МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА
И ПРОДОВОЛЬСТВИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ОСНОВЫ ПРОГРАММИРОВАНИЯ (в визуальной среде Delphi)

Учебно-методический комплекс по дисциплине «Основы программирования» для студентов специальностей 1-25 01 07 «Экономика и управление на предприятии», 1-26 02 02 «Менеджмент»

В двух частях

Часть 2

Минск
БГАТУ
2010

УДК 004.4(07)
ББК 22.18я7
О-75

Рекомендовано научно-методическим советом факультета предпринимательства и управления БГАТУ.
Протокол № 4 от 25 марта 2010 г.

Составители:
профессор Р. И. Фурунжиев,
старший преподаватель Е. М. Исаченко,
старший преподаватель Т. В. Ероховец

Рецензенты:
доцент кафедры программного обеспечения САПР и АСУ Учреждения образования «Белорусский национальный технический университет», кандидат технических наук, доцент *Н. Н. Гурский*;
доцент кафедры вычислительной техники Учреждения образования «Белорусский государственный аграрный технический университет», кандидат технических наук, доцент *А. И. Шакирин*

Основы программирования (в визуальной среде Delphi).
О-75 В 2 ч. Ч. 2 : учебно-методический комплекс / сост. : Р. И. Фурунжиев, Т. В. Ероховец, Е. Г. Исаченко. – Минск : БГАТУ, 2010. – 100 с.
ISBN 978-985-519-304-4.

Во второй части учебно-методического комплекса рассматриваются программирование с использованием записей, файлов; организация подпрограмм и модулей; средства для построения и отображения графиков.

Ориентирован на теоретическое изучение и приобретение навыков разработки различных программных приложений, реализующих элементы задач экономики и управления.

Предназначен для студентов высших учебных заведений экономического профиля.

УДК 004.4(07)
ББК 22.18я7

ISBN 978-985-519-304-4 (ч. 2)
ISBN 978-985-519-265-8

ВВЕДЕНИЕ

В основе идеологии визуальной среды программирования Delphi, принятой в настоящей учебной дисциплине для изучения основ программирования, лежит современная технология визуального проектирования и методология объектно-ориентированного событийного программирования.

Цель дисциплины – формирование необходимых знаний и навыков использования современных базовых компьютерных технологий разработки алгоритмов и создания Windows-приложений в визуальной среде программирования Delphi, обеспечивающих эффективное решение задач экономики и управления на предприятии; развитие логико-алгоритмического, пооперационного и системного мышления студентов.

Задачи дисциплины

Изучение дисциплины способствует формированию у студентов следующих компетенций:

- **академических**, включающих знания и умения по анализу, формализации, обобщению, алгоритмизации и программированию процессов окружающего мира, способность и умение учиться;
- **социально-личностных,** включающих культурно-ценностные ориентации, знание идеологических, нравственных ценностей общества и государства и умение следовать им;
- **профессиональных,** включающих знание основных особенностей информационных процессов, протекающих в обществе и их влияние на темпы экономического развития страны; умение формулировать проблемы, составлять алгоритмы и решать задачи, разрабатывать планы и обеспечивать их выполнение в избранной сфере профессиональной деятельности.

В результате освоения материала курса

студент должен знать:

- особенности современных технологий программирования;
- методику постановки, алгоритмизации и программирования на объектно-ориентированном алгоритмическом языке Object Pascal в визуальной среде программирования Delphi задач экономики и управления на предприятии;

уметь:

- конструировать интерфейс и формировать сценарий применения Windows-приложений;
- математически формулировать, алгоритмизировать и программировать задачи экономики и управления на предприятии в визуальной среде программирования Delphi.

При выполнении лабораторных работ студентам предлагаются индивидуальные задания трех уровней сложности. Задания первого уровня соответствуют усваиванию материала на репродуктивном уровне, задания второго уровня предполагают продуктивный уровень усваивания материала, задания третьего уровня сложности предназначены для творчески мыслящих студентов и отображают творческий уровень усваивания материала. Студент выбирает уровень сложности задания самостоятельно (в соответствии с самооценкой своих знаний).

Изучение дисциплины «Основы программирования» базируется на знании «Математики» и «Информатики» в объеме учебной программы общеобразовательной школы.

МОДУЛЬ 3

ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ЗАПИСЕЙ, ФАЙЛОВ, ПОДПРОГРАММ, МОДУЛЕЙ И СРЕДСТВ ДЛЯ ОТОБРАЖЕНИЯ ГРАФИЧЕСКОЙ ИНФОРМАЦИИ

В результате изучения тем модуля студент должен:

знать методы описания пользовательских типов данных, приемы работы с данными типа «запись» (record), средства создания и обработки файлов различных типов, свойства и методы компонентов OpenDialog, SaveDialog и др., изучить возможности Delphi для разработки подпрограмм и модулей, освоить средства отображения графической информации;

уметь описывать пользовательские типы данных, обрабатывать данные типа «запись» (record), создавать и обрабатывать файлы различных типов, конструировать алгоритмы и программы с применением записей и файлов, использовать компоненты OpenDialog и SaveDialog, создавать подпрограммы и модули, строить графики и диаграммы с помощью компонентов отображения графической информации; применять полученные теоретические и практические знания для создания программ.

Практические задания

МАТЕРИАЛЫ К ЛАБОРАТОРНОЙ РАБОТЕ № 1

1. Программирование с использованием записей

Цель работы – овладение навыками создания приложений, обрабатывающих данные типа «запись».

Обработка записей

«Запись» (record) – это тип структурированных данных, позволяющий создавать сртуктуры, состоящие из именованных разнотипных элементов данных, обрабатываемых как единое целое. Отдельные элементы записи называются ***полями*.** Таким образом тип «запись» позволяет объединять элементы данных различных типов в единое целое. Синтаксис объявления записи следующий:

```
type
  ИмяТипа = record
    ИмяПоля1: ТипПоля1;
    ИмяПоля2: ТипПоля2;
    ИмяПоля3: ТипПоля3;
    ...
    ИмяПоляN: ТипПоляN;
  end;
```

Замечание. Вместо элемента ИмяПоляN может быть приведен список имен однотипных полей.

Например, в приведенном ниже примере будет объявлен тип данных «запись» (record), имя типа «zap»:

```
type
 zap=record
  nzach :integer;
  fio:string[20];
  mat, fiz, inf: integer;
  srb:extended;
 end;
```

Запись состоит из 6 полей, четыре из которых имеют тип **integer**, одно **– string** и одно – **extended.**

Далее будет объявлен массив записей, то есть таблица.

```
var
  MZap: array[1..9] of zap;
```

Переменные типа «запись» могут участвовать в операторах присваивания, но никакие операции над ними выполняться не могут.

Арифметические или другие операции могут выполняться только над отдельными полями записи.

Обращение к соответствующим полям записи осуществляется с помощью ***составного имени***, в котором указывается имя переменной типа «запись» и имя поля, разделенные символом **.** (точка). Синтаксис обращения к полю записи выглядит следующим образом:

ПеременнаяТипаЗапись.ИмяПоля

Например, для присваивания значения полю fio первого элемента массива **MZap** типа **zap** необходим следующий оператор:

MZap[1]**.**fio:=’Смирнов’;

При использовании записей сложной структуры, например, когда один или несколько элементов записи в свою очередь может быть записью или при использовании массивов записей, обращение к элементам становится громоздким. Для упрощения записи таких элементов используется оператор **with**, который в общем виде выглядит следующим образом:

with простое или составное имя переменной типа запись **do**

```
begin
операторы
end;
```

Например, с учетом предыдущего объявления, можно записать:

```
with MZap[1] do
 begin
   nzach:=346578;              // или MZap[1].nzach:=346578;
   fio:=’Соловей В. Н.’;     // или MZap[1].fio:=’Соловей В. Н.’;
   mat:=9;                     // или MZap[1]. mat:=9;
   fiz:=10;                    // или MZap[1]. fiz:=10;
   inf:=9;                     // или MZap[1]. inf:=9;
   srb:=(mat+fiz+inf)/3;       // или MZap[1]. srb:=( MZap[1].mat+
                               // MZap[1].fiz+ MZap[1]. inf)/3;
end;
```


Обращение ко всей записи в целом, а не только к ее элементам, допускается в операторе присваивания, при условии, что записи идентичны, например:

Zap2:=Zap1; После выполнения этого оператора значения элементов записи Zap2 станут в точности равны значениям соответствующих элементов записи Zap1.

1.1. Пример создания приложения

Задание. Создать Windows-приложение для обработки ведомости об успеваемости студентов группы в количестве 5 человек. Каждая запись должна содержать номер зачетной книжки, фамилию и инициалы, а также оценки по математике, физике и информатике. Для каждого студента рассчитать средний балл. Вывести ведомость в порядке убывания среднего балла.

В отдельный список вывести информацию о неуспевающих студентах (студентах, получивших хотя бы одну двойку).

В отдельное поле вывести наибольший и наименьший номера зачетных книжек в группе, а так же фамилии студентов, которым они принадлежат.

1.1.1. Размещение компонентов на Форме

Один из возможных вариантов панели интерфейса создаваемого приложением показан на рисунке 1.1.

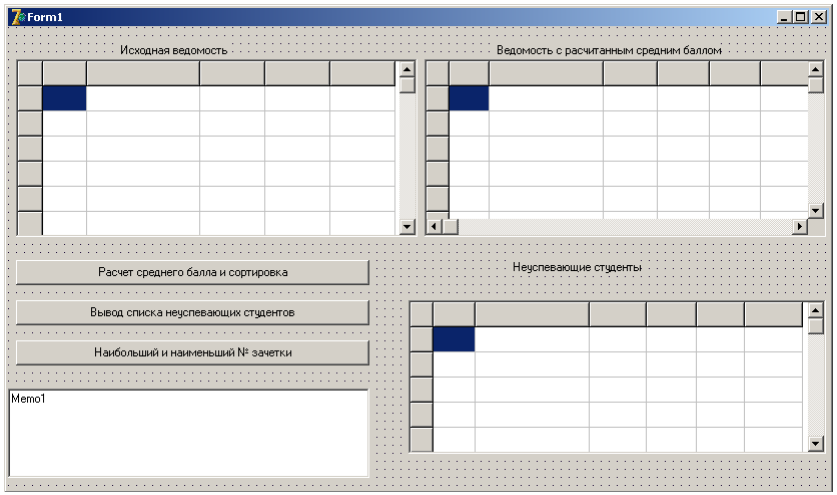


Рис. 1.1. Размещение компонентов на форме

При работе с записями ввод и вывод информации на экране удобно организовать с помощью компонента ***StringGrid***, который находится на странице ***Additional***.

Как видно, на форме размещены три компонента ***StringGrid***: первый (***StringGrid1***) предназначен для ввода исходной ведомости, второй (***StringGrid2***) – для вывода ведомости, содержащей средний балл в порядке убывания среднего балла, третий (***StringGrid3***) – для вывода списка неуспевающих студентов.

