Министерство образования Республики Беларусь

Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра «Вычислительные методы и программирование»

Шестакович В. П.

Электронный учебно-методический комплекс

по дисциплине

“ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ”

Для студентов специальностей

36 04 01 «Электронно-оптические системы и технологии»,
39 02 02 «Проектирование и производство радиоэлектронных средств»,
39 02 03 «Медицинская электроника»,
39 02 01 «Моделирование и компьютерное проектирование радиоэлектронных средств»,
38 02 03 «Техническое обеспечение безопасности».

Лабораторный практимум

Минск 2010

СОДЕРЖАНИЕ

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

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

1.1. Интегрированная среда разработчика DELPHI

Среда DELPHI визуально реализуется в виде нескольких одновременно раскрытых на экране монитора окон. Количество, расположение, размер и вид окон может меняться программистом в зависимости от его текущих нужд, что значительно повышает производительность работы. При запуске DELPHI вы можете увидеть на экране картинку, подобную представленной на рис. 1.1.

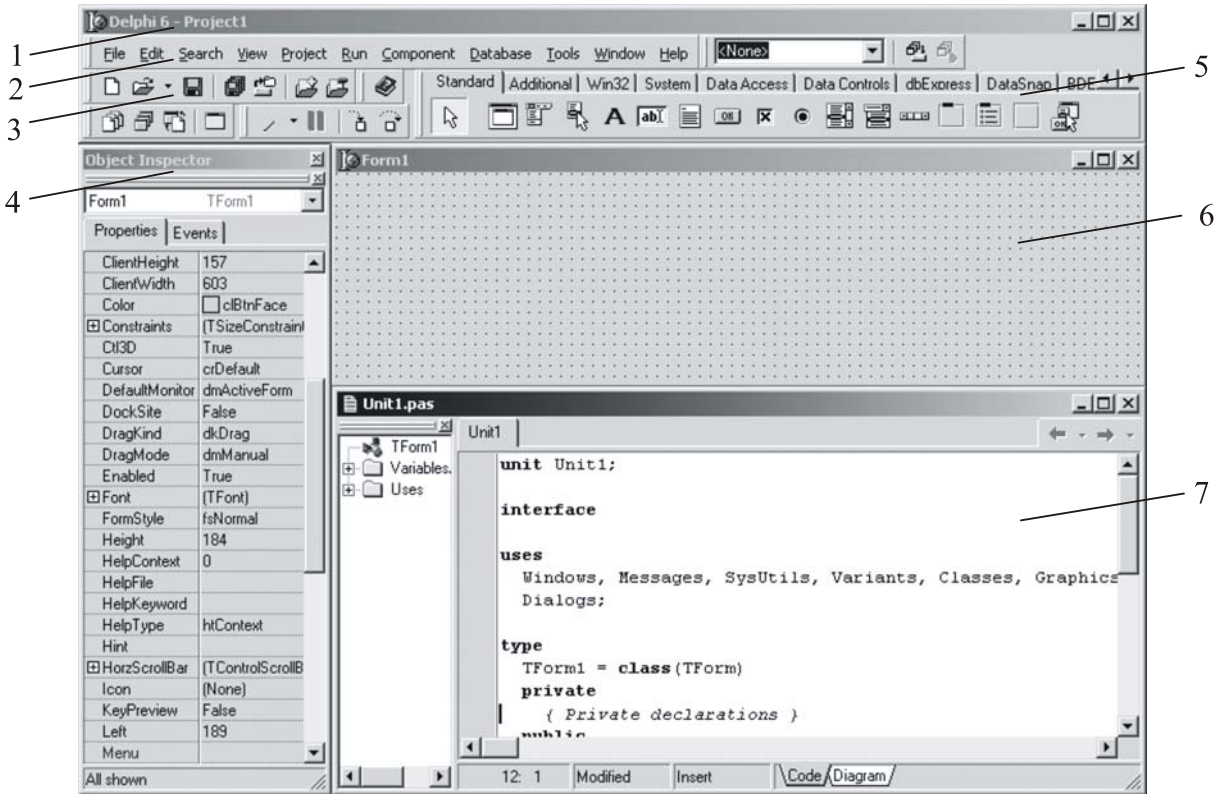


Рис.1.1.

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

Главное окно всегда присутствует на экране и предназначено для управления процессом создания программы. Основное меню содержит все необходимые средства для управления проектом. Пиктограммы облегчают

доступ к наиболее часто применяемым командам основного меню. Через меню компонентов осуществляется доступ к набору стандартных сервисных программ среды DELPHI, которые описывают некоторый визуальный элемент (компонент), помещенный программистом в окно формы. Каждый компонент имеет определенный набор свойств (параметров), которые программист может задавать. Например, цвет, заголовок окна, надпись на кнопке, размер и тип шрифта и др.

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

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

Окно текста программы предназначено для просмотра, написания и редактирования текста программы. В системе DELPHI используется язык программирования Object Pascal. При первоначальной загрузке в окне текста программы находится текст, содержащий минимальный набор операторов для нормального функционирования пустой формы в качестве Windows-окна. При помещении некоторого компонента в окно формы, текст программы автоматически дополняется описанием необходимых для его работы библиотек стандартных программ (раздел uses) и типов переменных (раздел type) (см. Листинг 1.1).

Программа в среде DELPHI составляется как описание алгоритмов, которые будут выполняться при возникновении того или иного события (например, щелчок мыши на кнопке – событие OnClick, создание формы – OnCreate). Для каждого обрабатываемого события, с помощью страницы Events инспектора объектов в тексте программы организуется процедура (procedure), между ключевыми словами begin и end которой программист записывает на языке Object Pascal требуемый алгоритм.

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

1.2. Структура программ DELPHI

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

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

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

Модуль имеет следующую структуру:

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

В разделе объявлений описываются типы, переменные, заголовки процедур и функций, которые могут быть использованы другими модулями, через операторы подключения библиотек (Uses). В разделе реализации располагаются тела процедур и функций, описанных в разделе объявлений, а также типы переменных, процедуры и функции, которые будут функционировать только в пределах данного модуля. Раздел инициализации используется редко и его можно пропустить.

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

1.3. Пример написания программы

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

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

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

1.3.1. Настройка формы

Для создания нового проекта выберите в основном меню пункт File–New–Application. Пустая форма в правом верхнем углу имеет кнопки управления, которые предназначены: для свертывания формы в пиктограмму , для разворачивания формы на весь экран и возвращения к исходному размеру и для закрытия формы . С помощью мыши, «захватывая» одну из кромок формы или выделенную строку заголовка отрегулируйте нужные размеры формы и ее положение на экране.

1.3.2. Изменение заголовка формы

Новая форма имеет одинаковые имя (Name) и заголовок (Caption) – Form1. Для изменения заголовка вызовите окно инспектора объектов и щелкните кнопкой мыши на форме. На странице Properties инспектора объектов найдите свойство Caption и в правой ячейке наберите «Иванов А. Гр. 710201 Линейный алгоритм».

1.3.3. Размещение строки ввода (TEdit)

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

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

Выберите в меню компонентов Standard пиктограмму  и щелкните мышью в том месте формы, где вы хотите ее поставить. Поместите три компонента TEdit в форму, в тексте программы (см. Листинг 1.1) появится три новых переменных – Edit1, Edit2, Edit3. Захватывая компоненты «мышью» отрегулируйте размеры окон и их положение. С помощью инспектора объектов установите шрифт и размер символов отражаемых в строке Edit (свойство Font). В свойстве Text инспектора задайте начальные значения переменных x,y,z.

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

1.3.4. Размещение надписей (TLabel)

На форме рис. 1.2 имеются четыре пояснительные надписи. Для нанесения таких надписей на форму используется компонент TLabel.

Выберите в меню компонентов Standard пиктограмму  и щелкните мышью в нужном месте формы (появится надпись Label1). Проделайте это для четырех надписей (в тексте программы автоматически появятся четыре новых переменных типа TLabel). Для каждой надписи, щелкнув на ней мышью, отрегулируйте размер и положение на форме. В свойство Caption введите строку,

например «Значения переменных», а также выберите размер символов (свойство Font).

1.3.5. Размещение многострочного окна вывода (TMemo)

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

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

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

Если выводятся данные, находящееся в переменных действительного или целого типа, то их надо предварительно преобразовать к типу String. Например, если переменная u:=100 целого типа, то метод Memo1.Lines.Add(‘Значение u=’+IntToStr(u)) сделает это, и в окне появится строка «Значение u=100». Если переменная u:=-256,38666 действительного типа, то при использовании метода Memo1.Lines.Add(‘Значение u=’+FloatToStrF(u,fffixed,8,2)) будет выведена строка «Значение u= -256,39». При этом под все число отводится восемь позиций, из которых две позиции занимает его дробная часть.

1.3.6. Написание программы обработки события создания формы (FormCreate)

После запуска программы, на некотором этапе ее выполнения, происходит создание спроектированной формы (событие OnCreate). Создадим подпрограмму – обработчик этого события (TForm1.FormCreate). Она очищает окно TMemo1. Для этого дважды щелкнем мышью на любом свободном месте формы. На экране появится текст, в котором автоматически внесен заголовок процедуры – обработчика события создания формы: ***Procedure TForm1.FormCreate(Sender:TObject)***. Между ***begin … end*** вставим текст программы (смотрите пример, расположенный ниже).

1.3.7. Написание программы обработки события нажатия кнопки (ButtonClick)

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

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

Наберите текст этой процедуры (см. Листинг 1).

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

Разместите на форме собственную кнопку для конца работы, поместив из папки Additional кнопку Bitbtn1 и установив в инспекторе для свойства Kind значение Close.

1.3.8. Запуск и работа с программой

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

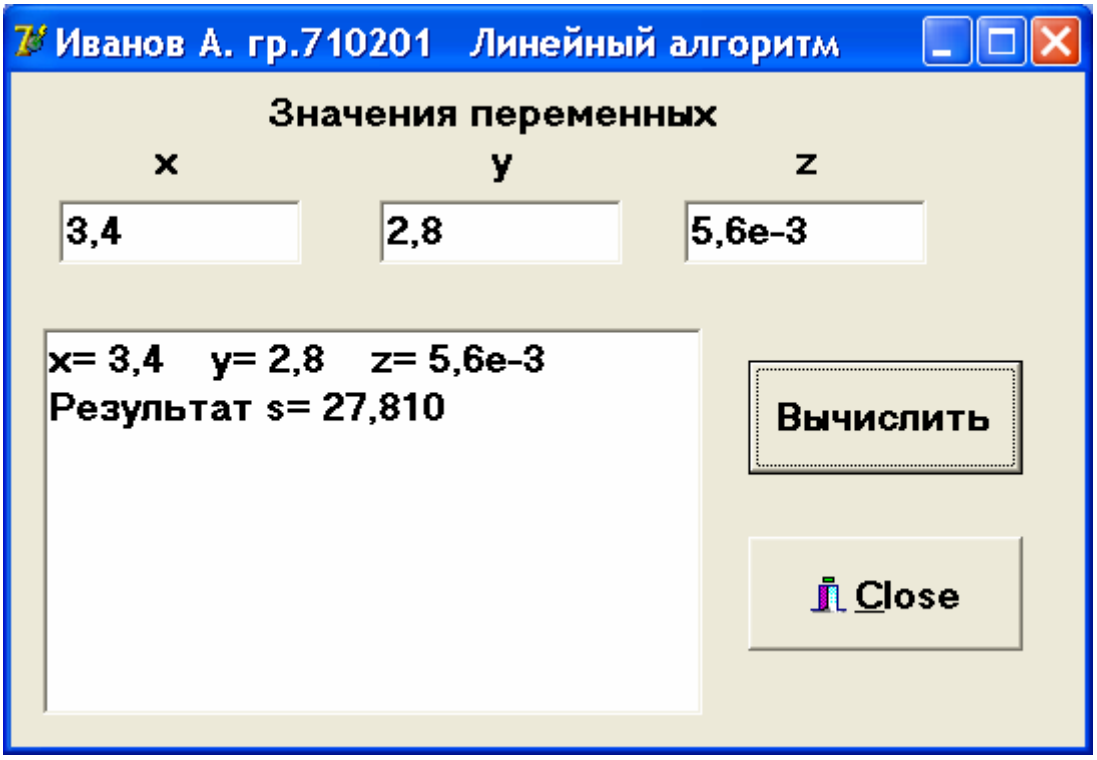


Рис.1.2

Работа с программой происходит следующим образом. Нажмите (щелкните мышью) кнопку «Вычислить». В окне Memo1 появляется результат. Измените исходные значения x, y, z в окнах Edit и снова нажмите кнопку «Вычислить» - появится новые результаты. Завершить работу программы можно или нажав кнопку  на форме или кнопку «Close» или, перейдя в окно DELPHI, выбрать в главном меню пункт Run – Program Reset. Последний способ выхода из программы обычно используют в случае ее зацикливания.

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

Листинг 1.1.

```
unit Unit1;
                    interface
uses
  Windows,Messages,SysUtils,Variants,Classes,Graphics,
Controls,Forms,Dialogs,Buttons,StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Memo1: TMemo;
    Button1: TButton;
    BitBtn1: TBitBtn;
    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
  memo1.Clear;    //   Очистка окна
end;

procedure TForm1.Button1Click(Sender: TObject);
  var x,y,z,a,b,c,s:extended;
begin
  x:=strtofloat(edit1.text);
  y:=strtofloat(edit2.text);
  z:=strtofloat(edit3.text);
  a:=sqr(sin(x+y)/cos(x+y));
  b:=abs(exp(3*y)-x*x);
  c:=sqrt(arctan(z)+ln(x));
  s:=a*b/c;
// Вывод исходных данных и результата в окно memo1
  memo1.Lines.Add('x='+edit1.Text+
     '    y='+edit2.Text+'    z='+edit3.Text);
  memo1.Lines.Add('Результат s=
'+floattostrF(s,fffixed,8,3));
end;

end.
```

1.4. Индивидуальные задания

По указанию преподавателя выберите индивидуальное задание. Установите необходимое количество окон Edit, меток label. Выберите необходимые типы переменных и функции их преобразования при вводе и выводе данных. С помощью инспектора объектов измените цвет формы, шрифт выводимых символов.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.1. Обработка событий

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

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

Каждый компонент имеет свой набор обработчиков событий, однако некоторые из них присуши большинству компонентов. Наиболее часто применяемые события представлены в табл. 2.1.

Таблица 2.1

Событие	Описание события
OnActivate	Форма получает это событие при активации
OnCreate	Возникает при создании формы (компонент TForm). В обработчике данного события следует задавать действия, которые должны происходить в момент создания формы, обычно установку начальных значений в окнах формы
OnKeyPress	Возникает при нажатии кнопки на клавиатуре. Параметр Key имеет тип Char и содержит ASCII-код нажатой клавиши (клавиша Enter клавиатуры имеет код #13, клавиша Esc - #27 и т.д.).
OnKeyUp	Является парным событием для OnKeyDown и возникает при отпускании ранее нажатой клавиши
OnClick	Возникает при нажатии кнопки мыши в области компонента
OnDblClick	Возникает при двойном нажатии кнопки мыши в области компонента

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

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

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

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

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

Например, если vib : integer, то после выполнения

```
case vib of
     0:u:=x+y;
     1,5:u:=x-y;
     2,4,6:u:=x*y;
         else u:=0;
end;
```

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

2.3. Кнопки-переключатели в Delphi

При создании программ в DELPHI для организации разветвлений часто используются компоненты в виде кнопок-переключателей. Состояние такой кнопки (включено - выключено) визуально отражается на форме. На форме (рис.2.1) представлены кнопки-переключатели двух типов (TCheckBox, TRadioGroup).

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

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

2.4. Пример написания программы

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

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

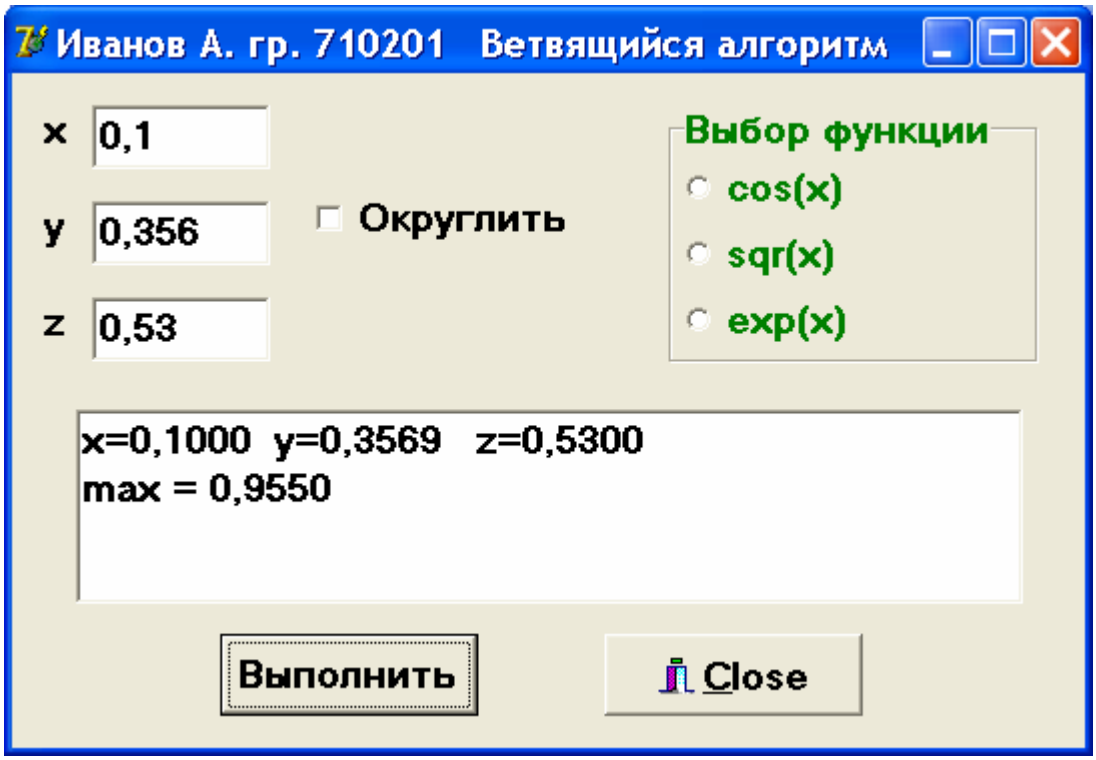


Рис. 2.1.

2.4.1. Создание формы

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

2.4.2. Работа с компонентом TCheckBox

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

2.4.3. Работа с компонентом TRadioGroup

Выберите в меню компонентов Standard пиктограмму  и поместите ее в нужное место формы. На форме появится окаймленный линией чистый прямоугольник с заголовком RadioGroup1. Замените заголовок (Caption) на «Выбор функции». Для того чтобы разместить на компоненте кнопки, необходимо свойство Columns установить равным единице (кнопки

размещаются в одном столбце). Дважды щелкните по правой части свойства Items «мышью», появится строчный редактор списка заголовков кнопок. Наберите три строки с именами: в первой строке – «cos(x)»,. во второй – «sqr(x)», в третьей – «exp(x)», нажмите ОК.

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

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

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

Листинг 2.1.

```
unit Unit2;
       interface
uses
  Windows,Messages,SysUtils,Variants,Classes,Graphics,
       Controls,Forms,Dialogs,StdCtrls,ExtCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    CheckBox1: TCheckBox;
    RadioGroup1: TRadioGroup;
    Label2: TLabel;
    Edit2: TEdit;
    Label3: TLabel;
    Edit3: TEdit;
    Label4: TLabel;
    Memo1: TMemo;
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    private   { Private declarations }
    public    { Public declarations }
    end;

var
  Form1: TForm1;

implementation
{$R *.dfm}

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

```
begin
  Edit1.text:='0,1';
  Edit2.text:='0,356';
  Edit3.text:='0,53';
  Memo1.Clear; ;  // Активна первая кнопка RadioGroup1
  RadioGroup1.ItemIndex:=0;
end;

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

2.5. Индивидуальные задания

В вариантах 1-10 вычислить выражение по одной из трех формул в зависимости от результата выполнения условия. В качестве f(x) использовать по выбору: sh(x) или x^2 или e^x. Отредактируйте вид формы и текст программы, в соответствии с полученным заданием. Используя теорию п. 2.1. создайте вместо обработчика Button1.Click обработчик Memo1Click.

.

