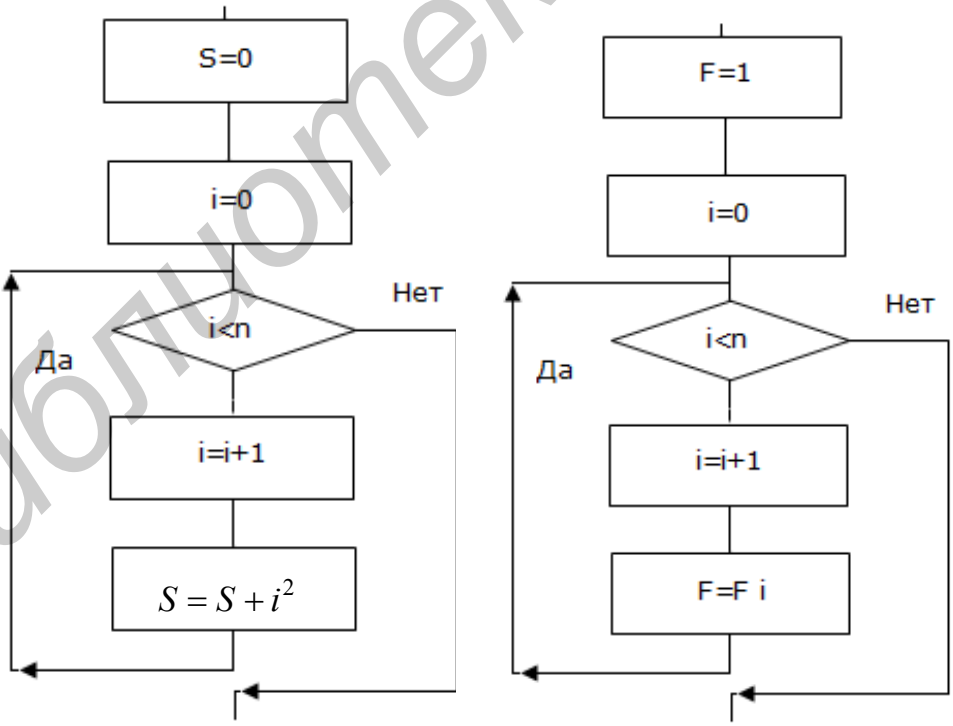


Рис. 3.3. Фрагменты блок-схем вычисления суммы и факториала

3.1.3. Оператор цикла с постусловием Repeat … Until

Формат оператора:

Repeat

оператор1;

…

Until **<логическое выражение>***;*

<логическое выражение>– условие окончания цикла.

Repeat…Until… – «повторять до тех пор, пока»

Выполнение оператора цикла:

1. Выполняются операторы цикла.
2. Вычисляется значение условия: если оно ложно (**False**), то происходит переход на пункт **1**, иначе (**True**) цикл завершается.

Примеры: вычислить значения

1) суммы $S = \sum_{i=1}^{n} i^2$

```
S:=0;
i := 0;
Repeat
   Inc(i);
   S:= S+ i*i;
  Until i=n;
```

2) факториала n!=1·2·3·…· n

```
F:=1;
i := 0;
Repeat
   Inc(i);
   F:= F*i;
   Until i=n;
```

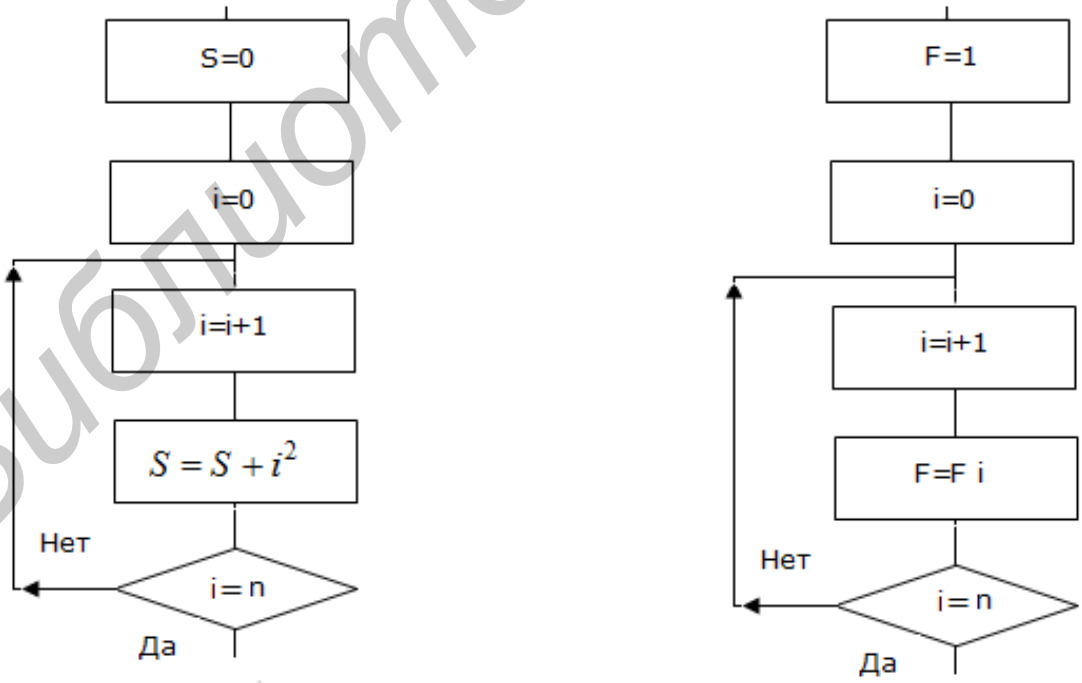


Рисунок 3.4. Фрагменты блок-схем вычисления суммы и факториала

Таблица 3.1

Отличия циклов While от Repeat...Until

Признак	While	Repeat...Until
Минимальное количество раз выполнения цикла	0	1
Значение условия при выполнении цикла	True	False
Местоположение условия	В начале цикла	В конце цикла
Применение конструкции Begin…End	Применяется, если в цикле более одного оператора	Не применяется

3.2. Операторы управления

Exit или **Return** – прерывают выполнение подпрограммы и осуществляют принудительный выход из подпрограммы в вызывающую программу.

Halt – останавливает выполнение программы и возвращает управление операционной системе.

Для управления работой циклов используются специальные операторы **Continue** и **Break**, которые можно вызывать только в теле цикла.

3.2.1 Оператор Break

Прерывает выполнение цикла и передает управление первому оператору, расположенному после цикла:

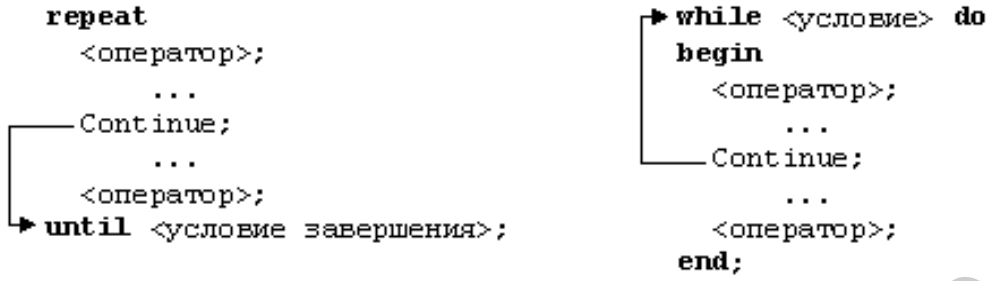
```
repeat
  <оператор>;
    ...
  Break;
    ...
  <оператор>;
until <условие завершения>;
<оператор>;
```

```
while <условие> do
begin
  <оператор>;
    ...
  Break;
    ...
  <оператор>;
end;
<оператор>;
```

При прерывании работы цикла **For** с помощью **Break** переменная цикла сохраняет свое текущее значение.

3.2.2. Оператор Continue

Прерывает работу текущей итерации цикла и передает управление оператору проверки условия, пропуская оставшуюся часть цикла:



3.3. Средства отладки программ в DELPHI

Практически в каждой вновь написанной программе обнаруживаются ошибки.

Ошибки первого уровня, или ошибки компиляции, связаны с неправильной записью операторов. При ее обнаружении 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 ошибке надо дважды щелкнуть на строке с ее описанием. Для получения более полной информации о характере ошибки необходимо вызвать справку клавишей **F1**. Следует учитывать то, что одна ошибка может повлечь за собой другие, которые исчезнут при ее исправлении. Поэтому рекомендуется исправлять ошибки последовательно сверху вниз и после исправления каждой ошибки компилировать программу снова.

Ошибки второго уровня, или ошибки выполнения, связаны с ошибками алгоритма или с неправильной его программной реализацией. Эти ошибки приводят к неверному результату, переполнению, делению на ноль и т.д. Поэтому программу надо протестировать, т.е. выполнить расчеты при таких значениях 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 редактора программы установить курсор в строке перед подозрительным участком и нажать клавишу **F4** (выполнение до курсора). Выполнение программы будет остановлено на строке с курсором. Для просмотра значения переменной нужно навести на нее курсор (на экране будет высвечено ее значение) или нажать одновременно **Ctrl–F7** и в появившемся окне ввести ее имя (с помощью данного окна можно также изменить значение переменной во время выполнения программы). Нажимая клавишу **F7** (пошаговое выполнение), можно построчно выполнять программу, контролируя изменение тех или иных переменных и правильность вычисления. При нахождении курсора внутри цикла после нажатия **F4** вычисления останавливаются после

каждого выполнения тела цикла. Для продолжения расчетов надо выбрать меню **Run** – **Run**.

Способы установки точек прерывания:

1. Щелчок мышью на левом краю окна редактирования: выбранная для остановки строка выделяется красной полосой, на ее левом краю появляется маленький значок.

2. При выполнении команд **Run** – **Add Breakpoint** – **Source Breakpoint...** появится диалоговая панель редактирования точек прерывания **Edit Breakpoint**. Можно задать параметр **Condition**, где ввести выражение, при истинности которого точка прерывания «сработает», иначе выполнение приложения не будет прервано при прохождении через эту строку или количество проходов, после которых точка прерывания переходит в активное состояние.

3.4. Порядок выполнения задания

Написать и отладить программу, которая выводит таблицу значений функции $y(x) = e^{-x}$ и ее разложение 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 удобно использовать рекуррентную 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 $-\frac{x}{k}$. 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. 3.5.

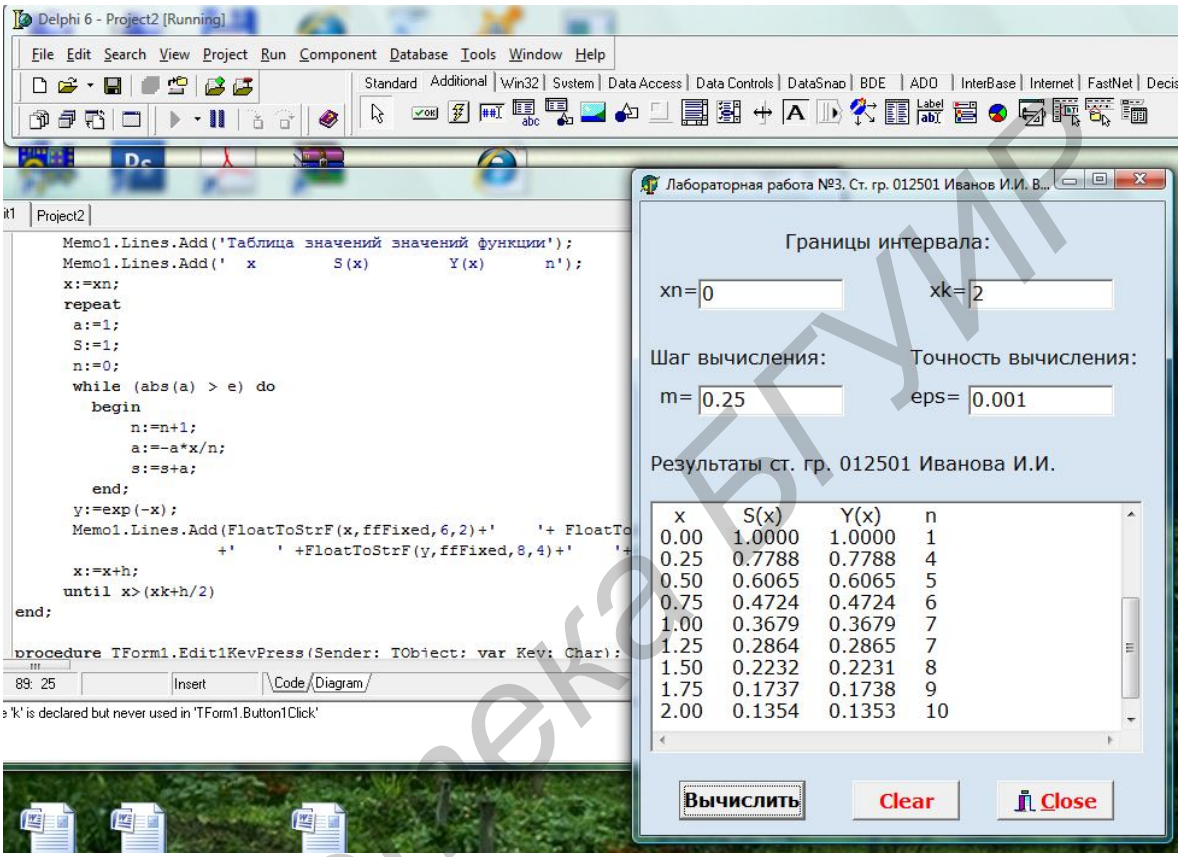


Рис. 3.5. Панель диалога

Блок-схема алгоритма представлена на рис. 3.6.

3.4.1 Блок-схема алгоритма

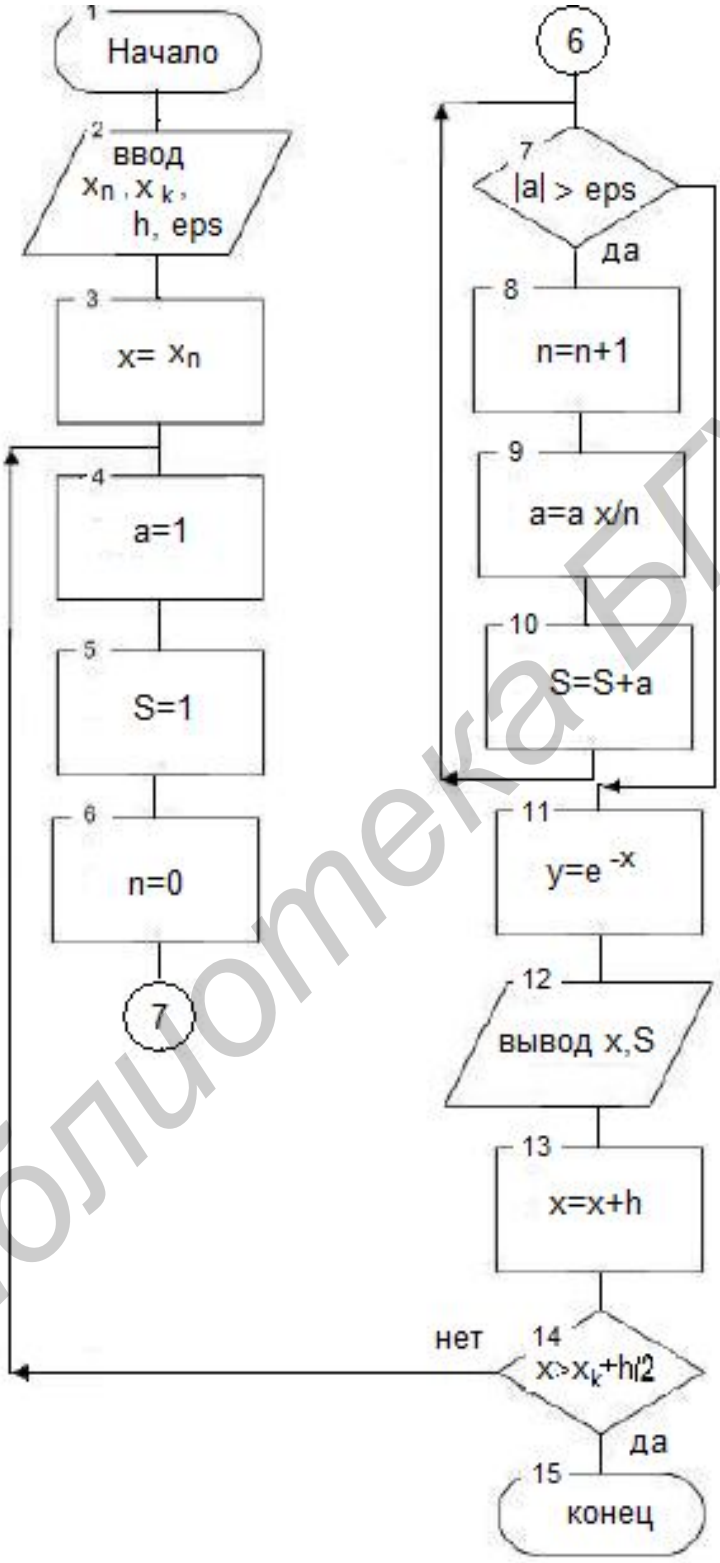


Рис. 3.6. Блок-схема алгоритма

3.4.2 Код программы

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms, Dialogs, Buttons, StdCtrls;
type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Edit3: TEdit;
    Label5: TLabel;
    Edit4: TEdit;
    Label6: TLabel;
    Memo1: TMemo;
    Label7: TLabel;
    Label8: TLabel;
    Button1: TButton;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Edit1KeyPress(Sender: TObject; var Key: Char);
    procedure Edit2KeyPress(Sender: TObject; var Key: Char);
    procedure Edit3KeyPress(Sender: TObject; var Key: Char);
    procedure Edit4KeyPress(Sender: TObject; var Key: Char);
  procedure BitBtn2Click(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';
   Memo1.Clear;
   Label6.Caption:='Результаты ст. гр. 012501 Иванова И.И.';
end;

procedure TForm1.Button1Click(Sender: TObject);
var
    xn,xk,x,h,e,a,s,y :Extended;
    n,k:Integer;
begin
   Memo1.Lines.Add('Вычисление таблицы значений функции');
   Memo1.Lines.Add('Исходные данные:');
   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('eps='+FloatToStrF(e,ffFixed,8,5));
   Memo1.Lines.Add('Таблица значений функции');
   Memo1.Lines.Add('  x      S(x)       Y(x)      n');
   x:=xn;
   Repeat
      a:=1;
      S:=1;
      n:=0;
      While Abs(a) > e Do
       Begin
         Inc(n)          // n:=n+1;
         a:=-a*x/n;
          s:=s+a;
        End;
       y:=Exp(-x);
       Memo1.Lines.Add(FloatToStrF(x,ffFixed,6,2)+'
     '+FloatToStrF(s,ffFixed,
```