Для соответствующих заголовков колонок и номеров строк используется фиксированная зона компонента ***StringGrid***, поэтому в Инспекторе Объектов значение ***FixedCols*** и ***FixedRows*** установите равными 1 для всех компонентов ***StringGrid***. В соответствии с заданием установите значение свойства ***ColCount*** = 6 (количество столбцов) для ***StringGrid1*** и ***ColCount*** = 7 для ***StringGrid2*** и ***StringGrid3***, а значение свойства ***RowCount*** = 6 (количество строк) для всех компонентов ***StringGrid***. Для наличия вертикальной или горизонтальной или обеих линеек прокрутки в компоненте ***StringGrid*** установите свойство ***ScrollBars*** в состояние *ssVertical*, *ssGorizontal* или *ssBoth* соответственно. Но старайтесь устанавливать такой размер поля компонента ***StringGrid***, чтобы вся таблица умещалась в поле.

Откройте список опций свойства ***+Options*** и установите значение ***goEditing*** в *True* – это даст возможность редактировать информацию в компоненте ***StringGrid*** с помощью клавиатуры и «мыши».

На форме также размешены три кнопки (***Batton***). Каждая из них выполняет функцию задания соответственно надписи.

Компонент ***Memo*** предназначена для вывода фамилий студентов с максимальным и минимальным номерами зачетных книжек.

Три компонента ***Label*** (Метка) служат для вывода пояснительных надписей.

1.1.2. Сохранение проекта

Для нового проекта создайте новую папку, например ***X:\41эи\Ленартович\Mod3\Lab1***.

Сохраните проект ***File | Save Project As….*** Сначала сохраните модуль с именем **UnZap.pas**, затем файл проекта под именем **PrZap.dpr.**

Последующие сохранения выполнять командами ***File | Save All.***

1.1.3. Создание процедур обработки событий FormCreate и ButtonClick

Двойным нажатием клавиши «мыши» на Форме и на кнопках ***Button1***, ***Button2***, ***Button3*** создайте соответствующие процедуры обработки событий. Используя текст модуля **UnZap**, внимательно наберите операторы этих процедур.

1.1.4. Работа с приложением

Запустите созданное приложение. Проанализируйте результаты. В случае необходимости откорректируйте приложение, снова сохраните его аналогичным образом. Выполните приложение и проанализируйте результаты (рисунок 1.2). После каждой корректировки следует выполнять сохранение приложения с помощью пунктов ***File*** | ***Save All***.

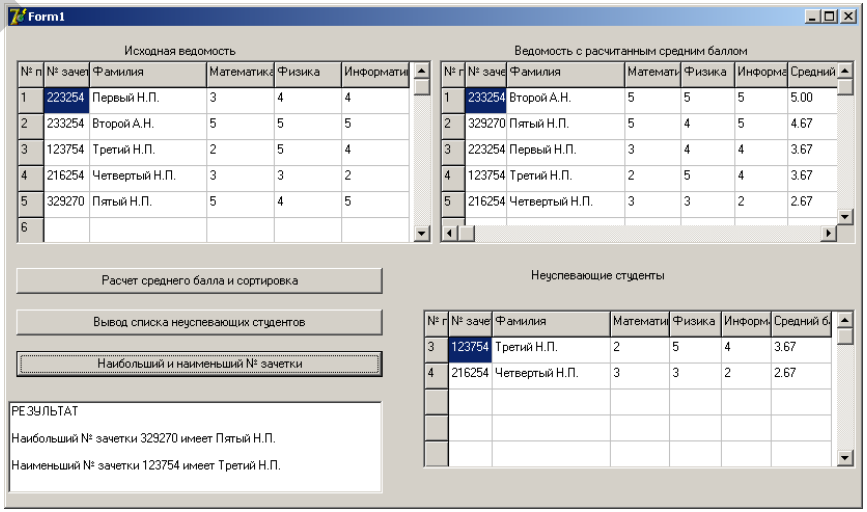


Рис. *1.2.* Интерфейс приложения после его выполнения

Текст программы приведен в листинге 1.1.

Листинг 1.1

```
Unit UnZap;
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls, Grids;
type
  TForm1 = class(TForm)
    StringGrid1: TStringGrid;
    StringGrid2: TStringGrid;
    Memo1: TMemo;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    StringGrid3: TStringGrid;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
implementation
{$R *.dfm}
type       // Создание типа данных «запись»(record), имя типа - zap
zap=record
```



```
nzach :integer;
fio:string[20];
mat,fiz,inf:integer;
srb:extended;
end;
var
MZap:array[1..5] of zap;   // объявление массива записей
{ Обработчик события создания формы}
procedure TForm1.FormCreate(Sender: TObject);
var
i: integer;
begin
 with StringGrid1 do
  begin                    // занесение информации в ячейки StringGrid1
   Cells[0,0]:='№ п/п';
   Cells[1,0]:='№ зачетной книжки';
   Cells[2,0]:='Фамилия';
   Cells[3,0]:='Математика';
   Cells[4,0]:='Физика';
   Cells[5,0]:='Информатика';
   for i:=1 to 5 do
   Cells[0,i]:=IntToStr(i);
  Cells[1,1]:='223254';Cells[2,1]:='Первый Н.П.'; Cells[3,1]:='5';
Cells[4,1]:='7';
Cells[5,1]:='8';
  Cells[1,2]:='233254';Cells[2,2]:='Второй А.Н.'; Cells[3,2]:='9';
Cells[4,2]:='9';
Cells[5,2]:='9';
  Cells[1,3]:='123754';Cells[2,3]:='Третий Н.П.'; Cells[3,3]:='2';
Cells[4,3]:='5';
Cells[5,3]:='4';
  Cells[1,4]:='216254';Cells[2,4]:='Четвертый Н.П.'; Cells[3,4]:='5';
Cells[4,4]:='6';
Cells[5,4]:='2';
  Cells[1,5]:='329270';Cells[2,5]:='Пятый Н.П.'; Cells[3,5]:='9';
Cells[4,5]:='8';
Cells[5,5]:='9';
    for i:=1 to 5 do
```

```
    with MZap[i] do
     begin                    // формирование полей массива записей
      nzach:=StrToInt(Cells[1,i]);
      fio:=Cells[2,i];
      mat:=StrToInt(Cells[3,i]);
      fiz:=StrToInt(Cells[4,i]);
      inf:=StrToInt(Cells[5,i]);
     end;
  end;
 end;
{ Сортировка записей в порядке убывания среднего балла}
procedure TForm1.Button1Click(Sender: TObject); var
i,j: integer;
rab: zap;        // Рабочая переменная для обмена значениями двух
элементов
                 //   массива при сортировке
begin
 for i:=1 to 5 do
  with StringGrid1,MZap[i] do
   begin   // формирование полей массива записей при нажатии Button1
    nzach:=StrToInt(Cells[1,i]);
    fio:=Cells[2,i];
    mat:=StrToInt(Cells[3,i]);
    fiz:=StrToInt(Cells[4,i]);
    inf:=StrToInt(Cells[5,i]);
    srb:=(mat+fiz+inf)/3         // Расчет среднего балла
   end;
for i:=2 to 5 do        // Сортировка массива записей по убыванию
среднего балла
  for j:=5 downto i do           // методом пузырька
   if MZap[j-1].srb<MZap[j].srb then
    begin
     rab:=MZap[j-1];
     MZap[j-1]:=MZap[j];
     MZap[j]:=rab;
    end;
   with StringGrid2 do            // Очистка ячеек StringGrid2
   for i:=0 to 5 do
```



```
    for j:=0 to 6 do
     Cells[j,i]:=' ';
  with StringGrid2 do
   begin          // Пересылка заголовков столбцов в первую строку
StringGrid2
    Cells[0,0]:='№ п/п';
    Cells[1,0]:='№ зачетной книжки';
    Cells[2,0]:='Фамилия';
    Cells[3,0]:='Математика';
    Cells[4,0]:='Физика';
    Cells[5,0]:='Информатика';
    Cells[6,0]:='Средний балл';
    for i:=1 to 5 do
     with MZap[i] do
      begin   // Занесение информации в остальные ячейки StringGrid2
      Cells[0,i]:=IntToStr(i);       // из рассортированного массива Mzap
       Cells[1,i]:=IntToStr(nzach);
       Cells[2,i]:=fio;
       Cells[3,i]:=IntToStr(mat);
       Cells[4,i]:=IntToStr(fiz);
       Cells[5,i]:=IntToStr(inf);
       Cells[6,i]:=floatToStrF(srb,ffFixed,5,2);
      end;
    end;
 end;
{ Вывод списка неуспевающих студентов}
procedure TForm1.Button2Click(Sender: TObject); var
i,j:integer;
begin
 for i:=1 to 5 do
  with StringGrid1,MZap[i] do
   begin                        // Занесение данных в массив записей
    nzach:=StrToInt(Cells[1,i]);
    fio:=Cells[2,i];
    mat:=StrToInt(Cells[3,i]);
    fiz:=StrToInt(Cells[4,i]);
    inf:=StrToInt(Cells[5,i]);
    srb:=(mat+fiz+inf)/3
```

```
  end;
  with StringGrid3 do            // Очистка ячеек  StringGrid3
   for i:=0 to 5 do
    for j:=0 to 6 do
     Cells[j,i]:=' ';
  with StringGrid3 do
   begin              // Занесение заголовков столбцов в первую строку
StringGrid3
    Cells[0,0]:='№ п/п';
    Cells[1,0]:='№ зачетной книжки';
    Cells[2,0]:='Фамилия';
    Cells[3,0]:='Математика';
    Cells[4,0]:='Физика';
    Cells[5,0]:='Информатика';
    Cells[6,0]:='Средний балл';
    j:=0;
    for i:=1 to 5 do // Занесение только данных о неуспевающих сту-
дентах в                    StringGrid3
     with MZap[i] do
     if (mat=2) or (fiz=2) or (inf=2) then
      begin
       j:=j+1;
       Cells[0,j]:=IntToStr(j);
       Cells[1,j]:=IntToStr(nzach);
       Cells[2,j]:=fio;
       Cells[3,j]:=IntToStr(mat);
       Cells[4,j]:=IntToStr(fiz);
       Cells[5,j]:=IntToStr(inf);
       Cells[6,j]:=floatToStrF(srb,ffFixed,5,2);
      end;
     end;
    end;
{ Определение наибольшего и наименьшего номеров зачетных книжек}
procedure TForm1.Button3Click(Sender: TObject);
var
i, min, max, k, l: integer;
begin
 for i:=1 to 5 do
```



```
  with StringGrid1,MZap[i] do
   begin        // Занесение данных в массив записей при нажатии
кнопки Batton3
    nzach:=StrToInt(Cells[1,i]);
    fio:=Cells[2,i];
    mat:=StrToInt(Cells[3,i]);
    fiz:=StrToInt(Cells[4,i]);
    inf:=StrToInt(Cells[5,i]);
    srb:=(mat+fiz+inf)/3
   end;
max:=MZap[1].nzach; k:=1;  // занесение № первой зачетки в ячейку max
 min:=MZap[1].nzach; l:=1; // занесение № первой зачетки в ячейку min
 for i:=1 to 5 do   // поиск максимального и минимального № зачеток
  with MZap[i] do                // и номеров в массиве их обладателей
   begin
    if MZap[i].nzach>max then
    begin
     max:=nzach;
     k:=i;
    end;
   if MZap[i].nzach<min then
    begin
     min:=nzach;
     l:=i;
    end;
  end;
   Memo1.Lines.Add('РЕЗУЛЬТАТ");  // Занесение результатов
в Memo     Memo1.Lines.Add (#13#10 + 'Наибольший № зачетки ' +
IntToStr(max) +          ' имеет ' + MZap[k].fio);
    Memo1.Lines.Add(#13#10+'Наименьший № зачетки
'+IntToStr(min)+
    ' имеет  ' + MZap[l].fio);
 end;

end.
```

1.2. Индивидуальные задания

По указанию преподавателя выберите вариант индивидуальной задачи. Выполните задания определенного уровня. Создайте приложение и протестируйте его работу.