1. $(f(x)+y)^2$, $xy>0$
   $f(x)*\cos(y)$, $xy<0$
   $f(x)+y$, $xy=0$

2. $\ln(y+2) + f(x)$, $x/y>0$
   $\ln|y| - tg(f(x))$, $x/y<0$
   $f(x)*y^3$, иначе

3. $e^{f(x)} + sh(y)$, $5<x<9$
   $ch(y) + f(x)^4$, $x>10$
   $|f(x)|*tg(y)$, иначе

4. $\ln|y+f(x)|$, $3<xy<8$
   $\cos(f(x))-y$, $xy>12$
   $sh(f(x)+cs(y))$, иначе

5. $f(x)^3+ctg(y)$, $xy>12$
   $sh(f(x)^3)+ y^2$, $xy<7$
   $\cos(x - f(x)^3)$, иначе

6. $f(x)^3+\sin(y)$, $xy<5$
   $ch(f(x)^3)+ y^2$, $xy>7$
   $\cos(x+f(x)^3)$, иначе

7. $(f(x)+y)^2$, $4>xy>1$
   $f(x)*tg(y)$, $8<xy<10$
   $f(x)+y$, иначе

8. $e^{f(x)} + \cos(y)$, $2<x<4$
   $sh(y) + f(x)^4$, $x>10$
   $|f(x)|*tg(y)$, иначе

9. $ctg(y) + f(x)$, $x/y>0$
   $\ln|y| + tg(f(x))$, $x/y<0$
   $f(x)*y^3$, иначе

10. $f(x)^3+\sin(y)$, $xy<0$
    $ch(f(x)^3)+ y^2$, $xy>11$
    $tg(x+f(x)^3)$, иначе

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

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

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

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

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

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

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

3.1. Средства отладки программ в DELPHI

Практически в каждой вновь написанной программе после запуска обнаруживаются ошибки. Ошибки первого уровня (ошибки компиляции) связаны с неправильной записью операторов (орфографические, синтаксические). При обнаружении ошибки компилятор DELPHI останавливается напротив первого оператора, в котором обнаружена ошибка. В нижней части экрана появляется текстовое окно, содержащее сведения обо всех ошибках найденных в проекте. Каждая строка этого окна содержит имя файла, в котором найдена ошибка, номер строки с ошибкой и характер ошибки. Для быстрого перехода к интересующей ошибке необходимо дважды щелкнуть на строке с ее описанием. Для получения более полной информации о характере ошибки необходимо обратится к HELP нажатием клавиши F1. Следует обратить внимание на то, что одна ошибка может повлечь за собой другие, которые исчезнут при ее исправлении. Поэтому следует исправлять ошибки последовательно, сверху вниз и, после исправления каждой ошибки компилировать программу снова. Ошибки второго уровня (ошибки выполнения) связаны с ошибками выбранного алгоритма решения или с неправильной программной реализацией алгоритма. Эти ошибки проявляются в том, что результат расчета оказывается неверным либо происходит переполнение, деление на ноль и др. Поэтому перед использованием отлаженной программы ее надо протестировать, т.е. сделать просчеты при таких комбинациях исходных данных, для которых заранее известен результат. Если тестовые расчеты указывают на ошибку, то для ее поиска следует использовать встроенные средства отладки среды DELPHI. В простейшем случае для локализации места ошибки рекомендуется поступать следующим образом. В окне редактирования текста установить курсор в строке перед подозрительным участком и нажать клавишу F4 (выполнение до курсора) или щелкнуть на серой полосе слева от оператора для обозначения точки прерывания (появится красная точка) и нажать клавишу F9. Выполнение программы будет остановлено на указанной строке. Для просмотра текущих значений можно поместить на нужную переменную курсор (на экране будет высвечено ее значение), либо нажать Ctrl-F7 (окно оценки и модификации) или Ctrl-F5 (окно наблюдения) и в появившимся диалоговом окне указать

интересующую переменную. Нажимая клавишу F7 (пошаговое выполнение), можно построчно выполнять программу, контролируя изменение тех или иных переменных и правильность вычислений. Если курсор находится внутри цикла, то после нажатия F4 расчет останавливается после одного выполнения тела цикла. Для продолжения расчетов следует нажать <Run> меню Run или F9.

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

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

Оператор

```
repeat
<операторы>
Until <условие>;
```

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

Оператор

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

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

Оператор

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

организует повторение операторов при нарастающем изменении переменной цикла i от начального значения i1 до конечного i2 с шагом единица. Заметим, что если i2>i1, то <операторы> не выполнятся ни разу. Модификация оператора

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

организует повторения при убывающем изменении i на единицу.

Для прекращения выполнение цикла используется процедура ***Break***, которая прерывает выполнение тела любого цикла и передает управление

следующему за циклом оператору. Для прерывания текущей итерации цикла и передачи управления следующей, используется процедура ***Continue.***

3.3. Пример написания программы

Задание: написать и отладить программу, которая выводит таблицу значений функции $y(x) = e^{-x}$ и ее разложения в ряд $s(x) = \sum_{k=0}^{\infty} a^k = \sum_{k=0}^{\infty} (-1)^k \frac{x^k}{k!}$ для x изменяющихся в интервале от a до b с шагом h. Функцию s(x) вычислять с точностью до 0,0001. Вывести число итераций, необходимое для достижения заданной точности. При составлении алгоритма вычисления удобно использовать рекуррентную последовательность (такую последовательность, каждое новое слагаемое которой зависит от одного или нескольких предыдущих). Для получения расчетной формулы рассмотрим значение слагаемого при различных значениях *k*: при $k = 0;\ a_0 = 1\frac{1}{1};$ при $k = 1;\ a_1 = -1\frac{x}{1};$ при $k = 2;\ a_2 = 1\frac{x \cdot x}{1 \cdot 2};$ при $k = 3;\ a_3 = -1\frac{x \cdot x \cdot x}{1 \cdot 2 \cdot 3}$ и т.д. Видно, что на каждом шаге слагаемое дополнительно умножается на $-1\frac{x}{k}$. Исходя их этого, формула рекуррентной последовательности будет иметь вид: $a_k = -a_{k-1}\frac{x}{k};$. Полученная формула позволяет избавиться от многократного вычисления факториала и возведения в степень. Если в выражении имеется нерекуррентая часть, то ее следует рассчитывать отдельно.

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

Рис. 3.1.

```
Unit Unit1;
             interface
uses
  Windows,Messages,SysUtils,Variants,Classes,Graphics,
        Controls,Forms,Dialogs,Buttons,StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Label2: TLabel;
    Edit4: TEdit;
    Label3: TLabel;
    Button1: TButton;
    Memo1: TMemo;
    BitBtn1: TBitBtn;
    Label4: TLabel;
    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
   memo1.Clear;
end;

procedure TForm1.Button1Click(Sender: TObject);
   var   a,b,h,x,w,s,eps:extended;
          k:integer;
begin
    a:=strtofloat(edit1.Text);
    b:=strtofloat(edit2.Text);
    h:=strtofloat(edit3.Text);
    eps:=strtofloat(edit4.Text);
    x:=a;
    repeat
          s:=1;   w:=1;   k:=0;
          repeat
              inc(k);
              w:=-w*x/k;
              s:=s+w;
          until abs(w)<eps;
          memo1.Lines.Add(floattostrf(x,fffixed,5,2)+
            '  '+floattostrf(s,fffixed,9,6)+
            '  '+floattostrf(exp(-x),fffixed,9,6)+
            '  '+inttostr(k));
          x:=x+h;ε
      until x>b+0.0000000001;
end;

end.
```

3.4. Индивидуальные задания

Вывести на экран таблицу значений функции *Y(x)* и ее разложения в ряд *S(x)* для x изменяющихся от *a* до *b* с заданным шагом *h* и точностью ε. Близость значений *S(x)* и *Y(x)* во всем диапазоне значений x указывает на правильность вычисления *S(x)* и *Y(x)*.

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

Таблица 3.1.

№	x_n	x_k	S(x)	e	Y(x)
1.	0.1	1	$\sum_{n=0}^{\infty}(-1)^n\frac{x^{2n+1}}{(2n+1)!}$	0.001	$\sin x$
2.	0.1	1	$\sum_{n=0}^{\infty}\frac{x^{2n}}{(2n)!}$	0.0001	$\frac{e^x+e^{-x}}{2}$
3.	0.1	1	$\sum_{n=0}^{\infty}\frac{\cos n\frac{\pi}{4}}{n!}x^n$	0.001	$e^{x\cos\frac{\pi}{4}}\cos(x\sin\frac{\pi}{4})$
4.	0.1	1	$\sum_{n=0}^{\infty}(-1)^n\frac{x^{2n}}{(2n)!}$	0.0001	$\cos x$
5.	0.1	0.7	$\sum_{n=0}^{\infty}\frac{2n+1}{n!}x^{2n}$	0.001	$(1+2x^2)e^{x^2}$
6.	0.1	1	$\sum_{n=0}^{\infty}\frac{x^{2n+1}}{(2n+1)!}$	0.0001	$\frac{e^x-e^{-x}}{2}$
7.	0.2	1	$\sum_{n=0}^{\infty}\frac{(\ln 9)^n}{n!}x^n$	0.001	9^x
8.	0.1	0.7	$\sum_{n=0}^{\infty}\frac{(2x)^n}{n!}$	0.0001	e^{2x}
9.	0.3	1	$\sum_{n=0}^{\infty}\frac{n^2+1}{n!}\left(\frac{x}{2}\right)^n$	0.001	$\left(\frac{x^2}{4}+\frac{x}{2}+1\right)e^{\frac{x}{2}}$
10	0.1	0.5	$\sum_{n=0}^{\infty}(-1)^n\frac{x^{2n+1}}{2n+1}$	0.0001	$arctgx$

11	0.2	1	$\sum_{n=0}^{\infty}(-1)^n \frac{2n^2+1}{(2n)!}x^{2n}$	0.001	$\left(1-\frac{x^2}{2}\right)\cos x - \frac{x}{2}\sin x$
12	0.1	1	$\sum_{n=1}^{\infty}(-1)^n \frac{(2x)^{2n}}{(2n)!}$	0.0001	$2(\cos^2 x - 1)$
13	-2	-0.1	$\sum_{n=0}^{\infty}(-1)^n \frac{x^{2n}}{(2n+1)!}$	0.0001	$\frac{\sin(x)}{x}$
14	0.2	0.8	$\sum_{n=1}^{\infty}\frac{n^2}{(2n+1)!}x^n$	0.0001	$\frac{1}{4}\left(\frac{x+1}{\sqrt{x}}sh\sqrt{x} - ch\sqrt{x}\right)$
15	0.1	0.8	$\sum_{n=1}^{\infty}(-1)^{n+1}\frac{x^{2n}}{2n(2n-1)}$	0.001	$xarctgx - \ln\sqrt{1+x^2}$

ТЕМА 4. ОБРАБОТКА ИСКЛЮЧИТЕЛЬНЫХ СИТУАЦИЙ. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ МАССИВОВ

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

4.1. Обработка исключительных ситуаций

Под исключительной ситуацией понимается некое ошибочное состояние, возникающее при выполнении программы и требующее выполнения определённых действий для продолжения работы или корректного ее завершения. Стандартный обработчик (метод **TApplication.HandleException**), вызываемый по умолчанию, информирует пользователя о возникновении ошибки и завершает выполнение программы. Для защиты от завершения в языке Object Pascal используется оператор **try**, который перехватывает исключительную ситуацию и дает возможность разработчику предусмотреть определенные действия при ее возникновении.

Конструкция блока **try… finally**:

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

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

Конструкция блока **try…except**:

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

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

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

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

Таблица 4.1.

Тип исключительной ситуации	Причина
EAbort	Намеренное прерывания программы, генерируемое процедурой **Abort**
EArrayError	Ошибка при операциях с массивами: использование ошибочного индекса массива, добавление слишком большого числа элементов в массив фиксированной длины (для использования требует подключения модуля mxarrays)
EConvertError	Ошибка преобразования строки в другие типы данных
EDivByZero	Попытка целочисленного деления на нуль
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 нуль 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+ })

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

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

4.2. Использование функций ShowMessage и MessageDlg

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

Таблица 4.2.

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

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

Таблица 4.3.

mbYes	Кнока «Yes»	mbRetry	Кнока «Retry»
mbNo	Кнока «No»	mbIgnore	Кнока «Ignore»
mbOK	Кнока «OK»	mbAll	Кнока «All»

mbCancel	Кнока «Cancel»	mbHelp	Кнока «Help»
mbAbort	Кнока «Abort»		

Параметр HelpCtx определяет номер контекстной справки для данного окна. Результатом выполнения функции является значение, соответствующее нажатой кнопке. Возвращаемое значение имеет имя, состоящее из букв mr и имени кнопки, например: mrYes, mrOK, mr Help.

4.3. Работа с массивами

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

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

```
Array [описание индексов] of <тип элемента массива>;
```

Примеры описания массивов:

```
Const Nmax=20; // максимальное значение индекса
 Type vek=array[1..Nmax] of word;
    // Описание типа одномерного массива
 Var a:vek; // a – массив типа vek
     s:array[1..10] of integer;
      // s–массив из 10 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[s[4]+1]*3;
a[n]:=1+sqrt(abs(a[n-1]));
```

4.4. Компонент TStringGrid

При работе с массивами ввод и вывод информации на экран удобно организовывать в виде таблиц. Компонент TStringGrid предназначен для отображения информации в виде двумерной таблицы, каждая ячейка которой представляет собой окно однострочного редактора (аналогично окну TEdit). Доступ к информации осуществляется с помощью свойства Cells[ACol : Integer; ARow : Integer] : String, где ACol, ARow - индексы элемента двумерного массива. Свойства ColCount и RowCount устанавливают количество столбцов и строк в таблице, а свойства FixedCols и FixedRows задают количество столбцов и строк фиксированной зоны. Фиксированная зона выделена другим цветом, и в нее запрещен ввод информации с клавиатуры. Для установки компонента TStringGrid на форму необходимо на странице Additional меню компонентов

щелкнуть мышью по пиктограмме  . После этого щелкните мышью в нужном месте формы. Захватывая кромки компонента, отрегулируйте его размер. В инспекторе объектов значения свойств ColCount и RowCount установите 4 (четыре строки и четыре столбца), а FixedCols и FixedRows установите 1 (один столбец и одна строка с фиксированной зоной). Т.к. компоненты StringGrid2 и StringGrid3 имеют только один столбец, то у них: ColCount= 1, RowCount=4, FixedCols=0 и FixedRows=1. По умолчанию в компонент TStringGrid запрещен ввод информации с клавиатуры, поэтому для компонентов StringGrid1 и StringGrid2 необходимо в инспекторе объектов раскрыть раздел Options (нажав на знак «+», стоящий слева от Options) и свойство goEditing установить в положение True.

4.5. Пример написания программы

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

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

Панель диалога приведена на рис. 4.1. Текст программы приведен ниже.

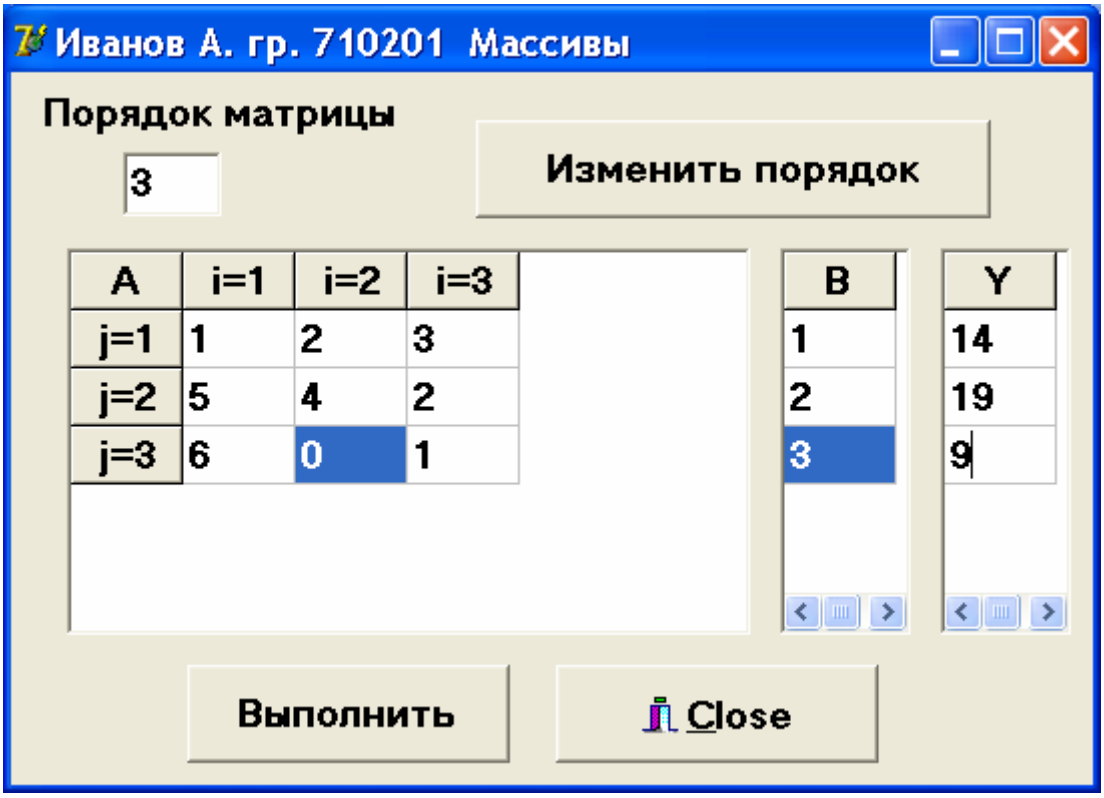


Рис. 4.1.

```
unit Unit4;
          interface
uses Windows,Messages,SysUtils,Variants,Classes,
Graphics,Controls,Forms,Dialogs,Grids,StdCtrls,
mxarrays;
type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Label1: TLabel;
    Button1: TButton;
    StringGrid1: TStringGrid;
    StringGrid2: TStringGrid;
    Button2: TButton;
    StringGrid3: TStringGrid;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

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

  implementation
  {$R *.dfm}

 procedure TForm1.FormCreate(Sender: TObject);
   begin
     N:=3;   Edit1.Text:=intToStr(N);
     {Задание числа строк и столбцов в таблицах}
     StringGrid1.RowCount:=N+1;
     StringGrid1.ColCount:=N+1;
     StringGrid2.RowCount:=N+1;
     StringGrid3.RowCount:=N+1;
     StringGrid1.Cells[0,0]:='   A';
     StringGrid2.Cells[0,0]:='   B';
     StringGrid3.Cells[0,0]:='   Y';
      {Заполнение   верхнего   и   левого   столбцов
  поясняющими подписями}
     for i:=1 to N do
       begin
       StringGrid1.Cells[0,i]:=' i='+IntToStr(i);
       StringGrid1.Cells[i,0]:=' j='+IntToStr(i);
       end;
   end;
end;

procedure TForm1.Button1Click(Sender: TObject);
  begin           // Изменить порядок
    N:=StrToInt(Edit1.Text);
   {Задание числа строк и столбцов в таблицах}
    StringGrid1.RowCount:=N+1;
    StringGrid1.ColCount:=N+1;
    StringGrid2.RowCount:=N+1;
    StringGrid3.RowCount:=N+1;
```

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

 procedure TForm1.Button2Click(Sender: TObject);
   begin   //   Выполнить
      {$R+}
   try
{Заполнение массива A элементами из таблицы StringGrid1}
for i:=1 to N do
  for j:=1 to N do
       A[i,j]:=StrToFloat(StringGrid1.Cells[j,i]);
{Заполнение массива B элементами из таблицы StringGrid2}
for i:=1 to N do
  B[i]:=StrToFloat(StringGrid2.Cells[0,i]);
except
  on ERangeError do
     begin
     ShowMessage('Уменьшите размер массива');
     Exit;
     end;
  on EConvertError do
     begin
     ShowMessage(` Прверьте значение числа');
     Exit;
     end;
         else
     begin
     ShowMessage('Возникла неизвестная
            исключительная ситуация!');
     Exit;
     end;
 end;
 try
          {Умножение массива A на массив B}
 for i:=1 to N do
     begin
     Y[i]:=0;
     for j:=1 to N do Y[i]:=Y[i]+A[i,j]*B[j];
```

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

4.6. Индивидуальные задания

Во всех заданиях переменные вводить и выводить с помощью компонента TEdit, массивы – с помощью компонента TStringGrid, в котором 0-й столбец и 0-ю строку использовать для отображения индексов массивов. Вычисления выполнять, после нажатия кнопки типа TButton. В местах возможного возникновения ошибок использовать конструкции для обработки исключительных ситуаций.

1. Задана матрица размером N строк и M столбцов. 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 строк и M столбцов. 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 строк и M столбцов. 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 строк и M столбцов. Определить количество «особых» элементов матрицы, считая элемент «особым», если он больше суммы остальных элементов своего столбца.

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

6. Дана матрица размером *N* строк и *M* столбцов. Упорядочить ее строки по возрастанию их первых элементов.

7. Дана матрица размером *N* строк и *M* столбцов. Упорядочить ее строки по возрастанию суммы их элементов.

8. Дана матрица размером *N* строк и *M* столбцов. Упорядочить ее строки по возрастанию их наибольших элементов.

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

10. Задана матрица A, размером *N* строк и *M* столбцов. Получить массив *B*, присвоив его *k*-му элементу значение максимального элемента в *k*–от столбце матрицы A.

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

12. В матрице размером *N* строк и *M* столбцов поменять местами строку, содержащую элемент с наибольшим значением со строкой, содержащей элемент с наименьшим значением.

13. Из матрицы *n*-го порядка получить матрицу порядка *n-1* путем удаления из исходной матрицы строки и столбца, на пересечении которых расположен элемент с наибольшим по модулю значением.

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

15. Дана матрица размером *N* строк и *M* столбцов. Поменять местами все четные и нечетные строки матрицы.

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

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

5.1. Динамическое распределение памяти

В языке Паскаль, наряду с обычным, статическим, возможна организация динамического распределения памяти, при которой оперативная память для размещения данных выделяется непосредственно во время выполнения программы по мере надобности. Если переменная, соответствующая этим данным, становится ненужной, то она удаляется, а выделенная под нее память освобождается. Обычно динамическое выделение и освобождение памяти используется при работе с массивами данных. Обеспечение работы с динамическими данными реализуется с помощью переменных типа ***указатель***.

5.2. Компонент TBitBtn

Компонент TBitBtn расположен на странице Additional палитры компонентов и представляет собой разновидность стандартной кнопки TButton. Его отличительная особенность – наличие растрового изображения на поверхности кнопки, которое определяется свойством Clyph. Кроме того, имеется свойство Kind, которое задает одну из 11 стандартных разновидностей кнопок. Кнопка bkClose закрывает главное окно и завершает работу программы.

5.3. Индивидуальные задания

Необходимый теоретический материал и примеры работы с динамическими массивами приведены в лекции «Указатели». Выделение памяти под массив организовать динамически. Ввод данных производить из StringGrid1. Вывод организовать, в зависимости от варианта, в StringGrid2, или в Edit1.

1. Дан массив, состоящий из символов. Расположить его элементы в обратном порядке.
2. Дан массив, состоящий из символов. Преобразовать его по следующему правилу: сначала должны находиться цифры, а затем все остальные символы, сохраняя при этом взаимное расположение символов в каждой из этих двух групп.
3. Дан массив, состоящий из символов. Вывести на экран цифру, наиболее часто встречающуюся в этом массиве.
4. Дан массив, состоящий из символов. Определить количество различных элементов массива (т.е. повторяющиеся элементы считать один раз).

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

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

7. Дан массив, состоящий из символов. Преобразовать массив по следующему правилу: все прописные латинские буквы перенести в начало, а все строчные латинские буквы – в конец, сохраняя исходное взаимное расположение.

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

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

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

11. Дан массив, состоящий из символов. Заменить в нем строчные латинские буквы прописными.

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

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

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

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

ТЕМА 6. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ПОДПРОГРАММ И МОДУЛЕЙ

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

6.1. Использование модулей

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

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

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

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

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

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

6.2. Создание модуля

В среде Delphi модули могут создаваться как со своей формой, так и без нее. Для создания нового модуля без своей формы необходимо в меню File выбрать New – Unit. В результате будет создан файл с заголовком Unit Unit2. Имя модуля можно изменить на другое, отвечающее внутреннему содержанию модуля, например Unit biblio. Для этого необходимо сохранить содуль с новым именем (например, biblio.pas). Следует обратить внимание на то, что имя файла должно совпадать с именем модуля.

6.3. Подключение модуля

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

6.4. Пример написания программы

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

Панель диалога будет иметь следующий вид.

Тексты модуля (Листинг 6.1.) и вызывающей программы (Листинг 6.2.) приведены ниже.

Листинг 6.1.