```
                    8,4)+'    ' + FloatToStrF(y,ffFixed,8,4)+ '    '+ In-
                    tToStr(n));
      x:=x+h;
    Until x>xk+h/2   //xk+h/2 применяется для исключения потери последнего x
  end;

procedure TForm1.BitBtn2Click(Sender: TObject);
begin     // Очистка полей ввода и вывода результата для нового расчета
  Edit1.text:=' ';
  Edit2.text:=' ';
  Edit3.text:=' ';
  Edit4.text:=' ';
  Memo1.Clear;
  Label6.Caption:=' ';
  end;

end.
```

После отладки программы составьте тест (n=2, x_n=0, x_k=1, h=3), установите курсор на оператор **Inc(n)**; нажмите клавишу **F4**. После этого нажимая клавишу **F7**, выполните пошаговую программу и проследите, как меняются все переменные в процессе выполнения.

3.5. Индивидуальные задания

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 eps. Близость значений ***S*(*x*)** и ***Y*(*x*)** во всем диапазоне значений x указывает на правильность вычисления ***S*(*x*)** и ***Y*(*x*)**.

Составить блок-схему алгоритма. Спроектировать панель диалога и написать код программы.

Создать вместо обработчика **Button1.Click** обработчик **Memo1Click** или **Label1DblClick**.

После написания программы и исправления ошибок трансляции изучить средства отладки программ, для чего установить курсор на первый оператор и нажать клавишу **F4**. После этого, нажимая клавишу **F7**, выполнить пошагово программу и проследить, как меняются все переменные в процессе ее выполнения.

Таблица 3.2

Индивидуальные задания

№ зад.	x_n	x_k	$S(x)$	n	$Y(x)$
1	2	3	4	5	6
1	0,1	1	$x-\frac{x^3}{3!}+...+(-1)^n\frac{x^{2n+1}}{(2n+1)!}$	16	$\sin x$
2	0,1	1	$1+\frac{x^2}{2!}+...+\frac{x^{2n}}{(2n)!}$	10	$\frac{e^x+e^{-x}}{2}$
3	0,1	1	$1+\frac{\cos\frac{\pi}{4}}{1!}x+...+\frac{\cos n\frac{\pi}{4}}{n!}x^n$	12	$e^{x\cos\frac{\pi}{4}}\cos(x\sin\frac{\pi}{4})$
4	0,1	1	$1-\frac{x^2}{2!}+...+(-1)^n\frac{x^{2n}}{(2n)!}$	8	$\cos x$
5	0,1	1	$1+3x^2+...+\frac{2n+1}{n!}x^{2n}$	14	$(1+2x^2)e^{x^2}$
6	0,1	1	$x+\frac{x^3}{3!}+...+\frac{x^{2n+1}}{(2n+1)!}$	8	$\frac{e^x-e^{-x}}{2}$
7	0,1	1	$\frac{x^3}{3}-\frac{x^5}{15}+...+(-1)^{n+1}\frac{x^{2n+1}}{4n^2-1}$	12	$\frac{1+x^2}{2}\operatorname{arctg}x-\frac{x}{2}$
8	0,1	1	$1+\frac{2x}{1!}+...+\frac{(2x)^n}{n!}$	10	e^{2x}
9	0,1	1	$1+2\frac{x}{2}+...+\frac{n^2+1}{n!}\left(\frac{x}{2}\right)^n$	14	$\left(\frac{x^2}{4}+\frac{x}{2}+1\right)e^{\frac{x}{2}}$
10	0,1	0,5	$x-\frac{x^3}{3}+...+(-1)^n\frac{x^{2n+1}}{2n+1}$	15	$\operatorname{arctg}x$
11	0,1	1	$1-\frac{3}{2}x^2+..+(-1)^n\frac{2n^2+1}{(2n)!}x^{2n}$	10	$\left(1-\frac{x^2}{2}\right)\cos x-\frac{x}{2}\sin x$
12	0,1	1	$-\frac{(2x)^2}{2}+\frac{(2x)^4}{24}-...+(-1)^n\frac{(2x)^{2n}}{(2n)!}$	8	$2(\cos^2 x-1)$
13	-2	-0,1	$-(1+x)^2+\frac{(1+x)^4}{2}+...+(-1)^n\frac{(1+x)^{2n}}{n}$	16	$\ln\frac{1}{2+2x+x^2}$

Продолжение табл. 3.2

1	2	3	4	5	6
14	0,2	0,8	$\frac{x}{3!}+\frac{4x^2}{5!}+\ldots+\frac{n^2}{(2n+1)!}x^n$	12	$\frac{1}{4}\left(\frac{x+1}{\sqrt{x}}sh\sqrt{x}-ch\sqrt{x}\right)$
15	0,1	0,8	$\frac{x^2}{2}-\frac{x^4}{12}+\ldots+(-1)^{n+1}\frac{x^{2n}}{2n(2n-1)}$	18	$xarctgx-\ln\sqrt{1+x^2}$
№	x_n	x_k	$S(x)$	eps	$Y(x)$
16	0,1	0,9	$\sum_{i=1}^{\infty}\frac{x^{2^{i-1}}}{1-x^{2^i}}$	0,001	$\frac{x}{1-x}$
17	0,1	3	$\sum_{i=1}^{\infty}(-1)^{i-1}\frac{\cos\left((2i-1)x\right)}{2i-1}$	0,001	$\pi/4$
18	0,2	1,2	$\sum_{i=1}^{\infty}\frac{(-1)^i 2^i\cos(ix)}{i}$	0,0001	$-\frac{1}{2}\ln\left(5+4\cos(x)\right)$
19	1	3	$\sum_{i=0}^{\infty}\frac{\left(x\ln(2)\right)^i}{i!}$	0,0001	2x
20	0,1	1,5	$\sum_{i=1}^{\infty}(-1)^{i+1}\frac{2^{2i-1}x^{2i}}{(2i)!}$	0,0001	Sin2(x)
21	0,1	1,5	$\frac{1}{4}\sum_{i=0}^{\infty}(-1)^{i+1}\frac{3^{2i+1}-3}{(2i+1)!}x^{2i+1}$	0,0001	Sin3(x)
22	0,1	2	$\cos ec(x)\sum_{i=1}^{\infty}(-1)^{i+1}\frac{2^{2i}x^{4i-2}}{(4i-2)!}$	0,001	Sh(x)
23	0,1	1	$\sec(x)+\sec(x)\sum_{i=1}^{\infty}(-1)^i\frac{2^{2i}x^{4i}}{(4i)!}$	0,001	Ch(x)
24	0,1	1,5	$x\prod_{i=1}^{\infty}\left(1-\frac{x^2}{i^2\pi^2}\right)$	0,001	Sin(x)
25	0,1	1,5	$\prod_{i=1}^{\infty}\left(1-\frac{4x^2}{(2i+1)^2\pi^2}\right)$	0,001	Cos(x)

Окончание табл. 3.2

1	2	3	4	5	6
26	0,1	0,9	$x\prod_{i=1}^{\infty}\cos\left(\frac{x}{2^i}\right)$	0,001	Sin(x)
27	0,1	1,5	$\prod_{i=1}^{\infty}\left[1-\frac{4}{3}\sin^2\left(\frac{x}{3^i}\right)\right]$	0,001	$\frac{\sin(x)}{x}$
28	0,1	0,9	$\sum_{i=0}^{\infty}\frac{(2i)!}{2^{2i}(i!)^2(2i+1)}x^{2i+1}$	0,001	Arcsin(x)
29	0,1	1	$\sum_{i=0}^{\infty}\frac{2^{2i}(i!)^2 x^{2i+2}}{(2i+1)!(i+1)}x^{2i+1}$	0,001	$(\arcsin(x))^2$
30	0,1	2	$\ln(2)-\sum_{i=1}^{\infty}(-1)^i\frac{(2i-1)!}{2^{2i}(i!)^2}x^{2i}$	0,001	$\ln\left(1+\sqrt{1+x^2}\right)$

3.6. Задания повышенной сложности

1. Дано натуральное число *k*. Изменить в нем порядок следования цифр на обратный.
2. Удалить из записи натурального числа *n* все цифры, равные 0, сохранив порядок следования остальных цифр. Например, 5003 →53.
3. Найти наибольшую цифру в заданном натуральном числе k.
4. Определить, является ли натуральное число *n* палиндромом (палиндром – число, одинаково читаемое слева направо и справа налево). Например, 121, 2332 – палиндромы; 7664, 112 – не палиндромы.
5. Числа Фибоначчи (F_i) определяются по формулам: $F_0 = F_1 = 1$; $F_i = F_{i-1} + F_{i-2}$ при i=2, 3, ... Найдите первое из чисел Фибоначчи, которое превосходит заданное число M ($M > 0$).
6. Пифагоровыми называются тройки натуральных чисел a, b, c, удовлетворяющие условию: $a^2+b^2=c^2$. Например, пифагоровой является тройка чисел 6, 8, 10. Найдите все тройки пифагоровых чисел, не превышающих 25.
7. Совершенными называются числа, равные сумме своих делителей. Например, совершенным является число 28=1+2+4+7+14. Найдите все совершенные числа в интервале [1, 1000].

Лабораторная работа№ 4. Обработка исключительных ситуаций. Программирование с использованием одномерных массивов

Цель работы: изучить свойства компонента **TStringGrid**. Составить блок-схему, написать и отладить программу с использованием одномерных массивов, обработать возможные исключительные ситуации.

4.1. Обработка исключительных ситуаций

Под исключительной ситуацией понимается 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 обработки исключительных ситуаций внутри программы 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 определенных исключительных ситуаций>
End;
```

При возникновении исключительной ситуации управление передается в блок **Except,** иначе 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.

Для отладки программы, содержащей обработку исключительных ситуаций, надо отключить опцию **Stop on Delphi Exceptions**, находящуюся в **Tools – Debbuger Options …** , закладка **Language Exceptions**.

Возникновение исключительной ситуации можно инициировать преднамеренно. Для этого применяются процедуры **Abort**, **Assert (b : Boolean),** а также ключевое слово **Raise**:

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

Таблица 4.1

Типы исключительных ситуаций

Тип исключительной ситуации	Причина
1	**2**
EAbort	Намеренное прерывание 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+ }**)

4.2. Функции ShowMessage и MessageDlg

Для вывода сообщений применяются 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).

Например**: If MessageDlg ('Вы действительно хотите завершить работу?', mtConfirmation, [mbOk, mbNo], 0) = mrNo Then CanClose:=False;**

Функция **MessageDlg** отображает диалоговое окно в центре экрана. Первый ее параметр сообщение, выводимое в окне. Второй параметр – тип окна – может принимать следующие значения: **mtWarning** (предупреждение), **mtError** (ошибка), **mtlnformation** (информация), **mtConfirmation** (подтверждение). Третий параметр задает кнопки в диалоговом окне, его значения: **mbYes**, **mbNo**, **mbOK**, **mbCancel**, **mbAbort**, **mbRetry**, **mblgnore**, **mbAll**, **mbHelp**. Четвертый параметр определяет темы помощи, появляющиеся при щелчке на кнопке помощи или нажатии **F1** во время отображения диалогового окна.

Таблица 4.2

Вид отображаемого окна, определяемого параметром **DlgType**

Значения типа окна	**Вид отображаемого окна**
mtWarning	Заголовок: «**Warning**». Значок: желтый треугольник с восклицательным знаком внутри
mtError	Заголовок: «**Error**». Значок: красный круг с перечеркиванием внутри
mtInformation	Заголовок: «**Information**». Значок: символ «**i**» на голубом поле.
mtConfirmation	Заголовок: «**Confirmation**». Значок: символ «**?**» на зеленом поле.
mtCustom	Заголовок соответствует имени выполняемого файла. Без значка.

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

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**, **mrHelp**.

Таблица 4.3

Значения параметра **Buttons**

Значение параметра	Тип кнопки	Значение параметра	Тип кнопки
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»		

4.3. Работа с массивами

Массив – пронумерованная структура данных, элементы которой обладают одним и тем же типом.

Например:

1 5 6 0 7 9 2 – *массив целых чисел*

a_1 a_2 a_3 a_4 a_5 a_6 a_7

1,2 5,1 7,9 2,1 5,5 9,3 4,8 – *массив вещественных чисел*

Каждый элемент массива обозначается именем, за которым в квадратных скобках указывается 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.

4.3.1. Объявление массива

Тип массива или сам массив определяются соответственно в разделе типов (**Type**) или переменных (**Var**) с помощью следующей конструкции:

Array [описание индексов] **Of** <тип элемента массива>;

Например:

```
Const
      N_Max = 100;
Type
      Mas = Array [1 ..  N_Max] Of тип элементов массива;
Var
      A: Mas;
      i, n : Byte; {i – порядковый номер (индекс) элемента в массиве A;
                    n – количество элементов массива A}
```

Примеры описания массивов:

```
Const
   N_Max=20;      // Задание максимального значения размерности массива
```

```
Type
   Mas =Array[1..N] Of Byte;    // Описание типа одномерного массива
 Var
     a: Mas;                          // A – массив типа Mas;
     b:Array[1..10] Of Integer;       // b – массив из десяти целых чисел
     y:Array[1..5,1..10] Of Char;     // y– двумерный массив символьного типа
```

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

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

Для доступа к элементу массива необходимо указать его имя и индекс (порядковый номер элемента в массиве): имя_массива [индекс]

4.3.2. Примеры программ

Пример 1. Подсчитать сумму и количество отрицательных и положительных элементов массива.

```
Const
  N_Max = 100;
Type
  Mas = Array [1 .. N_Max] Of Extended;
Var
    a: Mas;
    i,  n, K_Pol, K_Otr : Byte; // K_Pol – количество положительных элементов,
    Sum_P,  Sum_O  :  Extended;  // K_Otr  –  количество отрицательных элементов
                                 // Sum_P – сумма положительных элементов,
                                 // Sum_O – сумма отрицательных элементов
Begin
   // Ввод элементов массива
  Sum_P := 0;       // Инициализация переменных
  K_Pol := 0;
  Sum_O := 0;
  K_Otr := 0;
  For i :=1 To n Do  Begin     // Подсчет сумм и количества
   If A [i] > 0 Then  Begin    // Если элемент положительный, то
     Inc (K_Pol);   // подсчитываем количество
     Sum_P := Sum_P + A[i];     // и сумму
     End;
    If A [i] < 0 Then Begin                 // Если элемент отрицательный, то
     Inc (K_Otr);   // подсчитываем количество
     Sum_O := Sum_O + A[i]; // и сумму
```

```
        End;
    End;  { For  i }
  // Вывод результатов
  End;
```

Пример 2. Изменить значения элементов массива по заданному правилу:

- если значение элемента массива четное и положительное, то увеличить его **в** 2 раза,
- если кратно 3 – уменьшить на 10.

```
Const
  N_Max = 100;
Type
  Mas = Array [1 .. N_Max] Of Integer ;
Var
  A: Mas;
  i, n : Byte;
Begin
// Ввод элементов массива
  For i :=1 To n Do Begin      //  Изменение значений элементов
    If (A [i] > 0) And (Not Odd (A[i])) Then A[i]:= A[i] *2;
    If A [i] Mod 3 = 0 Then  A[i]:=  A[i] - 10;
  End;    { For  i }
    // Вывод результатов
  End;
```

Пример 3. Записать положительные элементы массива **A** в массив **B**, а отрицательные элементы – в массив **C**, не меняя порядка следования элементов.

```
        i_b :=0;        // i_b – количество элементов массива B,
        i_c :=0;        // i_c – количество элементов массива C
        For i :=1 to n Do Begin       // Формирование новых массивов
          If A [i] > 0 Then  Begin
            Inc (i_b);          // Увеличиваем индекс
            B [i_b] := A [i];   // Записываем в массив B
            End;
          If A [i] <  0 Then Begin
            Inc (i_c);          // Увеличиваем индекс
            C [i_c] := A [i];   // Записываем в массив
            End;
        End;    { For }
```

Пример 4. Упорядочить элементы массива по возрастанию их значений, т.е. для всех элементов массива должно выполняться условие: $\mathbf{a_i < a_{i+1}}$.

```
        For k :=1 To n-1 Do       // k – номер просмотра массива
```

```
For i :=1 To n-k Do          // Просмотр элементов массива
If  A [i] > A [i+1] Then   Begin  // Сравнение элементов массива
   Tmp := A [i];          // Перестановка элементов
   A [i]:=A[i+1];         // a_i и a_{i+1}, если они
   A [i+1] := Tmp;        // стоят неправильно
   End;
```

Пример 5. Удалить из одномерного массива все отрицательные элементы.

Для решения данной задачи необходимо выполнить следующие действия:

– найти индекс удаляемого элемента;
– сдвинуть элементы, стоящие после него, в начало на один индекс;
– уменьшить размерность на 1.

```
…
i :=1;
While  i<=n Do Begin
   If  B [i] < 0 Then Begin     {1}// Если найден отрицательный элемент, то
      For j := i DownTo n-1 Do {2}  // сдвинуть все элементы, стоящие после
      B [j] := B [j+1];             // удаляемого, на одну позицию
      Dec (n);                      {3}// Уменьшить размер массива
      Dec (i);                         // Возврат к предыдущему индексу
      End;
   Inc(i);
   End;
```

Пример 6. Циклический сдвиг элементов массива на k позиций

- *влево*:

```
k := k Mod n;
For j :=1 To k Do Begin
   Tmp := A [1];
   For i :=1 To n-1 Do
   A [i] := A [i+1] ;
   A [n] := Tmp;
   End;
```

- *вправо*:

```
k := k Mod n;
For j :=1 To k Do Begin
   Tmp := A [n];
   For i :=n  DownTo 2 Do
   A [i] := A [i-1] ;
   A [1] := Tmp;
   End;
```

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 двумерной таблицы, каждая ячейка которой представляет собой окно однострочного редактора (аналогично строке ввода **TEdit**). Доступ к информации осуществляется с помощью свойства:

Cells[ACol:Integer; **ARow**:Integer**]** : String,

где **ACol**, **ARow** – соответственно номер столбца и строки элемента двумерного массива. Нумерация осуществляется с **нуля**.

Свойства **ColCount** и **RowCount** устанавливают соответственно 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. Порядок выполнения задания

В массиве **A** из **n** элементов поменяйте местами максимальный и минимальный элементы.

Значения **n** вводить в компонент **TEdit**, **A** – в компонент **TStringGrid**. Результат после нажатия кнопки типа **TButton** вывести в компонент **TStringGrid**.

4.5.1. Настройка компонента TStringGrid

1. На вкладке **Additional** в меню компонентов щелкните мышью по пиктограмме  для установки на форму компонентов **StringGrid1** и **StringGrid2**.
2. Установите компонент в нужном месте формы (ЛКМ на форме).
3. Захватывая кромки компонента, отрегулируйте его размер.
4. В инспекторе объектов установите значения свойств **TStringGrid**:

   ColCount равным **7** (семь столбцов),

   RowCount равным **2** (две строки),

   FixedCols – **0** (нет столбцов с фиксированной зоной),

   FixedRows – **1** (одна строка с фиксированной зоной для подписи порядковых номеров (индексов) элементов массива).
5. Раскройте раздел **Options** (нажав на знак «+», стоящий слева от **Options**) и установите свойство **goEditing** в положение **True** для компонентов **StringGrid1** и **StringGrid2** (по умолчанию в компонент **TStringGrid** запрещен ввод информации с клавиатуры).

Панель диалога приведена на рис. 4.1.

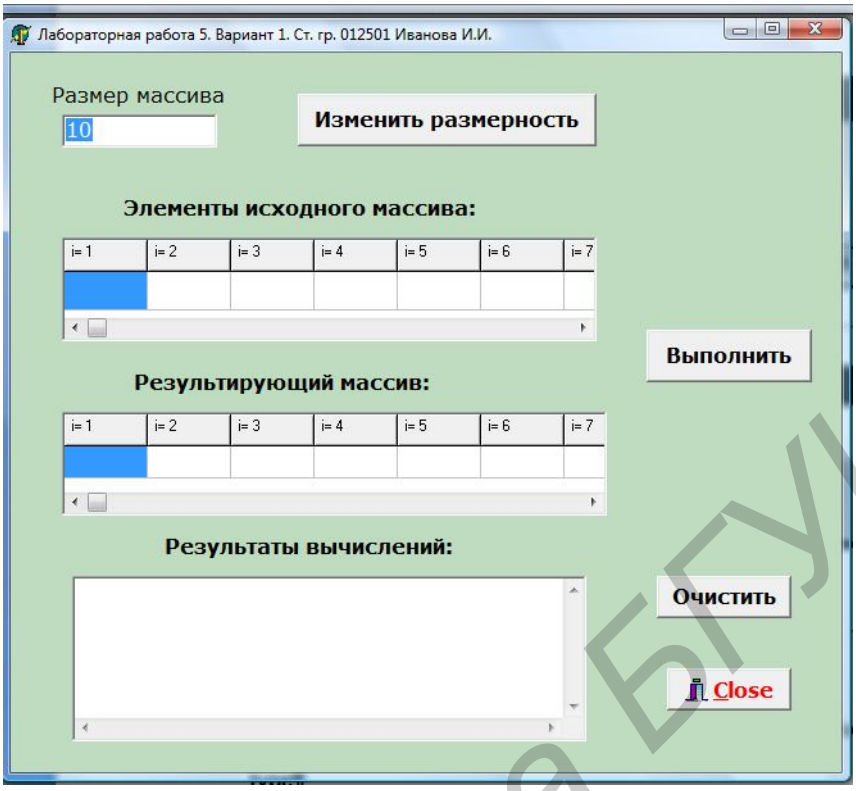


Рис. 4.1. Панель диалога

4.5.2. Блок-схема алгоритма

Блок-схема алгоритма представлена на рис. 4.2.