Задачи

1. Информация о сотрудниках предприятия содержит: Ф.И.О., номер отдела, должность, дату поступления на работу.

а) Вывести списки сотрудников по отделам (в порядке возрастания).

б) Выбрать список сотрудников, поступивших на работу в текущем году с подсчетом их количества.

в) Рассчитать количество сотрудников, работающих в каждой должности и вывести информацию в алфавитном порядке должностей.

2. В библиотеке хранится информация о наличии книг. Имеются следующие данные о каждой книге: инвентарный номер, фамилии авторов, название книги, год издания, количество страниц, цена.

а) Вывести имеющуюся информацию о книгах в алфавитном порядке авторов книг.

б) Выбрать список книг, изданных ранее заданного года и определить общую стоимость этих книг.

в) Вывести фамилию автора, количество книг которого в библиотеке наибольшее.

3. Для участия в конкурсе исполнителей необходимо заполнить следующую анкету: Ф.И.О., год рождения, название страны, класс музыкального инструмента (гитара, фортепиано, скрипка, виолончель).

а) Вывести список участников конкурса по классам инструментов.

б) Рассчитать количество молодых участников конкурса, родившихся не раньше заданного года и вывести список в алфавитном порядке фамилий.

в) Определить, по какому классу инструмента в конкурсе участвует наибольшее количество музыкантов.

4. Для получения места в общежитии формируется список студентов, который включает Ф.И.О. студента, группу, средний балл, доход на члена семьи. Общежитие в первую очередь предоставляется тем, у кого доход на члена семьи меньше двух минимальных зарплат, и у кого средний балл не меньше 3.

а) Вывести список в алфавитном порядке фамилий.

б) Вывести список первоочередников, претендующих на места

в общежитии в порядке возрастания дохода на члена семьи.

в) Определить группу, содержащую наибольшее количество очередников.

5. Сведения о наличии лекарственных средств в аптеках города содержат следующие данные: наименование лекарства, фирма-изготовитель, номер аптеки, количество упаковок, стоимость одной упаковки.

а) Вывести сведения в алфавитном порядке по названиям фирм-изготовителей.

б) Получить список аптек, в которых можно купить N упаковок заданного лекарства, включив все имеющиеся сведения о лекарстве. Список вывести в порядке возрастания цены за упаковку.

в) Вывести название фирмы-изготовителя, которая поставила наибольшее число наименований лекарств, зарегистрированных в рассматриваемой информации.

6. Фирма реализует изделия клиентам. Имеются следующие данные о продажах: изделие, цена, клиент, количество, дата продажи.

а) Вывести сведения о продажах с подсчетом вырученной суммы в порядке возрастания даты продажи.

б) Для заданного изделия вывести сведения о продажах его клиентам в порядке убывания количества проданных изделий. Рассчитать общую вырученную сумму от продаж изделия.

в) Определить клиента, купившего наибольшее суммарное количество изделий, и вывести его название вместе с рассчитанным наибольшим количеством.

7. В справочной автовокзала хранится расписание движения автобусов. Для каждого рейса указаны его номер, тип автобуса, пункт назначения, время отправления и прибытия.

а) Вывести расписание в алфавитном порядке пунктов назначения.

б) Вывести информацию о рейсах, которыми можно воспользоваться для прибытия в заданный пункт назначения раньше заданного времени в порядке убывания времени прибытия.

в) Определить пункт назначения, обеспеченный наибольшим количеством рейсов.

8. На междугородней АТС информация о разговорах содержит дату разговора, код и название города, время разговора, тариф, номер телефона в этом городе и номер телефона абонента.

а) Вывести все данные с расчетом стоимости разговора в алфавитном порядке названий городов.

б) Вывести все разговоры за день, заданный с клавиатуры датой,

общую стоимость этих разговоров и самый короткий из них.

в) Определить город, общая стоимость переговоров с которым наибольшая.

9. Информация о сотрудниках фирмы включает: код подразделения, Ф.И.О., табельный номер, количество отработанных часов за месяц, почасовый тариф. Рабочее время свыше 144 часов считается сверхурочным и оплачивается в двойном размере.

а) Подсчитать размер заработной платы каждого сотрудника фирмы за вычетом подоходного налога, который составляет 12 % от суммы заработка. Вывести ведомость в алфавитном порядке фамилий сотрудников.

б) Вывести ведомость по заданному подразделению.

в) Вывести список подразделений с указанием суммарной величины заработной платы по подразделению.

10. Разработать программу формирования ведомости об успеваемости студентов. Каждая запись этой ведомости должна содержать: номер группы, Ф.И.О. студента, оценки за последнюю сессию (4 дисциплины) средний балл.

а) Вывести списки в алфавитном порядке фамилий с расчетом общего (суммарного) балла для каждого студента.

б) Выбрать список студентов заданной группы и рассчитать среднее значение общего балла по группе.

в) Определить группу с наибольшим средним баллом.

11. Различные цеха завода выпускают продукцию нескольких наименований. Сведения о выпущенной продукции включают номер цеха, наименование продукции, ее количество.

а) Вывести информацию по всем цехам в порядке возрастания номера цеха.

б) Вывести информацию об изделиях, выпуск которых составил 1000 и более единиц с подсчетом общего количества таких изделий.

в) Определить номер цеха, выпускающего наибольшее количество наименований изделий.

12. Ведомость абитуриентов, сдавших вступительные экзамены в университет, содержит Ф.И.О., город проживания, оценки по 4-м дисциплинам.

а) Вывести список абитуриентов в алфавитном порядке фамилий с подсчетом общего балла для каждого абитуриента.

б) Определить количество абитуриентов, проживающих в г. Минске и набравших общий балл не ниже 32, вывести всю информацию об этих абитуриентах.

в) Определить город, абитуриенты которого набрали наибольший суммарный общий балл.

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

а) Вывести расписание в порядке возрастания времени вылета.

б) Вывести расписание вечерних рейсов для заданного пункта назначения (время вылета >=17-00) с подсчетом количества таких рейсов.

в) Для каждого пункта назначения рассчитать количество рейсов и вывести список в алфавитном порядке наименований пунктов назначения.

14. В магазине имеется список поступивших в продажу автомобилей. Каждая строка этого списка содержит следующие характеристики: марка автомобиля, количество, стоимость, расход топлива на 100 км, надежность (число лет безотказной работы), комфортность (отличная, хорошая, удовлетворительная), страна-производитель.

а) Вывести список в порядке возрастания стоимости автомобилей.

б) Получить перечень автомобилей, удовлетворяющих требованиям покупателя, которые вводятся с клавиатуры в виде некоторого интервала допустимых значений с подсчетом количества таких автомобилей.

в) Определить, какой стране принадлежит производитель, поставивший наибольшее количество автомобилей.

15. Для участия в конкурсе на замещение вакантной должности сотрудника фирмы желающие подают следующую информацию: Ф.И.О, год рождения, образование (среднее, специальное, высшее), знание иностранных языков (английский, немецкий, французский), владение компьютером (Да, Нет), стаж работы.

а) Вывести список претендентов в алфавитном порядке фамилий.

б) Получить список претендентов в соответствии с требованиями руководства фирмы, которые вводятся с клавиатуры, с подсчетом количества претендентов.

в) Рассчитать количество претендентов каждого вида образования и вывести в виде списка.

Индивидуальные задания 1-го уровня

Выполнить задание (а) в соответствии с вариантом.

Индивидуальные задания 2-го уровня

Выполнить задания (а, б) в соответствии с вариантом.

Индивидуальные задания 3-го уровня

Выполнить задания (а, б, в) в соответствии с вариантом.

Вопросы для самоконтроля

1. Что такое пользовательский тип данных?
2. Дайте определение типа данных «запись» (Record).
3. Как организуется обращение к полям записи?
4. Как объявляется массив записей?
5. Могут ли элементы записи иметь разный тип данных?

МАТЕРИАЛЫ К ЛАБОРАТОРНОЙ РАБОТЕ № 2

2. Программирование с использованием файлов

Цель работы **–** изучить процесс организации и обработки файлов в языке Object Pascal, освоить применение компонентов ***OpenDialog*** и ***SaveDialog***.

Обработка файлов

Для организации данных на внешнем носителе используются файлы.

Файл – это поименованная совокупность логически связанных данных, хранящихся на запоминающем устройстве компьютера.

Object Pascal располагает средствами создания и обработки файлов различных типов. Для того, чтобы получить доступ к файлу, нужно иметь возможность связать созданное в Delphi приложение с некоторым файлом для чтения или записи информации. Эта связь создается при помощи переменных файлового типа – файловых переменных.

В Object Pascal существует три файловых типа:

TextFile – текстовый файл, представляющий собой набор символьных строк переменной длины;

File of <тип> – типизированный файл, представляющий собой набор данных указанного типа;

File – нетипизированный файл, представляющий собой набор неструктурированных данных.

Рассмотрим некоторые приемы работы с типизированными файлами.

Перед использованием файловой переменной она должна быть связана с внешним файлом с помощью вызова процедуры ***AssignFile***:

AssignFile(<файловая переменная>, <имя файла>);

Здесь <файловая переменная> – имя переменной, объявленной в программе как переменная файлового типа;

<имя файла> – символьная строка, содержащая имя файла.

Если файл располагается не в одной папке с программой, то необходимо указать полный путь к файлу.

Когда связь с внешним файлом установлена, его можно открыть для ввода или вывода данных с помощью процедуры Reset:

Reset (<имя файла>);

Эта процедура открывает существующий внешний файл, имя которого было связано с файловой переменной.

Новый файл можно создать и открыть для записи с помощью процедуры Rewrite:

Rewrite(<файловая переменная>);

Последовательный доступ к записям файла осуществляется с помощью процедур Read и Write.

Read (<файловая переменная>, список ввода) чтение записи файла

Write (<файловая переменная>, список вывода) вывод записи в файл.

Список ввода и список вывода должны иметь данные того же типа, что и компоненты файла.

Прямой доступ к типизированным файлам можно организовать с помощью стандартной процедуры ***Seek***, которая перемещает указатель файла к заданному элементу. Для определения текущей записи в файле и текущего размера файла используются стандартные функции ***FilePos*** и ***FileSize.***

Procedure Seek(var F; N: Longint); перемещает текущую позицию в типизированном файле, связанном с файловой переменной F к компоненту с номером N. Нумерация компонентов в файле начинается с 0.

function FilePos(var F): Longint; Возвращает номер текущего компонента в файле, связанном с файловой переменной F.

function FileSize(var F): Integer; Возвращает количество компонентов в файле, связанном с файловой переменной F.