```
unit biblio;

                    interface
function y(x:extended):extended;
```

```
 procedure sum(x,eps:extended; var s:extended; var:word);

                implementation
   function y;
      begin
        result:=exp(-x);
      end;

   procedure sum;
         var w:extended;
      begin
        s:=1;  w:=1; k:=0;
        repeat
          inc(k);
          w:=-w*x/k;
          s:=s+w;
        until abs(w)<eps;
      end;

  end.
```

Листинг 6.2.

```
unit Unit1;
                  interface
uses
   Windows,Messages,SysUtils,Variants,Classes,Graphics,
       Controls,Forms,Dialogs,Buttons,StdCtrls,biblio;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Label2: TLabel;
    Edit4: TEdit;
    Label3: TLabel;
    Button1: TButton;
    Memo1: TMemo;
    BitBtn1: TBitBtn;
    Label4: TLabel;
    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
   memo1.Clear;
end;

procedure TForm1.Button1Click(Sender: TObject);
     var a,b,h,x,eps,s:extended;
           it:word;
begin
  a:=strtofloat(edit1.Text);
  b:=strtofloat(edit2.Text);
  h:=strtofloat(edit3.Text);
  eps:=strtofloat(edit4.Text);
  x:=a;
  repeat
    sum(x,eps,s,it);
    memo1.Lines.Add(floattostrf(x,fffixed,5,2)+
      '    '+floattostrf(s,fffixed,9,6)+
      '    '+floattostrf(y(x),fffixed,9,6)+
      '    '+inttostr(it));
    x:=x+h;
  until x>b+0.0000000001;
end;

end.
```

6.5. Индивидуальные задания

Необходимый теоретический материал и примеры работы с подпрограммами и модулями изложены в лекции «Подпрограммы и модули». Выберите вариант задачи из заданий, приведенных в теме 3. В местах возможного возникновения ошибок использовать конструкции для обработки исключительных ситуаций.

ТЕМА 7. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ МНОЖЕСТВ И СТРОК

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

7.1. Системы счисления

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

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

где a_i – цифры, p – основание системы счисления. Количество цифр равно p. Для записи цифр в общем случае может быть использован любой набор p символов. Обычно для p<10 используют символы 0…9, для $p \geq 10$ добавляют буквы латинского алфавита A, B, C, D, E, F, которые в десятичной системе представляют числами 10, 11, 12, 13, 14, 15. Например

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

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

В компьютерной технике обычно используются системы с основанием равным степени двойки: двоичная, восьмеричная и шестнадцатеричная. Имеются процессоры, реализующие троичную систему счисления. Для удобства пользователей ввод – вывод и операции над числами в компьютере производят в десятичной системе счисления. При переводе числа из десятичной системы счисления в другую систему счисления, целая и дробная часть числа переводятся различным образом. При переводе целой части, она делится на основание новой системы счисления, остаток представляет очередную цифру $a_0, a_1, \ldots, a_n$, а частное снова делится на основание. Процесс повторяется до тех пор, пока частное не станет равным нулю. Заметим, что цифры получаются в порядке, обратном порядку их следования в записи числа. При переводе дробной части она умножается на основание системы счисления. Целая часть полученного числа представляет очередную цифру $a_{-1}, a_{-2}, \ldots, a_{-k}$, а дробная часть опять умножается на основание системы. Расчеты ведут до получения требуемого количества цифр.

7.2. Индивидуальные задания

Необходимый теоретический материал и примеры приведены в лекциях «Работа с множествами» и «Строки». Для ввода строк и работы с ними использовать компонент TEdit. Ввод строки заканчивать нажатием клавиши Enter. Алогритм оформить в виде подпрограммы.

1. Дана строка символов, состоящая из слов, разделенных пробелами. Найти количество слов с пятью символами.

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

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

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

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

6. Дана строка, состоящая из слов, разделенных пробелами. Найти и вывести на экран самое короткое слово.

7. Дана строка, состоящая из слов, отделенных друг от друга одним или несколькими разделителями (пробелы, точки, запятые, скобки и пр.). Подсчитать количество символов в самом длинном слове.

8. Дана строка, состоящая из слов, отделенных друг от друга одним или несколькими разделителями (пробелы, точки, запятые, скобки и пр.). Найти и вывести на экран слова с четным количеством символов.

9. Дана строка, состоящая из слов, отделенных друг от друга одним или несколькими разделителями (пробелы, точки,запятые, скобки и пр.). Подсчитать количество разных символов в словах.

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

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

12. Дана строка символов, состоящая из произвольного текста, слова разделены одним или несколькими 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, состоящая из слов, отделенных друг от друга одним или несколькими разделителями (пробелы, точки,запятые, скобки и пр.). Вывести на экран порядковый номер слова минимальной длины.

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

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

ТЕМА 8. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ЗАПИСЕЙ И ФАЙЛОВ

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

8.1. Компоненты TOpenDialog и TSaveDialog

Компоненты TOpenDialog и TSaveDialog находятся на странице DIALOGS. Все компоненты этой страницы являются невизуальными, т.е. не видны в момент работы программы. Поэтому их можно разместить в любом удобном месте формы. Оба рассматриваемых компонента имеют идентичные свойства и отличаются только внешним видом. После вызова компонента появляется диалоговое окно, с помощью которого выбирается имя программы и путь к ней. В случае успешного завершения диалога имя выбранного файла и маршрут поиска содержатся в свойстве FileName. Для фильтрации файлов, отображаемых в окне просмотра, используется свойство Filter, а для задания расширения файла, в случае, если оно не задано пользователем, – свойство DefaultExt. Если необходимо изменить заголовок диалогового окна, используется свойство Title.

8.2. Настройка компонентов TOpenDialog и TSaveDialog

Для установки компонентов TOpenDialog и TSaveDialog на форму, необходимо на странице Dialogs меню компонентов щелкнуть мышью соответственно по пиктограммам  или  и поставить их в любое свободное место формы. Настройка фильтра производится следующим образом. Выбрав соответствующий компонент, дважды щелкнуть по правой части свойства Filter инспектора объектов. Появится окно Filter Editor, в левой части которого записывается текст, характеризующий соответствующий фильтр, а в правой части – маску. Для OpenDialod1 установим значения маски как показано на рис. 8.1. Формат *.dat означает что, будут видны все файлы с расширением *dat*, а формат *.* - что будут видны все файлы (с любым именем и с любым расширением).

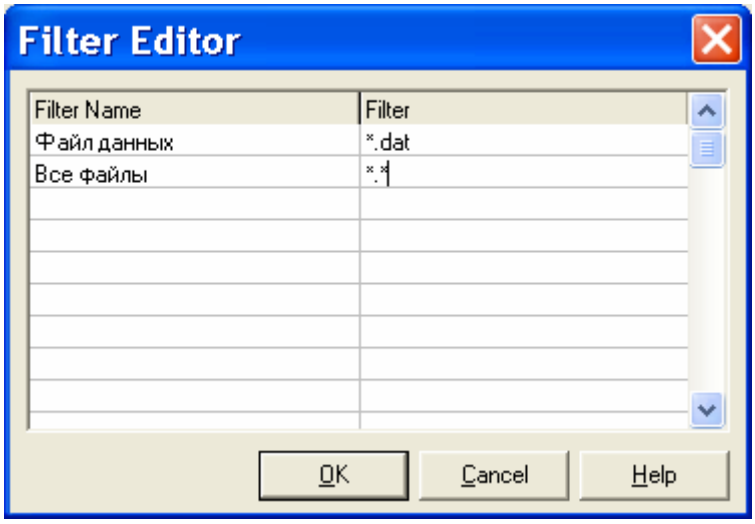
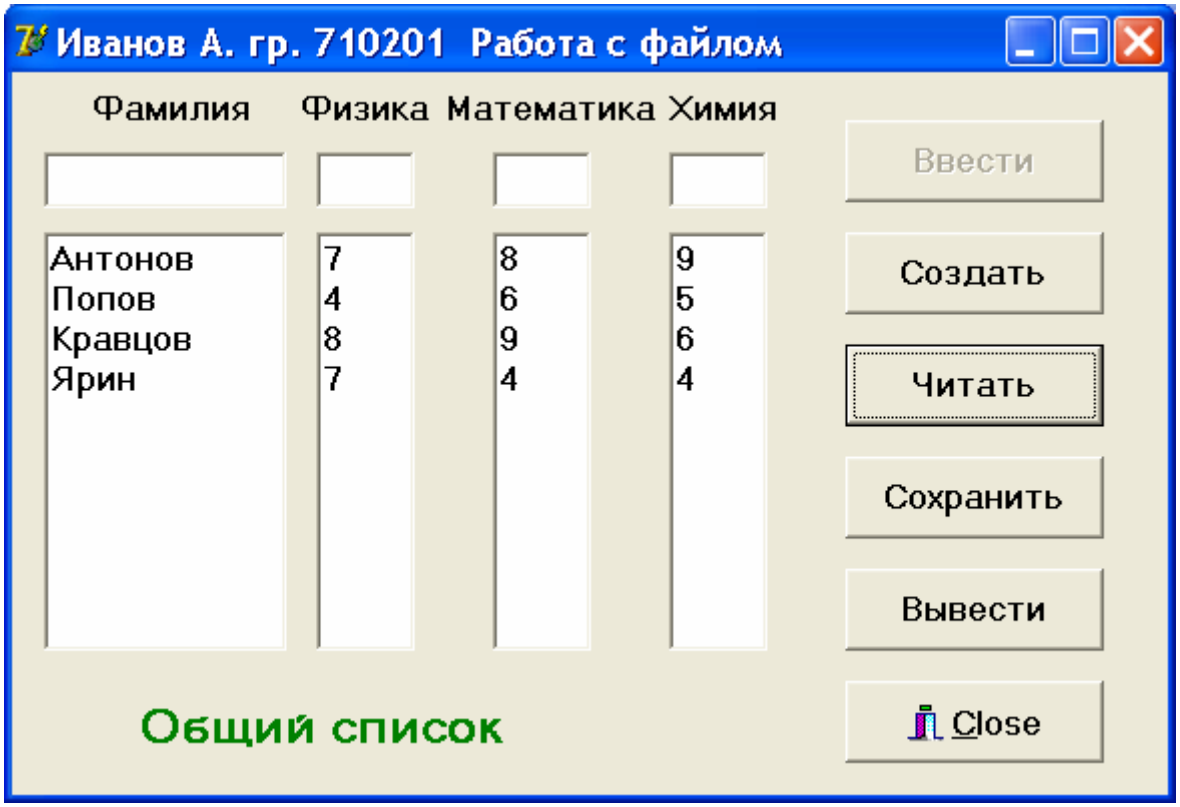


Рис. 8.1.

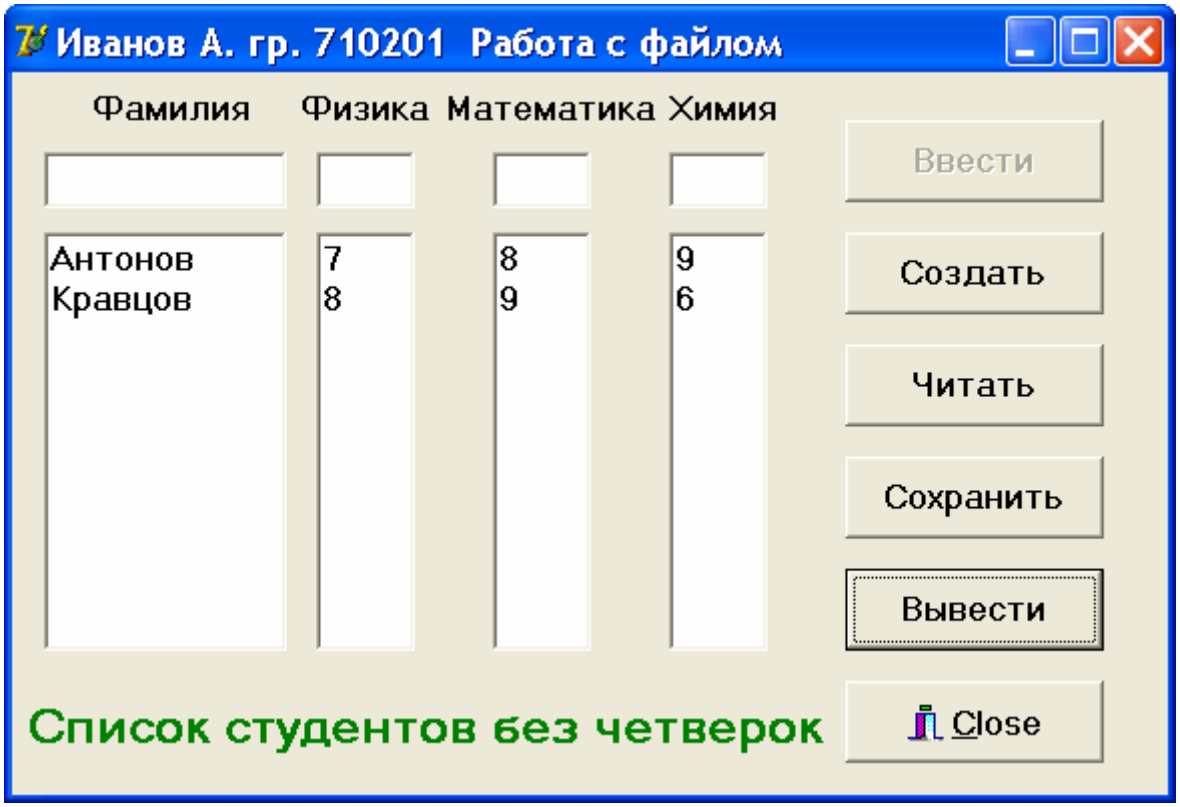
Для того, чтобы файл автоматически записывался, например, с расширением *dat*, в свойстве DefaultExt компонета SaveDialog запишем требуемое расширение – .dat (для текстового файла – .txt).

8.3. Пример написания программы

Задание: написать программу, содержащую режимы ***создания*** типизированного файла (каждая запись содержит фамилию студента и оценки по физике, математике и химии), ***чтения*** ранее созданного файла, ***записи*** содержимого в текстовый файл и ***вывода*** списка студентов, не имеющих четверок.



Вид формы после нажатия кнопки «Читать».



Вид формы после нажатия кнопки «Вывести».

Текст программы приведен на Листинге 8.1.

Листинг 8.1.

```
    unit Unit1;
                  interface
uses  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,Dialogs, StdCtrls, Buttons;
type
  TForm1 = class(TForm)
    OpenDialog1: TOpenDialog;
    SaveDialog1: TSaveDialog;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    Button5: TButton;
    BitBtn1: TBitBtn;
    Memo1: TMemo;
    Memo2: TMemo;
    Memo3: TMemo;
    Memo4: TMemo;
    Label5: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
type stud=record
       fam:string[20];
       oc:array[1..3] of byte;
       end;
var
  Form1: TForm1;
  f:file of stud;
  ft:textfile;
  w:stud;
  fname,fnamet:string;
  zak:boolean;

       implementation

{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
begin
 memo1.Clear; memo2.Clear; memo3.Clear; memo4.Clear;
 edit1.clear; edit2.clear; edit3.clear; edit4.clear;
 button1.Enabled:=false;
 label5.Caption:='';
 zak:=false;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin       // Ввести
   w.fam:=edit1.text;
   w.oc[1]:=strtoint(edit2.text);
   w.oc[2]:=strtoint(edit3.text);
   w.oc[3]:=strtoint(edit4.text);
   write(f,w);
   memo1.lines.add(w.fam);
   memo2.lines.add(inttostr(w.oc[1]));
   memo3.lines.add(inttostr(w.oc[2]));
   memo4.lines.add(inttostr(w.oc[3]));
   edit1.clear; edit2.clear; edit3.clear; edit4.clear;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin    //  Создать
  SaveDialog1.Title:='Создать файл';
  SaveDialog1.DefaultExt:='.dat';
  if SaveDialog1.Execute then
```

```
      begin
      fname:=SaveDialog1.FileName;
      AssignFile(f,fname);
      Rewrite(f);
      end;
   button1.Enabled:=true;
    zak:=true;
   label5.Caption:='        Общий список';
  end;

procedure TForm1.Button3Click(Sender: TObject);
begin       // Читать
  OpenDialog1.Title:='Открыть файл';
  if OpenDialog1.Execute then
      begin
      fname:=OpenDialog1.FileName;
      AssignFile(f,fname);
      Reset(f);
      end;
  memo1.clear; memo2.clear; memo3.clear; memo4.clear;
  while not eof(f) do
     begin
     read(f,w);
     memo1.lines.add(w.fam);
     memo2.lines.add(inttostr(w.oc[1]));
     memo3.lines.add(inttostr(w.oc[2]));
     memo4.lines.add(inttostr(w.oc[3]));
     end;
  closeFile(f);
  label5.Caption:='        Общий список';
end;

procedure TForm1.Button4Click(Sender: TObject);
begin   // Сохранить
  SaveDialog1.Title:='Сохранить в текстовом файле';
  SaveDialog1.DefaultExt:='.txt';
  if SaveDialog1.Execute then
      begin
      fnamet:=SaveDialog1.FileName;
      AssignFile(ft,fnamet);
      Rewrite(ft);
      end;
  writeln(ft,'        Общий список студентов');
  writeln(ft,'   Фамилия  Физика  Математика  Химия ');
```