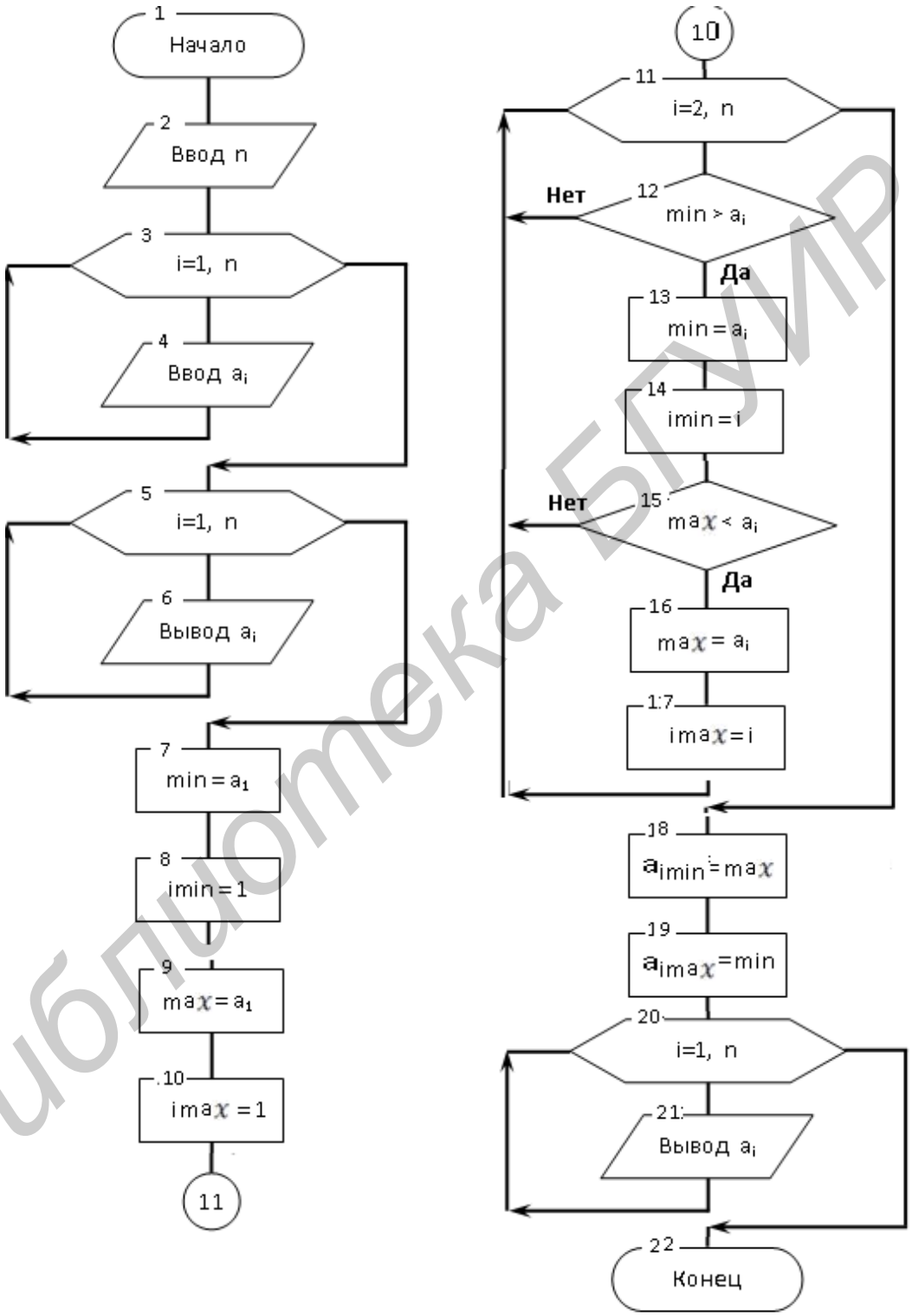


Рис. 4.2. Блок-схема алгоритма

4.5.3. Код программы

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, Grids, MxArrays;
Type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Button1: TButton;
    StringGrid1: TStringGrid;
    Label2: TLabel;
    Button2: TButton;
    BitBtn1: TBitBtn;
    StringGrid2: TStringGrid;
    Label3: TLabel;
    Button3: TButton;
    Memo1: TMemo;
    Label4: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
Const
      Nmax=100;          // Максимальный размер массива
Type                     // Объявление типа одномерного массива
    Mas = Array[1..Nmax] Of Extended;   // размерностью Nmax
Var
  Form1: TForm1;
  A:Mas;             // Объявление одномерного массива
  i, n: Byte;
Implementation
{$R *.dfm}
```

```
procedure TForm1.FormCreate(Sender: TObject);
Begin
   Edit1.Text:='10';              // Ввод размерности массива
   n:=StrToInt (Edit1.Text);
   Memo1.Clear;
   StringGrid1.ColCount:=n;    // Задание числа столбцов в таблицах
   StringGrid2.ColCount:=n;
   For i:=0 To N-1 Do Begin                    //Заполнение верхней строки
      StringGrid1.Cells[i, 0]:=' i= '+IntToStr(i+1); // поясняющими подписями
      StringGrid2.Cells[i, 0]:=' i= '+IntToStr(i+1);
   End;
End;

procedure TForm1.Button1Click(Sender: TObject); // Изменить размер
Begin                                          // таблицы
   n:=StrToInt(Edit1.Text);      // Количество элементов в массиве
   StringGrid1.ColCount:=n;
   StringGrid2.ColCount:=n;
   For i:=0 To n -1 Do Begin                   // Заполнение верхней строки
      StringGrid1.Cells[i, 0]:=' i= '+IntToStr(i+1); // поясняющими подписями
      StringGrid2.Cells[i, 0]:=' i= '+IntToStr(i+1);
   End;
End;

procedure TForm1.Button3Click(Sender: TObject);   // Очистить
begin
   Memo1.Clear;
   Edit1.Text:=' ';
   For i:=0 To n -1 Do Begin
      StringGrid1.Cells[i,1]:=' ';
      StringGrid2.Cells[i,1]:=' ';
      End;
   End;

procedure TForm1.Button2Click(Sender: TObject);
Var
      iMax, iMin : Byte;
      Max, Min, t :  Extended;
Begin
{$R+}
   Try          {Заполнение массива A элементами из таблицы StringGrid1}
      Memo1.Lines.Add('Исходный массив:');
      For i:=0 To n -1 Do
```

```
      A[i+1]:=StrToFloat(StringGrid1.Cells[i,1]);
      For i:=1 To n Do
      Memo1.Lines.Add('A['+IntToStr(i)+']='+FloatToStrF(A[i],ffFixed,6,0));
      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 ситуа-
ция!');
         Exit;
         End;
      End;
   Try
{ Поиск минимального и максимального элементов массива и их индексов}
      Max:=A[1];              // Инициализация значений
      iMax:=1;
      Min:=A[1];
      iMin:=1;
      For i:=1 To n Do Begin
         If Max<A[i] Then Begin   // Поиск максимального элемента и его индек-
са
            Max:=A[i];
            iMax:=i;
            End;
         If Min>A[i] Then Begin   // Поиск минимального элемента и его индекса
            Min:=A[i];
            iMin:=i;
            End;
         End;
      A[iMax]:=Min;
      A[iMin]:=Max;
      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;
    For i:=0 To n -1 Do      { Вывод результата в таблицу StringGrid2 }
    StringGrid2.Cells[i,1]:=FloatToStrF(A[i+1],ffFixed,6,0);
    Memo1.Lines.Add('Min='+FloatToStrF(Min,ffFixed,6,0)+',  iMin='+
                        IntToStr(iMin));
    Memo1.Lines.Add('Max='+FloatToStrF(Max,ffFixed,6,0)+',  iMax='+
                        IntToStr(iMax));
    Memo1.Lines.Add('Результирующий массив:');
    For i:=0 To n -1 Do
    Memo1.Lines.Add('A['+IntToStr(i)+']='+FloatToStrF(A[i+1],ffFixed,6,0));
    End;
End.
```

4.6. Индивидуальные задания

В соответствии с индивидуальным вариантом выполнить вариант индивидуального задания. Составить блок-схему алгоритма. Во всех заданиях переменные вводить и выводить с помощью компонента TEdit, массивы – с помощью компонента TStringGrid. Вычисления выполнять после нажатия кнопки типа TButton. Вывести исходные данные и полученный результат. В местах возможного возникновения ошибок использовать конструкции для обработки исключительных ситуаций.

1. Найти произведение элементов массива, расположенных между максимальным и минимальным элементами.
2. Найти сумму элементов массива, расположенных после первого нулевого элемента.
3. Найти среднее арифметическое элементов массива, расположенных до максимального элемента.

4. Найти количество элементов массива, расположенных после минимального элемента.

5. Найти произведение и количество элементов массива, расположенных до первого отрицательного элемента.

6. Найти сумму и количество элементов массива, расположенных после первого положительного элемента.

7. Найти среднее арифметическое положительных, кратных трем элементов массива, расположенных до минимального элемента.

8. Найти произведение четных отрицательных элементов массива, расположенных после минимального элемента.

9. Найти сумму и количество нечетных элементов массива, расположенных до последнего положительного элемента.

10. Найти среднее арифметическое модулей, кратных пяти, элементов массива, расположенных после максимального элемента.

11. Найти произведение модулей элементов массива, расположенных после минимального элемента.

12. Найти сумму модулей элементов массива, расположенных после последнего нулевого элемента.

13. Найти среднее арифметическое модулей четных элементов массива, расположенных между первым отрицательным и последним положительным элементами.

14. Найти максимальное значение между суммами четных и нечетных элементов массива, расположенных до минимального элемента.

15. Элементы массивов B размерностью n и C размерностью m преобразовать по правилу: значение b_i замените наибольшим из элементов b_i и c_i, а значение c_i — наименьшим из элементов b_i и c_i .

16. Элементы массива, кратные трем 3, заменить средним арифметическим двух соседних элементов (предшествующим и последующим). Первый и последний элементы оставить без изменений.

17. Даны массивы A размерностью n и B размерностью m, в которых числа не повторяются. Записать в новый массив элементы, встречающиеся как в массиве A, так и в B.

18. В массиве из n вещественных чисел найти второй по счету максимальный элемент и его индекс.

19. Удалить из массива A размерностью n те элементы, которые не входят в массив B размерностью m.

20. Заданы два массива из n и m вещественных чисел. Найти наименьшее среди тех элементов первого массива, которые не входят во второй.

21. В заданном одномерном массиве удалить каждый третий положительный элемент.

22. В заданном массиве A размерностью n, все элементы которого попарно различны, найти наибольший отрицательный элемент.

23. Дана последовательность из n вещественных чисел. Найти наиболее длинную последовательность подряд идущих нулей.

24. Дана последовательность из n вещественных чисел. Преобразовать ее по следующему правилу: все четные элементы перенести в начало, а остальные – в конец, сохраняя исходное взаимное расположение как среди четных, так и среди остальных элементов.

25. Дана последовательность из n действительных чисел. Оставить ее без изменения, если она упорядочена по неубыванию или по невозрастанию; в противном случае получить положительные элементы исходной последовательности, упорядоченные по возрастанию.

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

27. В заданном одномерном массиве B вставить после каждого отрицательного, кратного трем элемента его модуль.

28. Записать в новый массив C элементы массива B, большие по модулю вводимого значения T, и найти среди них максимальное значение.

29. Удалить из массива X элементы, входящие в него более одного раза.

30. В одномерном массиве определить значение элемента, наиболее часто встречающегося в нем, и записать в новый массив те элементы, значения которых меньше найденного.

Лабораторная работа № 5. Программирование с использованием двумерных массивов

***Цель работы**:* изучить свойства компонента **TBitBtn**. Написать программу с использованием двумерных массивов.

5.1. 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 **Glyph**. Кроме того, имеется свойство **Kind**, которое задает одну из 11 стандартных разновидностей кнопок. Нажатие любой из них, кроме **bkCustom** и **bkHelp**, закрывает модальное окно и возвращает в программу результат **mr*** (*например* **bkOk** – **mrOk**). Кнопка **bkClose** закрывает главное окно и завершает работу программы.

5.2. Примеры программ

Пример 1. Поменять местами максимальный и минимальный элементы в каждой строке двумерного массива.

```
Const
  N_Max = 10;
Type
  Matr = Array [1 ..  N_Max ,  1 ..  N_Max ] Of Extended;
Var
  A  : Matr;
  i, n, j, m, j_Min, j_Max:Byte;  // j_Min – номер столбца минимального элемента,
  Min, Max  : Extended;          // j_Max – номер столбца максимального  элемента
Begin
// Ввод элементов массива
  For i := 1 To n Do Begin
    Min := A[i, 1];       //Предположения о начальных значениях максимального
    j_Min := 1;           //и минимального элементов i-й строки
    Max := A[i, 1];
    j_Max := 1;
    For j:=2 To m Do // Поиск максимального и минимального элементов
    Begin             // в i-й строке
      If A [i, j] > Max Then Begin
        Max := A[i, j];
        j_Max := j;
        End;
```

```
        If A [i, j] < Min Then Begin
           Min := A[i, j];
           j_Min := j;
           End;
        End;            // For  j
     A [i, j_Max] :=Min;       //Перестановка максимального и минимального
     A [i, j_Min] :=Max;    // элементов в i-й строке
     End;     // For  i
   // Вывод элементов массива
  End;
```

Пример 2. Сформировать новый массива С, состоящий из отрицательных и кратных трем элементов массива А размерностью n·m.

```
Const
  N_Max = 10;
Type
  Mas = Array [1..N_Max* N_Max] Of Integer ;
  Matr = Array [1..N_Max, 1..N_Max] Of Integer ;
Var
      A          : Matr;
     C          : Mas;
   i, j, k, n, m : Byte;              // k – количество элементов массива C
Begin
    // Ввод элементов массива
  k:=0;
  For i := 1 To n Do
  For j := 1 To m Do
  If (A [i, j] < 0) And ( A [i, j] Mod 3 = 0) Then // Поиск отрицательных и
  Begin                  // кратных трем элементов массива A
     Inc (k); // k:= k +1;
     C [k] := A [i, j];
     End;
   // Вывод элементов массива
  End;
```

Пример 3. Отсортировать элементы каждой строки массива А по возрастанию, применяя улучшенный метод пузырька.

```
Const
  N_Max = 10;
Type
  Matr = Array [1..N_Max, 1..N_Max ] Of Extended;
Var
```

```
   A : Matr;
   i,  n, j, m : Byte;
   t   : Extended;       // Переменная для перестановки элементов массива
   Fl : Boolean;         // Флаг для определения: отсортирована строка или нет
Begin
    // Ввод элементов массива
   For i := 1 To n Do
   Repeat
      Fl := True;      // Предположение о том, что строка отсортирована
      For j := 1 To m-1 Do // Просмотр элементов i-ой строки массива
      if A [i, j] > A [i, j+1] Then Begin   // Если элементы стоят по убыванию,
         t := A [i, j];                     // то поменять их местами
         A [i, j] := A [i, j+1] ;
         A [i, j+1] := t;
         Fl:= False; // Строка не отсортирована
         End;
      Until   Fl; // Если Fl=True, то выйти из цикла, т.к. строка отсортирована
    // Вывод элементов массива
   End;
```

Пример 4. Найти максимальный элемент, стоящий на главной диагонали, и минимальный элемент, стоящий на побочной диагонали массива A.

К элементу, стоящему на главной диагонали, обращаются как A[i,i] или A[j,j], так как индексы строки и столбца у него совпадают (A[1,1], A[2,2],…, A[n, n]); а к элементу побочной диагонали – A[i, n-i+1].

```
Max:= A[1, 1];       //Предположения о начальных значениях максимального
Min := A[1, n];      //и минимального элементов диагоналей
For i := 2 To n Do Begin    // Поиск максимального и минимального элементов
   If A[i, i]> Max Then Max := A[i, i];
   If A[i, n-i+1] < Min Then Max := A[i, n-i+1];
   End;
```

Пример 5. Найти сумму нечетных элементов массива, расположенных ниже главной диагонали, и произведение четных элементов, расположенных выше ее.

```
Sum := 0;
For i := 2 To n Do        //Поиск нечетных элементов массива,
For j := 1 To i-1 Do // расположенных ниже главной диагонали
   If  Odd(A[i, j]) Then Sum := Sum + A[i, j];
Pr := 1 ;
Kol:= 0;
For i := 1 To n-1 Do      //Поиск четных элементов массива,
```