По завершении обработки файла он должен закрываться с помощью стандартной процедуры ***CloseFile***.

CloseFile(<файловая переменная>);

При закрытии файла обеспечивается сохранение в файле всех новых записей и регистрация файла в папке. Процедура ***CloseFile*** не разрывает связь файла с файловой переменной, поэтому файл можно открывать снова без повторного использования процедуры ***AssignFile***.

2.1. Пример создания приложения

Задание. Создать Windows-приложение для формирования списка очередности предоставления мест в общежитии. Исходные данные, которые следует организовать в виде файла, должны содержать следующую информацию о студентах: фамилию и инициалы студента, доход на одного члена семьи и четыре оценки, полученные на экзаменах последней сессии. Общежитие в первую очередь предоставляется тем студентам, у которых доход на одного члена семьи не превышает прожиточного минимума. Рассортировать список в порядке возрастания дохода на члена семьи. Вывести список первоочередников в порядке убывания среднего балла экзаменационных оценок. Создать текстовый файл исходных данных.

2.1.1. Размещение компонентов на Форме

Один из возможных вариантов панели интерфейса создаваемого приложения показан на рисунке 2.1.

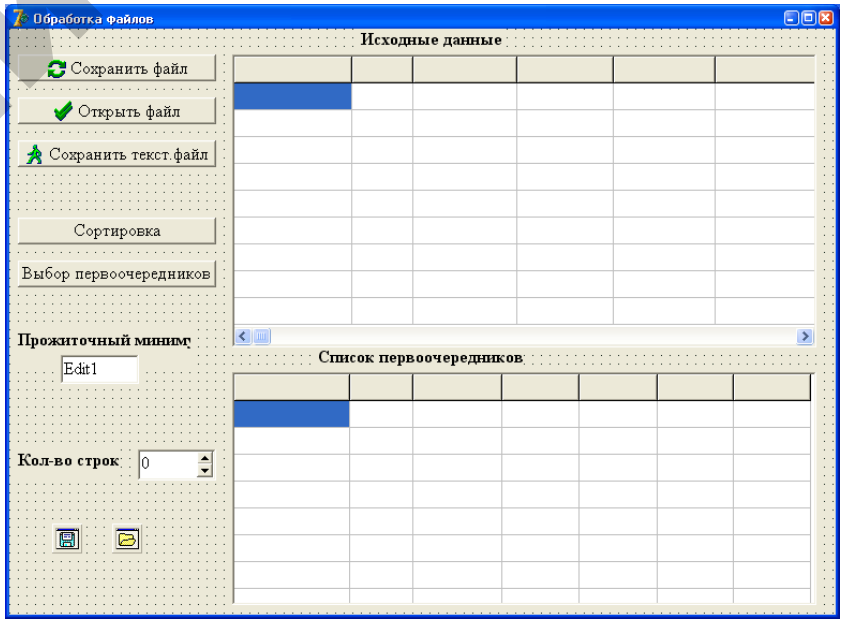


Рис. 2.1. Размещение компонентов на форме

При работе с файлами чтение и запись информации удобно организовывать с помощью компонентов ***OpenDialog*** и ***SaveDialog***.

Компоненты ***OpenDialog*** и ***SaveDialog*** находятся на странице

Dialogs. Все компоненты этой страницы являются не визуальными, т. е. не видны в момент выполнения приложения. Поэтому их можно разместить в любом удобном месте Формы. Оба рассматриваемых компонента имеют идентичные свойства и отличаются только внешним видом.

Для установки компонентов ***OpenDialog*** и ***SaveDialog*** на Форму необходимо на странице Dialogs Палитры Компонентов щелкнуть «мышью» соответственно по пиктограмме  или  и разместить их в любом свободном месте Формы. При выполнении приложения в момент вызова компонента появляется диалоговое окно, с помощью которого пользователь выбирает имя файла и маршрут к нему. В случае успешного завершения диалога имя выбранного файла и маршрут поиска содержится в свойстве ***FileName***.

Пользователь имеет возможность настроить параметры окна диалога по своему усмотрению. В частности, изменить заголовок окна можно с помощью свойства *Title*. В свойстве ***DefaultExt*** можно указать расширение файла, если оно не задано пользователем. Свойство *Filter* используется для поиска (фильтрации) файлов, отображенных в окне. Установка фильтра производится следующим образом. Выделив соответствующий компонент, необходимо дважды щелкнуть по правой (белой) части свойства Filter Инспектора Объектов. В появившемся окне редактора фильтра – ***Filter Editor*** необходимо в колонке *Filter Name* набрать текст, характеризующий соответственный фильтр, а в колонке *Filter*-маску. Для компонента ***OpenDialog1*** установим значение масок, как показано на рисунке 2.2.

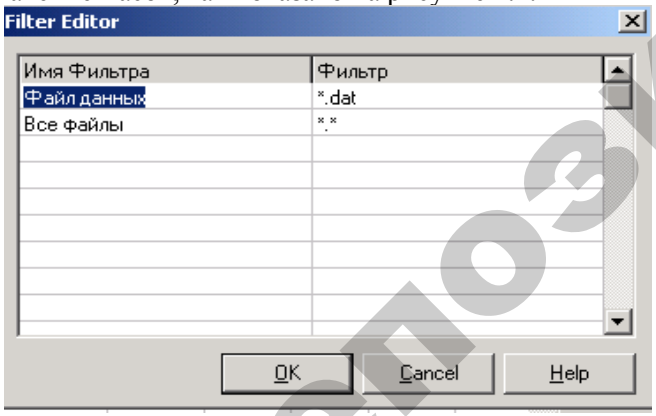


Рис. 2.2. Окно редактора фильтра

Маска *.dat означает, что будут видны файлы данных с любым именем и с расширением dat, а маска *.* – что будут видны все файлы (с любым именем и с любым расширением).

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

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

Кроме того, на Форме размещаются два компонента ***StringGrid***. Первый предназначен для ввода, корректировки, добавления, удаления, просмотра исходной информации, второй – для вывода списка первоочередных претендентов на места в общежитии в порядке убывания среднего балла. Следует разрешить ввод данных в компоненты ***StringGrid***, установив в True значение опции *goEditing* свойства +***Options.***

Три компонента ***BitBtn*** (переносятся со страницы ***Additional*** палитры компонентов) предназначены для обработки файлов: первый – для сохранения файла на внешний носитель, второй – для открытия файла, третий – для создания текстового файла.

Для удобства работы с несколькими различными процедурами обработки событий в свойстве ***Name*** каждого компонента ***BitBtn*** заменить программные имена кнопок: ***BitBtn1*** – на ***BitBtnNew***, ***BitBtn2*** – на ***BitBtnOpen***, ***BitBtn3*** – на ***BitBtnSave***. В свойстве ***Caption*** каждого компонента ввести надпись, указывающую выполняемую процедуру.

Кроме того, на Форму наносятся две кнопки, выполняющие процедуры: сортировки записей исходного списка и вывод списка первоочередников. В свойстве ***Caption*** каждого из этих компонентов также ввести надпись, указывающую выполняемую процедуру.

Для ввода минимальной зарплаты размешается компонент ***Edit***. Компонент ***SpinEdit*** предназначен для изменения количества записей исходной информации и переносится со страницы ***Samples***.

2.1.2. Сохранение проекта

Для нового проекта создайте новую папку, например ***X:\41эи\Ленартович\Mod3\Lab2***.

Сохраните проект ***File | Save Project As*** Сначала сохраните модуль с именем **UnFile.pas**, затем файл проекта под именем **PrFile.dpr.**

Последующие сохранения выполнять командами ***File | Save All.***

2.1.3. Создание процедур обработки событий

Двойным нажатием клавиши мыши в свободном поле Формы открываем процедуру обработки события ***FormCreate*** и вводим текст процедуры.

Процедуры обработки событий нажатия кнопок создаются так же, как в предыдущих работах. Вывод этих процедур на экран осуществляется двойным щелчком клавиши мыши по компонентам. Процедура обработки события ***SpinEditChange*** выводится двойным щелчком клавиши мыши по компоненту ***SpinEdit***.

Пользуясь текстом модуля ***UnFile***, внимательно наберите операторы этих процедур.

2.1.4. Работа с приложением

Выполните созданное приложение. Занесите в соответствующе поля компонента ***StringGrid1*** исходную информацию. Кнопкой «Сохранить файл» сохраните данные в файле. Завершите выполнение приложения.

Вновь запустите приложение, кнопкой «Открыть файл» откройте только что созданный файл. Убедитесь, что информация не содержит ошибок. При необходимости обнаруженные ошибки можно исправить, а также дополнить ведомость новой информацией и снова нажать кнопку «Сохранить файл».

Для сортировки ведомости в порядке возрастания дохода на члена семьи нажмите кнопку «Сортировка» и сохраните отсортированную информацию в текстовый файл (кнопка «Сохранить текстовый файл»). Далее выведите список первоочередников, воспользовавшись соответствующей кнопкой.

Результат выполнения приложения приведен на рисунке 2.3.

Еще раз завершите и вновь запустите приложения.

Кнопкой «Открыть файл» откройте файл и убедитесь, что в нем теперь содержится ведомость, отсортированная в порядке возрастания дохода на члена семьи. Кнопкой «Создать текстовый файл» сохраните информацию в текстовом файле. Для просмотра содержимого текстового файла воспользуйтесь, например, приложением «Microsoft Word».

Используя все управляющие компоненты панели, убедитесь в правильном функционировании приложения во всех предусмотренных режимах работы. После каждой корректировки приложения следует выполнять сохранение приложения с помощью пунктов ***File*** | ***Save All***.

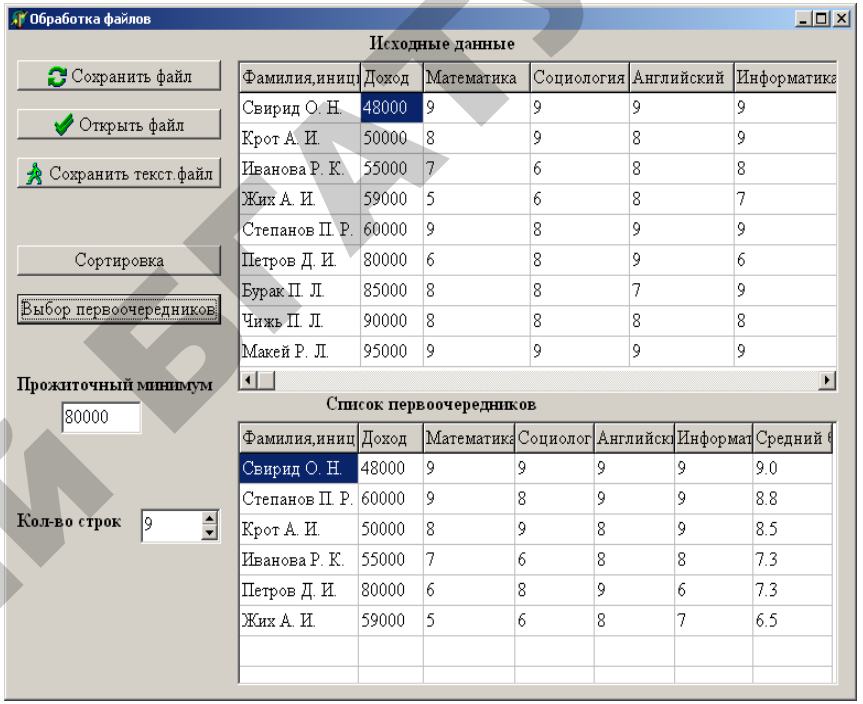


Рис. 2.3. Интерфейс приложения после его выполнения

Текст программы приведен в листинге 2.1.