```
  reset(f);
  while not eof(f) do
     begin
     read(f,w);
     with w do
        writeln(ft,fam:10,oc[1]:6,oc[2]:10,oc[3]:9);
     end;
  closeFile(f);
  closeFile(ft);
end;

procedure TForm1.Button5Click(Sender: TObject);
   var k,m:integer;
begin    //  Вывести
  memo1.clear; memo2.clear; memo3.clear; memo4.clear;
  reset(f);
  while not eof(f) do
     begin
     read(f,w);
     m:=0;
     for k:=1 to 3 do
        if w.oc[k]=4 then m:=1;
     if m=0 then
          begin
          memo1.lines.add(w.fam);
          memo2.lines.add(inttostr(w.oc[1]));
          memo3.lines.add(inttostr(w.oc[2]));
          memo4.lines.add(inttostr(w.oc[3]));
          end;
     end;
   closeFile(f);
   label5.Caption:='Список студентов без четверок';
end;

procedure TForm1.BitBtn1Click(Sender: TObject);
begin
  if zak then closeFile(f);
end;

end.
```

8.4. Индивидуальные задания

Необходимый теоретический материал и примеры программ приведены в лекции «Файлы». В программе предусмотреть сохранение вводимых данных в файле и возможность чтения из ранее сохраненного файла. Результаты выводить в окно просмотра и в текстовой файл.

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

2. Список товаров, имеющихся на складе, включает в себя наименование товара, количество и дату поступления товара на склад. Вывести в алфавитном порядке список товаров, хранящихся больше месяца.

3. Для получения места в общежитии формируется список студентов, который включает фамилию студента, 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 задается пункт назначения, время отправления и прибытия. Вывести информацию о рейсах, которыми можно воспользоваться для прибытия в пункт назначения раньше заданного времени.

5. Информация о сотрудниках фирмы включает: фамилию и количество проработанных часов за месяц. Рабочее время свыше 144 часов считается сверхурочным и оплачивается в двойном размере. Ввести почасовой тариф и вывести размер заработной платы каждого сотрудника фирмы за вычетом подоходного налога, который составляет 12% от суммы заработка.

6. Информация об участниках спортивных соревнований содержит: название команды, фамилию игрока и возраст. Вывести информацию о самой молодой команде.

7. Для книг, хранящихся в библиотеке, задаются: автор, название, год издания. Вывести список книг с фамилиями авторов в алфавитном порядке, изданных после заданного года.

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

9. Информация о сотрудниках предприятия содержит: фамилию, номер отдела и дату начала работы. Вывести списки сотрудников по отделам в порядке убывания стажа.

10. Ведомость абитуриентов, сдавших вступительные экзамены в университет, содержит: фамилию, адрес, три оценки. Определить количество абитуриентов, проживающих в г. Минске и сдавших экзамены со средним баллом не ниже 4.5, вывести их фамилии в алфавитном порядке.

11. В справочной аэропорта хранится расписание вылета самолетов на следующие сутки. Для каждого рейса указаны: номер рейса, пункт назначения, время вылета. Вывести все номера рейсов и времена вылета для заданного пункта назначения в порядке возрастания времени вылета.

12. У администратора железнодорожных касс хранится информация о свободных местах в поездах дальнего следования в следующем виде: пункт назначения, время отправления, число свободных мест. Оргкомитет международной конференции обращается к администратору с просьбой зарезервировать *заданное* количество мест до *заданного* города с временем отправления поезда не позднее *заданного*. Вывести время отправления или сообщение о невозможности выполнить заказ в полном объеме.

13. Ведомость абитуриентов, сдавших вступительные экзамены в университет, содержит: фамилию абитуриента, три оценки. Определить средний балл по университету и вывести список абитуриентов, средний балл которых выше среднего балла по университету.

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

15. На междугородной АТС информация о разговорах содержит название города, время разговора и номер телефона абонента. Ввести поминутный тариф и вывести по каждому городу общее время разговоров с ним и сумму.

ТЕМА 9. ПРОГРАММИРОВАНИЕ С ОТОБРАЖЕНИЕМ ГРАФИЧЕСКОЙ ИНФОРМАЦИИ

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

9.1. Как рисуются изображения

Нарисовать картинку в среде Delphi можно на многих компонентах (например, на форме, на TPaintBox), однако наиболее удобно использовать компонент TImage (страница Additional). Нарисованную в Image1 картинку можно перенести в отчет, используя процедуру `ClipBoard.Assign(Image1.Picture)` (модуль Clipbrd). Для рисования используют класс TCanvas, который является свойством многих компонентов, и представляет собой прямоугольный холст в виде матрицы из пикселей и набор инструментов для рисования на нем. Каждый пиксел имеет координтау (x, y), где x – порядковый номер пиксела, начиная от левой границы холста, а y – порядковый номер пиксела, начиная от верхней границы холста. Левый верхний угол холста имеет координату (0, 0), а правый `(Image1.Width -1, Image1.Height - 1)`.

9.2. Как строится график с помощью компонента TChart

Обычно результаты расчетов представляются в виде графиков и диаграмм. Система DELPHI имеет мощный пакет стандартных программ вывода на экран и редактирования графической информации, который реализуется с помощью визуально отображаемого на форме компонента TChart. Построение графика (диаграммы) производится после вычисления таблицы значений функции y=f(x). Полученная таблица передается в специальный двумерный массив Chart**I**.SeriesList[**k**] (**k** – номер графика (0,1,2,...)) компонента TChart с помощью метода AddXY. Компонент TChart осуществляет всю работу по отображению графиков, переданных в объект Chart**I**.SeriesList[**k**]**:** строит и размечает оси, рисует координатную сетку, подписывает название осей и самого графика, отображает переданную таблицу в виде всевозможных графиков или диаграмм. При необходимости, с помощью встроенного редактора EditingChart компоненту TChart передаются данные о толщине, стиле и цвете линий, параметрах шрифта подписей, шагах разметки координатной сетки и другие настройки. В процессе работы программы изменение параметров возможно через обращение к соответствующим свойствам компонента TChart. Так, например, свойство Chart1.BottomAxis содержит значение максимального предела нижней оси графика. Перенести график в отчет можно через буфер обмена, используя процедуру Chart**I**.CopyToClipboardMetafile(True). Для изменения параметров компонента

TChart необходимо дважды щелкнуть по нему мышью в окне формы. Появится окно редактирования EditingChat1 (рис. 9.1). Для создания нового объекта Series1 щелкнуть по кнопке Add на странице Series. В появившемся диалоговом окне TeeChart Gallery выбрать пиктограмму с надписью Line (график выводится в виде линий). Если нет необходимости представления графика в трехмерном виде, отключить независимый переключатель 3D. После нажатия на кнопку OK появится новая серия с название Series1. Для изменения названия нажать кнопку Title. Закладка Legend задает список обозначений даиграммы (можно убирать с экрана). Название графика вводится на странице Titles. Разметка осей меняется на странице Axis. Страница Series задает характеристики (цвет, толщина линий) для определенного графика. Нажимая различные кнопки меню, познакомтесь с другими возможностями EditingChat.

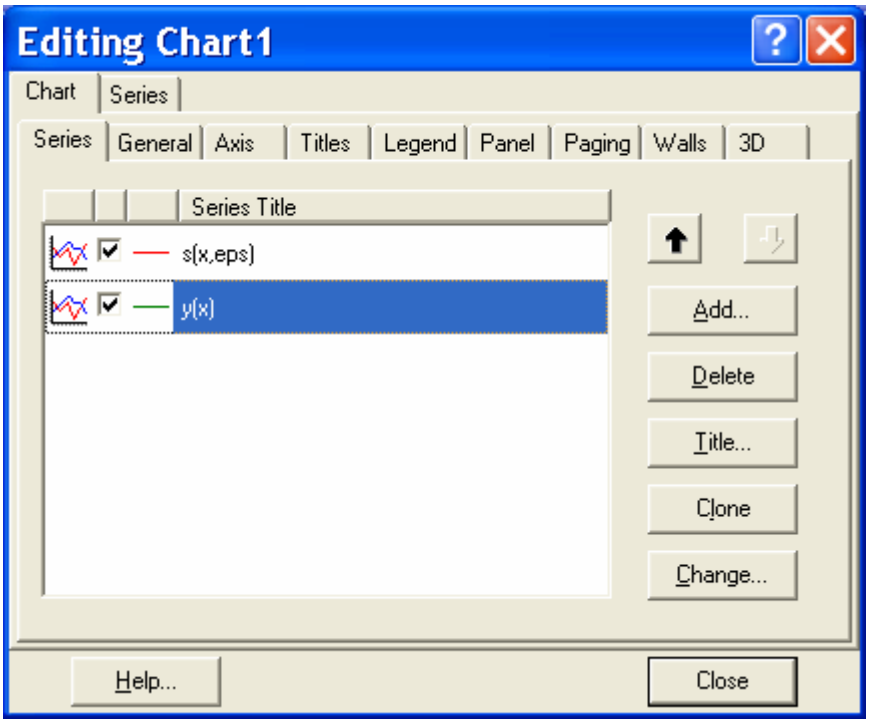


Рис. 9.1.

9.3. Индивидуальные задания

Необходимый теоретический материал и примеры программ приведены в лекции «Графика». Постройте графики двух функций s(x) и y(x) для вариантов из темы 3. Таблицу данных получить, задав интервал и шаг таблицы. Ввод исходных данных организовать через окна TEdit. Самостоятельно выбрать удобные параметры настройки.

По указанию преподавателя выберите вариант задачи. Решите задачу, и используя функции класса TCanvas нарисуйте соответствующие геометрические фигуры. Расположите все рисунки в центре TImage, так чтобы

они занимали 2/3 области окна. Все исходные данные имеют действительный тип. Используйте масштабирование.

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

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

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

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

5. Определить количество точек с целочисленными координатами, лежащих внутри окружности радиуса *R* с центром в точке (x_0, y_0).

6. Найти координаты точек пересечения двух окружностей радиуса R_1 и R_2 с центрами в точках (x_1, y_1) и (x_2, y_2) соответственно.

7. Найти координаты точки, симметричной данной точке *M* с координатами (x_1, y_1) относительно прямой $Ax+By+C=0$.

8. Даны две точки $M_1(x_1, y_1)$, $M_2(x_2, y_2)$ и прямая $Ax+By+C=0$. Необходимо найти на этой прямой такую точку $M_0(x_0, y_0)$, чтобы суммарное расстояние от нее до двух данных точек было минимально.

9. Даны три точки с координатами (x_1, y_1), (x_2, y_2), (x_3, y_3), которые являются вершинами некоторого прямоугольника со сторонами, параллельными осям координат. Найти координаты четвертой точки.

10. Даны координаты четырех точек (x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4). Необходимо определить, образуют ли они выпуклый четырехугольник.

11. Даны координаты четырех точек (x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4). Необходимо определить, образуют ли они: а) ромб; б) квадрат; в) трапецию.

12. Даны координаты двух вершин (x_1, y_1) и (x_2, y_2). некоторого квадрата. Необходимо найти возможные координаты других его вершин.

13. Даны координаты двух вершин (x_1, y_1) и (x_2, y_2) некоторого квадрата, которые расположены на диагонали, и точка (x_3, y_3). Необходимо определить, лежит или не лежит точка внутри квадрата.

14. Даны координаты трех вершин (x_1, y_1), (x_2, y_2), (x_3, y_3) треугольника. Необходимо найти координаты точки пересечения его медиан.

15. Даны координаты трех вершин (x_1, y_1), (x_2, y_2), (x_3, y_3) треугольника. Необходимо найти длины его высот.

ТЕМА 10. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ РЕКУРСИИ

Цель лабораторной работы: изучить способы программирования алгоритмов с использованием рекурсии.

10.1. Понятие рекурсии

Рекурсия – это такой способ организации вычислительного процесса, при котором подпрограмма в ходе выполнения составляющих ее операторов обращается сама к себе.

10.2. Пример решения задачи

Написать программу вычисления максимального значения массива и квадратного корня с использованием рекурсии и без. Листинг программы приведен ниже.

Листинг 10.1

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

```
    StringGrid1: TStringGrid;
    Edit1: TEdit;
    BitBtn1: TBitBtn;
    Button2: TButton;
    Label1: TLabel;
    Button3: TButton;
    Label2: TLabel;
    Button5: TButton;
    Edit2: TEdit;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

type vek=array[1..50] of byte;
var  Form1: TForm1;
     a:vek;
     n,k:integer;
                implementation
{$R *.dfm}

function maxRn( x:vek; n:integer ):byte;
 begin
  if n=1 then result:=x[1]
         else
          if maxRn(x,n-1)>x[n] then result:=maxRn(x,n-1)
                               else result:=x[n];
 end;

function max( x:vek; n:integer ):byte;
  begin
    result:=x[1];
    for k:=1 to n do
      if x[k]>result then result:=x[k];
  end;
```

```
function sqrtR( x:extended; n:integer  ):extended;
 begin
   if n=0 then result:=x
          else result:=0.5*(sqrtR(x,n-1)+x/sqrtR(x,n-1));
 end;

function sqrtA( x:extended; n:integer  ):extended;
 begin
   result:=x;
   for k:=1 to n do
       result:=0.5*(result+x/result);
 end;

procedure TForm1.FormCreate(Sender: TObject);
 begin
   n:=5;  edit1.Text:=inttostr(n);
   StringGrid1.colcount:=n;
   randomize;
   for k:=0 to n-1 do
     StringGrid1.cells[k,0]:=inttostr(random(100));
 end;

procedure TForm1.Button2Click(Sender: TObject);
  begin   // Изменить
    n:=strtoint(edit1.Text);
    StringGrid1.colcount:=n;
    randomize;
    for k:=0 to n-1 do
       StringGrid1.cells[k,0]:=inttostr(random(100));
  end;

procedure TForm1.Button3Click(Sender: TObject);
  begin
    for k:=1 to n do
     a[k]:=strtoint( StringGrid1.cells[k-1,0]);
    label1.Caption:='Максимальный элемент '+
       inttostr(maxRn(a,n));
    label2.Caption:='Максимальный элемент '+
       inttostr(max(a,n))+'  рек';
  end;


procedure TForm1.Button5Click(Sender: TObject);
  begin
```

```
label5.Caption:='sqrt(x)='+
floattostrf(sqrtR(strtofloat(edit2.Text),n),fffixed,9,5)
       +'  рек';
 label6.Caption:='sqrt(x)= '+
floattostrf(sqrtA(strtofloat(edit2.Text),n),fffixed,9,5);
end;