```
For j := i+1 To n Do // расположенных выше главной диагонали
   If  Not Odd(A[i, j]) Then Begin
      Pr:= Pr * A[i, j];
      Inc(Kol); // Kol:= Kol +1;
      End;
   If  Kol = 0Then Pr := 0; //Если четных элементов нет, то произведение равно 0
```

5.3. Пример выполнения задания

Составить программу для вычисления значения вектора $\vec{Y} = A * \vec{B}$,
где A – квадратная матрица размерностью n x n,
а Y, B – одномерные массивы размерностью n элементов.

Элементы вектора **Y** определяются по формуле $Y_i = \sum_{j=1}^{N} A_{ij} \cdot B_i$.

Следует значения n вводить в компонент TEdit, A и B – в компонент TStringGrid. После нажатия кнопки типа TButton, результат вывести в компонент TStringGrid.

Панель диалога приведена на рис. 5.1.

5.3.1. Настройка компонента TStringGrid

1. Установите на форму три компонента типа **TStringGrid**: **StringGrid1, StringGrid2** и **StringGrid3** (вкладка **Additional**, ЛКМ по пиктограмме  и ЛКМ в нужном месте формы).

2. Захватывая кромки компонентов, отрегулируйте их размер.

3. В инспекторе объектов установите значения свойств **StringGrid1**:

ColCount равным **2** (два столбца),

RowCount равным **2** (две строки),

FixedCols – 1 (один столбец с фиксированной зоной для подписи номеров строк),

FixedRows – 1 (одна строка с фиксированной зоной для подписи индексов столбцов).

Установите значения свойств **StringGrid2** и **StringGrid3**:

ColCount равным **1** (один столбец),

RowCount равным **2** (две строки),

FixedCols – 0 (нет столбца с фиксированной зоной),

FixedRows – 1 (одна строка с фиксированной зоной для подписи имени массива).

4. Раскройте раздел **Options** (нажав на знак «+», стоящий слева от **Options**) и установите свойство **goEditing** в положение **True** для компонентов

StringGrid1, **StringGrid2** и **StringGrid3** (по умолчанию в компонент **TStringGrid** запрещен ввод информации с клавиатуры).

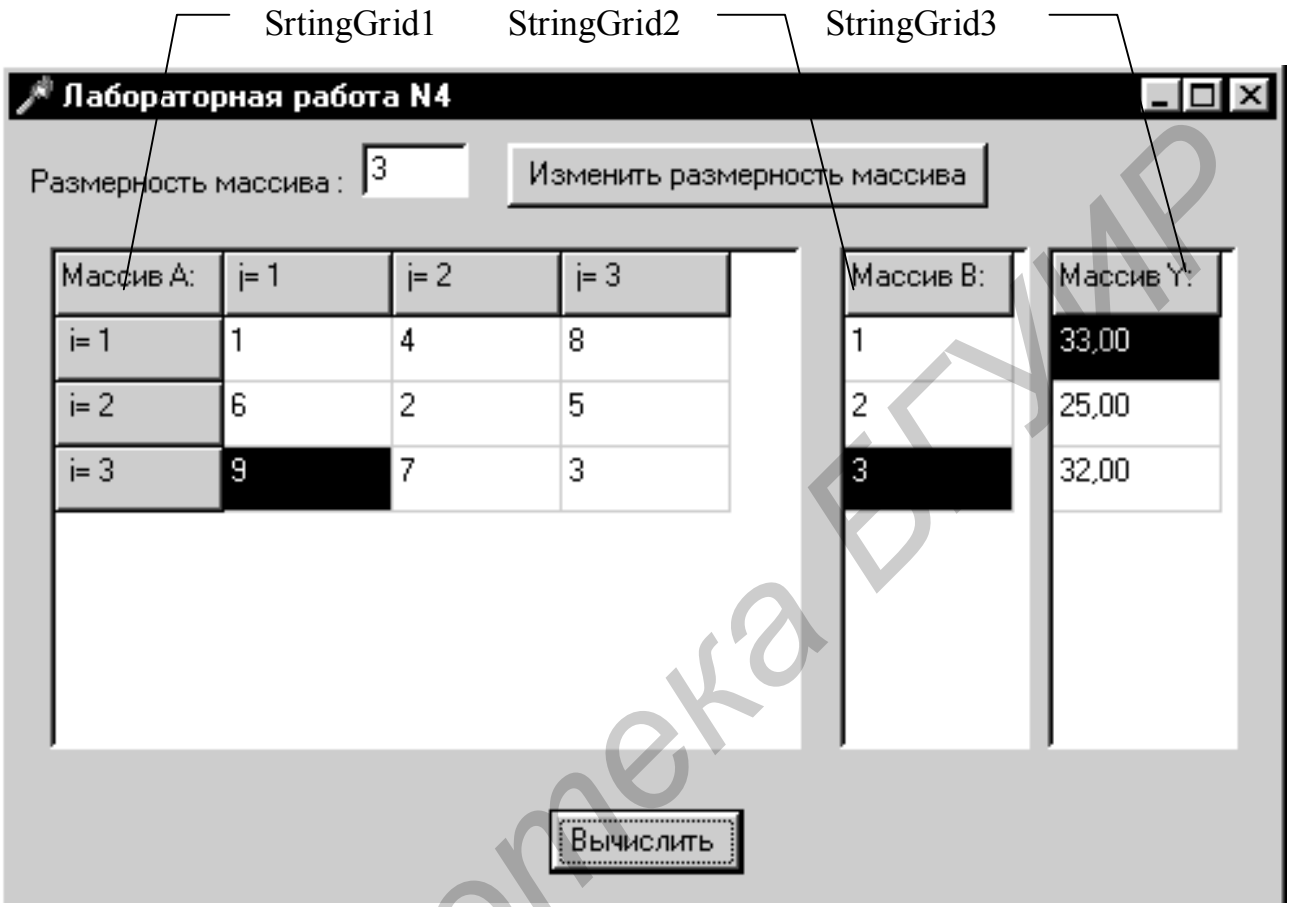


Рис. 5.1. Панель диалога формы

Панель диалога приведена на рис. 5.1.

5.3.2. Код программы

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Grids;
type
  TForm1 = class(TForm)
```

```
    Label1: TLabel;
    Edit1: TEdit;
    Button1: TButton;
    Button2: TButton;
    StringGrid1: TStringGrid;
    StringGrid2: TStringGrid;
    StringGrid3: TStringGrid;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
Const
  Nmax=10;
Type                                                    // Объявление типа
  Mas2 = Array[1..Nmax,1..Nmax] Of Extended; // двумерного массива и
  Mas1 = Array[1..Nmax] Of Extended;              // одномерного массива
Var
  Form1: TForm1;
  A : Mas2;        // Объявление двумерного массива
  B, Y : Mas1;    // Объявление одномерных массивов
  n, i, j : Integer;
implementation
{$R *.DFM}
procedure TForm1.FormCreate(Sender: TObject);
begin
    n:=3;      // Размерность массива
    Edit1.Text:= IntToStr(n);
    StringGrid1.ColCount:=n+1;       //Задание числа столбцов и
    StringGrid1.RowCount:= n +1;    // строк в таблицах,
    StringGrid2.RowCount:= n +1;// включая фиксированную зону (+1) для под-
писей
    StringGrid3.RowCount:= n +1;
    StringGrid1.Cells[0,0]:='Массив A:';  // Ввод в левую верхнюю ячейку
    StringGrid2.Cells[0,0]:='Массив B:';   // таблицы названия массива
    StringGrid3.Cells[0,0]:='Массив Y:';
    For i:=1 To n Do Begin// Заполнение таблицы поясняющими подписями:
       StringGrid1.Cells[0,i]:=' i= '+IntToStr(i);     //ввод номера строки
       StringGrid1.Cells[i,0]:=' j= '+IntToStr(i);    // ввод номера столбца
       End;
    end;
procedure TForm1.Button1Click(Sender: TObject); //Изменить размерность
```

```
begin
   n:=StrToInt(Edit1.Text);
   StringGrid1.ColCount:= n +1;     // Задание числа столбцов и
   StringGrid1.RowCount:= n +1;     // строк в таблицах, включая
   StringGrid2.RowCount:= n +1; // фиксированную зону (+1) для подписей
   StringGrid3.RowCount:= n +1;
   For i:=1 To n Do Begin                          // Ввод поясняющих подписей:
      StringGrid1.Cells[0,i]:=' i= '+IntToStr(i);    // номера строки
      StringGrid1.Cells[i,0]:=' j= '+IntToStr(i);    // номера столбца
   End;
 end;
procedure TForm1.Button2Click(Sender: TObject); // Вычислить
Var
   s: Extended;
begin
   For i:=1 To n Do // Заполнение массива A элементами из таблицы StringGrid1
   For j:=1 To n Do
   A[i,j]:=StrToFloat(StringGrid1.Cells[j,i]);
   For i:=1 To n Do   // Заполнение массива B элементами из таблицы StringGrid2
   B[i]:=StrToFloat(StringGrid2.Cells[0,i]);
   For i:=1 To n Do Begin     // Умножение массива A на массив B
      s:=0;
      For j:=1 To n Do
      s:=s+A[i,j]*B[j];
      Y[i]:=s;
      StringGrid3.Cells[0,i]:=FloatToStrf(Y[i],ffFixed,6,2); // Вывод результата
      End;
   End;
```

5.4. Индивидуальные задания

В соответствии с индивидуальным вариантом выполнить вариант индивидуального задания. Во всех заданиях скалярные переменные вводить с помощью компонента TEdit с соответствующим пояснением в виде компонента TLabel. Скалярный результат выводить в виде компонента TLabel. Массивы представлять на форме в виде компонентов TStringGrid, в которых 0-й столбец и 0-ю строку использовать для отображения индексов массивов. Вычисления выполнять после нажатия кнопки типа TBitBtn. Для выхода из программы использовать кнопку Close.

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

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

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

4. Задана матрица размером $n \cdot m$. Определить k – количество «особых» элементов матрицы, считая элемент «особым», если он больше суммы остальных элементов своего столбца.

5. Задана матрица размером $n \cdot m$. Определить k – количество «особых» элементов матрицы, считая элемент «особым», если в его строке слева от него находятся элементы, меньшие его, а справа – большие.

6. Задана символьная матрица размером $n \cdot m$. Определить k – количество различных элементов матрицы (повторяющиеся элементы считать один раз).

7. Дана матрица размером $n \cdot m$. Упорядочить ее строки по неубыванию их первых элементов.

8. Дана матрица размером $n \cdot m$. Упорядочить ее столбцы по неубыванию суммы их элементов.

9. Найти в каждом столбце матрицы максимальный элемент и записать их в новый массив. Среди найденных элементов определить минимальный.

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

11. Определить количество отрицательных элементов, расположенных выше главной диагонали матрицы.

12. Определить количество положительных элементов, расположенных ниже побочной диагонали матрицы.

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

14. В матрице размером $n \cdot m$ поменять местами строку, содержащую элемент с наибольшим значением, со строкой, содержащей элемент с наименьшим значением.

15. Определить сумму элементов, расположенных на главной диагонали матрицы, и произведение элементов, расположенных на побочной диагонали.

16. Для матрицы размером $n \cdot m$ вывести на экран все ее седловые точки. Элемент матрицы называется седловой точкой, если он является наименьшим в своей строке и одновременно наибольшим в своем столбце, или наоборот.

17. 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.

18. В матрице n-го порядка переставить строки так, чтобы на главной диагонали матрицы были расположены элементы, наибольшие по абсолютной величине.

19. В двумерном массиве поменять местами столбцы, содержащие соответственно максимальный и минимальный элементы массива.

20. Пребразовать двумерную матрицу размерности $n \cdot n$ в одномерный массив путем обхода всех элементов двумерной матрицы по скручивающейся против часовой стрелки спирали.

21. Определить количество столбцов матрицы, состоящих только из одних нулей.

22. Определить произведение элементов тех столбцов матрицы, последний элемент которых равен нулю.

23. Из двумерного массива размерности $n \cdot n$ построить одномерный путем обхода его элементов по скручивающейся по часовой стрелке спирали.

24. Пребразовать двумерную матрицу размерности $n \cdot n$ в одномерный массив путем обхода всех элементов двумерной матрицы по скручивающейся по часовой стрелке спирали.

25. Для каждого столбца матрицы определить количество элементов, совпадающих с первым элементом столбца.

26. Дана матрица размером $n \cdot m$. Найти в каждой строке наибольший элемент и поменять его местами с элементами главной диагонали.

27. Дана матрица размером $n \cdot m$. Подсчитать количество столбцов матрицы, сумма элементов которых кратна трем, но не кратна пяти.

28. В каждом столбце матрицы определить количество элементов, больших среднего арифметического значения соответствующего столбца.

29. Поменять местами элементы четных и нечетных столбцов матрицы.

30. В каждой строке матрицы подсчитать количество отрицательных элементов, кратных пяти, и записать найденные элементы в новый одномерный массив.

Лабораторная работа №6. Программирование с использованием строк

Цель работы: изучить правила работы с компонентами **TListBox** и **TComboBox**. Составить и отладить программу работы со строками.

6.1. Типы данных для работы со строками

6.1.1. Короткие строки типа ShortString и String[N]

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

6.1.2. Длинная строка типа AnsiString

При работе с этим типом данных память выделяется по мере необходимости (динамически) и может занимать до двух Гбайт памяти, но для каждого символа отводится один байт.

Процедуры и функции для работы с короткими и длинными строками представлены в прил. 4.

6.1.3. Широкая строка типа WideString

Этот тип аналогичен типу **AnsiString**, но для каждого символа выделяется по два байта памяти, что обеспеченивает совместимость с кодировкой **Unicode**.

6.1.4. Нуль-терминальная строка типа PChar

Представляет собой цепочку символов, ограниченную символом **#0**. Нуль-терминальные строки широко используются при обращениях к **API**-функциям Windows (**API** – **Application Program Interface** – интерфейс прикладных программ).

6.1.5. Представление строки в виде массива символов

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

```
Var
    MasC : Array[1…100] Of Char;
```

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

6.2. Компонент TListBox

Компонент **TListBox** представляет собой список, элементы которого выбираются при помощи клавиатуры или мыши. Список элементов задается свойством **Items**, методы **Add**, **Delete** и **Insert** которого используются для добавления, удаления и вставки строк. Объект **Items** (**TString**) хранит строки, находя-

щиеся в списке. Для определения номера выделенного элемента используется свойство **ItemIndex**.

6.3. Компонент TComboBox

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

6.4. Обработка событий

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

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

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

Таблица 6.1

Наиболее часто применяемые события

Событие	Описание события
1	2
OnActivate	Событие наступает при активации компонента
OnCreate	Возникает при создании компонента (выделения динамической памяти). В обработчике данного события следует задавать действия, которые должны происходить в момент создания компонента, например установка начальных значений
OnEnter	Компонент получает фокус, т.е. становится активным
OnExit	Компонент теряет фокус, т.е. теряет активность

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

6.5. Пример выполнения задания

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

Панель диалога приведена на рис. 6.1.

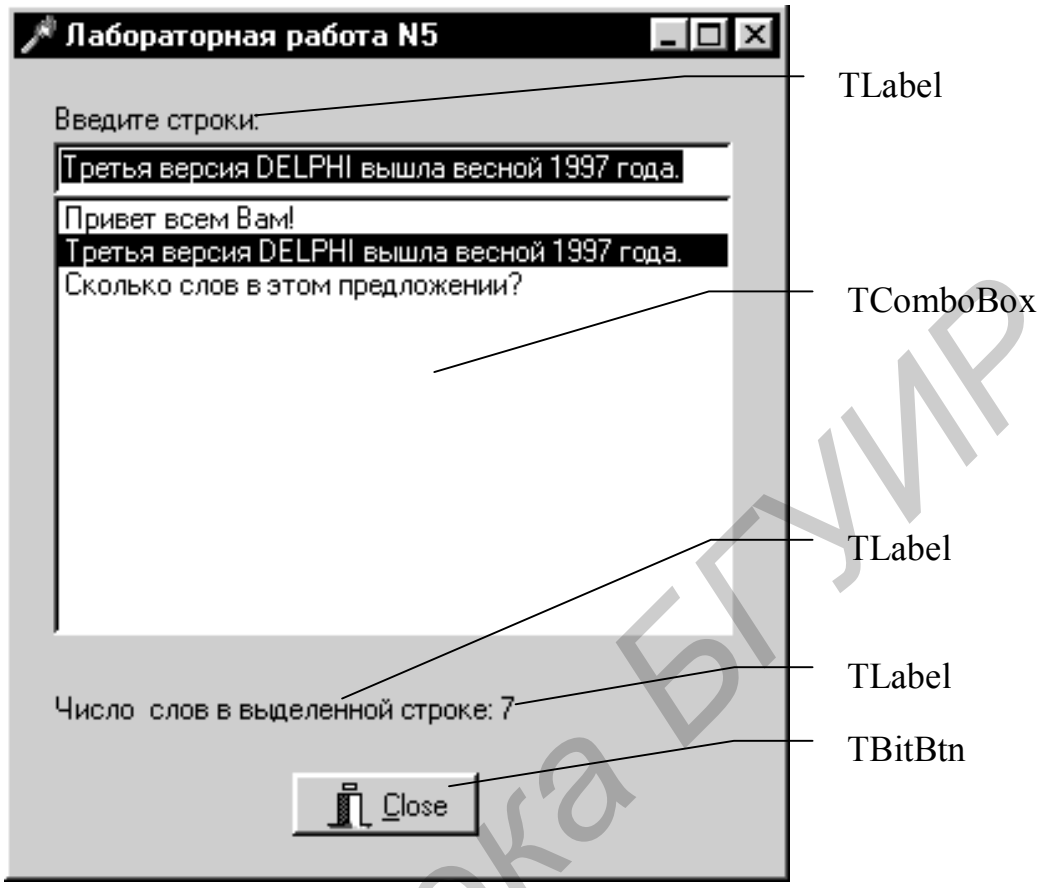


Рис. 6.1. Панель диалога формы

6.5.1. Код программы

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons;
type
  TForm1 = class(TForm)
    Label2: TLabel;
    Label3: TLabel;
    BitBtn1: TBitBtn;
    ComboBox1: TComboBox;
    Label1: TLabel;
    procedure FormActivate(Sender: TObject);
    procedure ComboBox1KeyPress(Sender: TObject; var Key: Char);
    procedure ComboBox1Click(Sender: TObject);
  private
    { Private declarations }
  public
```

```
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}

procedure TForm1.FormActivate(Sender: TObject); // Обработка события
begin                                           // активизации формы
   ComboBox1.SetFocus;      // Передача фокуса ComboBox1
end;
procedure TForm1.ComboBox1KeyPress(Sender: TObject; var Key: Char);
begin                    // Обработка события нажатия левой клавиши мыши
   If Key=#13 Then Begin                  // При нажатии клавиши Enter:
                                          // строка из окна редактирования
      ComboBox1.Items.Add(ComboBox1.Text)  // заносится в список выбора,
      ComboBox1.Text:='';                 // очищается окно редактирования
      End;
   end;

procedure TForm1.ComboBox1Click(Sender: TObject);
Var
   st : String;
    n,i,nst,ind: Integer;
Begin
   n:=0;   // Число слов равно 0
   ind:=0;
   nst:=ComboBox1.ItemIndex;  // Определение номера выбранной строки
   st:=ComboBox1.Items[nst];    // Занесение выбранной строки в переменную st
   For i:=1 To Length(st) Do Begin    // Просмотр всех символов строки st
      Case ind Of
         0: If st[i]<>' ' Then Begin     // Если после пробела встретился символ, то
            ind:=1;
            Inc(n);                      // число слов увеличивается на единицу
            End;
         1: If st[i]=' ' Then ind:=0;     // Если после символов встретился пробел
         End;   // Case
   End;      // For
   Label3.Caption:=IntToStr(n);         // Вывод числа слов в Label3
  End;
```

6.6. Индивидуальные задания

В соответствии с индивидуальным вариантом выполнить вариант индивидуального задания. Во всех заданиях исходные данные вводить с помощью компонента TEdit в компонент TListBox либо с помощью свойства Text в свой-

ство Items компонента TComboBox. Скалярный результат выводить с помощью компонента TLabel. Ввод строки заканчивать нажатием клавиши Enter. Для выхода из программы использовать кнопку Close. При выполнении задания ввести несколько различных строк.

1. Найти количество слов, состоящих из пяти символов.

2. Найти самое короткое слово в строке.

3. Дана строка символов, состоящая из произвольного текста, слова разделены пробелами. Преобразовать последовательность, заменив пробелы между словами на символ звездочки *** .**

4. Определить количество символов во втором слове.

5. Дана строка символов, среди которых есть двоеточия. Выведите все символы, расположенные до первого двоеточия.

6. Подсчитать количество слов, начинающихся с буквы «а».

7. Найти слова, содержащие букву «**s**».

8. Найти самое длинное слово в строке.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

23. Вывести на экран римскими цифрами заданное целое число от 1 до 1999.

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

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

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

27. В строке переставить слова в обратном порядке.

28. В строке расположить слова в порядке увеличения их длины.

29. В коде программы заменить слова Begin и End на Начало и Конец.

30. В строке поменять порядок символов на обратный.

Лабораторная работа № 7. Программирование с использованием записей и файлов

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

7.1. Программирование с использованием переменных типа запись

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