Листинг 2.1

```
unit UnFile;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, Grids, StdCtrls, Buttons, Spin;
type
  TForm1 = class(TForm)
    StringGrid1: TStringGrid;
    StringGrid2: TStringGrid;
    OpenDialog1: TOpenDialog;
```

```
    SaveDialog1: TSaveDialog;
    Button1: TButton;
    Button2: TButton;
    Edit1: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    BitBtnNew: TBitBtn;
    BitBtnOpen: TBitBtn;
    BitBtnSave: TBitBtn;
    SpinEdit1: TSpinEdit;
    procedure FormCreate(Sender: TObject);
    procedure BitBtnNewClick(Sender: TObject);
    procedure BitBtnOpenClick(Sender: TObject);
    procedure BitBtnSaveClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure SpinEdit1Change(Sender: TObject);
    private
     { Private declarations }
  public
     { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
type
zap=record          // объявление записи
    fio    :string[20];
    dohod: integer;
    mat,soch,english,inf: integer;
    end;
zap1=record         // объявление записи
    fio    : string[20];
    dohod: integer;
    mat,soch,english,inf: integer;
```



```
    srball: Real;
    end;
var
MZap: array[1..10]of zap; //объявление массива записей
MZap1: array[1..10]of zap1;
FileZap: file of zap;          // объявление файла записей
FileText:TextFile;             // объявление текстового файла
FileNameZap,FileNameText: string; // имена файла записей и текстового файла
n: integer;    // текущее количество исходных записей
{ Обработчик события создания Формы}
procedure TForm1.FormCreate(Sender: TObject);
begin
Edit1.Text:='80000';  // начальное значение X (прожиточного минимума)
SpinEdit1.Text:='9'; // начальное значение для  количества записей
n:=9;
 with StringGrid1 do
 begin
 Cells[0,0]:='Фамилия, инициалы';
 Cells[1,0]:='Доход';
 Cells[2,0]:='Математика';
 Cells[3,0]:='Социология';
 Cells[4,0]:='Английский';
 Cells[5,0]:='Информатика';
end;
 with StringGrid2 do
 begin
 Cells[0,0]:='Фамилия, инициалы';
 Cells[1,0]:='Доход';
 Cells[2,0]:='Математика';
 Cells[3,0]:='Социология';
 Cells[4,0]:='Английский';
 Cells[5,0]:='Информатика';
 Cells[6,0]:='Средний балл';
end;

BitBtnSave.Hide; // Спрятать кнопку "Создать текстовый файл"
end;
```

```
{ Сохранения файла записей}
procedure TForm1.BitBtnNewClick(Sender: TObject);
var
i:integer;
begin
 if MessageDlg('Содержимое существующего файла будет уничто-
жено.Вы уверены?',mtConfirmation,mbYesNoCancel,0)=mrYes then
 begin
 for i:=1 to n do
  with StringGrid1,MZap[i] do
   begin
    fio:=Cells[0,i];
    dohod:=StrToInt (Cells[1,i]);
    mat:=StrToInt(Cells[2,i]);
    soch:=StrToInt(Cells[3,i]);
    english:=StrToInt(Cells[4,i]);
    inf:=StrToInt(Cells[5,i]);
   end;
  with OpenDialog1 do
   begin
    Title:='Создание файла';
     if Execute then
      begin
      FileNameZap:=FileName;
      AssignFile(FileZap,FileNameZap);
      ReWrite(FileZap);
      for i:=1 to n do
       write(FileZap,MZap[i]);
       CloseFile(FileZap);
      end;
   end;
 end;
end;
{ Открытие файла}
procedure TForm1.BitBtnOpenClick(Sender: TObject);
var
i:integer;
begin
```



```
 with OpenDialog1 do
  begin
   Title:='Открытие файла'; // заголовок окна диалога
   if Execute then        // выполнение стандартного диалога выбора
имени файла
   begin
   FileNameZap:=FileName; // присваивание имени файла переменной
   AssignFile(FileZap,FileNameZap);
   ReSet(FileZap);        // открыть файл для чтения
   n:=0;
   while not EoF(FileZap) do
    begin
     n:=n+1;              // расчет количества записей в файле
     read(FileZap,MZap[n]); // чтение записей файла в массив записей
    end;
   SpinEdit1.Text :=IntToStr(n);
   StringGrid1.RowCount :=n+1;
   StringGrid2.RowCount :=n+1;
   for i:=1 to n do
    with StringGrid1,MZap[i] do
     begin
      Cells[0,i]:=fio;
      Cells[1,i]:=IntToStr(dohod);
      Cells[2,i]:=IntToStr(mat);
      Cells[3,i]:=IntToStr(soch);
      Cells[4,i]:=IntToStr(english);
      Cells[5,i]:=IntToStr(inf);
     end;
    CloseFile(FileZap);          // закрытие файла
    end;
  end;
  BitBtnSave.Show;   // показать кнопку «сохранить текстовый файл»
  end;
{ Запись текстового файла}
procedure TForm1.BitBtnSaveClick(Sender: TObject);//запись тек-
стового файла
Var
i:integer;
```

```
begin
with SaveDialog1 do
if Execute then // выполнение стандартного диалога выбора имени
файла
 begin
 FileNameText:=FileName;//присваивание имени файла
 AssignFile(FileText,FileNameText);//назначить файлу FileText имя
//FileNameText
  ReWrite(FileText);     //открыть текстовый файл на запись
  for i:=1 to n do
   with MZap[i] do        //запись в текстовый файл
   writeln(FileText, i:3, fio:20, mat:5, english:5, soch:5, inf:5);
  CloseFile(FileText);//закрытие текстового файла по окончании
записи
  end;
 end;
{ Сортировка исходных записей по возрастанию дохода на члена
семьи}
procedure TForm1.Button1Click(Sender: TObject);
var
i,j:integer;
vper:zap;
begin
for i:=1 to n do        // пересылка данных в массив записей
 with StringGrid1,MZap[i] do
 begin
 fio:=Cells[0,i];
 dohod:=StrToInt(Cells[1,i]);
 mat:=StrToInt(Cells[2,i]);
 soch:=StrToInt(Cells[3,i]);
 english:=StrToInt(Cells[4,i]);
 inf:=StrToInt(Cells[5,i]);
 end;
{Сортировка методом пузырька}
 for i:=2 to n do
  for j:=n downto i do
   if MZap[j-1].dohod>MZap[j].dohod then
                 begin
```



```
                  vper:=MZap[j-1];
                  MZap[j-1]:=MZap[j];
                  MZap[j]:=vper;
                 end;
{Пересылка рассортированных данных из массива записей в
StringGrid1}
 for i:=1 to n do
  with stringgrid1,mzap[i] do
   begin
    Cells[0,i]:=fio;
    Cells[1,i]:=IntToStr(dohod);
    Cells[2,i]:=IntToStr(mat);
    Cells[3,i]:=IntToStr(soch);
    Cells[4,i]:=IntToStr(english);
    Cells[5,i]:=IntToStr(inf);
   end;
  end;
  {Вывод списка первоочередников в массив Mzap1 и его сортировка
в порядке убывания среднего балла}
procedure TForm1.Button2Click(Sender: TObject);
var i,j:integer;
dd:integer;
k: integer;  // кол-во первоочередников
vper: Zap1;
begin
dd:=StrToInt(Edit1.Text); // ввод значения прожит. минимума в пе-
ременную   dd
 for i:=1 to n do
  with StringGrid2 do // очистка StringGrid2
   begin
    Cells[0,i]:='';
    Cells[1,i]:='';
    Cells[2,i]:='';
    Cells[3,i]:='';
    Cells[4,i]:='';
    Cells[5,i]:='';
   end;
 for i:=1 to n do
```

```
 with StringGrid1,MZap[i] do    // ввод данных в массив записей
  begin
   fio:=Cells[0,i];
   dohod:=StrToInt (Cells[1,i]);
   mat:=StrToInt(Cells[2,i]);
   soch:=StrToInt(Cells[3,i]);
   english:=StrToInt(Cells[4,i]);
   inf:=StrToInt(Cells[5,i]);
  end;
 k:=0;
for i:=1 to n do
   if MZap[i].dohod<=dd then        // выбор списка первой очереди
    begin
     k:=k+1;
     MZap1[k].fio:=MZap[i].fio;
     MZap1[k].dohod:=MZap[i].dohod;
     MZap1[k].mat:=MZap[i].mat;
     MZap1[k].soch:=MZap[i].soch;
     MZap1[k].english:=MZap[i].english;
     MZap1[k].inf:=MZap[i].inf;

MZap1[k].srball:=(MZap[i].mat+MZap[i].soch+MZap[i].english+MZap
[i].inf)/4
    end;
 {Сортировка списка первоочередников по убыванию среднего балла}
 for i:=2 to k do
  for j:=k downto i do
   if MZap1[j-1].srball<MZap1[j].srball then
                 begin
                  vper:=MZap1[j-1];
                  MZap1[j-1]:=MZap1[j];
                  MZap1[j]:=vper;
                 end;
  for i:=1 to k do       // вывод списка первоочередников в StringGrid2
   with stringgrid2, mzap1[i] do
    begin
     Cells[0,i]:=fio;
     Cells[1,i]:=IntToStr(dohod);
```



```
     Cells[2,i]:=IntToStr(mat);
     Cells[3,i]:=IntToStr(soch);
     Cells[4,i]:=IntToStr(english);
     Cells[5,i]:=IntToStr(inf);
     Cells[6,i]:=FloatToStrF(srball,ffFixed,2,1);
    end;
   end;
  {Изменение количества исходных записей}
procedure TForm1.SpinEdit1Change(Sender: TObject);
var
 i, m: integer;
begin
 m:=StrToInt(SpinEdit1.Text);
 With StringGrid1 do
 begin
  RowCount:=m+1;
  if m>n then
   for i:=n+1 to m do
    begin
     Cells[0,i]:='';
     Cells[1,i]:='';
     Cells[2,i]:='';
     Cells[3,i]:='';
     Cells[4,i]:='';
     Cells[5,i]:='';
    end;
 end;
 With StringGrid2 do
 begin
  RowCount:=m+1;
  if m>n then
   for i:=n+1 to m do
    begin
     Cells[0,i]:='';
     Cells[1,i]:='';
     Cells[2,i]:='';
     Cells[3,i]:='';
     Cells[4,i]:='';
```

```
    Cells[5,i]:='';
   end;
  end;
  n:=m;
 end;
end.
```

2.2. Индивидуальные задания

По указанию преподавателя выберите свое индивидуальное задание. Создайте приложение и протестируйте его работу.