end.
```

10.3. Индивидуальные задания

Необходимый материал и примеры приведены в лекции «Рекурсия». Решить поставленные задачи двумя способами – с применением рекурсии и без нее.

1. Вычислить факториал n!=1*2*3*…*(n-1)*n, 0!=1.

2. В упорядоченном массиве целых чисел a_i, $i=1...n$ найти номер элемента ***c*** методом двоичного поиска, используя очевидное соотношение: если $c \le a_{n/2}$, тогда $c \in [a_1 ... a_{n/2}]$ иначе $c \in [a_{n/2+1} ... a_n]$. Если элемент c отсутствует в массиве,то вывести соответствующее сообщение.

3. Найти наибольший общий делитель чисел *M* и *N*. Используйте теорему Эйлера (алгоритм Евклида): Если *M* делится на *N,* то *НОД (N, M)=N,* иначе *НОД (N, M)=НОД (M mod N, N)*.

4. Вычислить число Фибоначчи *Fb(n)*. Числа Фибоначчи определяются следующим образом: *Fb(0)=1; Fb(1)=1; Fb(n)=Fb(n-1)+Fb(n-2)*.

5. Найти сумму элементов массива .

6. Вычислить значение полинома степени n по схеме, $P_n = \sum_{i=0}^{n} a_i x^i = a_0 + x\left(a_1 + ... x\left(a_{n-1} + x a_n\right)...\right)$.

7. Вычислить значение $x = \sqrt{a}$, используя рекуррентную формулу $x_n = \frac{1}{2}\left(x_{n-1} + a / x_{n-1}\right)$, в качестве начального приближения использовать значение $x_0=0.5(1+a)$.

8. Найти максимальный элемент в массиве $a_1...a_n$, используя очевидное соотношение $max\ (a_1...a_n)=max\ (max\ (a_1...a_{n-1}),\ a_n)$.

9. Найти максимальный элемент в массиве $a_1...a_n$, используя соотношение (метод деления пополам) $max\ (a_1...a_n)=max\ (max\ (a_1...a_{n/2}),\ max\ (a_{n/2+1},\ a_n))$.

10. Вычислить $y(n) = \sqrt{n + \sqrt{n-1 + \sqrt{n-2 + ... + \sqrt{2 + \sqrt{1}}}}}$.

11. Вычислить $y(n)=\cfrac{1}{n+\cfrac{1}{(n-1)+\cfrac{1}{(n-2)+\cfrac{1}{.. ..}}}}$

$$....+\cfrac{1}{1+\cfrac{1}{2}}.$$

12. Вычислить произведение n≥2 (n-четное) сомножителей $y=\frac{2}{1}\cdot\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{5}\cdot\frac{6}{7}\cdot....$

13. Вычислить $y=x^N$ по следующему алгоритму: $y=(x^{N/2})^2$, если N четное и $y=x\cdot x^{N-1}$, если N нечетное.

14. Вычислить $C_n^m=\frac{n!}{m!(n-m)!}$, $m\le n$.

15. Найти значение функции Аккермана *A(m, n)*, которая определяется для всех неотрицательных целых аргументов m и n следующим образом:

A(o, n)=n+1;

A(m, o)=A(m-1, 1); (m>o); A(m, n)=A(m-1, A(m, n-1)); (m>o; n>o).

ТЕМА 11. РАБОТА СО СПИСКАМИ НА ОСНОВЕ ДИНАМИЧЕСКИХ МАССИВОВ

Цель лабораторной работы: изучить способы работы с динамическими массивами данных и организацию работы со списками, используя указатели.

11.1. Работа со списками

Необходимый теоретический материал и примеры программ приведены в лекциях «Поиск и сортировка» и «Списки на основе динамических массивов».

11.2. Индивидуальные задания

Организовать работу со списками записей на основе динамического массива. Индивидуальные варианты типов записей использовать те же, что и в лабораторной работе № 8. Обеспечить добавление и удаление элементов в массив на заданную позицию, сортировку и поиск по ключу.

ТЕМА 12. СПИСОК НА ОСНОВЕ РЕКУРСИВНЫХ ДАННЫХ В ВИДЕ СТЕКА

Цель лабораторной работы: получить навыки программирования однонаправленных списков в виде стека.

12.1. Работа со списками

Необходимый теоретический материал и примеры программ приведены в лекции «Связанные списки».

12.2. Индивидуальные задания

Организовать стек. Обеспечить добавление и удаление элементов в стек, предусмотреть возможность сортировки рекурсивным методом слияния.

1. Создать стек со случайными целыми числами в диапазоне –50 до +50 и преобразовать его в два стека. Первый должен содержать только положительные числа, а второй - отрицательные. Порядок чисел должен быть сохранен, как в первом стеке.
2. Создать стек из случайных целых чисел и удалить из него записи с четными числами.
3. Создать стек из случайных целых чисел в диапазоне от –10 до 10 и удалить из него записи с отрицательными числами.
4. Создать стек из случайных целых чисел и поменять местами крайние элементы.
5. Подсчитать, сколько элементов стека, построенного из случайных чисел, превышает среднее значение от всех элементов стека.
6. Создать стек из случайных целых чисел и найти в нем максимальное и минимальное значение.
7. Создать стек из случайных целых чисел и определить, сколько элементов стека, начиная с вершины, находится до элемента с максимальным значением.
8. Создать стек из случайных целых чисел и определить, сколько элементов стека, начиная от вершины, находится до элемента с минимальным значением.
9. Создать стек из случайных чисел и определить, сколько элементов стека находится между минимальным и максимальным элементами.
10. Создать стек из случайных чисел и определить, сколько элементов стека имеют значения меньше среднего значения от всех элементов стека.
11. Создать стек из случайных чисел и поменять местами минимальный и максимальный элементы.

12. Создать стек из случайных целых чисел и из него сделать еще два стека. В первый поместить все четные, а во второй - нечетные числа.
13. Создать стек из случайных целых чисел в диапазоне от 1 до 10 и определить наиболее часто встречающееся число.
14. Создать стек из случайных целых чисел и удалить из него каждый второй элемент.
15. Создать стек из случайных целых чисел и получить из него транспонированный.

ТЕМА 13. ОРГАНИЗАЦИЯ ОДНОНАПРАВЛЕННЫХ СПИСКОВ В ВИДЕ ОЧЕРЕДИ

Цель лабораторной работы: получить навыки программирования однонаправленных списков, организованных в виде очереди.

13.1. Работа со списками

Необходимый материал и примеры программ приведены в лекции «Связанные списки».

13.2. Индивидуальные задания

Организовать однонаправленный список. Обеспечить добавление и удаление элементов в начало и конец очереди, предусмотреть возможность сортировки рекурсивным методом слияния.

1. Создать стек со случайными целыми числами в диапазоне –50 до +50 и преобразовать его в два стека. Первый должен содержать только положительные числа, а второй - отрицательные. Порядок чисел должен быть сохранен, как в первом стеке.
2. Создать стек из случайных целых чисел и удалить из него записи с четными числами.
3. Создать стек из случайных целых чисел в диапазоне от –10 до 10 и удалить из него записи с отрицательными числами.
4. Создать стек из случайных целых чисел и поменять местами крайние элементы.
5. Подсчитать, сколько элементов стека, построенного из случайных чисел, превышает среднее значение от всех элементов стека.
6. Создать стек из случайных целых чисел и найти в нем максимальное и минимальное значение.
7. Создать стек из случайных целых чисел и определить, сколько элементов стека, начиная с вершины, находится до элемента с максимальным значением.
8. Создать стек из случайных целых чисел и определить, сколько элементов стека, начиная от вершины, находится до элемента с минимальным значением.
9. Создать стек из случайных чисел и определить, сколько элементов стека находится между минимальным и максимальным элементами.
10. Создать стек из случайных чисел и определить, сколько элементов стека имеют значения меньше среднего значения от всех элементов стека.
11. Создать стек из случайных чисел и поменять местами минимальный и максимальный элементы.

12. Создать стек из случайных целых чисел и из него сделать еще два стека. В первый поместить все четные, а во второй - нечетные числа.
13. Создать стек из случайных целых чисел в диапазоне от 1 до 10 и определить наиболее часто встречающееся число.
14. Создать стек из случайных целых чисел и удалить из него каждый второй элемент.
15. Создать стек из случайных целых чисел и получить из него транспонированный.

ТЕМА 14. ОРГАНИЗАЦИЯ ДВУНАПРАВЛЕННЫХ СПИСКОВ В ВИДЕ ОЧЕРЕДИ

Цель лабораторной работы: получить навыки программирования двунаправленных списков, организованных в виде очереди.

14.1. Работа со списками

Необходимый теоретический материал и примеры программ приведены в лекции «Связанные списки».

14.2. Индивидуальные задания

Организовать двунаправленный список. Обеспечить добавление и удаление элементов в начало и конец очереди, предусмотреть возможность сортировки рекурсивным методом слияния, вывод списка с начала и конца очереди.

1. Создать стек со случайными целыми числами в диапазоне –50 до +50 и преобразовать его в два стека. Первый должен содержать только положительные числа, а второй - отрицательные. Порядок чисел должен быть сохранен, как в первом стеке.
2. Создать стек из случайных целых чисел и удалить из него записи с четными числами.
3. Создать стек из случайных целых чисел в диапазоне от –10 до 10 и удалить из него записи с отрицательными числами.
4. Создать стек из случайных целых чисел и поменять местами крайние элементы.
5. Подсчитать, сколько элементов стека, построенного из случайных чисел, превышает среднее значение от всех элементов стека.
6. Создать стек из случайных целых чисел и найти в нем максимальное и минимальное значение.
7. Создать стек из случайных целых чисел и определить, сколько элементов стека, начиная с вершины, находится до элемента с максимальным значением.
8. Создать стек из случайных целых чисел и определить, сколько элементов стека, начиная от вершины, находится до элемента с минимальным значением.
9. Создать стек из случайных чисел и определить, сколько элементов стека находится между минимальным и максимальным элементами.
10. Создать стек из случайных чисел и определить, сколько элементов стека имеют значения меньше среднего значения от всех элементов стека.

11. Создать стек из случайных чисел и поменять местами минимальный и максимальный элементы.
12. Создать стек из случайных целых чисел и из него сделать еще два стека. В первый поместить все четные, а во второй - нечетные числа.
13. Создать стек из случайных целых чисел в диапазоне от 1 до 10 и определить наиболее часто встречающееся число.
14. Создать стек из случайных целых чисел и удалить из него каждый второй элемент.
15. Создать стек из случайных целых чисел и получить из него транспонированный.

ТЕМА 15. МЕТОДЫ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

Цель лабораторной работы: изучить основные методы и алгоритмы решения систем линейных алгебраических уравнений и их реализацию в среде DELPHI.

15.1. Методы и алгоритмы решения систем линейных алгебраических уравнений

Необходимый теоретический материал приведен в лекции «Алгоритмы решения систем линейных алгебраических уравнений». Основные схемы алгоритмов приведены ниже.

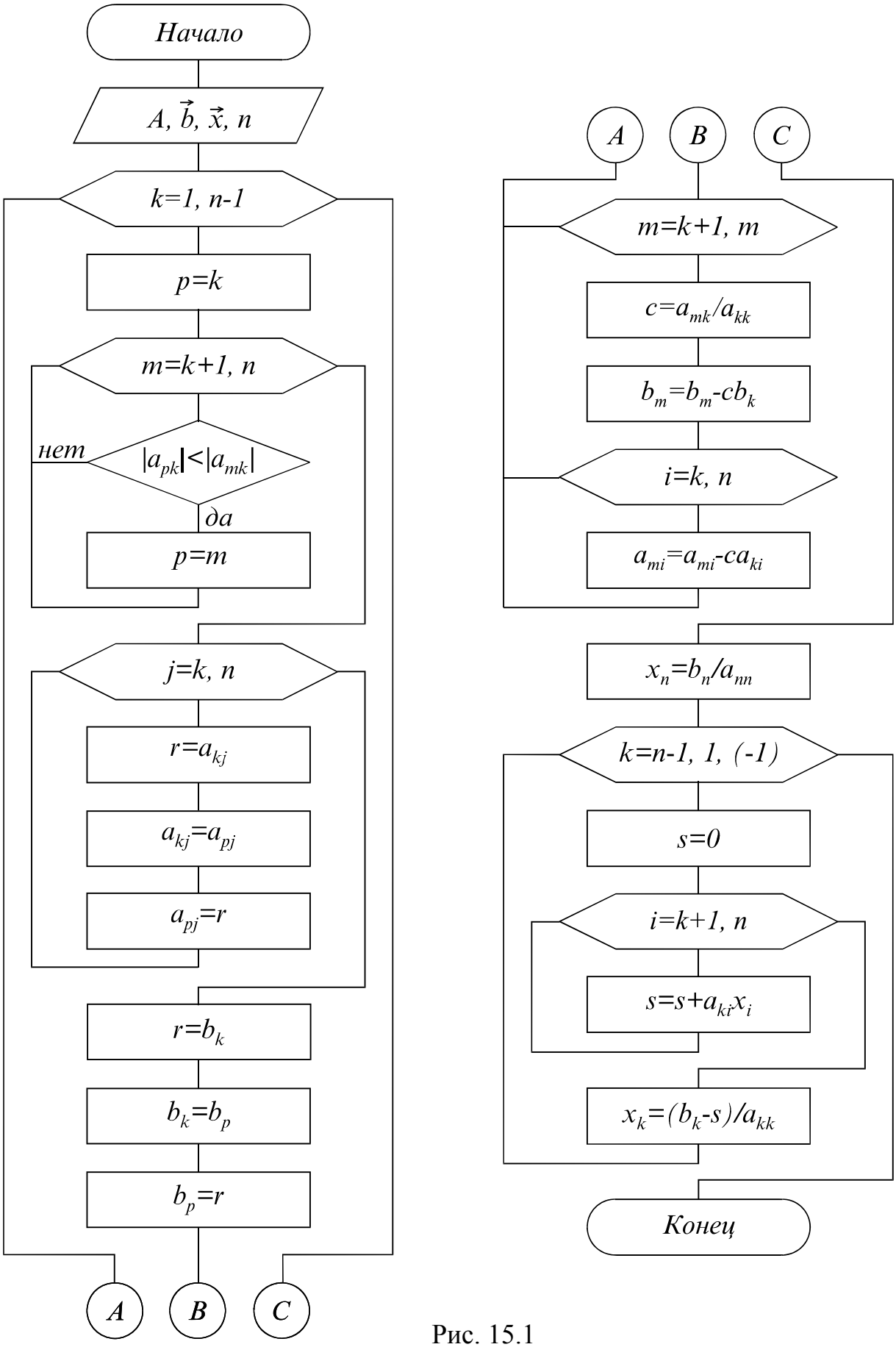


Рис. 15.1
Схема алгоритма метода Гаусса

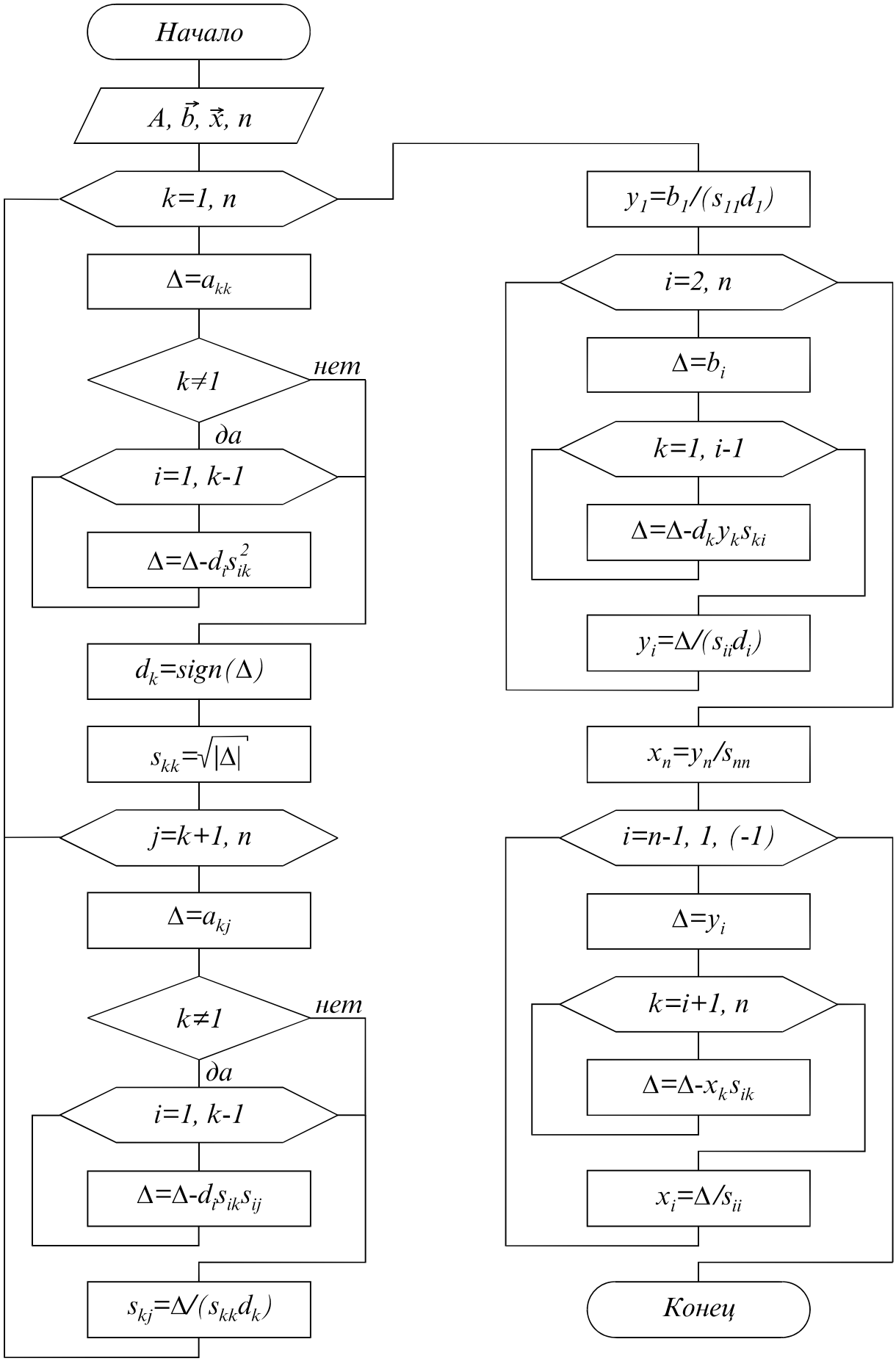


Рис. 15.2 Схема алгоритма метода квадратного корня

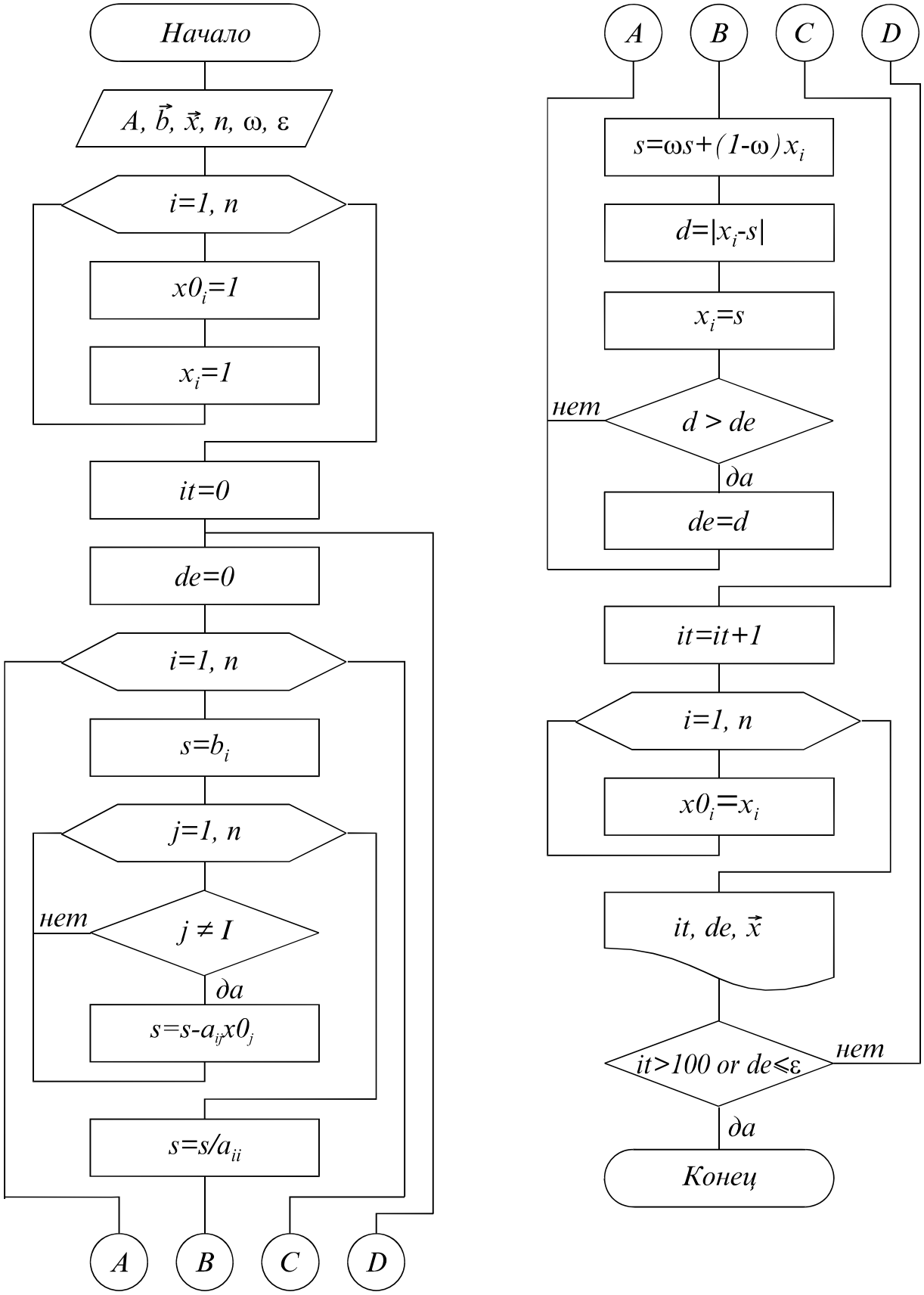
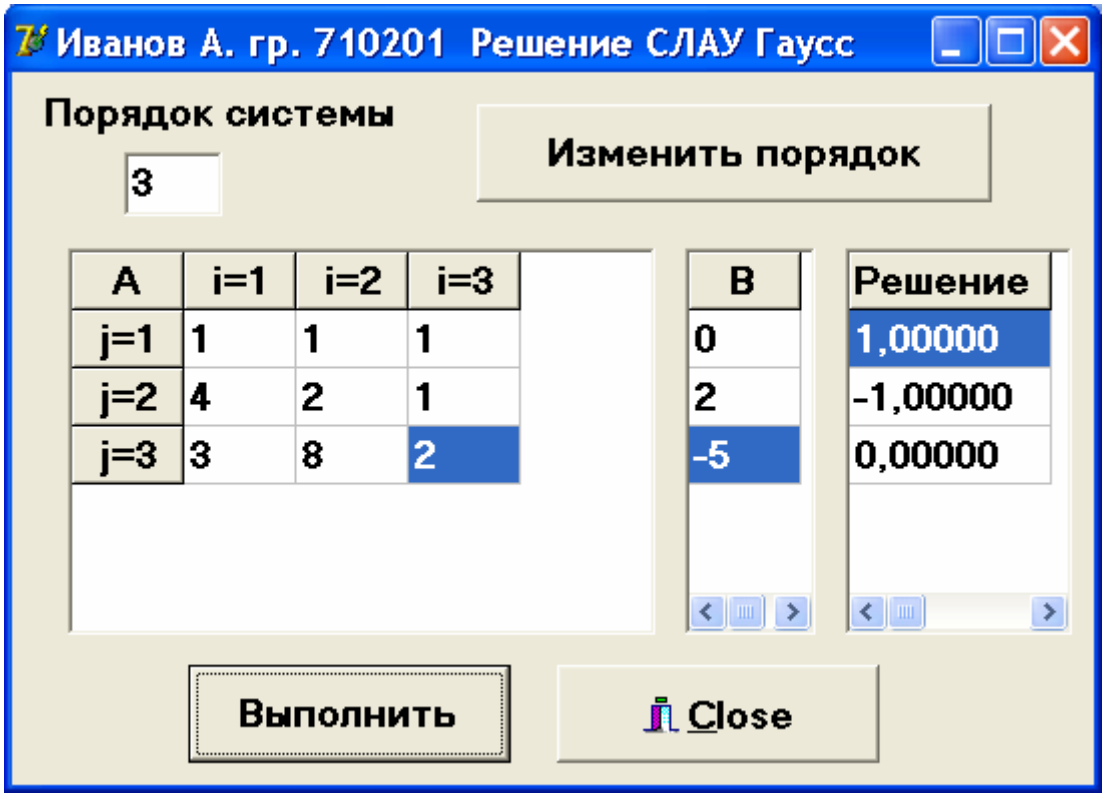


Рис. 15.3 Схема алгоритма метода простой итерации

15.2. Пример написания программы

Изображение формы и текст программы решения системы линейных алгебраических уравнений методом Гаусса приведены ниже. Сам метод Гаусса оформлен в виде процедуры Gauss(a,b,n,x), помещенной в библиотечный модуль.



Листинг 15.1

```
unit biblio;
            interface
  type mat=array[1..30,1..30] of extended;
       vek=array[1..30] of extended;
  procedure Gauss(a:mat; b:vek; n:integer; var x:vek);

              implementation
  procedure Gauss;
    var k,m,p,i,j:integer;
        r,c,s:extended;
    begin
      for k:=1 to n-1 do
        begin
        p:=k;
        for m:=k+1 to n do
```

```
          if abs(a[p,k])<abs(a[m,k]) then p:=m;
        for j:=k to n do
          begin
          r:=a[k,j];
          a[k,j]:=a[p,j];
          a[p,j]:=r;
          end;
        r:=b[k];
        b[k]:=b[p];
        b[p]:=r;
        for m:=k+1 to n do
          begin
          c:=a[m,k]/a[k,k];
          b[m]:=b[m]-c*b[k];
          for i:=k to n do
            a[m,i]:=a[m,i]-c*a[k,i];
          end;
      end;
      x[n]:=b[n]/a[n,n];
      for k:=n-1 downto 1 do
        begin
        s:=0;
        for i:=k+1 to n do
          s:=s+a[k,i]*x[i];
        x[k]:=(b[k]-s)/a[k,k];
        end;
  end;
end.
```

Листинг 15.2

```
unit Unit1;
           interface
uses
 Windows,Messages,SysUtils,Variants,Classes,Graphics,
 Controls,Forms,Dialogs,Buttons,Grids,StdCtrls,biblio;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Button1: TButton;
    Button2: TButton;
    StringGrid1: TStringGrid;
    StringGrid2: TStringGrid;
    StringGrid3: TStringGrid;
    BitBtn1: TBitBtn;
```

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

var
  Form1: TForm1;
  a:mat;
  b,x:vek;
  n,i,j:integer;
                  implementation
{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
begin
  n:=3; edit1.Text:=inttostr(n);
  stringgrid1.Cells[0,0]:='   A';
  stringgrid2.Cells[0,0]:='   B';
  stringgrid3.Cells[0,0]:='Решение';
  stringgrid1.RowCount:=n+1;
  stringgrid1.ColCount:=n+1;
  stringgrid2.RowCount:=n+1;
  stringgrid3.RowCount:=n+1;
  for i:=1 to n do
    begin
    stringgrid1.Cells[i,0]:='  i='+inttostr(i);
    stringgrid1.Cells[0,i]:='  j='+inttostr(i);
    end;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  n:=strtoint(edit1.Text);
  stringgrid1.RowCount:=n+1;
  stringgrid1.ColCount:=n+1;
  stringgrid2.RowCount:=n+1;
  stringgrid3.RowCount:=n+1;
  for i:=1 to n do
    begin
    stringgrid1.Cells[i,0]:='  i='+inttostr(i);
```

```
    stringgrid1.Cells[0,i]:='  j='+inttostr(i);
    end;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
  for i:=1 to n do
    for j:=1 to n do
      a[i,j]:=strtofloat(stringgrid1.Cells[j,i]);
  for i:=1 to n do
      b[i]:=strtofloat(stringgrid2.Cells[0,i]);
  Gauss(a,b,n,x);
  for i:=1 to n do
 stringgrid3.Cells[0,i]:=floattostrf(x[i],fffixed,9,5);
end;

end.
```

15.3. Индивидуальные задания

Составить программу решения СЛАУ *n*-го порядка одним из методов по указанию преподавателя. Вычисления оформить в виде подпрограммы, помещенной в библиотечный модуль

1. Метод Гаусса;
2. Метод квадратного корня;
3. Метод прогонки;
4. Метод простой итерации;
5. Метод Зейделя.

ТЕМА 16. АППРОКСИМАЦИЯ ФУНКЦИЙ. ВЫЧИСЛЕНИЕ ОПРЕДЕЛЕННЫХ ИНТЕГРАЛОВ

Цель лабораторной работы: изучить основные методы и алгоритмы аппроксимации функций и вычисления определенных интегралов и их реализацию в среде DELPHI.

16.1. Методы и алгоритмы аппроксимации функций и вычисления определенных интегралов

Необходимый теоретический материал приведен в лекциях «Аппроксимация функций» и «Вычисление производных и интегралов». Основные схемы алгоритмов приведены ниже.