```
Type
    TStudent = Record                  //Объявление типа запись
      Fio: String[20];                 //Поле фио
      Group: Integer;                  //Поле номера студ. группы
      Ocn: Array[1..3] Of Integer;     //Поле массива оценок
      end;
Var
    Student: TStudent;                 //Объявление переменной типа запись
```

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

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

Доступ к полям можно осуществлять также при помощи оператора **With**:

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

7.2. Работа с файлами

Файл – это именованный набор данных на внешнем физическом носителе. В Delphi 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. Каждая строка в таком файле заканчивается двумя специальными символами: **#10** – конец строки и **#13** – переход на следующую строку. Примером текстового файла может служить файл исходного кода программы в DELPHI (расширение ***.pas**). Для работы с текстовым файлом должна быть описана соответствующая файловая переменная, *например*: **Var F: TextFile;**.

Типизированные файлы состоят из записей одинаковой длины, которые имеют строго заданную в описании **Record** структуру. Это свойство типизиро-

ванных файлов позволяет получить доступ к любой записи файла по его порядковому номеру. В описании файловой переменной указывается ее тип, *например*: **Var F: TStudent;**.

Нетипизированные файлы похожи на типизированные, только одна запись здесь называется блоком, который воспринимается как последовательность байт. Длина блока по умолчанию – **128** байт. Файловая переменная объявляется как, *например:* **Var F: File;**.

Порядок работы с файлами следующий:

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

7.3. 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 компонента 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 – текстовых.

Write(F,v1[,v2,…vn]) – запись значений переменных начиная с текущей позиции для типизированных файлов и строк – 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 (по умолчанию размер буфера – 128 байт).

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

Truncate(Var F) – урезает файл начиная с текущей позиции.

LoResult: Integer – код результата последней операции ввода-вывода.

FilePos(Var F): Longint – для нетекстовых файлов возвращает номер текущей позиции. Отсчет ведется от нуля.

FileSize(Var F): Longint – для нетекстовых файлов возвращает количество компонентов в файле.

Eoln(Var F: TextFile): Boolean – возвращает **True**, если достигнут конец строки.

Eof(Var F)): Boolean – возвращает ***True***, если достигнут конец файла.

SeekEoln(Var F: TextFile): Boolean – возвращает **True**, если пройден последний значимый символ в строке или файле, отличный от пробела или знака табуляции.

SeekEof(Var F: TextFile): Boolean – то же, что и **SeekEoln**, но для всего файла.

BlockRead(Var F: File; Var Buf; Count: Word[; Result: Word]) ,

BlockWrite(Var F: File; Var Buf; Count: Word[; Result: Word]) – соответственно процедуры чтения и записи переменной **Buf** с количеством ***Count*** блоков.

7.4. Компоненты TOpenDialog и TSaveDialog

Компоненты **TOpenDialog**  и **TSaveDialog**  находятся на странице **DIALOGS** (с. прил. 2). Все компоненты этой страницы вызываются методом **Execute** и 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**.

7.5. Пример выполнения задания

Составить программу для ввода в файл или чтения из файла ведомости абитуриентов, сдавших вступительные экзамены. Сведения об абитуриенте содержат: фамилию и оценки по физике, математике и русскому языку. Вывести список абитуриентов, отсортированный в порядке уменьшения их среднего балла, и записать эту информацию в текстовой файл.

7.5.1. Настройка компонентов TOpenDialog и TSaveDialog

Для установки компонентов **TOpenDialog** и **TSaveDialog** на форму необходимо на странице **Dialogs** меню компонентов щелкнуть мышью соответственно по пиктограммам  или  и поставить их в любое свободное место формы.

Установка фильтра производится следующим образом:

1. Выбрав соответствующий компонент (**TOpenDialog** или **TSaveDialog**), 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 – маска.

2. Значение маски для **OpenDialog1** устанавливается как показано на рис. 7.1. 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).

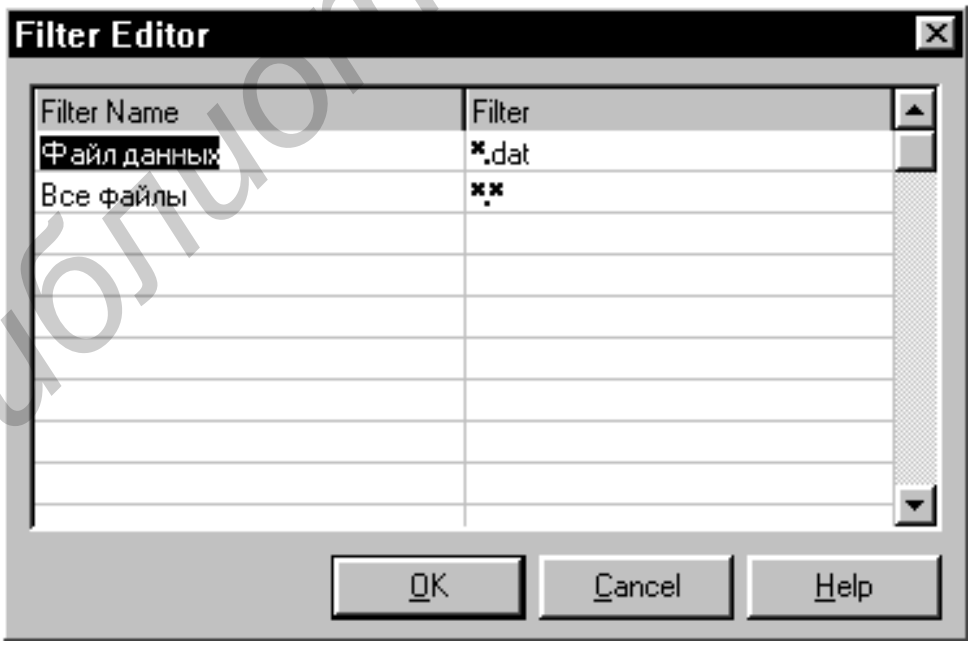


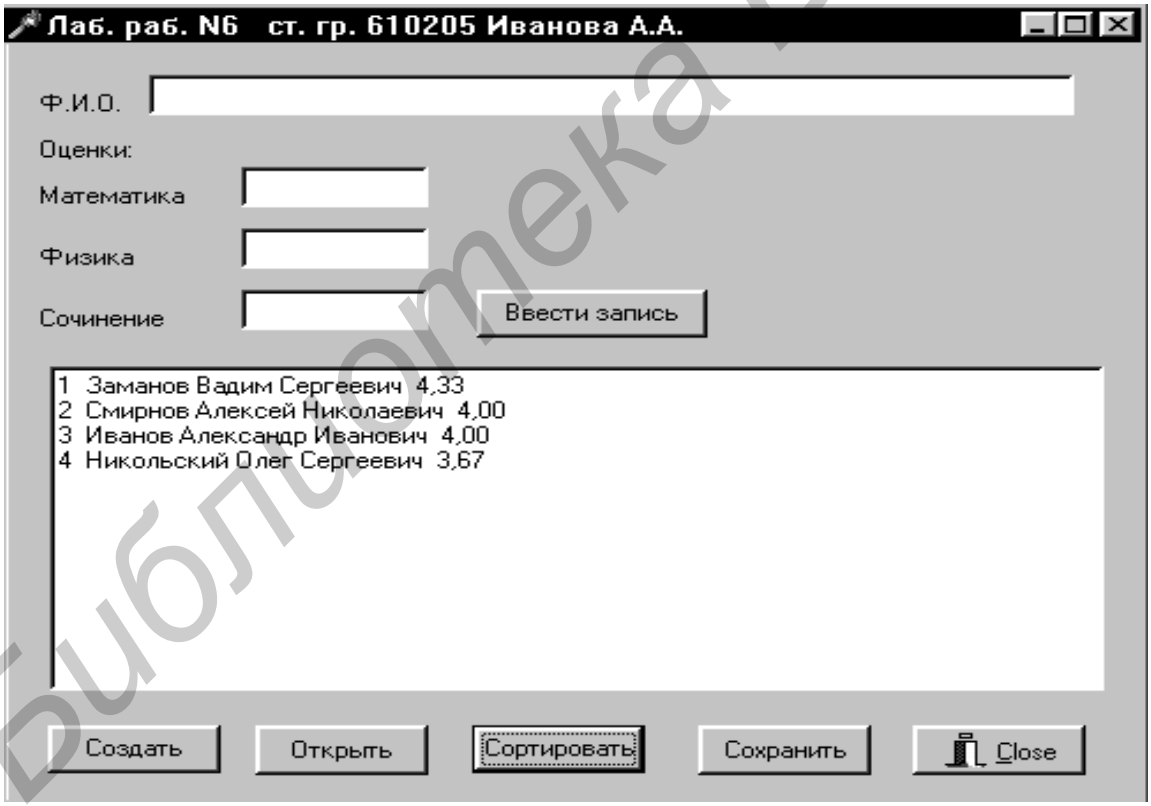
Рис. 7.1. Установка значения маски для **TOpenDialog**

3. Для того чтобы файл автоматически записывался с расширением ***.dat***, в свойстве ***DefaultExt*** надо записать требуемое расширение – ***.dat***.

Аналогичным образом настраивается ***SaveDialog1*** для текстового файла (расширение ***.txt***).

7.5.2. Работа с программой

После запуска программы на выполнение появится диалоговое окно программы. Кнопка «***Ввести запись***» не будет видна. 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***» ведомость абитуриентов будет отсортирована по убыванию среднего балла и диалоговое окно примет вид, показаный на рис. 7.2. 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 ✕.



7.2**.** Вид диалогового окна после нажатия на кнопку «Сортировка»

7.5.3. Код программы

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, ExtCtrls;

type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Memo1: TMemo;
    Button1: TButton;
    Button3: TButton;
    Splitter1: TSplitter;
    Button5: TButton;
    BitBtn1: TBitBtn;
    SaveDialog1: TSaveDialog;
    Button2: TButton;
    OpenDialog1: TOpenDialog;
    Button4: TButton;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

Type
  TStudent = Record
    FIO: String[40];              // Поле ФИО.
    Otc: Array[1..3] Of Word;     // Поле массива оценок
```

```
    Sball : Extended;              // Поле среднего балла
  End;
Var
  Fz  : File Of  Tstudent;         // Файл типа запись
  Ft  : TextFile;                  // Текстовой файл
  Stud : Array[1..100] of TStudent; // Массив записей
  nzap : Integer;                  // Номер записи
  FileNameZ, FileNameT : String;   // Имя файла
Var
  Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.FormCreate(Sender: TObject);
Begin
   Edit1.Text:=' ';
   Edit2.Text:=' ';
   Edit3.Text:=' ';
   Edit4.Text:=' ';
   Memo1.Clear;
   Button1.Hide;    // Сделать невидимой кнопку «Ввести запись»
   nzap:=0;
   End;
procedure TForm1.Button1Click(Sender: TObject);   // Ввести новую запись
Begin
   nzap:=nzap+1;
   With Stud[nzap] Do Begin
      FIO:=Edit1.Text;
      Otc[1]:=StrToInt(Edit2.Text);
      Otc[2]:=StrToInt(Edit3.Text);
      Otc[3]:=StrToInt(Edit4.Text);
      Sball:=(Otc[1]+Otc[2]+Otc[3])/3;
      Memo1.Lines.Add(Fio+'  '+IntToStr(Otc[1])+'  '+ IntToStr(Otc[2])+
                              '   '+IntToStr(Otc[3]));
      End;
   Write(Fz,Stud[nzap]);              // Запись в файл
   Edit1.Text:=' ';
   Edit2.Text:=' ';
   Edit3.Text:=' ';
   Edit4.Text:=' ';
   End;
procedure TForm1.Button2Click(Sender: TObject); // Создание нового файла
                                                  записей
Begin
   OpenDialog1.Title := 'Создать новый файл'; // Изменение заголовка окна
```

```
                                                    // диалога
   If OpenDialog1.Execute Then // Выполнение стандартного диалога выбора
   Begin                                  // 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 существующий
Begin                                                    // файл
   If OpenDialog1.Execute then Begin      // Выполнение стандартного диалога
                                          // выбора имени файла
      FileNameZ:= OpenDialog1.FileName;// Запоминание имени дискового
   файла
      AssignFile(Fz, FileNameZ);    // Связывание файловой переменной Fz с
                                          // именем файла
\     Reset(Fz);                        // Открытие существующего файла
      End;
   While Not Eof(Fz) Do Begin
      nzap:=nzap+1;
      Read(Fz,Stud[nzap]);      // Чтение записи из файла
      With Stud[nzap] Do
      Memo1.Lines.Add (Fio+'  '+IntToStr(Otc[1])+'  '+IntToStr(Otc[2])+
                              '  '+IntToStr(Otc[3]));
      End;
   Button1.Show;                  // Сделать видимой кнопку «Ввести запись»
   End;
procedure TForm1.Button4Click(Sender: TObject); // Сортировка записей
 Var
      i,j :  Word;
      st : TStudent;
Begin
   For i:=1 To nzap-1 Do                 // Сортировка массива записей
   For j:=i+1 To nzap Do
   if Stud[i].Sball < Stud[j].Sball Then Begin
      st:=Stud[i];
      Stud[i]:=Stud[j];
      Stud[j]:=st;
      End;
   Memo1.Clear;
   For i:=1 To nzap Do     // Вывод в окно Memo1 отсортированных записей
   With Stud[i] Do
   Memo1.Lines.Add (IntToStr(i)+'  '+Fio+'  '+FloatToStrf(Sball,ffFixed,5,2));
```

```
  end;
procedure TForm1.Button5Click(Sender: TObject); // Сохранение результатов
Var                                            //сортировки в текстовом
файле
   I : Word;
Begin
  If SaveDialog1.Execute Then Begin  // Выполнение стандартного диалога
                                     // выбора имени файла
    FileNameT:= SaveDialog1.FileName; // Возвращение имени дискового
  файла
    AssignFile(Ft, FileNameT); // Связывание файловой переменной Ft с име-
  нем
                               // файла
    Rewrite(Ft);               // Открытие нового текстового файла
    End;
  For i:=1 To nzap Do
  With Stud[i] Do
  WriteLn (Ft, i:4, '.', Fio:40, Sball:6:2);// Запись в текстовой файл
  CloseFile(Ft);       // Закрытие текстового файла
  End;
procedure TForm1.BitBtn1Click(Sender: TObject);
Begin
  CloseFile(Fz);       // Закрытие файла записей при нажатии на кнопку
  «Close»
  End;
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
Begin
  CloseFile(Fz);       // Закрытие файла записей при нажатии на кнопку
  End;
End.
```

7.6. Индивидуальные задания

В соответствии с индивидуальным вариантом выполнить вариант индивидуального задания. В программе предусмотреть сохранение вводимых данных в файл и возможность чтения из ранее сохраненного файла. Результаты выводить в окно просмотра и в текстовой файл.

1. В магазине формируется список лиц, записавшихся на покупку товара повышенного спроса. Каждая запись этого списка содержит: порядковый номер, ФИО, домашний адрес покупателя и дату постановки на учет. Удалить из списка все повторные записи, проверяя ФИО и домашний адрес.

2. Список товаров, имеющихся на складе, включает в себя: наименование товара, количество единиц товара, цену единицы и дату поступления товара на склад. Вывести в алфавитном порядке список товаров, хранящихся больше месяца, стоимость которых превышает 1000000 руб.

3. Для получения места в общежитии формируется список студентов, который включает: ФИО студента, группу, средний балл, доход на члена семьи. Общежитие в первую очередь предоставляется тем, у кого доход на члена семьи меньше двух минимальных зарплат, затем остальным в порядке уменьшения среднего балла. Вывести список очередности предоставления мест в общежитии.

4. В справочной автовокзала хранится расписание движения автобусов. Для каждого рейса указаны: его номер, тип автобуса, пункт назначения, время отправления и прибытия. Вывести информацию о рейсах, которыми можно воспользоваться для прибытия в пункт назначения раньше заданного времени.

5. На междугородной АТС информация о разговорах содержит: дату разговора, код и название города, время разговора, тариф, номер телефона в этом городе и номер телефона абонента. Вывести по каждому городу общее время разговоров с ним и сумму.

6. Информация о сотрудниках фирмы включает: ФИО, табельный номер, количество отработанных часов за месяц, почасовой тариф. Рабочее время свыше 144 часов считается сверхурочным и оплачивается в двойном размере. Вывести размер заработной платы каждого сотрудника фирмы за вычетом подоходного налога, который составляет 12 % от суммы заработка.

7. Информация об участниках спортивных соревнований содержит: наименование страны, название команды, ФИО игрока, его игровой номер, возраст, рост и вес. Вывести информацию о самой молодой, рослой и легкой команде.

8. Для книг, хранящихся в библиотеке, задаются: регистрационный номер книги, автор, название, год издания, издательство, количество страниц. Вывести список книг с фамилиями авторов в алфавитном порядке, изданных после заданного года.

9. Различные цехи завода выпускают продукцию нескольких наименований. Сведения о выпущенной продукции включают: наименование, количество, номер цеха. Для заданного цеха необходимо вывести количество выпущенных изделий по каждому наименованию в порядке убывания количества.

10. Информация о сотрудниках предприятия содержит: ФИО, номер отдела, должность, дату начала работы. Вывести списки сотрудников по отделам в порядке убывания стажа.

11. Ведомость абитуриентов, прошедших централизованное тестирование и поступающих в университет, содержит: ФИО, адрес, баллы по тестированию. Определить количество абитуриентов, проживающих в г.Минске и набравших количество баллов не ниже 280, вывести их фамилии в алфавитном порядке.

12. В справочной аэропорта хранится расписание вылета самолетов на следующие сутки. Для каждого рейса указаны: номер рейса, тип самолета, пункт на-

значения, время вылета. Вывести все номера рейсов, типы самолетов и время вылета для заданного пункта назначения в порядке возрастания времени вылета.

13. У администратора железнодорожных касс хранится информация о свободных местах в поездах дальнего следования на ближайшую неделю в следующем виде: дата выезда, пункт назначения, время отправления, число свободных мест. Оргкомитет международной конференции обращается к администратору с просьбой зарезервировать m мест до города N на k-й день недели со временем отправления поезда не позднее t часов вечера. Вывести время отправления или сообщение о невозможности выполнить заказ в полном объеме.

14. Ведомость студентов, сдавших сессию, содержит: ФИО студента и его оценки по пяти предметам. Определить средний балл по университету и вывести список студентов, средний балл которых выше среднего балла по университету. Первыми в списке должны идти студенты, сдавшие все экзамены на 9 и 10.

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

16. Ведомость об успеваемости студентов содержит: номер группы, ФИО студента, оценки за последнюю сессию. Вывести списки студентов по группам в порядке убывания среднего балла.

17. В исполкоме формируется список учета нуждающихся в улучшении жилищных условий. Каждая запись списка содержит: порядковый номер, ФИО, величину жилплощади на одного члена семьи и дату постановки на учет. По заданному количеству квартир, выделяемых в течение года, вывести список очередников с указанием ожидаемого года получения квартиры.

18. Имеется список женихов и список невест. Каждая запись списка содержит: пол, имя, возраст, рост, вес, а также требования к партнеру: наименьший и наибольший возраст, вес, рост. Объединить эти данные в список пар с учетом требований к партнерам без повторений женихов и невест.

19. В библиотеке имеется список книг, каждая запись которого содержит: фамилию автора, название книги, год издания. Вывести информацию о книгах, в названии которых встречается заданное ключевое слово (ввести с клавиатуры).

20. В магазине имеется список поступивших в продажу автомобилей. Каждая запись этого списка содержит: марку автомобиля, стоимость, расход топлива на 100 км, надежность (число лет безотказной работы), комфортность (отличная, хорошая, удовлетворительная). Вывести перечень автомобилей, удовлетворяющих требованиям покупателя, которые вводятся с клавиатуры в виде некоторого интервала допустимых значений.

21. Каждая запись списка вакантных рабочих мест содержит: наименование организации, должность, квалификацию (разряд или образование), стаж ра-

боты по специальности, заработную плату, наличие социального страхования (да/нет), продолжительность ежегодного оплачиваемого отпуска. Вывести список рабочих мест в соответствии с требованиями клиента.

22. В технической службе аэропорта имеется справочник, содержащий записи следующей структуры: тип самолета, год выпуска, расход горючего на 1000 км. Для определения потребности в горючем техническая служба запрашивает расписание полетов. Каждая запись расписания содержит следующую информацию: номер рейса, пункт назначения, дальность полета. Вывести суммарное количество горючего, необходимое для обеспечения полетов на следующие сутки.

23. Поля шахматной доски характеризуются записью:

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

Вывести шахматную доску, пометив крестиками все поля, которые «*бьет*» ферзь, стоящий на поле с координатами $\mathbf{Ver}_i$ и $\mathbf{Hor}_i$, и ноликами – остальные поля.

24. Для участия в конкурсе на замещение вакантной должности сотрудника фирмы желающие подают следующую информацию: ФИО, год рождения, образование (среднее, специальное, высшее), знание иностранных языков (английский, немецкий, французский, владею свободно, читаю и перевожу со словарем), владение компьютером (MS DOS,Windows и т.д.), стаж работы, наличие рекомендаций. Вывести список претендентов в соответствии с требованиями руководства фирмы.

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

26. Список студентов группы содержит следующую информацию: ФИО, рост и вес. Вывести ФИО студентов, рост и вес которых чаще всего встречается в списке.

27. В магазине имеется перечень ноутбуков с указанием марки, цены, типа процессора, объема оперативной памяти, объема дисковой памяти и их количества. Вывести полную стоимость всех компьютеров.

28. В роддоме ведется учет родившихся детей. В списке указывается: ФИО матери ребенка, дата и время его рождения, ФИО врача, принимавшего

роды, пол ребенка, его вес и окружность головы, дата выписки из роддома. Определить число детей, родившихся с заданным весом.

29. В метеоцентре ведется наблюдение за погодой и каждый день на состояние 12:00 часов делается запись, включающая: дату, атмосферное давление, направление и силу ветра, облачность, осадки, влажность. Определить дни с минимальным давлением.

30. В домоуправлении ведется учет жильцов данного района, который включает следующие сведения: улица, номер дома, номер квартиры, метраж, число комнат; для каждого проживающего указывается: ФИО, дата рождения, дата прописки и выписки, отношение к владельцу квартиры. Определить число жильцов, проживающих в однокомнатных квартирах.

Лабораторная работа №8. Программирование с использованием подпрограмм и модулей. Построение графиков функций

Цель работы: изучить возможности DELPHI для написания подпрограмм и создания библиотечных модулей, а также построения графиков функций с помощью компонента отображения графической информации **TChart**. Написать и отладить программу построения на экране графика заданной функции, использующую внешний модуль **UNIT** с подпрограммой.