Индивидуальные задания 1-го уровня

В задаче предусмотреть сохранение исходных данных в типизированном файле и возможность чтения из созданного файла.

1. В магазине формируется список лиц, записавшихся на покупку товара повышенного спроса. Каждая запись этого списка содержит: порядковый номер, Ф.И.О., домашний адрес покупателя и дату постановки на очередь, дату выполнения заказа. Вывести список выполненных заказов в порядке возрастания даты выполнения.

2. Список товаров, имеющихся на складе, включает в себя наименование товара, количество единиц товара, цену единицы и дату поступления товара на склад. Вывести список товаров, стоимость которых превышает 1000000 руб., а также общую стоимость всех таких товаров.

3. Информация об участниках спортивных соревнований содержит: наименование страны, название команды, Ф.И.О. игрока, игровой номер, возраст, рост, вес. Вывести информацию о самом молодом участнике, а также список участников, рост которых больше двух метров.

4. Для книг, хранящихся в библиотеке, задаются: регистрационный номер книги, автор, название, год издания, издательство, количество страниц. Сформировать и вывести в алфавитном порядке названий список книг, изданных за последние 4 года.

5. Ведомость абитуриентов, сдавших вступительные экзамены в университет, содержит Ф.И.О. абитуриента, оценки по трем дисциплинам. Определить средний балл каждого абитуриента и вывести список абитуриентов в порядке убывания среднего балла.

6. В радиоателье хранятся квитанции о сданной в ремонт радио-

аппаратуре. Каждая квитанция содержит следующую информацию: наименование группы изделий (телевизор, радиоприемник и т. п.), марка изделия, дата приемки в ремонт, состояние готовности заказа (выполнен, не выполнен). Вывести список выполненных заказов, рассортировав их по группам изделий.

7. В исполкоме формируется список учета нуждающихся в улучшении жилищных условий. Каждая запись этого списка содержит: порядковый номер, Ф.И.О., величину жилплощади на одного члена семьи и дату постановки на очередь. Вывести список очередников по возрастанию величины жилплощади на каждого члена семьи.

8. Для определения потребности в горючем техническая служба запрашивает расписание полетов. Каждая запись расписания содержит следующую информацию: дату вылета, номер рейса, пункт назначения, дальность полета, расход горючего на 1000 км пути. Вывести суммарное количество горючего, необходимое дли обеспечения полетов на следующие сутки.

9. В библиотеке имеется список книг. Каждая запись этого списка содержит фамилию автора, название книги, год издания. Вывести информацию о книгах заданного автора, изданных в текущем году, в алфавитном порядке наименований.

10. Имеется информация о работе бригад рабочих на строительном объекте. Каждая строка содержит название бригады, фамилию рабочего, количество отработанных дней, дневной тариф. Вывести данные в алфавитном порядке наименований бригад с подсчетом заработанных сумм по формуле (дневной тариф · количество отработанных дней).

Индивидуальные задания 2-го уровня

Использовать текст задач 1–15. В задаче предусмотреть сохранение вводимых данных в типизированном и текстовом файлах и возможность чтения из созданного типизированного файла. Просмотреть полученный текстовый файл в текстовом редакторе «Microsoft Word».

11. Имеется список женихов и список невест. Каждая запись списка содержит пол, имя, возраст, рост, вес, а также требования к партнеру: наименьший и наибольший возраст, наименьший и наибольший вес, наименьший и наибольший рост. Для первого в списке (по алфавиту имени) жениха составить список возможных невест.

12. Есть информация о поступлениях на склад сельскохозяйственной продукции за месяц: дата, наименование продукта, количе-

ство, цена, наименование производителя. Вывести наименование продукта, поступление которого на склад было наибольшим в стоимостном выражении.

13. Имеется информация о деятельности медицинского диагностического центра в текущем месяце: дата, фамилия врача, фамилия пациента, услуга, стоимость. Вывести информацию об оказании заданной услуги в текущем месяце в алфавитном порядке по фамилии врача.

14. Имеется следующая информация о сотрудниках предприятия: Ф.И.О., должность, подразделение, дата поступления на работу, год рождения. Вывести в алфавитном порядке список сотрудников, работающих в заданной должности.

15. Имеется информация о проведении уборочной компании зерновых культур в некотором хозяйстве: дата, бригада, культура, убрано (т), убранная площадь. Вывести информацию об уборке заданной культуры в порядке убывания убранного количества.

Индивидуальные задания 3-го уровня

Использовать текст задач 16–20. В задаче предусмотреть сохранение исходных данных в типизированном и текстовом файлах и возможность чтения из созданного типизированного файла. Просмотреть полученный текстовый файл в текстовом редакторе «Microsoft Word».

16. Каждая запись списка вакантных рабочих мест содержит: район города, наименование организации, должность, квалификацию (разряд), заработную плату, наличие социального страхования (да/нет), продолжительность ежегодного оплачиваемого отпуска, количество вакантных ставок. Рассчитать суммарное количество рабочих мест в каждом районе. Вывести список рабочих мест в соответствии с требованиями клиента.

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

18. Список студентов содержит следующую информацию: группа, Ф.И.О., рост и вес. Рассчитать количество студентов в каждой группе. Вывести Ф.И.О. студентов, рост и вес которых совпадает с наименьшим.

19. Имеется следующая информация о функционировании предприятия, занимающегося грузоперевозками: дата, автомобиль, пункт назначения, наименование груза, вес груза. Вывести информацию о перевозках в заданный пункт назначения. Рассчитать суммарный вес груза, перевезенный в каждый из пунктов назначения.

20. Имеется информация о состоянии вкладов в банке на текущий момент: дата, клиент, вид вклад, сумма вклада, процентная ставка. Определить самый популярный вид вклада и суммарное количество вложений для каждого вида вклада.

Вопросы для самоконтроля

1. Какова сущность понятия файл?
2. Какими средствами организации файлов располагает Delphi?
3. Какое назначение компонентов ***OpenDialog*** и ***SaveDialog***?
4. Какие функции выполняет компонент ***SaveDialog***?
5. На какой странице находятся компоненты ***OpenDialog*** и ***SaveDialog***?

МАТЕРИАЛЫ К ЛАБОРАТОРНОЙ РАБОТЕ № 3

3. Программирование с использованием подпрограмм и модулей

Цель работы **–** изучить возможности Delphi для составления подпрограмм и создания модулей. Освоить методику применения подпрограмм в качестве параметров.

Концепция модульного программирования

Термин модульное программирование означает разработку программ, состоящих из отдельных модулей исходного кода. В каждом модуле может находиться произвольное количество подпрограмм. Структура результирующей программы должна отражать структуру алгоритма решаемой задачи. ***Структурирование программы*** **–** это ее разбивка на отдельные модули и подпрограммы. Программы, написанные на Object Pascal, структурированы по своей природе, поскольку сам язык строго структурирован. Более того, все приложения, разработанные в среде Delphi, являются модульными программами, потому что среда разработки создает отдельные модули исходного кода.

Разработка программ на основе принципов модульного программирования предоставляет ряд существенных преимуществ.

Во-первых, средства модульного программирования позволяют логически организовать программу путем разделения задачи на более простые подзадачи. Такое разделение задачи называется ***пошаговым решением*****,** оно является составной частью методологии ***разработки сверху вниз***. Например, программист может создать программу, состоящую из трех подпрограмм: одна подпрограмма содержит процедуры ввода, другая выполняет вычисления, а третья выводит результаты вычислений на экран или в файлы.

Во-вторых, модульное программирование облегчает отладку и делает код более понятным.

В-третьих, программу можно модернизировать путем замены отдельных модулей, не затрагивая другие модули.

И, наконец, концепция модульного программирования позволяет использовать разработанные коды повторно. Повторное использование кодов – важнейший принцип компьютерного программирования.

Примеры применения ***системных*** подпрограмм рассматривались в предыдущих разделах – например, стандартной (встроенной) процедуры ShowMessage, функций StrToInt, FloatToStr, математических функций и др. ***Системными*** они называются потому, что созданы одновременно с системой Delphi и являются ее неотъемлемой частью. При отсутствии нужных аналогов в библиотеках Delphi для разработки прикладных программ приходится разрабатывать свои (***подпрограммы программиста)***.

Подпрограммы

Подпрограмма – это именованная, определенным образом оформленная группа операторов, которая может быть вызвана любое количество раз из любой точки программы. Подпрограмма используется в том случае, когда одна и та же последовательность операций программируемого алгоритма повторяется многократно. Последовательность операторов, программирующих эти операции, представляется подпрограммой, к которой осуществляется обращение из произвольной области основной программы.

Подпрограммы также применяются для создания специализированных библиотечных модулей, содержащих набор подпрограмм определенного назначения, для использования их другими программистами.

Вызов подпрограммы – это активизация требуемой подпрограммы. Оператор, с помощью которого осуществляется вызов подпрограммы, называется ***оператором вызова***. Операторы, находящиеся внутри подпрограммы, выполняются только тогда, когда подпрограмма явно вызвана. Подпрограммы могут быть вложенными – допускается вызов подпрограммы и из других подпрограмм.

Обычно данные передаются подпрограмме в виде параметров **(*аргументов*)**, которые описываются в заголовке подпрограммы. Во время создания подпрограммы заранее неизвестно, какие конкретно параметры она будет получать. Поэтому в качестве переменных, выступающих в роли ее аргументов в заголовке, могут использоваться произвольные допустимые имена.

Параметры, которые указываются в заголовке подпрограммы, называются ***формальными***. Они нужны только для написания операторов подпрограммы. Параметры, которые указываются в момент вызова подпрограммы, называются ***фактическими***. При выполнении операторов подпрограммы формальные параметры заменяются фактическими.

Подпрограммы подразделяются на ***процедуры*** и ***функции***.

Процедура имеет следующую структуру:

procedure <**имя процедуры**> ([список формальных параметров с указанием их типов]);

```
  const [описание используемых констант];
  type  [описание используемых типов];
  var   [описание используемых переменных];
begin
  // операторы процедуры
end;
```

Объявления формальных параметров, а также констант, типов и переменных действительны только в пределах данной процедуры. Такие данные называются ***локальными***. Внутри процедуры можно использовать любые глобальные данные, объявленные в вызывающей процедуре, а также вызывать любые функции и процедуры.