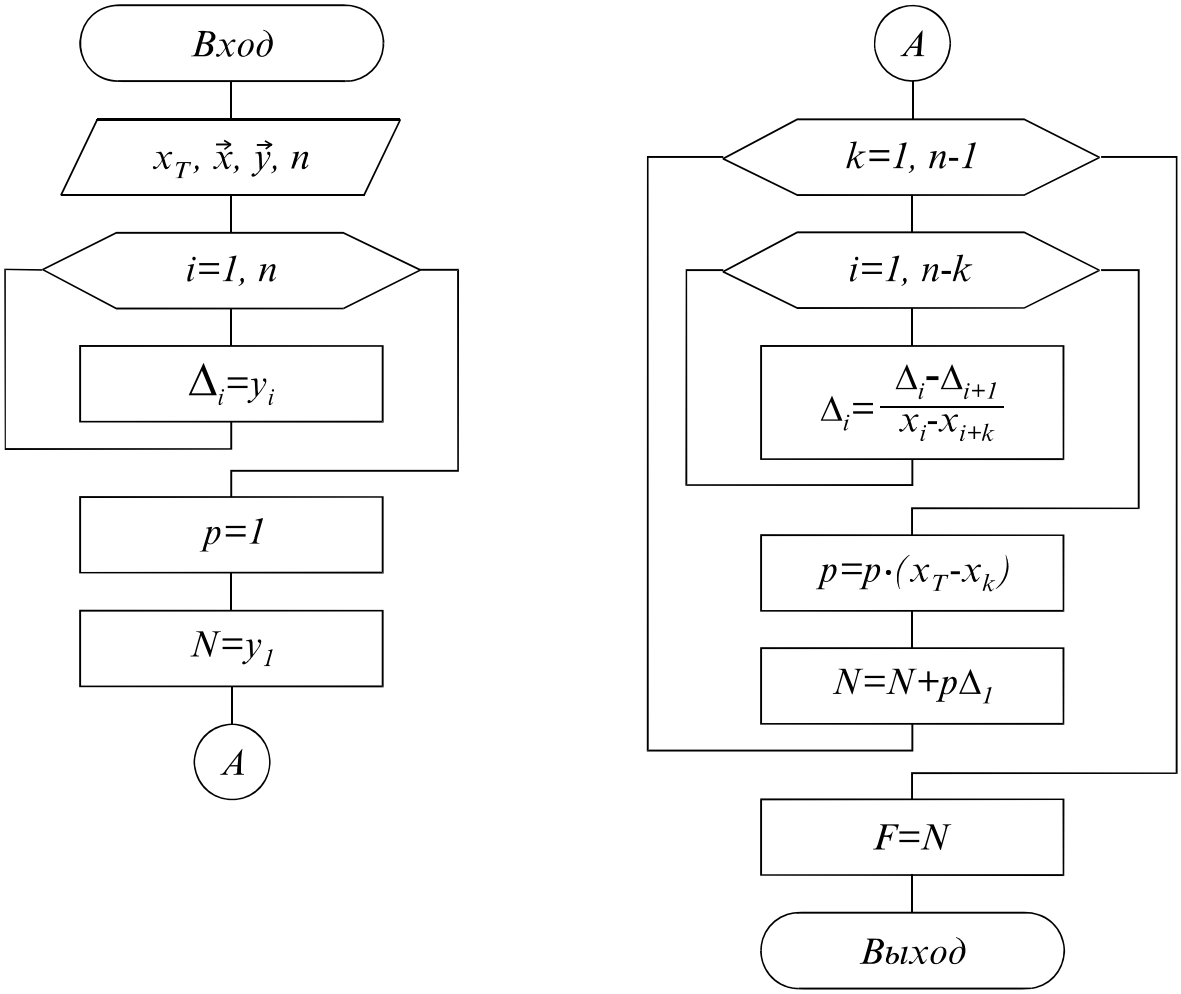


Рис. 16.1 Схема алгоритма расчета многочлена Ньютона

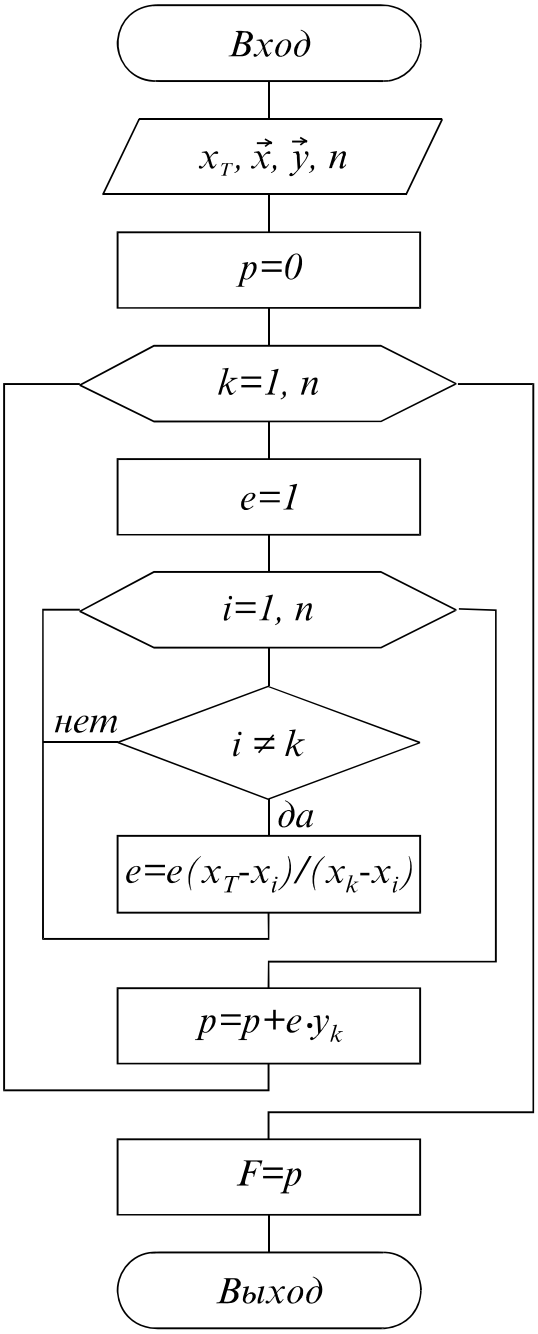


Рис 16.2. Схема алгоритма расчета интерполяционного полинома Лагранжа

16.2. Пример написания программы

Изображение формы и текст программы аппроксимации функции полиномом Лагранжа и вычисления определенного интеграла методом Гаусса с двумя узлами приведены ниже. Вычисления оформлены в виде процедур lag(xt,x,y,n) и Gauss2(a,b,m,f), помещенных в библиотечный модуль.

Листинг 16.1

```
unit biblio;
             interface
type vek=array[1..30] of extended;
     fun=function(x:extended):extended;
function lag(xt:extended; x,y:vek; n:integer):extended;
function gauss2(a,b:extended; n:integer;
                              f:fun):extended;

            implementation
  function lag;
    var i,k:integer;
        s,p:extended;
    begin
      s:=0;
      for k:=1 to n do
        begin
        p:=y[k];
```

```
        for i:=1 to n do
          if i<>k then p:=p*(xt-x[i])/(x[k]-x[i]);
        s:=s+p;
        end;
      result:=s;
    end;

  function gauss2;
     const c=0.5773502692;
     var x,h,s:extended;
         k:integer;
     begin
       h:=(b-a)/n;
       x:=a+h/2;  s:=0;
       for k:=1 to n do
         begin
         s:=s+f(x-h/2*c)+f(x+h/2*c);
         x:=x+h;
         end;
       result:=h*s/2;
     end;
end.
```

Листинг 16.2

```
unit Unit1;
                interface
uses
  Windows, Messages, SysUtils, Variants, Classes,
  Graphics, Controls, Forms,  Dialogs, TeEngine,
  Series, Buttons, StdCtrls, ExtCtrls, TeeProcs,
  Chart,biblio;
 type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Label2: TLabel;
    Chart1: TChart;
    Button1: TButton;
    Button2: TButton;
    BitBtn1: TBitBtn;
    Label3: TLabel;
    Series1: TLineSeries;
```

```
    Series2: TLineSeries;
    Edit4: TEdit;
    Label4: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}
  function f(x:extended):extended;
    begin
      f:=4*x-20*cos(3*x);
    end;

  procedure TForm1.Button1Click(Sender: TObject);
   var a,b,h,xt:extended; // Аппроксимировать
       n,k,m:integer;
       x,y:vek;
  begin
   a:=strtofloat(edit1.Text);
   b:=strtofloat(edit2.Text);
   n:=strtoint(edit3.Text);
   m:=strtoint(edit4.Text);
   h:=(b-a)/(n-1);
   for k:=1 to n do
     begin
     x[k]:=a+(k-1)*h;
     y[k]:=f(x[k]);
     end;
   series1.Clear; series2.Clear;
   for k:=0 to m do
     begin
     xt:=a+k*(b-a)/m;
     series1.AddXY(xt,f(xt),'',clred);
     series2.AddXY(xt,lag(xt,x,y,n),'',clblue);
     end;
```

```
  end;

procedure TForm1.Button2Click(Sender: TObject);
   var a,b:extended;  // Интеграл
       m:integer;
begin
   a:=strtofloat(edit1.Text);
   b:=strtofloat(edit2.Text);
   m:=strtoint(edit4.Text);
   label3.Caption:='Интеграл= '+
      floattostrf(gauss2(a,b,m,f),fffixed,9,6);
end;

end.
```

16.3. Индивидуальные задания

Составить программу аппроксимации функции *f(x)* и вычисления определенного интеграла одним из методов по указанию преподавателя. Во всех вариантах требуется аппроксимировать заданную исходную функцию *f(x)* многочленом на интервале *[a, b]*. Задано количество неизвестных параметров *n*, вид аппроксимации и *m* - количество точек, в которых задана функция. Вычисления оформить в виде подпрограммы, помещенной в библиотечный модуль. Провести исследование факторов, влияющих на точность аппроксимации.

1. Аппроксимация общего вида
2. Аппроксимация полиномом Лагранжа
3. Аппроксимация полиномом Ньютона
4. Аппроксимация методом наименьших квадратов
5. Линейная аппроксимация
6. Квадратичная аппроксимация

Вычислить интеграл:

1. Методом средних прямоугольников
2. Методом трапеций
3. Методом Симпсона
4. Методом с автоматическим выбором шага
5. Методом Гаусса с двумя узлами
6. Методом Гаусса с тремя узлами

ТЕМА 17. МЕТОДЫ РЕШЕНИЯ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

Цель лабораторной работы: изучить основные методы и алгоритмы решения нелинейных уравнений и их реализацию в среде DELPHI.

17.1. Методы и алгоритмы решения нелинейных уравнений

Необходимый теоретический материал приведен в лекции «Методы решений нелинейных уравнений». Основные схемы алгоритмов приведены ниже.

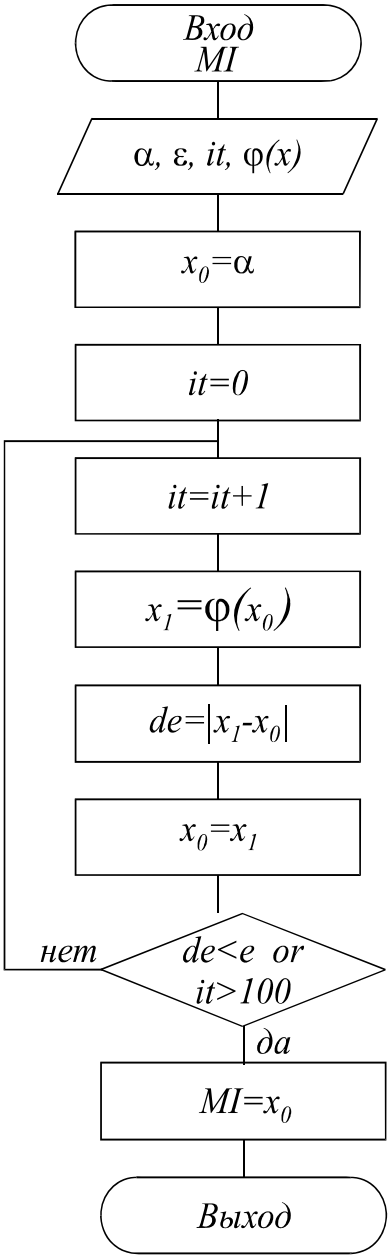


Рис 17.1. Схема алгоритма метода простой итерации

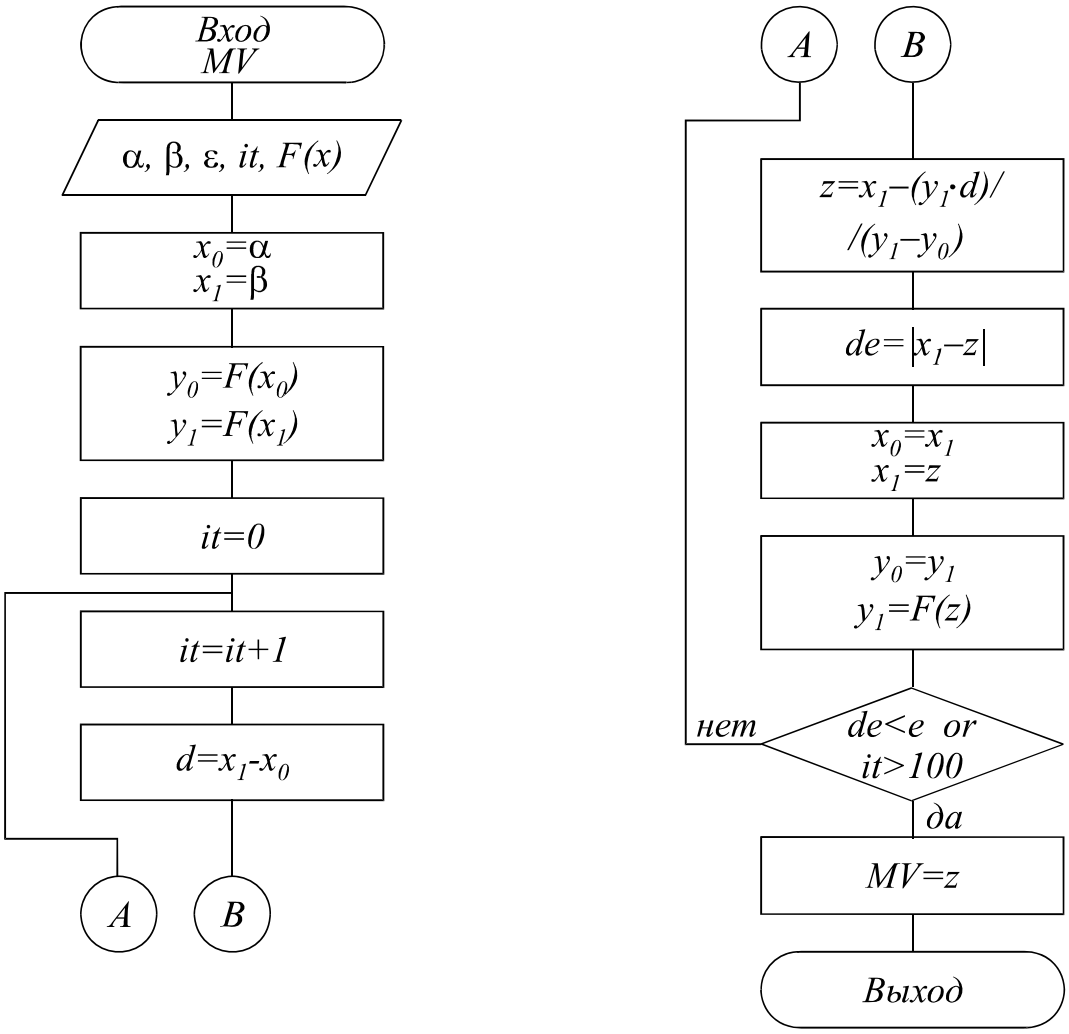
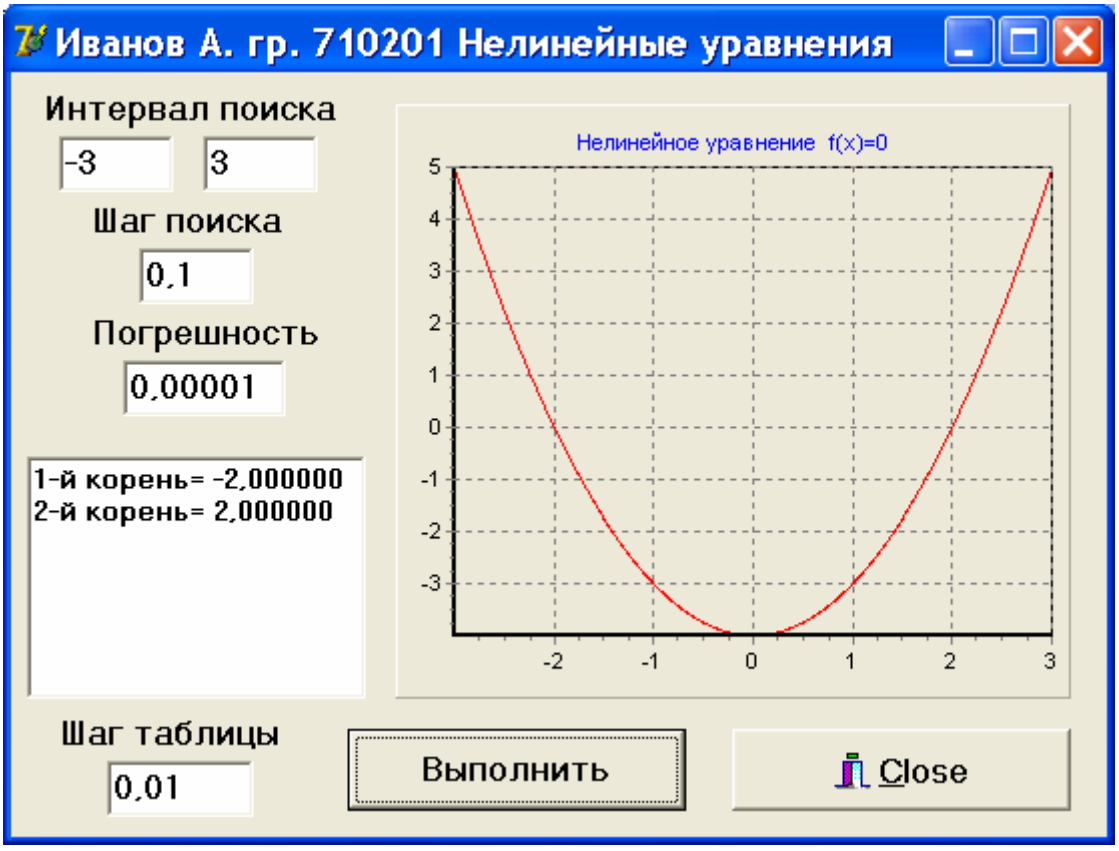


Рис. 17.2 Схема алгоритма метода Вегстейна

17.2. Пример написания программы

Изображение формы и текст программы нахождения простых корней нелинейного уравнения методом Ньютона приведены ниже. Вычисления оформлены в виде функций f(x), fp(x) и nyton(x,eps,f,fp) (описание нелинейного уравнения, его производной и самого метода Ньютона).



Листинг 17.1

```
unit Unit1;
                  interface
uses Windows, Messages, SysUtils, Variants, Classes,
 Graphics, Controls, Forms,  Dialogs, TeEngine,
 Series, Buttons, StdCtrls, ExtCtrls, TeeProcs,  Chart;
 type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Label2: TLabel;
    Chart1: TChart;
```

```
    Button1: TButton;
    BitBtn1: TBitBtn;
    Series1: TLineSeries;
    Memo1: TMemo;
    Edit4: TEdit;
    Label3: TLabel;
    Label4: TLabel;
    Edit5: TEdit;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var Form1: TForm1;

              implementation

{$R *.dfm}

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

  function f(x:extended):extended;
    begin  // Задание нелинейного уравнения
      f:=x*x-4;
    end;

  function fp(x:extended):extended;
    begin  // Производная
      fp:=2*x;
    end;

  function nyton(x,eps:extended; f,fp:fun):extended;
      var w:extended;
    begin   // Метод Ньютона
      repeat
         w:=x;
         x:=x-f(x)/fp(x);
      until abs(w-x)<eps;
      result:=x;
    end;
```

```
  procedure TForm1.FormCreate(Sender: TObject);
    begin
      memo1.Clear;
    end;

  procedure TForm1.Button1Click(Sender: TObject);
   var a,b,h,x,eps:extended; // Выполнить
       n:integer;
    begin
   a:=strtofloat(edit1.Text);
   b:=strtofloat(edit2.Text);
   h:=strtofloat(edit3.Text);
   eps:=strtofloat(edit4.Text);
   series1.Clear;
   x:=a;
   repeat
     series1.AddXY(x,f(x),'',clred);
     x:=x+strtofloat(edit5.Text);
   until x>b+0.0000000001;
   x:=a;   n:=0;
   repeat
    if f(x)*f(x+h)<0 then  // отсечение корня
      begin  inc(n);
      memo1.Lines.Add(inttostr(n)+'-й корень= '+
       floattostrf(nyton(x+h/2,eps,f,fp),fffixed,9,6));
      end;
      x:=x+h;
   until x>b+0.0000000001;
  end;

end.
```

17.3. Индивидуальные задания

Составить программу нахождения всех простых корней уравнения *f(x)=0* в заданном интервале *[a, b]* одним из методов по указанию преподавателя. Вычисления оформить в виде подпрограммы.

1. Метод простой итерации
2. Метод Ньютона
3. Метод секущих
4. Метод Вегстейна
5. Метод парабол
6. Метод деления отрезка пополам

ТЕМА 18. МЕТОДЫ НАХОЖДЕНИЯ МИНИМУМА ФУНКЦИИ ОДНОЙ ПЕРЕМЕННОЙ

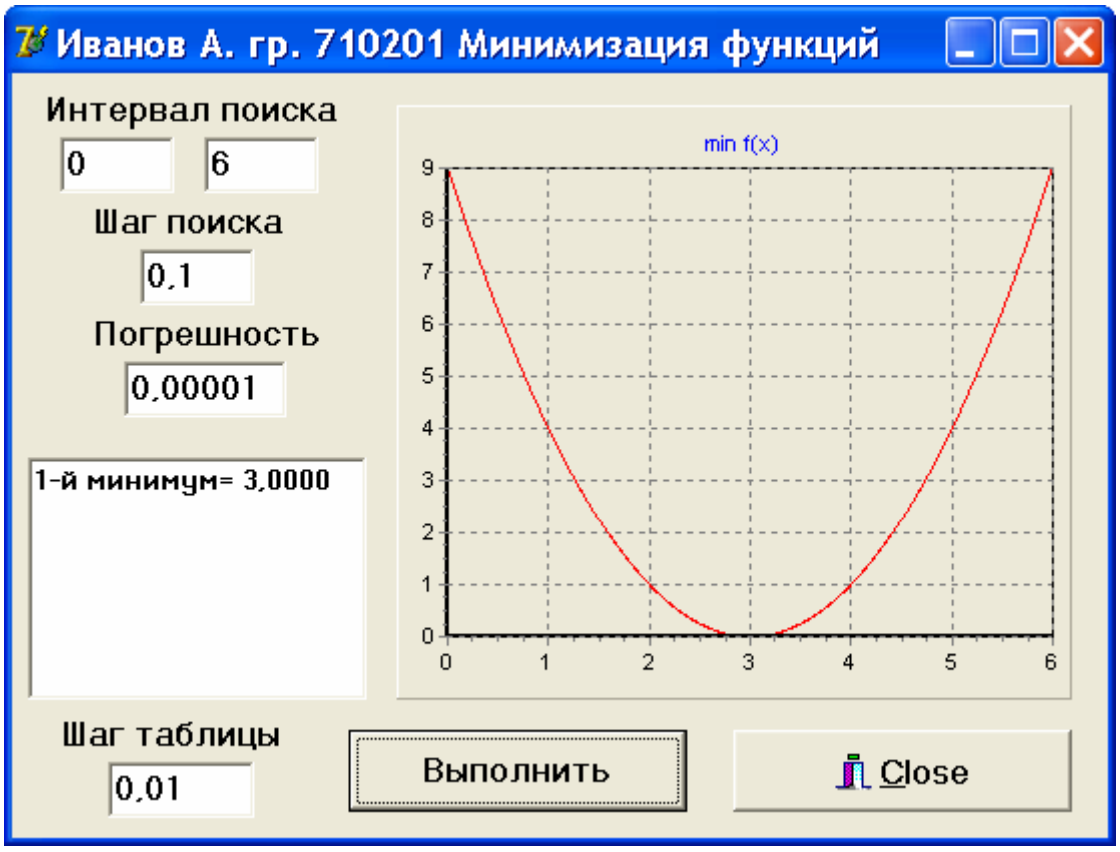
Цель лабораторной работы: изучить основные методы и алгоритмы нахождения минимума функции одной переменной и их реализацию в среде Delphi.