8.1. Построение графика функции с помощью компонента TChart

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

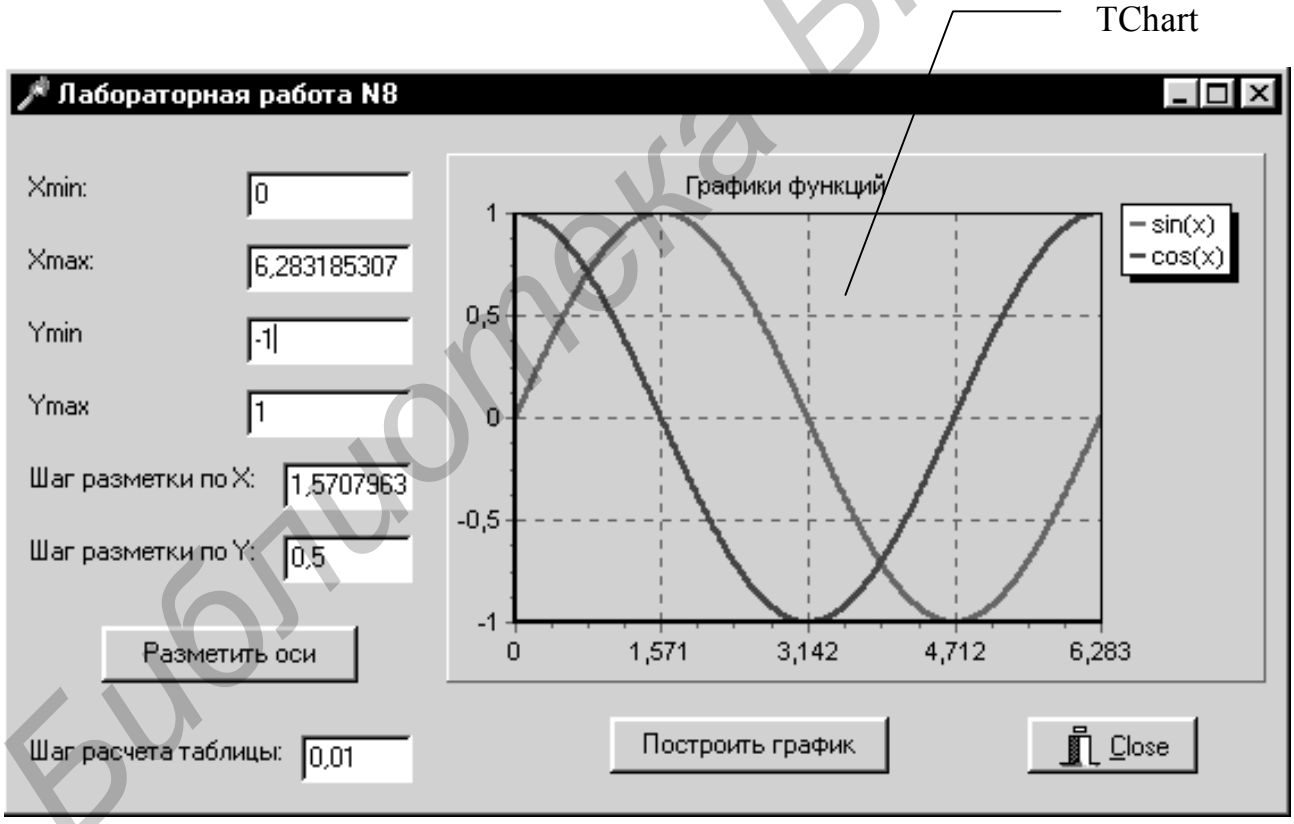


Рис. 8.1. Вывод графиков функций

Построение графика (диаграммы) производится после вычисления таблицы значений функции y=f(x) на интервале [Xmin, Xmax] с заданным шагом. Полученная таблица передается в специальный двумерный массив **Series<u>k</u>** (**k** – но-

мер графика) компонента **TChart** с помощью метода **Add**. Компонент **TChart** осуществляет всю работу по отображению графиков, переданных в объект **Series**__**k**__: строит и размечает оси, рисует координатную сетку, подписывает название осей и самого графика, отображает переданную таблицу в виде всевозможных графиков или диаграмм. При необходимости с помощью встроенного редактора **EditingChart** компоненту **TChart** передаются данные о толщине, стиле и цвете линий, параметрах шрифта подписей, шагах разметки координатной сетки и другие настройки. В процессе работы программы изменение параметров возможно через обращение к соответствующим свойствам компонента **TChart**. *Например*, свойство **Chart1**.**BottomAxis** содержит значение максимального предела нижней оси графика и при его изменении во время работы программы автоматически изменяется изображение графика.

8.2. Использование подпрограмм

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

Подпрограммы используются в том случае, когда одна и та же последовательность операторов в коде программы повторяется несколько раз. Эта последовательность заменяется вызовом подпрограммы, содержащей необходимые операторы. Подпрограммы также применяются для создания библиотечных модулей с целью использования их другими программистами.

Подпрограммы подразделяются на процедуры и функции.

Процедура имеет следующую структуру:

```
Procedure <имя процедуры> ([список формальных параметров]); [директива;]
Const     [описание используемых констант];
Type      [описание используемых типов];
Var       [описание используемых переменных];
Begin
   // Операторы
End;
```

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

```
Function <имя функции> ([список формальных параметров]) :
                                                  <тип результа-
                                                  та>;[директива;]
Const     [описание используемых констант];
Type      [описание используемых типов];
Var       [описание используемых переменных];
Begin
  // Операторы
  Result:= …;   // Занесение результата вычислений в Result
End;
```

Каждой запускаемой программе операционной системой выделяется четыре сегмента памяти:

1) кодовый сегмент;
2) сегмент глобальных данных;
3) сегмент стека;
4) сегмент динамической памяти.

При вызове любой подпрограммы в стек записываются:

– адрес возврата в вызывающую программу;
– список передаваемых в подпрограмму параметров;
– выделяется память для всех внутренних переменных и массивов.

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.

Существует пять видов формальных параметров:

1. Параметры-значения. Перед такими параметрами не ставится никаких ключевых слов. Значения таких параметров передаются через стек, и в вызывающую программу они не возвращаются. Такими параметрами не могут быть файловые переменные и структуры, их содержащие.

2. Параметры-переменные. Перед такими параметрами записывается ключевое слово Var. В этом случае через стек в подпрограмму передается адрес фактического параметра, и поэтому изменение этого параметра в подпрограмме приводит к его изменению и для вызывающей программы.

3. Параметры-константы. Перед такими параметрами записывается ключевое слово Const. Эти параметры похожи на внутренние константы или параметры, доступные только для чтения. Параметры константы подобны параметрам-переменным, только внутри подпрограммы им нельзя присваивать никаких значений и их нельзя передавать в другие подпрограммы, как параметры-переменные.

4. Выходные параметры. Перед такими параметрами записывается ключевое слово Out. Эти параметры имеют такие же свойства, как и параметры-переменные, однако им не присваиваются какие-либо значения при вызове подпрограммы.

5. Нетипированные параметры. Для таких параметров не указывается тип параметра. При этом через стек передается адрес фактического параметра, а внутри подпрограммы по этому адресу можно расположить переменную любого типа.

При использовании подпрограмм нужно придерживаться следующих правил:

– фактическими параметрами при вызове подпрограмм могут быть переменные, константы и целые выражения;

– формальными параметрами при описании подпрограмм могут быть только имена переменных;
– фактические и формальные параметры должны быть согласованы по типу, порядку следования и количеству.

Подпрограммы могут иметь директивы, которые записывают после объявления подпрограммы. Эти директивы могут определять правила вызова подпрограмм и передачи в них параметров. Директивой по умолчанию является директива **Register,** которая так же, как и **Pascal,** определяет передачу параметров в стек **слева направо**, т.е. так, как они записаны в определении подпрограммы.

Директивы **Cdecl, Stdcall** и **Safecall** определяют передачу параметров наоборот, т.е. **справа налево**, как это принято в языке Си.

Для всех директив, кроме **Cdecl,** освобождение стека происходит до выхода из подпрограммы, а для **Cdecl** – после передачи управления вызывающей программе.

Директива **Register** передает первые три параметра через регистры процессора, в то время как остальные передают все параметры через стек.

Директива **Safecall** используется при работе с **OLE Automation** интерфейсами.

Директива **Forward** означает, что дано только описание параметров вызова процедуры, а само описание тела процедуры будет дано ниже. Такая ситуация возникает, когда, допустим, процедура **A** вызывает процедуру **B**, в то время как процедура **B** сама вызывает процедуру **A**. В этом случае можно сначала описать параметры процедуры **B** с директивой **Forward**, затем поместить процедуру **A**, а затем программный код процедуры **B**. Это связано с тем, что Delphi не разрешает использовать не определенные заранее процедуры. Директива **Forward** может быть использована только в секции **Implementation** модуля **Unit**.

Директива **External** используется при описании правил вызова процедуры, когда ее тело извлекается из динамической библиотеки (**DLL**) или из объектного модуля (**OBJ**) – результата трансляции с языка Си.

Директива **OverLoad** используется тогда, когда две процедуры имеют одно имя, но разный список параметров. При вызове таких процедур предварительно проверяется список фактических параметров и вызывается та процедура, список формальных параметров которой совпадает по типу со списком фактических параметров.

Имена процедур и функций могут быть использованы в качестве формальных параметров подпрограмм. Для этого определяется тип:

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

или **Type** <имя> = Procedure ([список формальных параметров]);

В функциях используется специальная переменная **Result**, интерпретируемая как значение, которое вернет в основную программу функция по окончании своей работы.

8.3. Использование модулей Unit

Модуль **Unit** является отдельной программной единицей, поскольку описывается в отдельном текстовом файле с расширением ***.pas** и транслируется отдельно. Результатом трансляции является машинный код, который записывается в файл с расширением ***.dcu**. Структура модуля **Unit** может иметь следующий вид:

```
Unit    Имя модуля;
Interface          // Интерфейсная часть модуля
Uses               // Имена подключаемых модулей
  // Объявления глобальных типов, констант, переменных,
  // заголовков процедур и функций, которые будут доступны
  // в других модулях, подключивших данный модуль
……
Implementation  // Секция реализации модуля
Uses               // Имена подключаемых модулей
  // Здесь могут определяться внутренние константы, типы, переменные,
  // процедуры и функции, которые будут доступны только внутри
  // данного модуля. Здесь же дается реализация всех процедур и функций,
  // объявленных в интерфейсной секции модуля
……...
Initialization     // Секция инициализации модуля
  // В этой секции записываются операторы, которые будут выполнены сразу
  // после загрузки программы в память ПЭВМ. Секция инициализации
  // будет выполняться в том порядке, в каком модули Unit описаны
  // в основной программе в разделе Uses
  // Секция инициализации может отсутствовать в модуле Unit
Finalization       // Секция завершения
  // Может присутствовать в модуле Unit, только если в нем есть секция
  // инициализации. Выполнение операторов этой секции происходит после
  // окончания работы программы перед выгрузкой ее из оперативной памяти
  // ЭВМ. Эта секция выполняется в обратном порядке по сравнению с порядком
  // выполнения секции инициализации
End.               // Конец модуля Unit.
```

Все объявления, сделанные в интерфейсной секции, являются **глобальными** для программ, использующих данный модуль. Подключение модулей к другим модулям осуществляется в разделе **Uses** в списке подключаемых моду-

лей. Если в интерфейсных секциях модулей есть определения с одинаковым именем, то воспринимается определение того модуля, который находится в конце списка модулей в разделе **Uses**.

8.4. Пример выполнения задания

Составить программу, отображающую графики функций **Sin(x)** и **Cos(x)** на интервале [**Xmin**, **Xmax**]. Предусмотреть возможность изменения разметки координатных осей, а также шага построения таблицы. Значения функций **Sin(x)** и **Cos(x)** вычислить в библиотечном модуле.

8.4.1. Настройка формы

Окно формы приведено на рис. 8.1.

Для ввода исходных данных используются строки ввода **TEdit**. Компонент **TChart** находится в меню компонентов **Standard** и обозначается пиктограммой  .

8.4.2. Работа с компонентом TChart

Для изменения параметров компонента TChart надо дважды щелкнуть по нему мышью в окне формы. Появится окно редактирования EditingChat1 (рис. 8.2). Для создания нового объекта Series1 следует 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, надо отключить независимый переключатель 3D. После нажатия на кнопку OK появится новая серия с название Series1. Для изменения названия необходимо нажать кнопку Title… В появившемся однострочном редакторе следует набрать имя отображаемой функции: «Sin(x)».

Аналогичным образом создается объект **Series2** для функции **Cos(x)**.

Для изменения надписи над графиком на странице **Titles** в многострочном редакторе надо ввести: «Графики функций».

Для разметки осей выбрать страницу **Axis** и установить параметры настройки осей.

Нажатие на различные кнопки меню способствует ознакомлению с другими возможностями **EditingChat**.

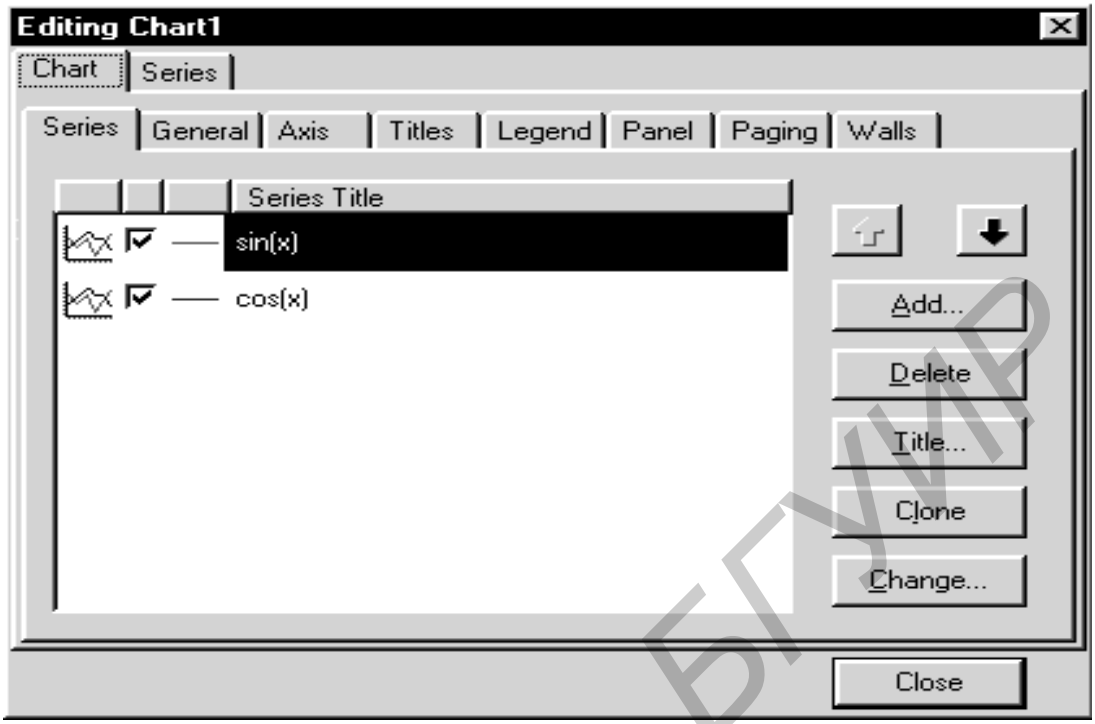


Рис. 8.2. Окно редактирования EditingChat1

8.4.3. Создание модуля

Создаваемый библиотечный модуль не должен иметь своей формы. Среда DELPHI при начальной загрузке автоматически создает шаблон программы, имеющий в своем составе форму, файл проекта и т. д. Так как модуль состоит только из одного файла, то необходимо перед его созданием уничтожить заготовку файла проекта и форму. Для этого следует выбирать: меню **File** – **Close All** – **файл проекта не сохранять**.

Для создания модуля необходимо выполнить: меню **File** – **File New** – пиктограмму Unit . В результате будет создан файл с заголовком **Unit Unit1**. Имя модуля можно изменить на другое, отвечающее внутреннему содержанию модуля, *например* **Unit Matfu**; Затем надо сохранить файл с именем, совпадающим с именем заголовка модуля: **Matfu**.**pas**. Следует обратить внимание на то, что имя файла должно совпадать с именем модуля, иначе DELPHI не сможет подключить его к другой программе.

8.4.4. Подключение модуля

Для подключения библиотечного модуля 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.

8.4.5. Написание программы обработки события создания формы

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

8.4.6. Написание программ обработки событий нажатия на кнопки

Процедура **TForm1**.**Button1Click** обрабатывает нажатие кнопки «**Установить оси**». Процедура **TForm1**.**Button2Click** обрабатывает нажатие кнопки «**Построить график**». Для добавления координат точек **(X,Y)** из таблицы значений в двумерный массив объекта **Series<u>k</u>** используется процедура **Series1**.**AddXY(Const AXValue, AYValue: Double; Const AXLabel: String; AColor: TColor) : Longint**;
где **AXValue**, **AYValue** – координаты точки по осям **X** и **Y**;
AXLabel может принимать значение ‘ ’;
AColor задает цвет линий (если равен **clTeeColor**, то принимается цвет, определенный при проектировании формы).

8.4.7. Код библиотечного модуля

```
Unit Matfu;

Interface
 Function Sinx (x : Extended) : Extended;      // Функция для вычисления синуса
 Function Cosx (x : Extended) : Extended;      // Функция для вычисления косинуса

Implementation
  Function Sinx;
  Begin
     Result := Sin(x);
  End;
  Function Cosx;
  Begin
     Result := Cos(x);
  End;
End.
```

8.4.8. Код основного модуля

```
Unit Unit1;

Interface

Uses
```

```
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, TeeProcs, TeEngine, Chart, Buttons, StdCtrls, Series, MatFu;
Type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Button1: TButton;
    Button2: TButton;
    BitBtn1: TBitBtn;
    Chart1: TChart;
    Series2: TLineSeries;
    Label6: TLabel;
    Edit6: TEdit;
    Label7: TLabel;
    Edit7: TEdit;
    Series1: TLineSeries;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
Var
  Form1: TForm1;
  Xmin, Xmax, Ymin, Ymax, Hx, Hy, h : Extended;
Implementation

{$R *.DFM}
Procedure TForm1.FormCreate(Sender: TObject);
Begin
    Xmin := 0;          //Установка начальных параметров координатных осей
    Xmax := 2*Pi;
    Ymin := -1;
    Ymax := 1;
    Hx := Pi/2;
```

```
  Hy := 0.5;
  h := 0.01;            // Установка шага расчета таблицы
  Edit1.Text := FloatToStr (Xmin);          //Вывод данных
  Edit2.Text := FloatToStr (Xmax);
  Edit3.Text := FloatToStr (Ymin);
  Edit4.Text := FloatToStr (Ymax);
  Edit5.Text := FloatToStr (Hx);
  Edit6.Text := FloatToStr (Hy);
  Edit7.Text := FloatToStr (h);
  Chart1.BottomAxis.Automatic := False;// Отключение автоматического
                                       // определения параметров нижней
  оси
  Chart1.BottomAxis.Minimum := Xmin;  // Установка левой границы нижней
  оси
  Chart1.BottomAxis.Maximum:=Xmax;// Установка правой границы нижней
  оси
  Chart1.LeftAxis.Automatic := False;      // Отключение автоматического
                                           // определения параметров левой оси
  Chart1.LeftAxis.Minimum := Ymin;         // Установка нижней границы левой
  оси
  Chart1.LeftAxis.Maximum := Ymax;         // Установка верхней границы левой
  оси
  Chart1.BottomAxis.Increment:=Hx;// Установка шага разметки по нижней
  оси
  Chart1.LeftAxis.Increment := Hy;    // Установка шага разметки по левой
  оси
  End;
Procedure TForm1.Button1Click(Sender: TObject);
Begin
  Xmin := StrToFloat (Edit1.Text);          //Ввод даннных
  Xmax := StrToFloat (Edit2.Text);
  Ymin := StrToFloat (Edit3.Text);
  Ymax := StrToFloat (Edit4.Text);
  Hx := StrToFloat (Edit5.Text);
  Hy := StrToFloat (Edit6.Text);
  Chart1.BottomAxis.Minimum := Xmin;  // Установка левой границы нижней
  оси
  Chart1.BottomAxis.Maximum:=Xmax;// Установка правой границы нижней
  оси
  Chart1.LeftAxis.Minimum := Ymin;         // Установка нижней границы левой
  оси
  Chart1.LeftAxis.Maximum := Ymax;         // Установка верхней границы левой
  оси
  Chart1.BottomAxis.Increment:=Hx;// Установка шага разметки по нижней
  оси
```

```
  Chart1.LeftAxis.Increment := Hy;      // Установка шага разметки по левой
  оси
  End;
Procedure TForm1.Button2Click(Sender: TObject);
Var
  x, y1, y2 : Extended;
Begin
  Series1.Clear;                        //Очистка графиков
  Series2.Clear;
  Xmin := StrToFloat (Edit1.Text);
  Xmax := StrToFloat (Edit2.Text);
  h := StrToFloat (Edit7.Text);         // Шаг расчета таблицы для графика
  x := Xmin;                            // Начальное значение по оси X
  Repeat
    y1 := Sinx(x);                      // Расчет функции
    Series1.AddXY (x,y1,' ',clTeeColor); // Вывод точки на график
    y2 := Cosx(x);                      // Расчет функции
    Series2.AddXY (x,y2,' ',clTeeColor); // Вывод точки на график
    x := x+h;                           // Увеличение значения x на величину
    шага
    Until x>Xmax;
  End;
End.
```

8.5. Индивидуальные задания

В соответствии с вариантом индивидуального задания составить программу построения графиков функций **Y(x)** и **S(x)**. Таблицу данных получить изменяя параметр **x** от $\mathbf{x_n}$ до $\mathbf{x_k}$ с шагом $\mathbf{h = (x_k - x_n)/m}$, где **m** –количество интервалов, на которые разбивается отрезок $\mathbf{[x_n , x_k]}$. Ввод исходных данных организовать через компоненты **TEdit**. Вычисление значений функций оформить в отдельном модуле Unit.