Вызов процедуры записывается в следующем виде:

<**имя процедуры**> ([список имен фактических параметров без указания их типов]);

Пример**.** Создать приложение для вычисления площади фигуры состоящей из отдельных кругов. Вычисления площади круга оформить как подпрограмму – процедуру. Фрагмент листинга программы приведен ниже.

```
{$R *.dfm}
...
procedure Area(r:extended;var z:extended);
  begin
    z:= pi*r*r;
  end;
procedure TForm1.Button1Click(Sender: TObject);
  var
     r1,r2,s1,s2,s:extended;
  begin
    r1:=StrToFloat(Edit1.Text);
    r2:=StrToFloat(Edit2.Text);
     area(r1,s1);
     area(r2,s2);
     s:=s1+s2;
     Panel1.Caption := FloatToStr(s);
```



```
  end;
end.
```

В рассмотренном примере обращение к процедуре Area осуществляется из процедуры TForm1.Button1Click программы. Передача результата выполняется через параметр *z*, объявленный в перечне формальных параметров вызываемой процедуры Area после ключевого слова **var**. Для ввода данных в примере используются компоненты Edit1 и Edit2, а для вывода – компонент Panel1. Значение слова **var** примера рассматривается далее.

Функции по структуре подобны процедурам, однако имеются некоторые отличия. Результатом работы функции является ***одно значение***, тип которого задан в заголовке функции. Функции (в отличие от процедур) могут использоваться в выражениях в качестве операнда.

Функция имеет следующую структуру:

function <**Имя Функции**> ([список имен формальных параметров с указанием их типов]) **:** <тип результата»;

```
  const [описание используемых  констант];
  type  [описание используемых типов];
  var   [описание используемых переменных];
begin
   Операторы функции
   <Имя Функции> :=  ... ;
     result := ...;
end;
```

Список формальных параметров, если они есть, содержит произвольные имена переменных, которые будут использоваться в качестве параметров, с указанием их типа. Обязательно объявление типа результата. Объявления формальных параметров, а также констант, типов и переменных (как в процедурах) действительны только в пределах данной функции как локальных переменных.

Внутри функции можно использовать любые глобальные (т. е. объявленные в вызывающей процедуре) данные, а также вызывать любые функции и процедуры.

Последним выполняемым оператором функции обязательно должен быть оператор присваивания, в котором результат вычис-

ления значения функции присваивается либо переменной ***Имя-Функции***, либо зарезервированной переменной ***result***.

Вызов функции осуществляется следующим образом:

<**Имя Функции** > ([список имен фактических параметров без указания их типов]);

Функция возвращает результат выполнения операции под своим именем, тип величины которой указан вслед за двоеточием в заголовке функции.

Пример.

```
function sec(x:extended):extended;
  begin   rezult := 1/cos(x);   end;
```

Пример обращения к этой подпрограмме функции:

```
z := sec(g) / sec(0.34);
```

Имя процедуры и функции должно быть уникальным в пределах программы. Список формальных параметров необязателен и может отсутствовать. Если же он есть, то в нем перечисляются через точку с запятой имена формальных параметров и их типы.

Имеется три вида формальных параметров:

параметры-значения,
параметры-переменные,
параметры-константы.

При вызове подпрограммы передача данных для этих видов осуществляется по-разному.

Параметры-значения копируются, и подпрограмма работает с их копией. В подпрограмму передаются значения фактических параметров, указанных при вызове подпрограммы, где создаются копии этих значений, и все вычисления производятся с этими копиями. Поэтому изменение значений формальных параметров внутри подпрограммы в этом случае не приводит к изменению соответствующих им фактических параметров после выполнения подпрограммы. При использовании параметров-значений не ставится никаких ключевых слов при описании формальных параметров. Применение параметров-значений требует дополнительных затрат памяти.

Параметры-переменные передаются в подпрограмму путем передачи их адресов, и она работает непосредственно с фактическими параметрами. В описании формальных параметров перед ними ставится ключевое слово **var**. Применение параметров-переменных не требует дополнительных затрат памяти. Все вычисления в подпрограмме производятся непосредственно с фактическими параметрами и, следовательно, изменение значений формальных параметров

в подпрограмме приводит к изменению фактических параметров, указанных при вызове подпрограммы. Передача результатов работы подпрограммы вызывающей программе организуется через параметры-переменные.

Параметры-константы передаются в подпрограмму путем передачи адресов констант. Она работает непосредственно с фактическими параметрами. Перед этими параметрами в списке формальных параметров записывается ключевое слово **const**. Параметры-константы подобны параметрам-переменным, но внутри подпрограмм таким формальным параметрам нельзя присваивать никаких значений и их нельзя передавать в другие подпрограммы, как параметры-переменные.

Если формальный параметр – параметр-константа или параметр-значение, то в качестве фактических параметров могут использоваться арифметические выражения.

Использование подпрограмм в качестве параметров

Подпрограммы могут использоваться в качестве формальных параметров подпрограмм. Переменные, которым в качестве значений присваиваются имена процедур и функций, называются соответственно ***переменными процедурных*** и ***функциональных типов***.

Для процедур или функций, которые будут передаваться как параметр, необходимо вначале объявить их тип. Объявление типов подпрограмм, применяемых в качестве параметров, осуществляется после ключевого слова **type**.

Для объявления типа переменных процедурного или функционального типа необходимо в разделе **type** указать ***имя типа***, знак «=» и заголовок процедуры или функции без указания имени:

type <имя>= **procedure**([список формальных параметров]);
type <имя>= **function**([список формальных параметров]):<тип результата>;

Например,

```
type
  functip1=function(x: integer): integer;
  proctip5=procedure(x:integer; var s:extended);
```

Здесь functip1 и proctip5 – имена типов переменных соответственно функционального и процедурного типов.

Возможно применение процедурных типов без параметров. Например:

type
ProcType = **procedure**;

Итак, при объявлении типа переменных процедурного и функционального типов их имена не записываются, но обязательно перечисляются типы их параметров. При объявлении функционального типа указывается тип результата. Имя процедурного или функционального типов должно быть уникальным в пределах программы. Список формальных параметров может отсутствовать.

Использование модулей

Модуль – автономно компилируемая программная единица, подобная программному модулю, включающая в себя процедуры, функции, а также различные компоненты раздела описаний, в том числе переменные процедурных функциональных типов. Разместив некоторые процедуры и функции в отдельном ***модуле программиста***, можно затем использовать их в различных программах, указав имя модуля в списке модулей соответствующих программ.

Список модулей, подключаемых к проекту или используемых в данном программном модуле других модулей, начинается с ключевого слова **uses**, после которого следует список имен модулей, разделенных запятыми. Этот список дополняется именем подключаемого модуля. Например:

uses Windows, SysUtils, Classes, Unit2, *MyUnit*;

Предложение **uses** в головном файле программы может также содержать после имени модуля ключевое слово **in**, после которого указывается имя файла, содержащего модуль. Например:

uses Unit1 **in** ‘Unit1.pas’, *MyUnit* **in** ‘d:\student*MyUnit*.pas’;

Модуль описывается в отдельном текстовом файле с расширением *.pas и компилируется отдельно. Имя модуля должно совпадать с именем файла, в котором хранится исходный текст модуля. Результатом компиляции является машинный код, который записывается в файл с расширением *.dcu.

Модуль имеет следующую структуру.

Unit имя модуля; *// заголовок модуля*
Interface *// интерфейсная часть*
uses имена модулей; *// подключение модулей*
const . . . ; *// глобальные константы*

type . . . ; *// глобальные типы*
var . . . ; *// глобальные переменные*
procedure имя(параметры); *// заголовки процедур*
function имя(параметры):тип; *// заголовки функций*
Implementation *// часть реализации (исполняемая часть)*
uses имена модулей; *// подключение модулей*
const . . . ; *// локальные константы*
type . . . ; *// локальные типы*
var . . . ; *// локальные переменные*
procedure имя(параметры); *// реализация процедур*
begin . . . end;
function имя(параметры):тип; *// реализация функций*
begin . . . end;
Initialization *// инициирующая часть (может отсутствовать)*
// Операторы инициализирующей части
Finalization *// завершающая часть (может отсутствовать)*
// Операторы завершающей части
end.

В секции **Interface** описываются глобальные данные, процедуры и функции, доступные для использования в основной программе и других модулях.

В секции **Implementation** описываются локальные данные, процедуры и функции, которые будут доступны только внутри данного модуля и недоступны основной программе и другим модулям. Здесь же дается реализация всех процедур и функций, объявленных в интерфейсной секции (Interface) модуля.

Операторы секции **Initialization** выполняются перед запуском основной программы и используются для подготовки ее работы, например, инициализируются переменные, открываются файлы и т.д.

Операторы секции **Finalization** выполняются после завершения основной программы. Если модуль не нуждается в инициализации и завершении, секции **Initialization** и **Finalization** могут отсутствовать.

Подключение модулей к основной программе и их компиляция осуществляются в порядке их объявления в **Uses**. При переходе к очередному модулю система предварительно находит все модули, на которые он ссылается.

3.1. Пример создания приложения с использованием подпрограмм

Задание. Создать Windows-приложение, которое обеспечивает вывод на экран таблицы значений функции $y(x) = e^x$, вычисленных по формуле и численно в виде разложения в степенной ряд $s(x) = 1 + 1/x + \ldots + x^n/n!$ с заданной точностью eps. Вычисления функции $y(x)$ оформить в виде подпрограммы-функции, а $s(x)$ – в виде подпрограммы-процедуры. Начальное значение аргумента $x - x_n$, конечное значение – x_k , n – требуемое количество вычисляемых значений функций задаются пользователем.

3.1.1. Размещение компонентов на Форме

Панель интерфейса для рассматриваемого примера может быть представлена в виде, показанном на рисунке 3.1.

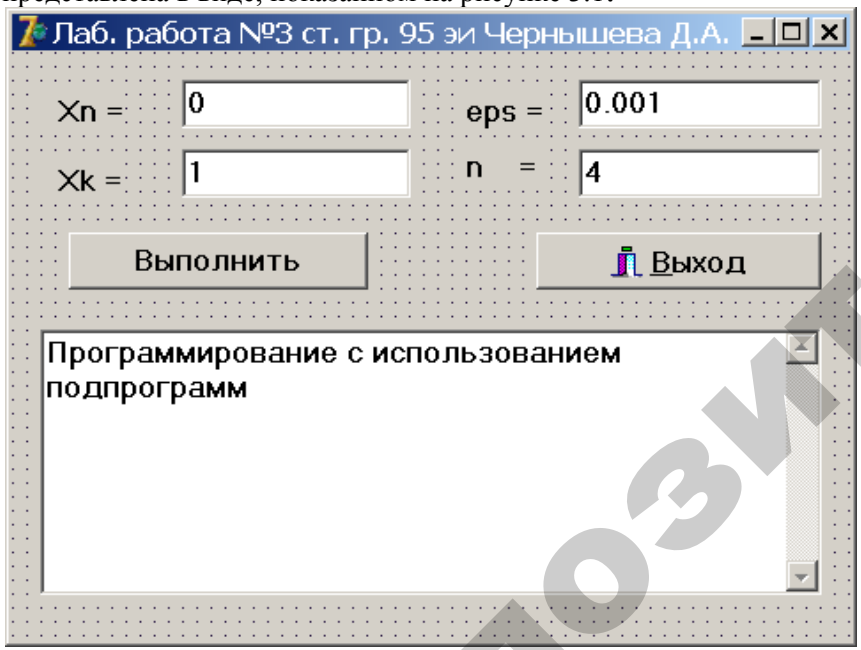


Рис. 3.1. Интерфейс приложения

Текст модуля UnLR3 приведен в листинге 3.1.