18.1. Методы и алгоритмы нахождения минимума функции одной переменной

Необходимый теоретический материал приведен в лекции «Методы нахождения минимума».

18.2. Пример написания программы

Изображение формы и текст программы нахождения минимума методом золотого сечения приведены ниже. Вычисления оформлены в виде функции gold(a,b,eps,f).



Листинг 18.1

```
unit Unit1;
                interface
uses  Windows, Messages, SysUtils, Variants, Classes,
 Graphics, Controls, Forms,  Dialogs, TeEngine,Series,
Buttons, StdCtrls, ExtCtrls, TeeProcs,  Chart;
 type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Label2: TLabel;
    Chart1: TChart;
    Button1: TButton;
    BitBtn1: TBitBtn;
    Series1: TLineSeries;
    Memo1: TMemo;
    Edit4: TEdit;
    Label3: TLabel;
    Label4: TLabel;
    Edit5: TEdit;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var Form1: TForm1;

              implementation

{$R *.dfm}

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

function f(x:extended):extended;
  begin  // Задание функции
    f:=sqr(x-3);
  end;
```

```
function gold(a,b,eps:extended; f:fun):extended;
    const w=0.381966011;
    var x1,x2,y1,y2:extended;
  begin   // Метод золотого сечения
    x1:=a+w*(b-a);   y1:=f(x1);
    x2:=b-w*(b-a);   y2:=f(x2);
    repeat
      if y1>y2 then
         begin
         a:=x1; x1:=x2; y1:=y2;
         x2:=b-w*(b-a); y2:=f(x2);
         end
               else
         begin
         b:=x2; x2:=x1; y2:=y1;
         x1:=a+w*(b-a); y1:=f(x1);
         end;

    until abs(b-a)<eps;
    result:=(a+b)/2;
  end;

  procedure TForm1.FormCreate(Sender: TObject);
    begin
      memo1.Clear;
    end;

  procedure TForm1.Button1Click(Sender: TObject);
   var a,b,h,x,eps:extended; // Выполнить
       n:integer;
    begin
   a:=strtofloat(edit1.Text);
   b:=strtofloat(edit2.Text);
   h:=strtofloat(edit3.Text);
   eps:=strtofloat(edit4.Text);
   series1.Clear;
   x:=a;
   repeat
     series1.AddXY(x,f(x),'',clred);
     x:=x+strtofloat(edit5.Text);
   until x>b+0.0000000001;
   x:=a;   n:=0;  memo1.Clear;
   repeat
    if (f(x)<f(x-h))and(f(x)<f(x+h))then
```

```
      begin  inc(n);
      memo1.Lines.Add(inttostr(n)+'-й минимум= '+
       floattostrf(gold(x-h,x+h,eps,f),fffixed,9,4));
      end;
      x:=x+h;
   until x>b+0.0000000001;
  end;

end.
```

18.3. Индивидуальные задания

Составить программу нахождения минимума функции *f(x)* в заданном интервале *[a, b]* одним из методов по указанию преподавателя. Вычисления оформить в виде подпрограммы.

1. Метод золотого сечения
2. Метод Фибоначи
3. Метод последовательного перебора
4. Метод квадратичной параболы
5. Метод кубической параболы
6. Метод деления отрезка пополам

ТЕМА 19. РЕШЕНИЕ ЗАДАЧИ КОШИ ДЛЯ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

Цель лабораторной работы: изучить основные методы и алгоритмы решения задачи Коши для обыкновенных дифференциальных уравнений и их реализацию в среде Delphi.

19.1. Методы и алгоритмы решения обыкновенных дифференциальных уравнений

Необходимый теоретический материал приведен в лекции «Решение задачи Коши для обыкновенных дифференциальных уравнений». Основные схемы алгоритмов приведены ниже.

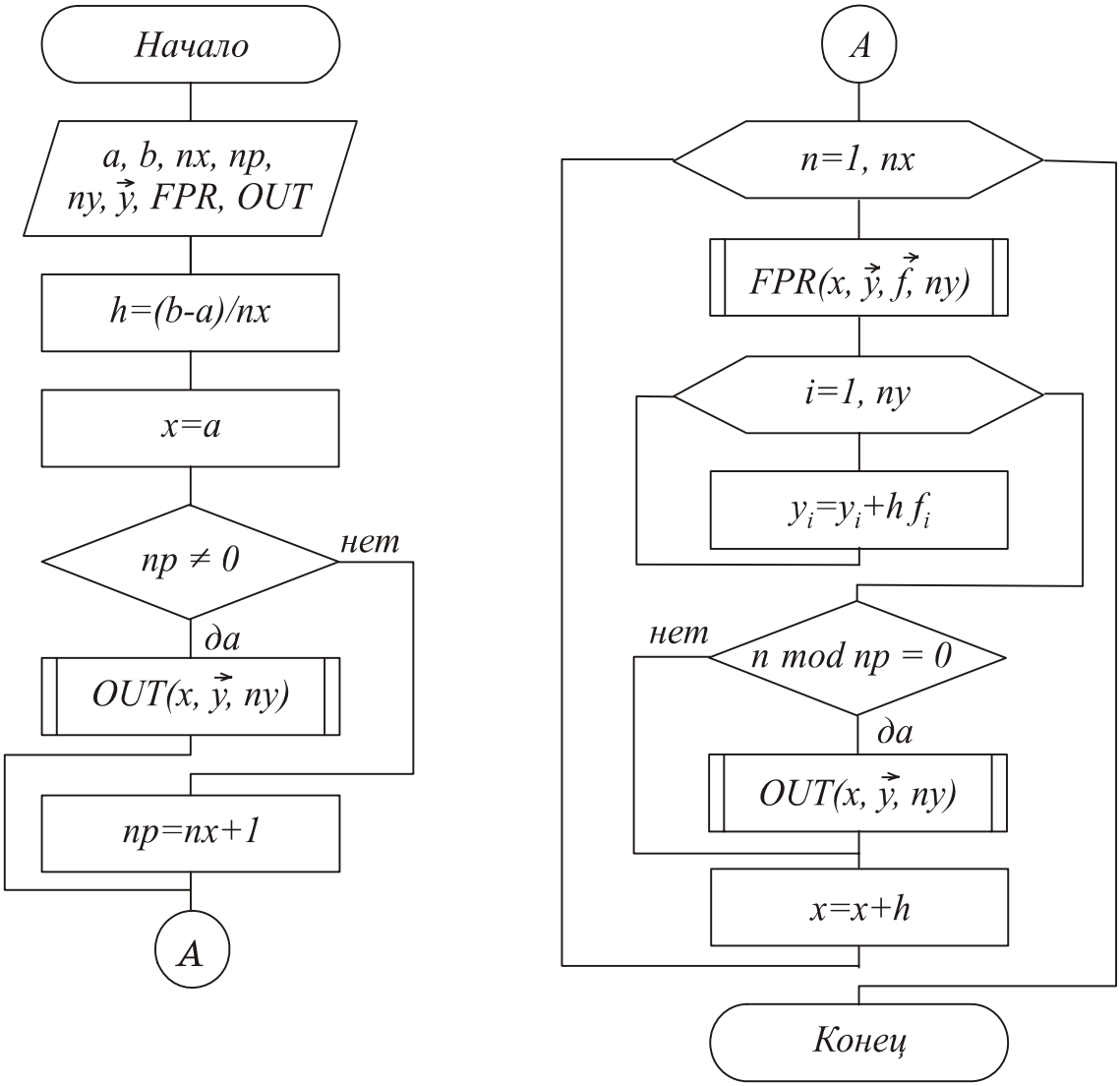


Рис.19.1. Схема алгоритма метода Эйлера

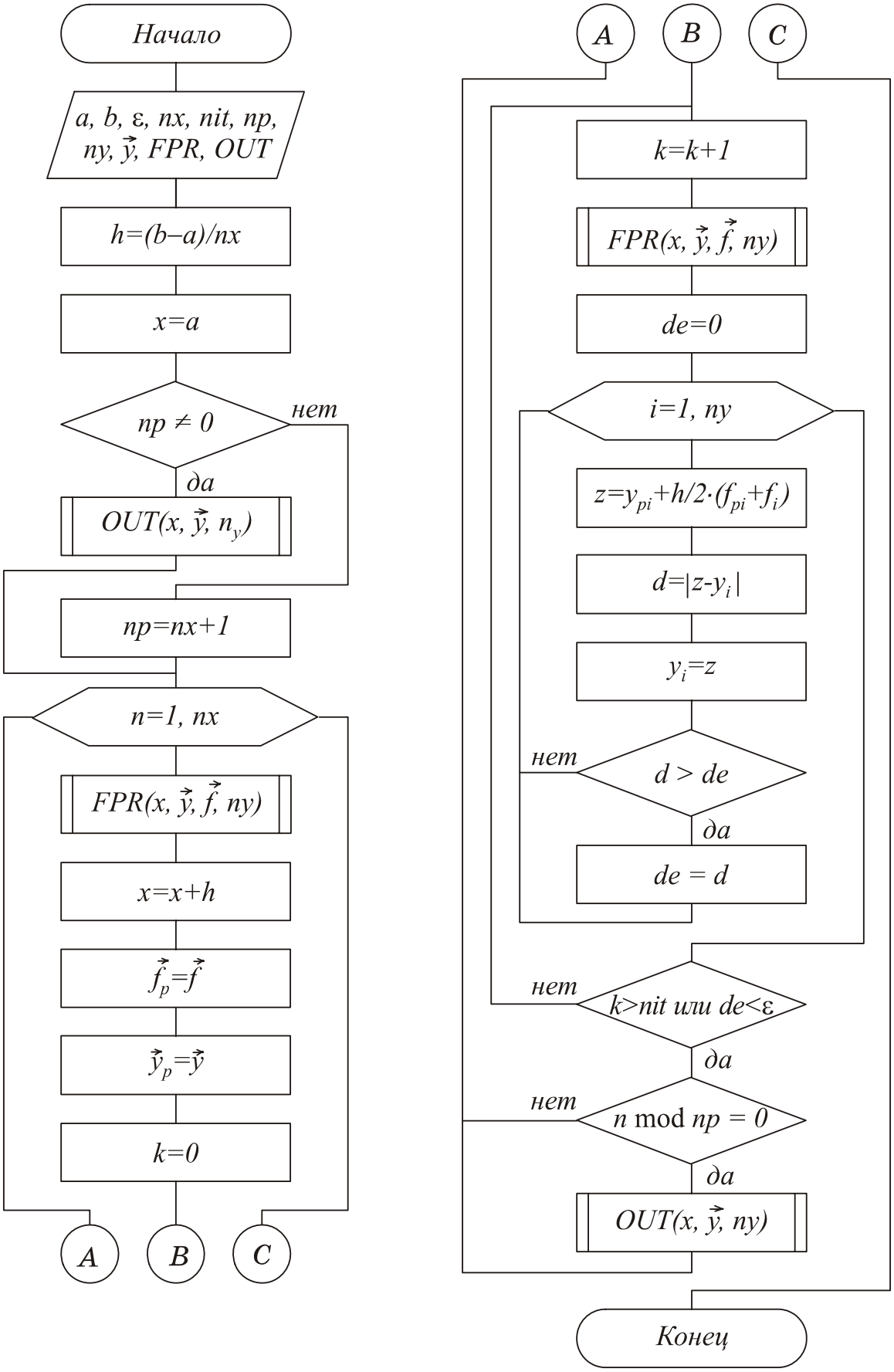


Рис. 19.2. Неявная схема 2-го порядка

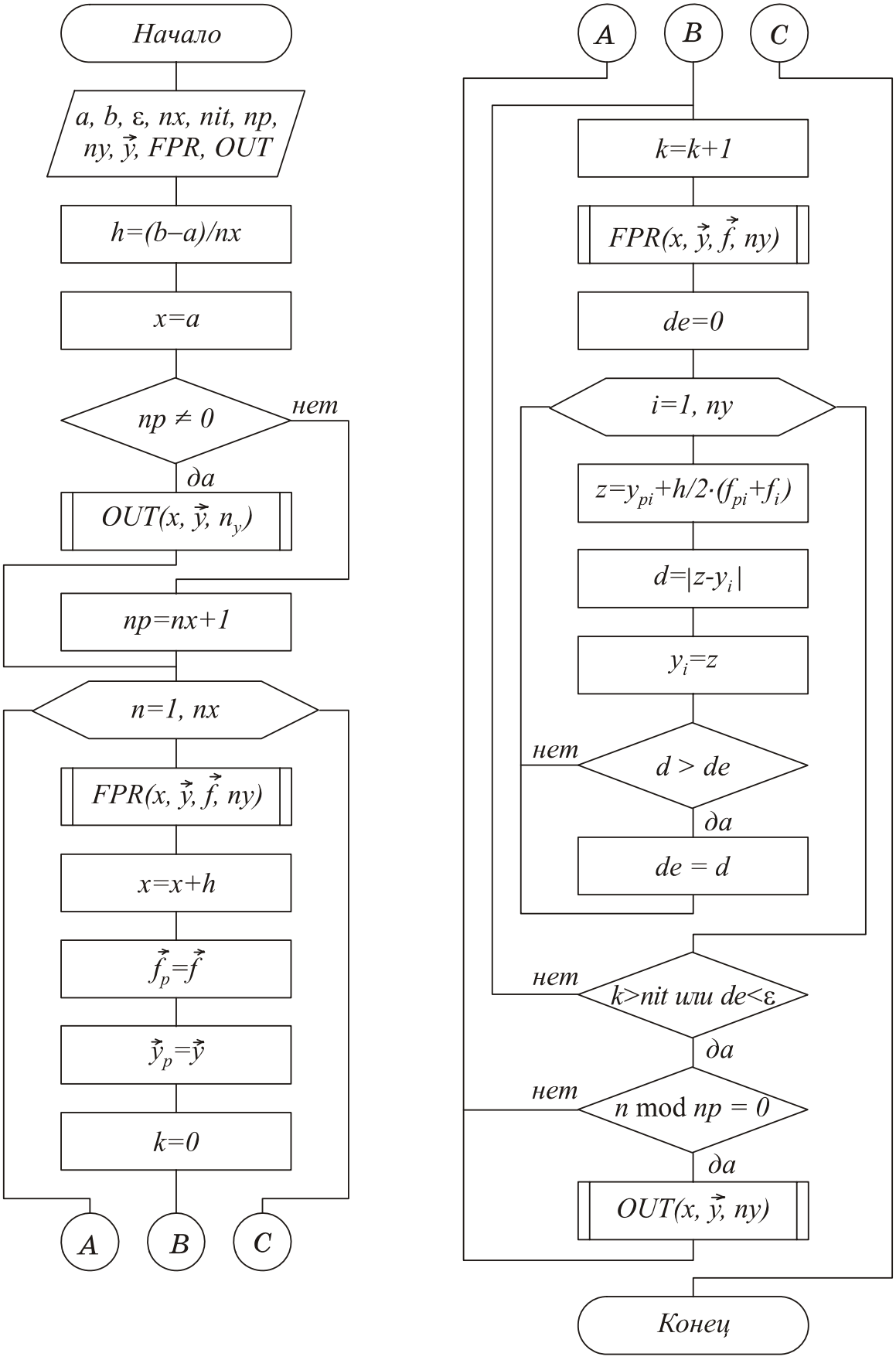


Рис. 19.3 Схема Рунге-Кутта 2-го порядка

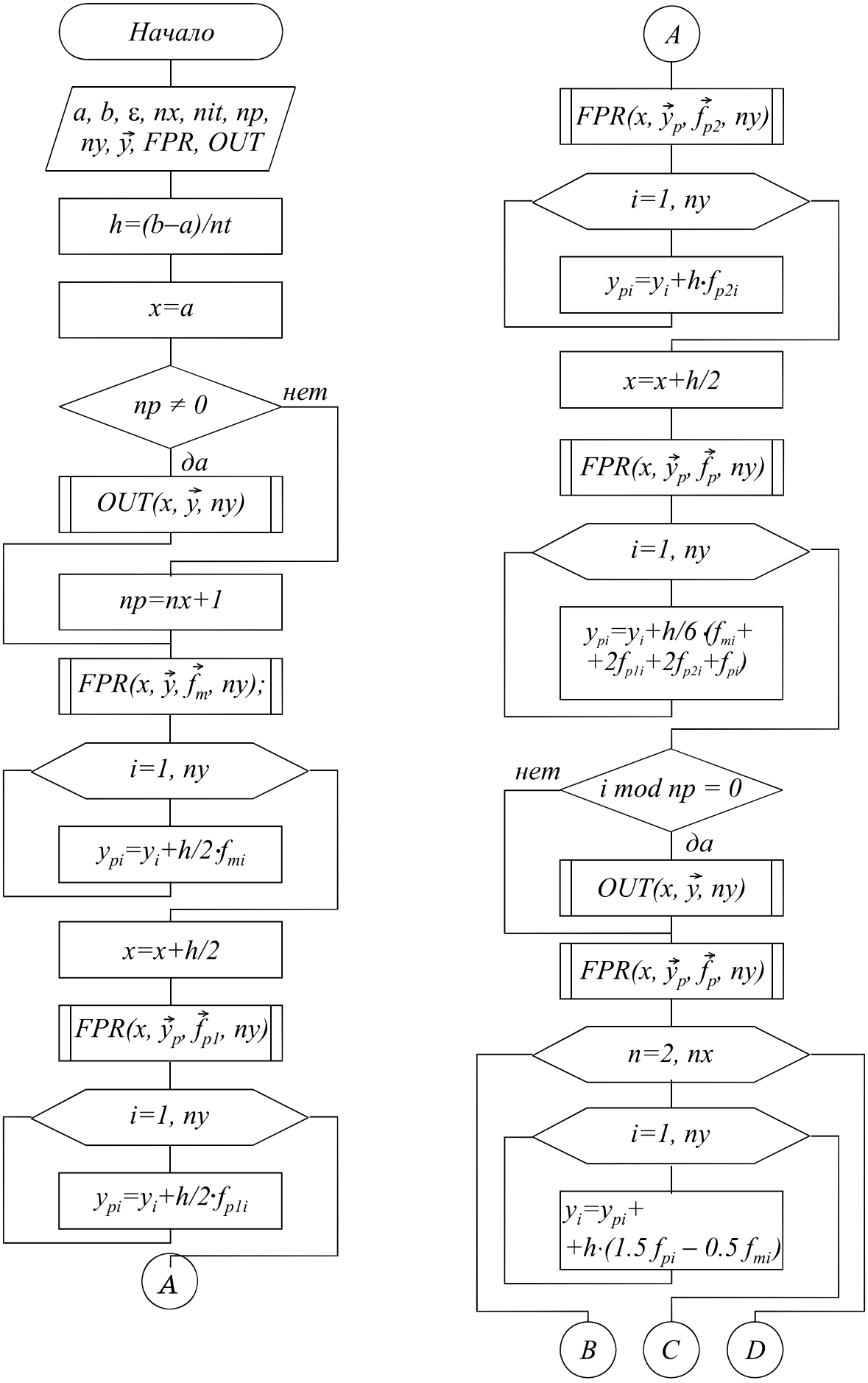
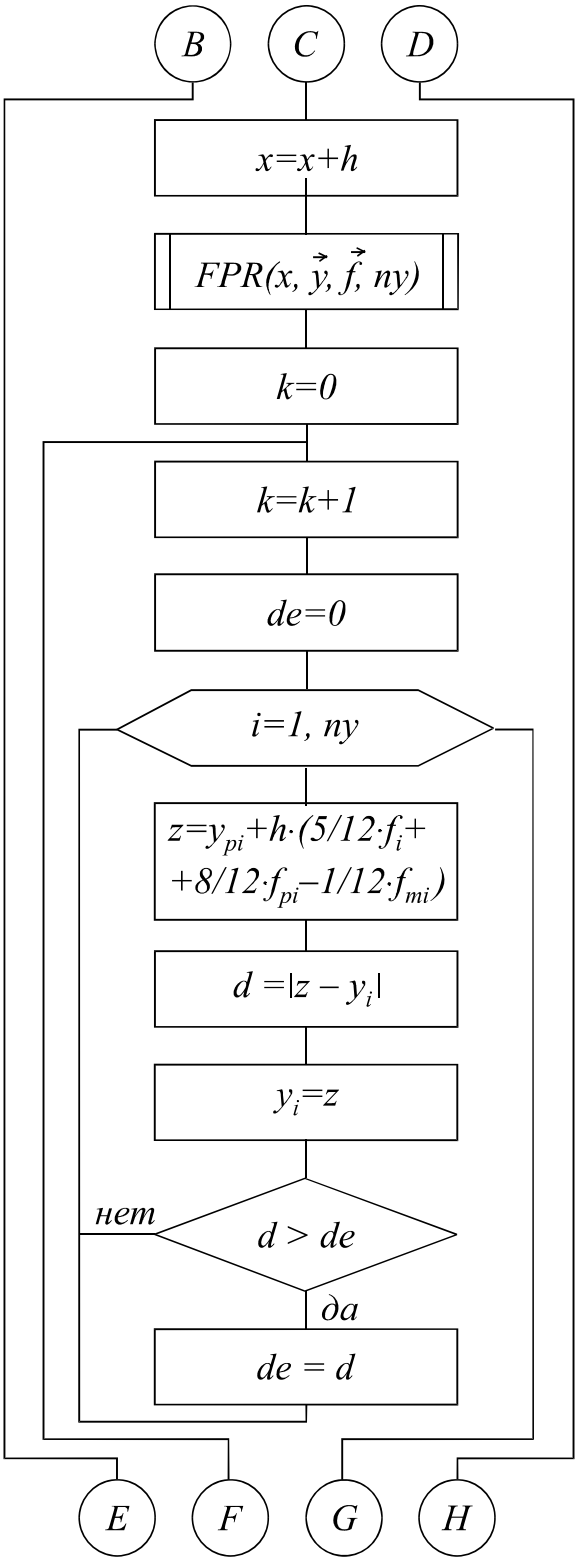