Выражения для функций **Y(x)** и **S(x)** следует взять из лабораторной работы №3 данного методического пособия.

ПРИЛОЖЕНИЕ 1

БЛОК-СХЕМА АЛГОРИТМА

Благодаря своей наглядности данный способ записи алгоритмов получил наибольшее распространение. При построении блок-схемы алгоритм изображается геометрическими фигурами (блоками), связанными линиями (направление потока информации) со стрелками. Внутри блоков записывается последовательность действий.

Ниже приведены виды и назначение основных блоков, применяемых при составлении блок-схемы алгоритма.

Таблица П 1.1

Виды и назначение основных блоков

Наименование блока	0бозначение блока	Функции блока
1	2	3
Процесс		Выполнение действий, изменяющих значение, форму представления или расположение данных
Ввод-вывод		Ввод или вывод данных
Условие		Выбор направления выполнения алгоритма в зависимости от заданного в блоке условия
Предопределенный процесс		Вызов подпрограммы
Пуск-остановка		Начало или конец описания алгоритма
Цикл с параметром		Цикл с параметром; внутри блока указывается изменение параметра (переменной цикла), *например* i=1,10
Внутристраничный соединитель		Переход между блоками в пределах данной страницы

Окончание табл. П 1.1

1	2	3
Межстраничный соединитель		Переход между блоками, расположенными на разных листах
Комментарий		Пояснение действий, указанных в блоке

Правила оформления блок-схем:

- в пределах одной схемы блоки изображают одинаковых размеров;
- все блоки нумеруются (номер ставится в верхнем левом углу блока с разрывом линии);
- линии, соединяющие блоки и указывающие последовательность связей между ними, проводятся параллельно линиям рамки;
- стрелка в конце линии не ставится, если линия направлена слева направо или сверху вниз;
- из блока «условие» могут выходить две линии, из других блоков – только одна линия;
- если схема занимает более одного листа, то в случае разрыва линии используется межстраничный соединитель;
- внутри каждого соединителя указывается номер блока (откуда или куда направлена соединительная линия). Внутри межстраничного соединителя в первой строке указывается номер листа, во второй – номер блока куда или откуда передается управление.

В зависимости от поставленной задачи выделяют три основных вида алгоритмов:

- линейный;
- разветвляющийся;
- циклический.

Линейным называется алгоритм, в котором все действия, указанные в блоках, выполняются по порядку их следования.

Пример блок-схемы алгоритма вычисления площадей прямоугольника и квадрата (рис. П 1.1).

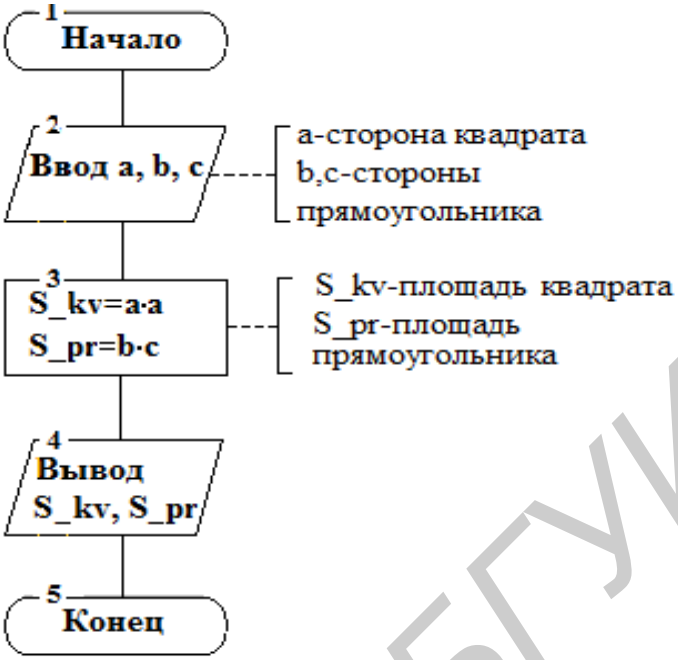


Рис. П 1.1. Блок-схема линейного алгоритма

Разветвляющимся называют алгоритм, в котором в зависимости от значения условия (выполняется или не выполняется) изменяется последовательность выполнения действий алгоритма.

Пример блок-схемы алгоритма определения принадлежности точки с координатами (x, y) номеру сектора (рис. П 1.2).

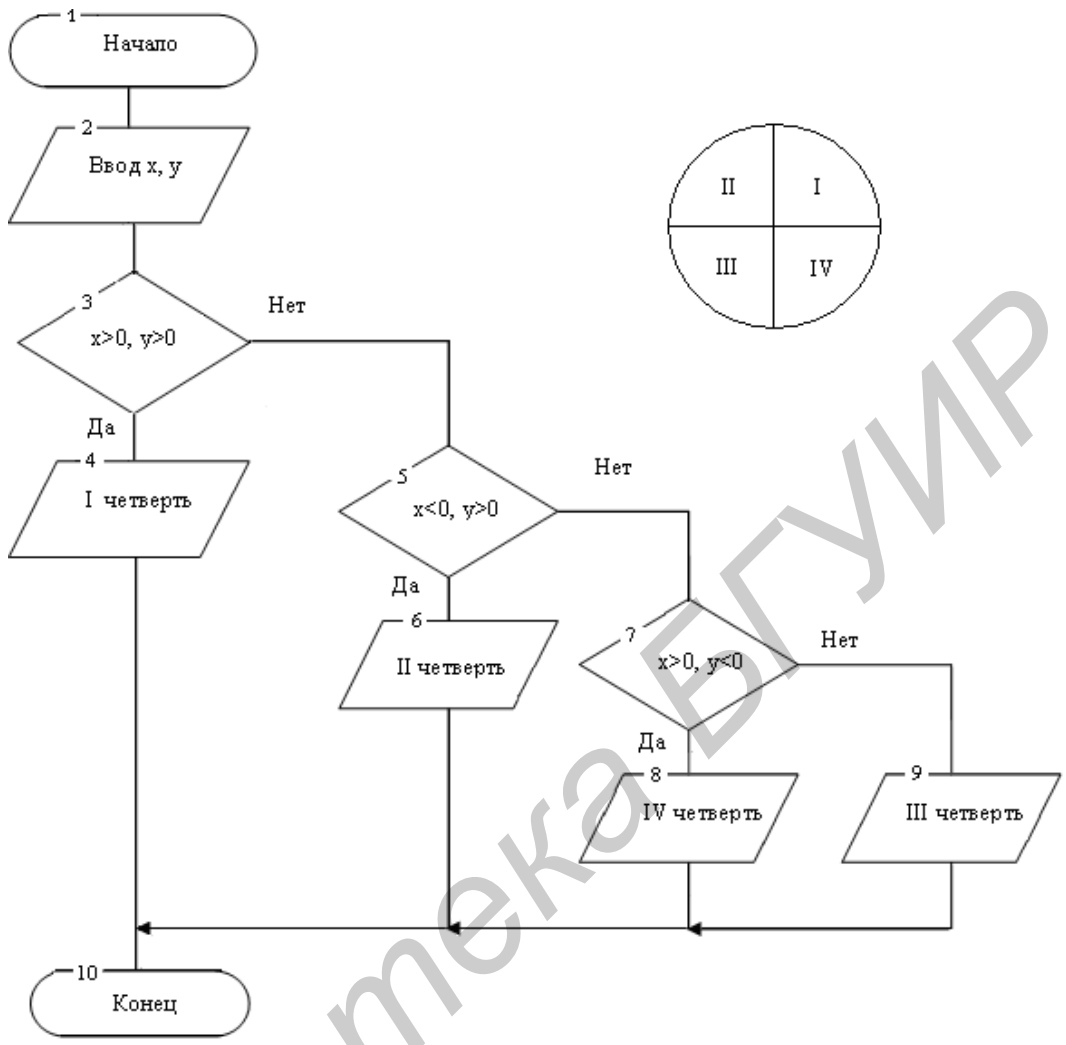


Рис. П 1.2. Блок-схема разветвляющегося алгоритма

Циклическим называется алгоритм, в котором некоторая последовательность действий повторяется определенное количество раз.

Тело цикла – действия, выполняемые в цикле.

Циклические алгоритмы могут быть:

- с предусловием – условие, при котором выполняется тело цикла, задается в начале цикла (рис. П 1.3);
- с постусловием – условие, при котором выполняется тело цикла, задается в конце цикла (рис. П 1.4);
- с параметром – в начале цикла задается правило изменения его параметра (рис. П 1.5).

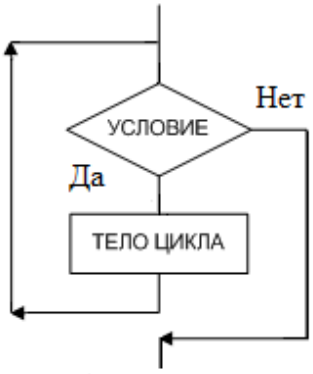


Рис. П1.3. Цикл постусловием

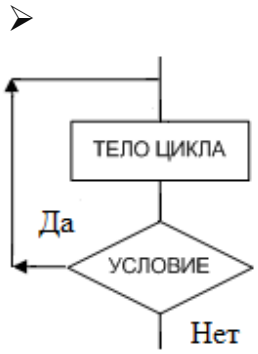


Рис. П1.4. Цикл с предусловием

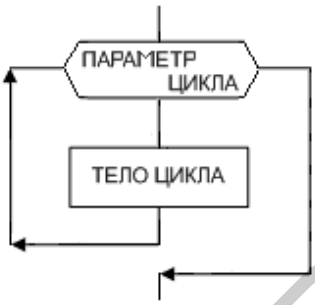


Рис. П1.5. Цикл с с параметром

Пример блок-схемы циклического алгоритма нахождения НОД (наибольшего общего делителя) двух значений a и b.

Правило нахождения НОД (a, b) формулируется следующим образом:

1) определяется наибольшее из значений a и b – max (a, b);
2) из наибольшего значения вычитается оставшееся значение;
3) п. 1 и 2 повторяются до тех пор, пока значения a и b не станут равными.

Полученное значение будет НОД (a, b).

Решение:

Предположим, что a = 20 и b = 15.

Тогда:

max (a, b) = max (20, 15) = 20 → a=20–15=5 b=15
max (a, b) = max (5, 15) = 15 → a=5 b=15–5=10
max (a, b) = max (5, 10) = 10 → a=5 b=10–5=5

a = b => НОД (a, b) = 5

Составим блок-схему алгоритма нахождения НОД (a, b) (рис. П 1.6):

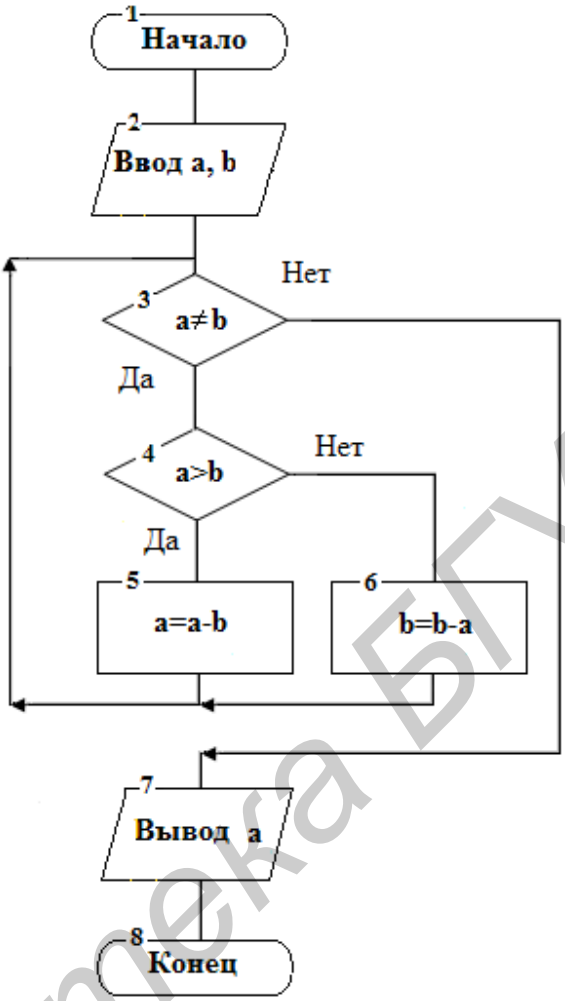


Рис. П 1.6. Блок-схема алгоритма нахождения НОД (a, b)

Для реализации алгоритма на компьютере необходимо описать его на языке программирования.

ПРИЛОЖЕНИЕ 2

МАТЕМАТИЧЕСКИЕ ФОРМУЛЫ

Таблица П 2.1

Стандартные математические функции

Функция	Назначение	Тип аргумента	Тип функции
Abs(x)	Вычисление абсолютного значения (модуля) x, |x|	Вещественный Целый	Такой же, как и тип аргумента
Sqr(x)	Вычисление квадрата x, x^2	Вещественный Целый	Такой же, как и тип аргумента
Sqrt(x)	Вычисление квадратного корня из x, $\sqrt{x}$, x>0	Вещественный Целый	Вещественный
Sin(x)	Вычисление синуса x, sin x	Вещественный Целый	Вещественный
Cos(x)	Вычисление косинуса x, cos x	Вещественный Целый	Вещественный
Arctan(x)	Вычисление арктангенса x, arctg x	Вещественный Целый	Вещественный
Exp(x)	Вычисление экспоненты (e≈2,71828) e^x ,e -> exp(1)	Вещественный Целый	Вещественный
Ln(x)	Вычисление натурального логарифма x, $\ln x$, x>0	Вещественный Целый	Вещественный
Log(x)	Вычисление десятичного логарифма, log x	Вещественный Целый	Вещественный
Pi	Число π≈3,1415…	Нет	Вещественный
Odd(x)	Проверка числа на нечетность (True)	Целый	Логический
Inc(x)	Увеличение x на 1	Целый	Целый
Inc(x, n)	Увеличение x на n	Целый	Целый
Dec(x)	Уменьшение x на 1	Целый	Целый
Dec(x, n)	Уменьшение x на n	Целый	Целый

Функции модуля Math

Для использования в программе расширенного списка основных математических функций следует в операторе **Uses** дописать в конец списка математический модуль **Math**:

```
Uses ……, Math;
```

Таблица П 2.2

Наиболее часто используемые функции модуля **Math**

Функция	**Описание**
Function **Power**(a,x:Extended): Extended;	$\mathbf{a^x}$ – возведение a в степень x
Function **ArcCos**(X: Extended): Extended;	**ArcCos**(x)
Function **ArcSin**(X: Extended): Extended;	**ArcSin**(x)
Function **Tan**(X: Extended): Extended;	**Tg (X)**, X <> 0
Function **ArcTan2**(Y, X: Extended): Extended;	Результат в диапазоне –π..+π
Function **Cotan**(X: Extended): Extended;	$\frac{1}{tgx}$
Function **Secant**(X: Extended): Extended;	$\frac{1}{\cos x}$
Function **Cosecant**(X: Extended): Extended;	$\frac{1}{\sin x}$
Function **Hypot**(X, Y: Extended): Extended;	$\sqrt{x^2 + y^2}$
Function **Cosh**(X:Extended): Extended;	$\frac{e^x + e^{-x}}{2}$
Function **Sinh**(X: Extended): Extended;	$\frac{e^x - e^{-x}}{2}$
Function **LogN**(N, X: Extended): Extended;	$Log_n(x)$
Function **Sign**(x:Extended):Extended;	1, если $x > 0$; 0, если $x = 0$; -1, если $x < 0$

Таблица П 2.3

Целочисленные операции

Операция	Назначение	Пример записи	Тип операндов	Тип результата
Div	вычисление частного при	c:=a Div b;	Целый	Целый
Mod	вычисление остатка от деления a на b	d:=a Mod b;	Целый	Целый
Shr	Побитный сдвиг вправо	N:= I shr 2	Целый	Целый
Shl	Побитный сдвиг влево	N=I shl 3	Целый	Целый

Примеры:

```
N:=18 div 7;    // результат N = 2
N:=18 mod 7;  // результат N = 4
N:=1 shl 3;      // результат N = 8
N:=8 shr 2;      // резальтат N = 2
```

Таблица П 2.4

Функции преобразования

Функция	Назначение	Тип аргумента	Тип функции
Trunc(x)	Нахождение целой части x (дробная часть числа отбрасывается)	Вещественный целый	LongInt
Round(x)	Округление x в сторону ближайшего целого по математическим правилам	Вещественный целый	LongInt
Int(x)	Вычисление целой части x	Вещественный	Вещественный
Frac(x)	Вычисление дробной части числа x	Вещественный	Вещественный

Примеры:

```
y := Trunc(13.999);      //переменной y присваивается 13
y := Trunc(13.111);      //переменной y присваивается 13
y := Round(3.145);       //переменной y присваивается 3
y := Round(23.5);        //переменной y присваивается 24
y := Round(-12.5);       //переменной y присваивается -13
y := Int(2.7);           //переменной y присваивается 2
y := Int(-32.3);         //переменной y присваивается -32
y := Frac (-32.3);       // переменной y присваивается -0.3
```

Таблица П 2.5

Получение случайных чисел

Функция	Назначение	Пример записи	Тип аргумента	Тип результата
Random	Получение (генерация) случайного числа в диапазоне 0<=…<1	y := Random;	Вещест-венный	Веществен-ный
Random(x)	Получение случайно-го числа в диапазоне 0<=…< x	y :=Random(39);	Целый	Целый

Таблица П 2.6

Функции, используемые для работы с порядковыми переменными

Функция	Назначение	Тип аргумента	Тип функции
1	2	3	4
Pred(x)	Определение предшественника взятого символа x	Порядковый	Порядковый
Succ(x)	Определение последующего символа за взятым символом	Порядковый	Порядковый

Окончание табл. П 2.6

1	2	3	4
Ord(x)	Определение кода символа, например, Ord(‘A’) ➲ 65	Порядковый	Целый
Chr(x)	Определение символа по коду, например, Chr(65) ➲ ’A’	Целый	Char

Таблица П 2.7

Процедуры и функции для работы со строкам

Функция	Описание
1	2
Function **Concat** (S1 [, S2, ..., SN]: String) : String;	Возвращает строку, представляющую собой сцепление строк-параметров **S1**. **S2**, … , **SN**
Function **Copy** (St: String; Index, Count: Integer) : String;	Копирует из строки **St Count** символов, начиная с символа с номером **Index**
Procedure **Delete** (St: String; Index, Count: Integer);	Удаляет **Count** символов из строки **St**, начиная с символа с номером **Index**
Procedure **Insert** (SubSt: String; St, Index: Integer);	Вставляет подстроку **SubSt** в строку **St**, начиная с символа с номером **Index**
Function **Length** (St: String): Integer;	Возвращает текущую длину строки **St**
Function **Pos** (SubSt, St: String): Integer;	Отыскивает в строке **St** первое вхождение подстроки **SubSt** и возвращает номер позиции, с которой она начинается. Если подстрока не найдена, то возвращается **ноль**
Procedure **SetLength** (St: String; NewLength: Integer);	Устанавливает новую (меньшую) длину **NewLength** строки **St**, если **NewLength** больше текущей длины строки; обращение к **SetLength** игнорируется

Продолжение табл. П 2.7

1	2
Подпрограммы преобразования строк в другие типы	
Function **StrToCurr** (St: String) : Currency;	Преобразует символы строки **St** в целое число типа **Currency**. Строка не должна содержать ведущих или ведомых пробелов
Function **StrToDate** (St: String) : TDateTime;	Преобразует символы строки **St** в дату. Строка должна содержать два или три числа, разделенных правильным для Windows разделителем даты (в русифицированной версии таким разделителем является «**.**») Первое число – день, второе – месяц, при задании третьего числа задается год
Function **StrToDateTime** (St: String) : TDateTime;	Преобразует символы строки **St** в дату и время. Строка должна содержать дату и время, разделенные пробелом
Function **StrToFloat** (St: String) : Extended;	Преобразует символы строки **St** в вещественное число. Строка не должна содержать ведущих или ведомых пробелов
Function **StrToInt** (St: String) : Integer;	Преобразует символы строки **St** в целое число. Строка не должна содержать ведущих или ведомых пробелов
Function **StrToIntDef** (St: String; Default: Integer) : Integer;	Преобразует символы строки **St** в целое число. Если строка не содержит правильного представления целого числа, то возвращается значение **Default**
Function **StrToIntRange** (St: String; Min, Max: Longint) : Longint;	Преобразует символы строки **St** в целое число и возбуждает исключение **ERangeError**, если число выходит из заданного диапазона **MmMax**