Рис. 19.4. (начало) Схема Адамса 3-го порядка



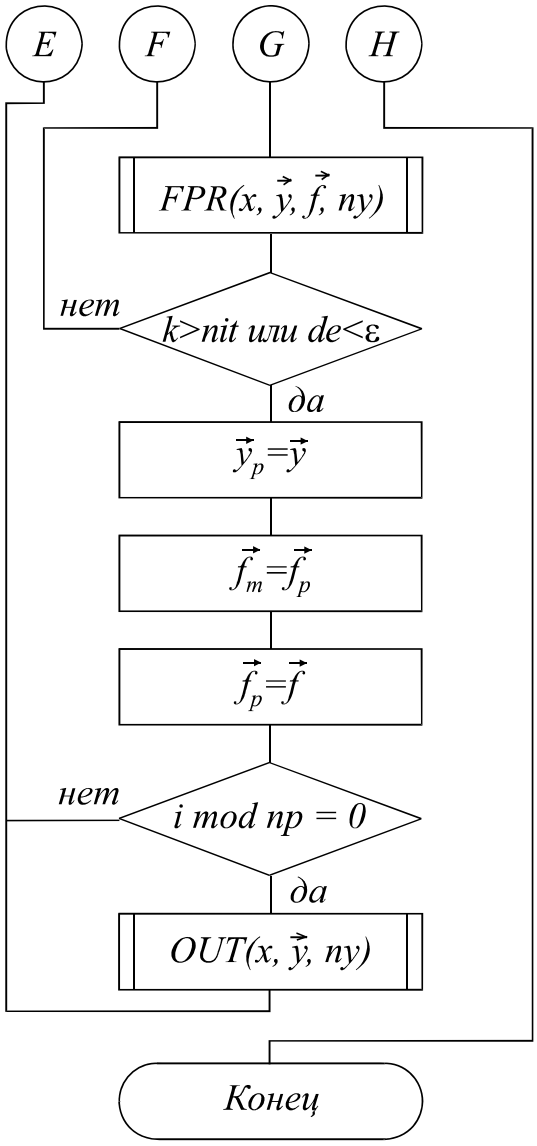
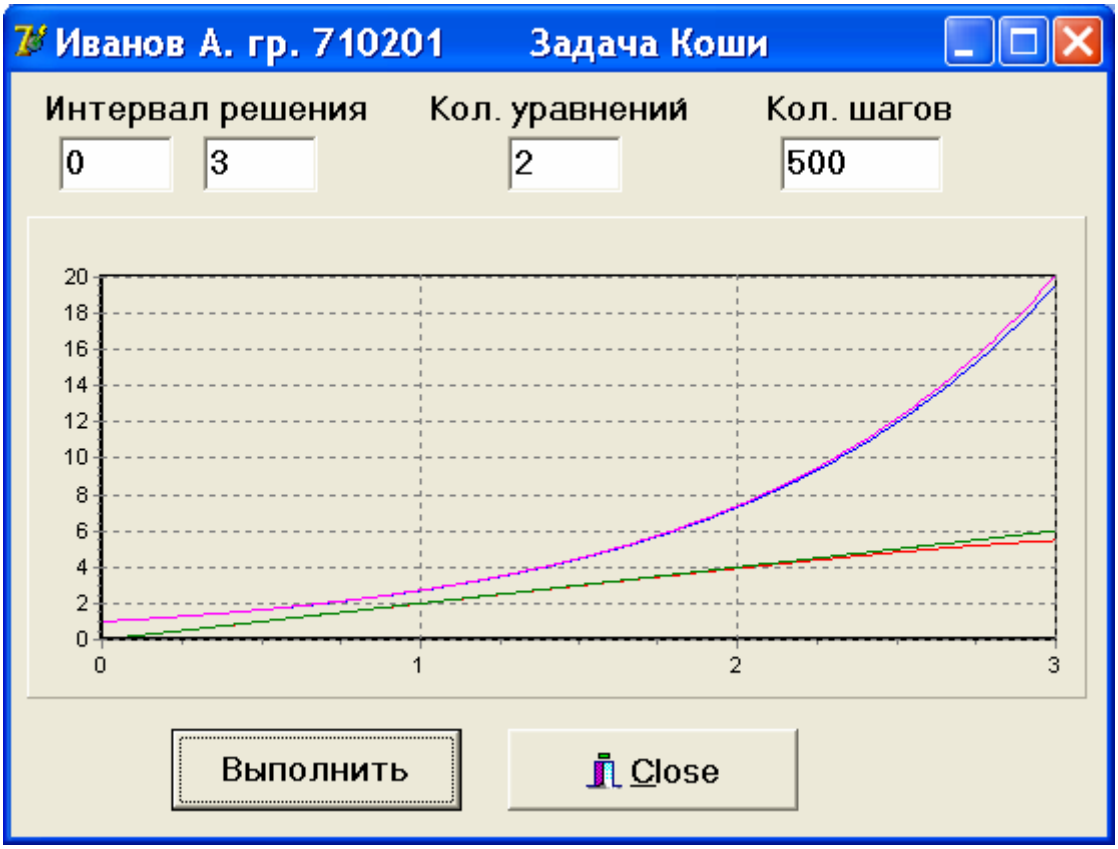


Рис. 19.4. (окончание) Схема Адамса 3-го порядка

19.2. Пример написания программы

Изображение формы и текст программы решения системы дифференциальных уравнений методом Эйлера приведены ниже. Метод Эйлера оформлен в виде процедуры.



Листинг 19.1

```
unit Unit1;
                    interface
uses
  Windows, Messages, SysUtils, Variants, Classes,
  Graphics, Controls, Forms,  Dialogs, TeEngine,
  Series,Buttons,StdCtrls,ExtCtrls,TeeProcs,Chart;
 type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Label2: TLabel;
    Chart1: TChart;
    Button1: TButton;
```

```
    BitBtn1: TBitBtn;
    Series1: TLineSeries;
    Edit4: TEdit;
    Label4: TLabel;
    Series2: TLineSeries;
    Series3: TLineSeries;
    Series4: TLineSeries;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var  Form1: TForm1;
                implementation
{$R *.dfm}

type vek=array[1..20] of extended;

procedure fpr(x:extended; var y,f:vek; ny:word);
  begin // Правые части системы диф. уравнений
    f[1]:=y[1]+y[2]-2*x-exp(x)+2;
    f[2]:=y[1]+y[2]-2*x;
  end;

 procedure wrt(x:extended; var y:vek; ny:word;
      chart:TChart);
  begin
   chart.series[0].AddXY(x,y[1],'',clred);
   chart.series[1].AddXY(x,y[2],'',clblue);
               // Точное решение
   chart.series[2].AddXY(x,2*x,'',clgreen);
   chart.series[3].AddXY(x,exp(x),'',$00FF00FF);
  end;

 procedure aler(a,b:extended; var y,f:vek;
  ny,nx:word; chart:TChart); // Метод Эйлера
    var x,h:extended;
        n,i:word;
  begin
    h:=(b-a)/nx;
    x:=a;
    wrt(x,y,ny,chart);
    for n:=1 to nx do
```

```
      begin
      fpr(x,y,f,ny);
      for i:=1 to ny do
        y[i]:=y[i]+h*f[i];
      x:=x+h;
      wrt(x,y,ny,chart);
      end;
  end;

 procedure TForm1.Button1Click(Sender: TObject);
   var a,b:extended; // Выполнить
       nx,ny:word;
       y,f:vek;
 begin
   a:=strtofloat(edit1.Text);
   b:=strtofloat(edit2.Text);
   ny:=strtoint(edit3.Text);
   nx:=strtoint(edit4.Text);
   series1.Clear;  series2.Clear;
   y[1]:=2*a;   y[2]:=exp(a); // Начальные условия
   aler(a,b,y,f,ny,nx,chart1);
 end;

end.
```

19.3. Индивидуальные задания

Составить программу решения задачи Коши для обыкновенных дифференциальных уравнений одним из методов по указанию преподавателя. Метод решения оформить в виде подпрограммы. Построить графики полученного и точного решений. С помощью этой программы решить задачу для системы двух уравнений в соответствии с вариантом из таблицы 19.1.

$$\frac{du_1}{dx} = f_1(x, u_1, u_2),$$

$$\frac{du_2}{dx} = f_2(x, u_1, u_2),$$

$$a \le x \le b,$$

$$u_1(a) = u_1^0,$$

$$u_2(a) = u_2^0$$

Точное решение для всех вариантов: $u_1 = 2x$, $u_2 = e^x$.

Методы решения задачи Коши

1. Метод Эйлера
2. Неявная схема 1-го порядка
3. Неявная схема 2-го порядка
4. Метод Рунге-Кутта 2-го порядка
5. Метод Рунге-Кутта 4-го порядка
6. Явная экстраполяционная схема Адамса 2-го порядка
7. Явная экстраполяционная схема Адамса 3-го порядка
8. Неявная схема Адамса 3-го порядка

Таблица 19.1

N	$f_1(x, u_1, u_2)$	$f_2(x, u_1, u_2)$	*[a, b]*	$U_1(a)$	$u_2(a)$	*Метод*
1	$u_1 / x - u_2 / e^x + 1$	$u_1 /(2x) + u_2 - 1$	[1, 3]	2	2.71	M1
2	$u_1 + u_2 - 2x - e^x + 2$	$u_1 + u_2 - 2x$	[1, 2]	2	2.71	M2
3	$u_1 + 2u_2 / e^x - 2x$	$u_1 /(2x) - e^x / u_2 + u_2$	[2, 3]	4	7.34	M3
4	$(u_1 \cdot e^x)/(x \cdot u_2)$	$2u_1 + u_2 - 4x$	[1, 4]	2	2.71	M4
5	$2u_1 + (u_2 + e^x)/e^x - 4x$	$2x \cdot u_2 / u_1$	[2, 4]	4	7.34	M5
6	$u_1 \cdot u_2 /(e^x \cdot x)$	$2x / u_1 + 2u_2 - e^x - 1$	[1, 3]	2	2.71	M6
7	$u_1 / 2x + u_2 / e^x$	$u_1 \cdot u_2 / 2x$	[2, 3]	4	7.34	M7
8	$u_1 / x + u_2 - e^x$	$2x / u_1 + u_2^2 / e^x - 1$	[1, 4]	2	2.71	M8
9	$u_1 + 2e^x / u_2 - 2x$	$u_1^2 / x^2 + u_2 - 4$	[1, 2]	2	2.71	M7
10	$4x / u_1 - u_2 + e^x$	$u_1 / 2x - u_2 / e^x + e^x$	[2, 4]	4	7.34	M6
11	$2x / u_1 + u_2 / e^x$	$u_1 \cdot e^{2x} /(u_2 \cdot 2x)$	[3, 4]	6	19.9	M5
12	$u_1 \cdot u_2 /(2e^x) - x + 2$	$u_1 + 2u_2 - 2x - e^x$	[1, 3]	2	2.71	M4
13	$u_1^2 + u_2 - 4x^2 - e^x + 2$	$u_1 \cdot e^x / u_2 + u_2 - 2x$	[1, 2]	2	2.71	M3
14	$u_1^2 / 2x^2 - u_2 + e^x$	$u_1 \cdot e^x / 2x + u_2 / e^x - 1$	[2, 4]	4	7.34	M2
15	$u_1 \cdot e^x /(x \cdot u_2)$	$2x / u_1 + u_2 - 1$	[3, 4]	6	19.9	M1

ПРИЛОЖЕНИЕ 1. ПРОЦЕДУРЫ И ФУНКЦИИ ДЛЯ ПРЕОБРАЗОВАНИЯ СТРОКОВОГО ПРЕДСТАВЛЕНИЯ ЧИСЕЛ

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

Подпрограммы преобразования строк в другие типы	
Function StrToFloat(St: String): Extended;	Преобразует символы строки St в вещественное число. Строка не должна содержать ведущих или ведомых пробелов
Function StrToInt(St: String): Integer;	Преобразует символы строки St в целое число. Строка не должна содержать ведущих или ведомых пробелов
Procedure Val(St: String; var X; Code: Integer);	Преобразует строку символов St во внутреннее представление целой или вещественной переменной X, которое определяется типом этой переменной. Параметр Code содержит ноль, если преобразование прошло успешно
Подпрограммы обратного преобразования	
Function FloatToStr(Value: Extended): String;	Преобразует вещественное значение Value в строку символов
Function FloatToStrF(Value: Extended; Format: TFloatFor-mat; Precision, Digits: Integer) : String;	Преобразует вещественное значение Value в строку символов с учетом параметров Precision и Digits (см. пояснения ниже)
Procedure Str(X [:width [:Decimals]]; var St: String);	Преобразует число X любого вещественного или целого типа в строку символов St; параметры Width и Decimals, если они присутствуют, задают формат преобразования

Правила использования параметров функции FloatToStrF

Значение Format	Описание
fFfExponent	Научная форма представления с множителем eXX. Precision задает общее количество десятичных цифр мантиссы. Digits - количество цифр в десятичном порядке XX.
ffFixed	Формат с фиксированным положением разделителя целой и дробной частей. Precision задает общее количество десятичных цифр в представлении числа. Digits - количество цифр в дробной части. Число округляется с учетом первой отбрасываемой цифры: 3,14
ffGeneral	Универсальный формат, использующий наиболее удобную для чтения форму представления вещественного числа. Соответствует формату ffFixed, если количество цифр в целой части меньше или равно Precision, а само число - больше или равно 0,00001, в противном случае соответствует формату ffExponent: 3,1416

ffNumber	Отличается от ffFixed использованием символа - разделителя тысяч при выводе больших чисел (для русифицированной версии Windows таким разделителем является пробел)
ffCurrency	Денежный формат. Соответствует ffNumber, но в конце строки ставится символ денежной единицы (для русифицированной версии Windows - символы «р.»). Для Value = π*1000 получим: 3 141,60р

ПРИЛОЖЕНИЕ 2. МАТЕМАТИЧЕСКИЕ ФОРМУЛЫ

Язык Object Pascal имеет ограниченное количество встроенных математических функций (|x| → abs (x), Arctg(x) → ArcTan(x), e^x → Exp(x), $\sqrt{x}$ → Sqrt(x), x^2 → Sqr(x), Ln(x), Cos(x), Sin (x) и др.). Поэтому при необходимости использовать другие функции следует применять известные соотношения или модуль Math. В таблице приведены выражения наиболее часто встречающихся функций.

Функция	Соотношение	Модуль Math
$Log_a(x)$	$\frac{Ln(x)}{Ln(a)}$	$LogN(a, x)$
x^a	$e^{a*Ln(x)}$	$Power(x,a)$
$Tg(x)$	$\frac{Sin(x)}{Cos(x)}$	$Tan(x)$
$Ctg(x)$	$\frac{Cos(x)}{Sin(x)}$	$CoTan(x)$
$ArcSin(x)$	$ArcTg\left(\frac{x}{\sqrt{1-x^2}}\right)$	$ArcSin(x)$
$ArcCos(x)$	$\frac{\pi}{2} - ArcSin(x)$	$ArcCos(x)$
$ArcCtg(x)$	$\frac{\pi}{2} - ArcTg(x)$	
$Sh(x)$	$\frac{e^x - e^{-x}}{2}$	$Sinh(\mathbf{x})$
$Ch(x)$	$\frac{e^x + e^{-x}}{2}$	$Cosh(x)$
$Sign(x)$	*1, если x>0;* *0, если x=0;* *-1, если x<0*	$Sign(x)$

ПРИЛОЖЕНИЕ 3. ТАБЛИЦЫ СИМВОЛОВ ASCII

Стандартная часть таблицы символов ASCII

КС	С	КС	С	КС	С	КС	С	КС	С	КС	С	КС	С	КС	С
0		16	►	32		48	0	64	@	80	P	96	`	112	p
1	☺	17	◄	33	!	49	1	65	A	81	Q	97	a	113	q
2	☻	18	↕	34	"	50	2	66	B	82	R	98	b	114	r
3	♥	19	‼	35	#	51	3	67	C	83	S	99	c	115	s
4	♦	20	¶	36	$	52	4	68	D	84	T	100	d	116	t
5	♣	21	§	37	%	53	5	69	E	85	U	101	e	117	u
6	♠	22	▬	38	&	54	6	70	F	86	V	102	f	118	v
7	•	23	↨	39	'	55	7	71	G	87	W	103	g	119	w
8	◘	24	↑	40	(	56	8	72	H	88	X	104	h	120	x
9	○	25	↓	41	)	57	9	73	I	89	Y	105	i	121	y
10	◙	26	→	42	*	58	:	74	J	90	Z	106	j	122	z
11	♂	27	←	43	+	59	;	75	K	91	[	107	k	123	{
12	♀	28	∟	44	,	60	<	76	L	92	\	108	l	124	|
13	♪	29	↔	45	-	61	=	77	M	93	]	109	m	125	}
14	♫	30	▲	46	.	62	>	78	N	94	^	110	n	126	~
15	☼	31	▼	47	/	63	?	79	O	95	_	111	o	127	⌂

Некоторые из вышеперечисленных символов имеют особый смысл. Так, например, символ с кодом 9 обозначает символ горизонтальной табуляции, символ с кодом 10 – символ перевода строки, символ с кодом 13 – символ возврата каретки.

Дополнительная часть таблицы символов

КС	С	КС	С	КС	С	КС	С	КС	С	КС	С	КС	С	КС	С
128	А	144	Р	160	а	176	▒	192	└	208	╨	224	р	240	Ё
129	Б	145	С	161	б	177	▒	193	┴	209	╤	225	с	241	ё
130	В	146	Т	162	в	178	▓	194	┬	210	╥	226	т	242	Є
131	Г	147	У	163	г	179	│	195	├	211	╙	227	у	243	є
132	Д	148	Ф	164	д	180	┤	196	─	212	╘	228	ф	244	Ї
133	Е	149	Х	165	е	181	╡	197	┼	213	╒	229	х	245	ї
134	Ж	150	Ц	166	ж	182	╢	198	╞	214	╓	230	ц	246	Ў
135	З	151	Ч	167	з	183	╖	199	╟	215	╫	231	ч	247	ў
136	И	152	Ш	168	и	184	╕	200	╚	216	╪	232	ш	248	°
137	Й	153	Щ	169	й	185	╣	201	╔	217	┘	233	щ	249	·

138	К	154	Ъ	170	к	186	╢	202	╩	218	┌	234	ъ	250	·
139	Л	155	Ы	171	л	187	╖	203	╦	219	█	235	ы	251	√
140	М	156	Ь	172	м	188	╜	204	╠	220	▄	236	ь	252	№
141	Н	157	Э	173	н	189	╛	205	═	221	▌	237	э	253	¤
142	О	158	Ю	174	о	190	╡	206	╬	222	▐	238	ю	254	■
143	П	159	Я	175	п	191	┐	207	╧	223	▀	239	я	255	

В таблицах обозначение **КС** означает "код символа", а **С** – "символ".

ЛИТЕРАТУРА

1. Архангельский А.Я. Программирование в Delphi 7. – М.: ЗАО “Издательство БИНОМ”, 2003.
2. Фаронов В.В. Delphi 6. Учебный курс. -М.: Издатель Молгачева С.В., 2001.
3. 3.Дж.Гленн Брукшир. Введение в компьютерные науки. – «Вильямс» М, С-П, Киев. 2001.
4. А.К. Синицын, С.В. Колосов, А.А. Навроцкий и др. Программирование алгоритмов в среде Delphi. Лаб. практикум. Ч. 1. – Мн., БГУИР, 2004.
5. Колосов С. В. Программирование в Delphi. Учеб. пособие – Мн., БГУИР, 2005.
6. Калиткин Н.Н. Численные методы. -М.: Наука, 1978. -512 с.
7. Бахвалов Н.С. Численные методы. -М.: Наука, 1975. -632 с.
8. Демидович В.П. и др. Численные методы анализа. -М.: Физматгиз, 1963. -400 с.
9. Волков Е.А. Численные методы. -М.: Наука, 1982. -255 с.
10. Крылов В.И. и др. Вычислительные методы высшей математики. Т.1. -Мн.: Вышэйшая школа, 1972. -584 с.
11. Крылов В.И. и др. Вычислительные методы высшей математики. Т.2. -Мн.: Вышэйшая школа, 1975. -672 с.
12. Форсайт Дж. и др. Машинные методы математических вычислений -М.: Мир, 1980. -280 с.
13. Шуп Т. Решение инженерных задач на ЭВМ. -М.: Мир, 1982. -238 с.
14. Самарский А.А. Введение в численные методы. -М.: Наука, 1982.-272 с.
15. 12. Березин И.С., Жидков Н.П. Методы вычислений. Т.2. -М.: Физматгиз, 1970. -620 с.
16. 13. Б.Банди. Методы оптимизации. Вводный курс. -М.: Мир, 1989. -277 с.