Продолжение табл. П 2.7

1	2
Function **StrToTime** (St: String) : TDateTime;	Преобразует символы строки **St** во время
Procedure **Val** (St: String; var X; Code: Integer);	Преобразует строку символов **St** во внутреннее представление целой или вещественной переменной **X**, которое определяется типом этой переменной. Параметр **Code** содержит **ноль**, если преобразование прошло успешно, и тогда в **X** помещается результат преобразования; в противном случае он содержит номер позиции в строке **St**, где обнаружен ошибочный символ, и в этом случае содержимое **X** не меняется. В строке **St** могут быть ведущие и (или) ведомые пробелы
Подпрограммы обратного преобразования	
Function **DateToStr** (Value: TDateTime) : String;	Преобразует дату из параметра **Value** в строку символов
Function **DateTimeToStr** (Value: TDateTime): String;	Преобразует дату и время из параметра **Value** в строку символов
Procedure **DateTimeToString** (Var St: String; Format: String; Value: TDataTime) ;	Преобразует дату и время из параметра **Value** в строку **St**
Function **FormatDateTime** (Format: String; Value: TDate-Time): String;	Преобразует дату и время из параметра **Value** в строку символов
Function **FloatToStr** (Value: Extended): String;	Преобразует вещественное значение **Value** в строку символов
Function **FloatToStrF** (Value: Extended; Format: TFloatFormat; Precision, Digits: Integer) : String;	Преобразует вещественное значение **Value** в строку символов с учетом параметров **Precision** и **Digits** (см. пояснения ниже)
Function **FormatFloat** (Format: String; Value: Extended): String;	Преобразует вещественное значение **Value** в строку
Function **IntToStr** (Value: Integer) : String;	Преобразует целое значение **Value** в строку символов

Окончание табл. П 2.7

1	2
Function **TimeToStr** (Value: TDateTime) : String;	Преобразует время из параметра **Value** в строку символов
Procedure **Str** (X [:Width [:Decimals]]; Var St: String);	Преобразует число **X** любого вещественного или целого типа в строку символов **St**; параметры **Width** и **Decimals**, если они присутствуют, задают формат преобразования: **Width** определяет общую ширину поля, выделенного под соответствующее символьное представление вещественного или целого числа **X**, а **Decimals** –количество символов в дробной части (этот параметр имеет смысл только в том случае, когда **X** – вещественное число)

Таблица П 2.8

Правила использования параметров функции **FloatToStrF**

Значение Format	Описание
ffExponent	Научная форма представления с множителем eXX («умножить на 10 в степени XX»). Precision задает общее количество десятичных цифр мантиссы. Digits – количество цифр в десятичном порядке XX. Число округляется с учетом первой отбрасываемой цифры, например 3.1416E+00
ffFixed	Формат с фиксированным положением разделителя целой и дробной частей. **Precision** задает общее количество десятичных цифр в представлении числа. **Digits** – количество цифр в дробной части. Число округляется с учетом первой отбрасываемой цифры. Например, при значении **Digits**, равном 2,будет выведено 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, например 3,1416
ffNumber	Отличается от **ffFixed** использованием символа – разделителя тысяч при выводе больших чисел (для русифицированной версии Windows таким разделителем является **пробел**). Для Value = π* 1000 получим 3 141,60
ffCurrency	Денежный формат. Соответствует ffNumber, но в конце строки ставится символ денежной единицы (для русифицированной версии Windows – символы «р.»). Для Value = π · 1000 получим 3 141,60р

ПРИЛОЖЕНИЕ 3

НАСТРОЙКА ПАРАМЕТРОВ СРЕДЫ DELPHI

Для упрощения работы с любой программой в среде **Delphi** следует задать новую переменную среды окружения, *например*, **Lab1**. Для этого надо запустить Delphi и пройти по основному меню путь: **Tools** – **Environmet Options** – **Environment Variables** и в окне **User Overrides** задать новый параметр с именем (**Variable Name**), *например* **Lab1**, и значением (**Variable Value**), описывающим путь к основному каталогу программы, *например* **d:\Work\Mylab\Lab1**.

Обычно при профессоинальном программировании в этом каталоге создают подкаталоги для хранения:

– исходного текста программы (например, Sourse), где будут храниться файлы с расширениями *.dpr, *.pas, *.dfm, *.res, *.cfg, *.dof;

– результатов трансляции модулей Unit (например, Lib), а это файлы с расширением *.dcu,

– готовой к выполнению программы и динамических библиотек (например Bin), а это файлы с расширением *.exe и *.dll.

В этом же каталоге можно создать подкаталог для описания самой программы и правил работы с ней и подкаталог исходных данных для различных вариантов расчетов по разрабатываемой программе.

Затем следует сохранить файл проекта, выполнив **File** – **Save Project as**, и определить директорию для файла проекта, например **D:\Work\Mylab\Lab1\Source\Project.dpr**.

Далее в основном меню Delphi выбрать **Project** – **Options** – **Directories**/ **Conditionals** и в окне **Directores** следует задать выходную директорию (**Output Directories**), например в таком виде: **$(Lab1)\BIN**. В этом же окне надо определить и директорию для результатов трансляции модулей (**Unit Output Directory**), например в таком виде: **$(Lab1)\LIB**.

После этого следует запустить трансляцию и выполнение текущей программы пунктом меню **Run**.

При сохрании проекта **File** – **Save** сохраняться и текущие настройки проекта.

Следующий запуск программы не потребует повторения указанных выше настроек.

Для режима отладки программы надо отключить режим оптимизации при работе трансляции, так как он иногда не позволяет проводить отладку программы. Для этого следует задать путь: **Project** – **Options** – **Compiler**. Вид формы при этом должен соответствовать рис. П 3.1.

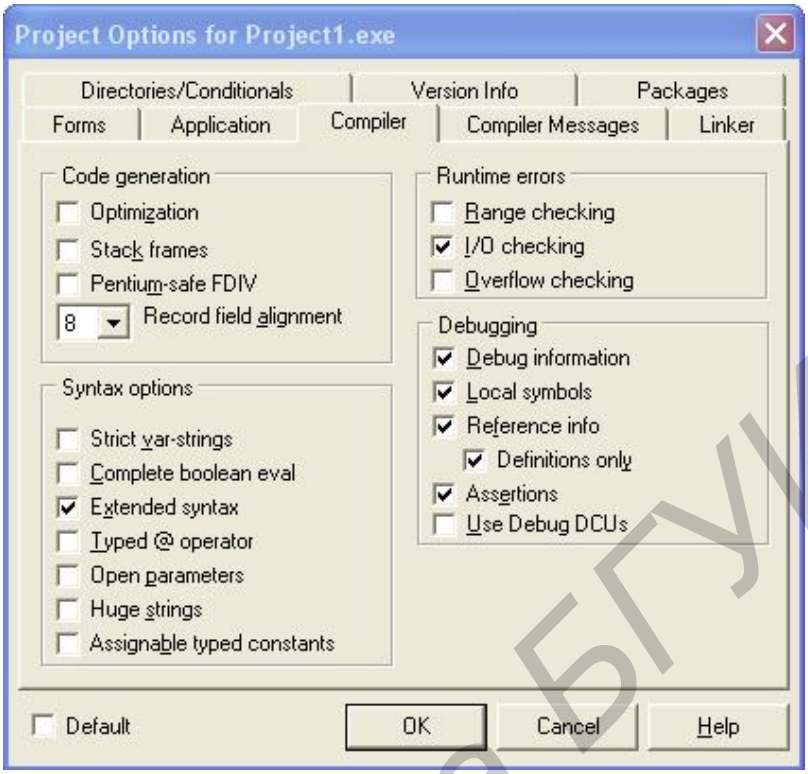


Рис. П 3.1. Вид формы окна вкладки Compiler

ПРИЛОЖЕНИЕ 4

СВОЙСТВА КОМПОНЕНТОВ

Многие стандартные визуальные компоненты имеют одинаковые свойства. Поэтому имеет смысл рассмотреть их отдельно.

Таблица П 4.1

Базовые свойства VCL-компонент

Свойство	Назначение
1	2
Alignment	Определяет горизонтальное выравнивание текста относительно границ компонента: **taCenter** (по центру), **taLeftJustify** (по левому краю), **taRightJustify**(по правому краю)
Caption или Text (TEdit)	Заголовок компонента
Color	Задает цвет фона компонента. Может быть выбран один из стандартных, перечисленных в списке или вводимых с клавиатуры. Например Color:=$00FF0000; определяет ярко голубой цвет. Младший байт определяет уровень красного цвета, второй байт – уровень зеленого цвета, а третий байт – уровень синего цвета (RGB)
Ctl3D	Задает вид компонента. Если значение этого свойства равно False, компонент имеет двумерный вид, если True – трехмерный (значение по умолчанию)
Cursor	Определяет вид курсора мыши в активной области компонента
DragCursor	Определяет вид курсора мыши при «перетаскивании» другого компонента в данный компонент
DragMode	Определяет режим поддержки протокола drag-and-drop. Возможны следующие значения: **DmAutomatic** – компонент можно «перетаскивать» мышью, **dmManual** – компонент не может быть «перетаскан» без вызова метода **BeginDrag**
Enabled	При значении **True** компонент реагирует на сообщения от мыши, клавиатуры и таймера; иначе (значение **False**) эти сообщения игнорируются

Продолжение табл. П 4.1

1	2
Font	Определяет шрифт текстовых элементов компонента. При создании компонента устанавливаются следующие параметры: Name=System, Size=10 и Color=clWindowText, Pitch= FpDefault. При нажатии на значок ⊞ раскрываются свойства шрифта, а на кнопку ··· – вызывается окно установки свойств шрифта
Height	Задает вертикальный размер компонента (в пикселах), вместе со свойствами **Width**, **Left** и **Top** задает его размер и положение
HelpContext	Задает номер контекста справочной системы (должен быть уникальным для каждого компонента). Если компонент активен (находится в фокусе), то нажатие **F1** выводит окно справочной системы, если оно существует для данного компонента
Hint	Задает текст, который будет отображаться при обработке события **OnHint**, если курсор находится в области компонента
Left	Задает горизонтальную координату левого угла компонента относительно формы в пикселах. Для формы это значение указывается относительно экрана
Name	Указывает внутреннее имя компонента, используемое в программном коде для обращения к объекту. Является идентификатором
ParentColor	Определяет цвет компонента: при значении **True** (по умолчанию) используется цвет родительского компонента, иначе (**False**) компонент использует значение собственного свойства **Color**. При смене свойства **Color** значение **ParentColor** автоматически меняется на **False**
ParentCtl3D	Указывает, каким образом компонент будет определять, является он трехмерным или нет. Если значение этого свойства равно **True**, то вид компонента задается значением свойства **Ctl3D** его владельца, если же значение этого свойства равно **False**, то – значением его собственного свойства **Ctl3D**

Окончание табл. П 4.1

1	2
ParentFont	Аналогично свойствам **ParentColor** и **ParentCtl3D**, но для шрифта. Если значение этого свойства равно **True**, используется шрифт, заданный у владельца компонента, иначе (**False)** шрифт задается значением его собственного свойства **Font**.
PopupMenu	Задает название локального меню, отображаемое при нажатии правой кнопки мыши. Локальное меню отображается, когда свойство **AutoPopup = True** или при вызове метода **Popup**.
ReadOnly	Запрещает редактирование текста, отображаемого в **TEdit** (значение **True**).
TabOrder	Задает очередность получения компонентами фокуса при нажатии клавиши **Tab**. По умолчанию определяется порядком размещения компонентов на форме: у первого компонента TabOrder=0, у второго – 1 и т.д. Компонент с TabOrder=0 получает фокус при выводе формы.
TabStop	Указывает возможность получения фокуса для компонента. Компонент получает фокус, если **TabStop** равно **True**.
Tag	«Привязывает» к любому компоненту значение типа **Longlnt**.
Top	Задает вертикальную координату левого верхнего угла интерфейсного элемента относительно формы в пикселах. Для формы это значение указывается относительно экрана.
Visible	Определяет видимость компонента на экране. Значением этого свойства управляют методы **Show** и **Hide**.
Width	Задает горизонтальный размер интерфейсного элемента или формы в пикселах.

Таблица П 4.2

Выравнивание компонента внутри родителя (свойство Align)

Значение	Расположение компонента
1	2
alNone	Выравнивание не используется. Компонент располагается на том месте, куда был помещен во время создания программы. Принимается по умолчанию.

Окончание табл. П 4.2

1	2
alTop	Компонент перемещается в верхнюю часть родительского окна, а его ширина становится равной ширине родительского окна. Высота компонента не изменяется.
alBottom	Компонент перемещается в нижнюю часть родительского окна, а его ширина становится равной ширине родительского окна. Высота компонента не изменяется.
alLeft	Компонент перемещается в левую часть родительского окна, а его высота становится равной высоте родительского окна. Ширина компонента не изменяется
alRight	Компонент перемещается в правую часть родительского окна, а его высота становится равной высоте родительского окна. Ширина компонента не изменяется
alClient	Компонент полностью занимает всю рабочую область родительского окна

Таблица П 4.3

Задание цвета фона компонента (свойство Color)

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

Таблица П 4.4

Системные цвета Windows, определяемые цветовой схемой

Значение	Цвет для элемента
clBackground	фон окна
clActiveCaption	заголовок активного окна
clInactiveCaption	заголовок неактивного окна
clMenu	фона меню
clWindow	фон Windows
clWindowFrame	рамка окна
clMenuText	текст элемента меню
clWindowText	текст внутри окна
cICaptionText	заголовок активного окна
clActiveBorder	рамка активного окна
clInactiveBorder	рамка неактивного окна
clAppWorkSpace	рабочая область окна
clHighlight	фон выделенного текста
clHightlightText	выделенный текст
clBtnFace	кнопка
clBtnShadow	фон кнопки
cIGrayText	недоступный элемент меню
clBtnText	текст кнопки

Таблица П 4.5

Свойства компонента TCheckBox
(позволяет выбрать или отменить определенную функцию)

Свойство	Назначение
1	2
Alignment	Определяет положение размещения надписи текста кнопки: **taLeftJustify** (с левой стороны компонента), **taRightJustify** (с правой стороны)
AllowGrayed	При значении **False** (по умолчанию) возможно два состояния флажка: выделен (не выделен), значение **True** – три состояния флажка: выделен, не выделен и промежуточное (серое окно индикатора и серая галочка –)
Caption	Надпись возле компонента **TCheckBox**

Окончание табл. П 4.5

1	2
Checked	Содержит выбор пользователя типа **Да**/**Нет**. При значении **True** компонент выделен (установлена черная галочка – ☑), **False** – не выделен (пустое окно индикатора – ☐)
State	Устанавливает значение кнопки, которая может находиться во включенном, выключенном и неактивном состоянии: **cbChecked** (выделен), **cbUnchecked** (не выделен) и **cbGrayed** (промежуточное значение) при значении **True** у свойства **AllowGrayed**

Таблица П 4.6

Свойства компонента **TListBox** (содержит список элементов, выбираемых мышью или клавиатурой)

Свойство	Назначение
Columns	Определяет количество столбцов в списке
MultiSelect	Разрешает (значение **True**) или отменяет (значение **False**) выбор нескольких строк из списка
SelCount	Определяет количество выделенных строк в списке. Свойство доступно только для чтения и не может быть изменено в программе, не отражается в инспекторе объектов
Selected[N]	Содержит признак выбора для элемента с номером **n** (нумерация строк в списке начинается с нуля): значение **True** – элемент выделен в списке. Не отражается в инспекторе объектов
ItemIndex	Содержит индекс выбранного элемента в списке; по умолчанию равен **-1**
Items	Содержит набор строк, показываемых в компоненте, определяет количество элементов списка и их содержимое
Sorted	Разрешает (**True**) или отменяет (**False**) сортировку строк списка в алфавитном порядке

Таблица П 4.7

Компоненты для реализации стандартных диалогов (страница **Dialogs**, компоненты невидимы во время выполнения, поэтому место их размещения на форме не имеет значения)

Пиктограмма	Компонент	Назначение
	OpenDialog	Создание окна диалога «*Открыть файл*»
	SaveDialog	Создание окна диалога «*Сохранить файл*»
	OpenPictureDialog	Создание окна диалога «*Открыть рисунок*»
	SavePictureDialog	Создание окна диалога «*Сохранить рисунок*»
	FontDialog	Создание окна диалога «*Шрифт*» – выбор атрибутов шрифта.
	ColorDialog	Создание окна диалога «*Цвет*» – выбор цвета
	PrintDialog	Создание окна диалога «*Печать*»
	PrintSetupDialog	Создание окна диалога «*Установка принтера*»
	FindDialog	Создание окна диалога «*Найти*» – контекстный поиск в тексте
	ReplaseDialog	Создание окна диалога «*Заменить*» – контекстная замена фрагментов текста
	PageSetupDialog	Создание окна диалога «*Параметры страницы*»

Таблица П 4.8

Свойства компонентов OpenDialog и SaveDialog

Свойства	Тип	Описание
1	2	3
DefaultExt	String	Задает расширение, автоматически подставляемое к имени файла, если не указано расширение
FileName	String	Указывает имя и полный путь файла, выбранного в диалоге
Filter	String	Задает маску имени файлов
FilterIndex	Integer	Указывает маску фильтра, отображаемую в списке
InitialDir	Integer	Определяет каталог, содержимое которого будет отображаться при вызове окна

Окончание табл. 4.8

1	2	3
Options	TOpenOptions	Применяется для настройки параметров, управляющих внешним видом и функциональными возможностями диалога
ofOverwritePrompt		Предупреждает, что файл уже существует и требует подтверждения
ofNoChangeDir		Вызывает текущий каталог при открытии
ofAllowMultiSelect		Разрешается одновременный выбор из списка нескольких файлов
ofPathMustExist		Задаются файлы только из существующих каталогов
ofFileMustExist		Задаются только существующие файлы
ofCreatePrompt		Формируется запрос на создание файла при вводе несуществующего имени файла
Title	String	Задает заголовок окна

Литература

1. Колосов, С. В. Программирование в среде Delphi : учеб. пособие по курсу «Программирование» / С. В. Колосов. – Минск : БГУИР, 2005. – 164 с.
2. Фаронов, В. В. Delphi. Программирование на языке высокого уровня / В. В. Фаронов. – СПб. : Питер, 2009. – 639 с.
3. Архангельский, А. Я. DELPHI 2006 / А. Я. Архангельский. – М. : Бином-Пресс, 2010. – 1152 с.
4. Культин, Н. Б. Delphi в задачах и примерах / Н. Б. Культин. – 2-е изд. – СПб. : BHV – СПб, 2008. – 288 с.
5. Окулов, С. М. Программирование в алгоритмах / С. М. Окулов. – М. : Бином. Лаборатория знаний, 2006. – 383 с.
6. Бакнелл, Дж. Фундаментальные алгоритмы и структуры данных в Delphi. Библиотека программиста / Дж. Бакнелл – М. : ООО «ДиаСофтЮт»; СПб. : Питер, 2006. – 557 с.
7. Долинский, М. С. Решение сложных и олимпиадных задач по программированию М. С. Долинский. – СПб. : Питер, 2006. – 366 с.
8. Окулов, С. М. Основы программирования / С. М. Окулов. – Бином. Лаборатория знаний, 2008.
9. Семакин, И. Г. Основы алгоритмизации и программирования / И. Г. Семакин, А. П. Шестаков. Академия, 2008.
10. Чеснокова, О. В. Delphi 2007. Алгоритмы и программы / О. В. Чеснокова. – Москва, НТ – Пресс, 2008. – 368 с.

Св. план 2011, поз.40

Учебное издание

Колосов Станислав Васильевич
Коренская Ирина Николаевна
Лущицкая Ирина Владимировна

ПРОГРАММИРОВАНИЕ В СРЕДЕ DELPHI. ПРАКТИКУМ ПО КУРСУ «ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ»

Учебно-методическое пособие

Редактор *Т. П. Андрейченко*
Корректор *И. П. Острикова*
Компьютерная верстка *Ю. Ч. Клочкевич*

Подписано в печать 14.05.2012. Формат 60x84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 7,56. Уч.-изд. л. 7,4. Тираж 150 экз. Заказ 196.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0494371 от 16.03.2009. ЛП №02330/0494175 от 03.04.2009.
220013, Минск, П. Бровки, 6