Листинг 3.1

```
unit UnLR3;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, StdCtrls, Buttons, Spin;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Button1: TButton;
    BitBtn1: TBitBtn;
    Memo1: TMemo;
    Label4: TLabel;
    Edit3: TEdit;
    Edit4: TEdit;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
 {$R *.dfm}
{Начальная информация интерфейса}
procedure TForm1.FormCreate(Sender: TObject);
begin
 Edit1.Text:='-0,1';          // начальное значение  xn
 Edit2.Text:='1';             // конечное  значение  xk
Edit3.Text:='0,001';     // точность eps
 Edit4.Text:='4';             // количество n
```

```
Memo1.Clear;
{Вывод строки в многострочный редактор Memo1}
Form1.Memo1.Lines.Add('Лаб. работа №3,ст.гр. 95 эи Чернышев Д.А. ');
end;
{Подпрограмма-функция }
Function Fx(x:extended):extended;
 begin
  Fx:=exp(x);
 end;
{Подпрограмма-процедура }
Procedure Sx(x,eps:extended;  var s:extended);
 var
  a: extended;
  k: integer;
 begin
 a:=1; s:=1; k:=0;
 while (abs(a) > eps) do
  begin
   k:=k+1;
   a:=a*x/k;
   s:=s+a;
  end;
 end;
{Процедура обработчика  события нажатия кнопки Выполнить}
procedure TForm1.Button1Click(Sender: TObject);
var
  xn,xk,eps: extended;
  i,n: integer;
  x,h,s,y:extended;
 begin
  xn:= StrToFloat(Edit1.Text);
  xk:= StrToFloat(Edit2.Text);
  eps:=StrToFloat(Edit3.Text);
  n:=  StrToInt(Edit4.Text);
  Memo1.Lines.Add(#9+'   x'+#9+'   s'+#9+'   y'); // заголовок таблицы
```



```
 h:=(xk-xn)/(n-1);              // вычисления величины шага  h
 x:=xn;
 for i:=1 to n do
 begin
  Sx(x,eps,s);          // вызов процедуры  Sx
  y:=Fx(x);             // вызов процедуры  Fx
  Form1.Memo1.Lines.Add(#9+FloatToStrF(x,ffFixed,6,4)+
                                                  // вывод  x
    #9+FloatToStrF(s,ffFixed,6,4)+                // вывод  s
     #9+FloatToStrF(y,ffFixed,6,4));              // вывод  y
  x:=x+h;
 end;
end.
```

3.1.2. Сохранение проекта

Для нового проекта создайте папку и сохраните проект, выполнив команду **File** | **Save Project As….** После этого сначала сохраните модуль под именем, например, UnLR3.pas, затем файл проекта под именем PrLR3.dpr**.**

3.1.3. Работа с приложением

Для запуска программы следует нажать пиктограмму **Run** на Панели инструментов или клавишу F9, либо из меню **Run** выбрать команду **Run**. При этом происходит компиляция модулей, компоновка проекта и создание выполняемого файла. На экране появляется панель интерфейса приложения с результатами, показанными на рисунке 3.2.

Близость значений функции $y(x)$ и ее разложения в степенной ряд $s(x)$ для значений x, изменяющихся от x_n до x_k, указывает на правильность вычислений.

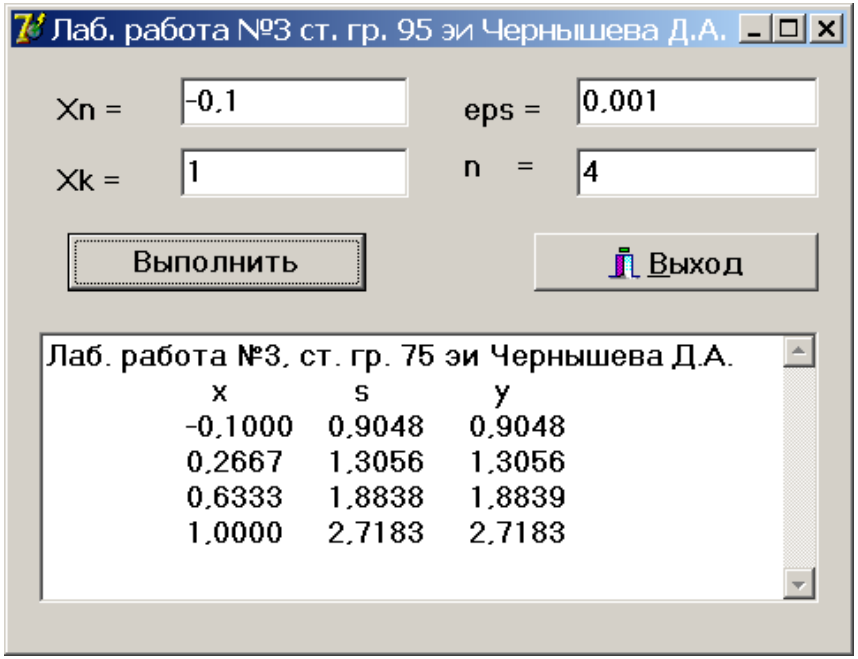


Рис. 3.2. Интерфейс приложения после нажатия кнопки «Выполнить»

3.2. Пример создания приложения с использованием подпрограмм в качестве параметров

Задание. Создать приложение, которое обеспечивает вывод на экран таблицы значений функций $y = e^x$ и $s = 1 + 1/x + \ldots + x^n/n!$, рассмотренных в предыдущем примере, с заданной точностью eps, а также вычисления функции гиперболического синуса $z(x) = Sh(x) = (e^x - e^{-x}) / 2$. Вычисления функций $y(x)$ и $z(x)$ обеспечить в подпрограммах-функциях, а $s(x)$ – в подпрограмме-процедуре. Начальное значение аргумента $x - x_n$, конечное значение – x_k, n – требуемое количество вычисляемых значений функций задаются пользователем. Для вычисления значений функций $y(x)$ и $z(x)$ использовать процедуру, обеспечивающую ввод этих подпрограмм в качестве параметров.

Размещение компонентов на Форме. Интерфейс приложения аналогичен интерфейсу предыдущего примера, показанного на рисунке 3.1.

Текст программы приведен в листинге 3.2.

Листинг 3.2

```
unit UnLR3_functip;
  {$R *.dfm}
  {Ввод начальной информации}
  procedure TForm1.FormCreate(Sender: TObject);
  begin
   Edit1.Text:='-0,1';     // значение  xn
   Edit2.Text:='1';        // значение  xk
   Edit3.Text:='0,001';    // точность eps
   Edit4.Text:='4';        // n (количество значений x)
   Memo1.Clear;
   {Вывод строки в Memo1}
Form1.Memo1.Lines.Add('Лаб. работа №3, ст.гр. 95эи Чернышев Д.А. ');
end;
  {Процедура численного вычисления exp(x)}
  Procedure Sx(x,eps:extended; var s:extended);
   var
    a: extended;
    k: integer;
  begin
   a:=1; s:=1; k:=0;
   while (abs(a) > eps) do
   begin
    k:=k+1;
    a:=a*x/k;
```

```
    s:=s+a;
   end;
  end;
 {Функция вычисления по формуле exp(x)}
 Function Fx(x:extended):extended;
  begin
   Fx:=exp(x);
  end;
 {Функция гиперболического синуса}
 Function Fz(x:extended):extended;
  begin
    Fz:=(exp(x)-exp(-x))/2;
  end;
{Процедура  обработчика события нажатия кнопки Выполнить}
 procedure TForm1.Button1Click(Sender: TObject);
 Type          // объявление функционального типа
  functip1=function(x:extended): extended;
 {Процедура вычисления значений функций}
 procedure Tabl(x:extended; f:functip1; var  y:extended);
   begin
     y := f(x);
   end;
 var
    xn,xk,eps: extended;
    i, n: integer;
```



```
    x,h,s,y,z:extended;
  begin
    xn:= StrToFloat(Edit1.Text);
    xk:= StrToFloat(Edit2.Text);
   eps:=StrToFloat(Edit3.Text);
    n:=  StrToInt(Edit4.Text);
    { Заголовок таблицы }
    Memo1.Lines.Add(#9+'  x'+#9+'  s'+#9+'  y'+#9+'  z');
    h:=(xk-xn)/(n-1);       // вычисление величины шага  h
    x:=xn;
   for i:=1 to n do
  begin
    Sx(x,eps,s);            // вызов процедуры Sx
    Tabl(x,Fx,y);       // вызов функции Fx
    Tabl(x,Fz,z);       // вызов функции Fz
Form1.Memo1.Lines.Add(#9+FloatToStrF(x,ffFixed,6,4)+ // вывод x
          #9+FloatToStrF(s,ffFixed,6,4)+          // вывод s
         #9+FloatToStrF(y,ffFixed,6,4)+           // вывод y
          #9+FloatToStrF(z,ffFixed,6,4));         // вывод z
    x:=x+h;
   end;
  end;
end.
```

Как видно, при обращении к процедуре Tabl в качестве фактического параметра используются имена функций *Fx* и *Fz*. После запуска программы на выполнение на экране отображается панель интерфейса с результатами, показанными на рисунке 3.3.

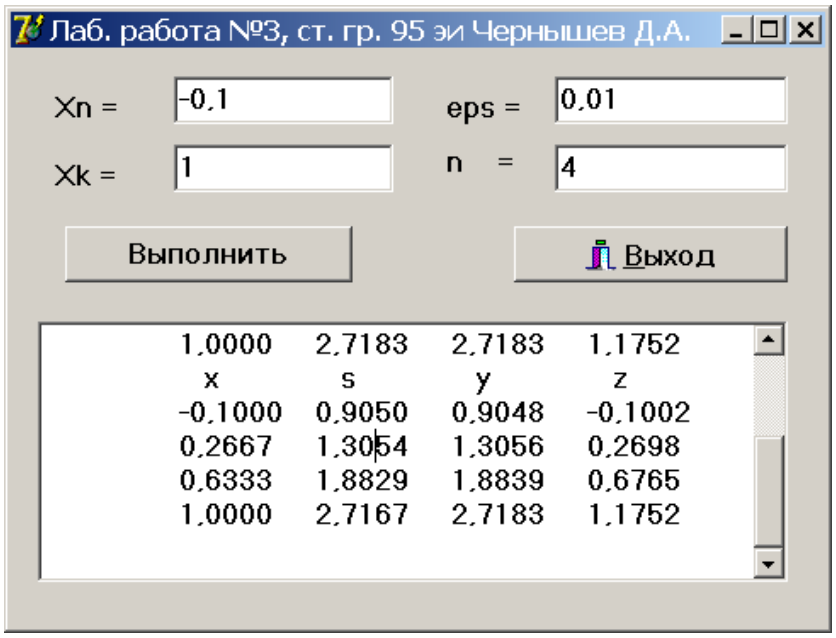


Рис. 3.3. Результаты приложения, в котором функции используются в качестве переменных

Как видно из рассмотренного примера, применение подпрограмм в качестве параметров позволяет легко модифицировать проектируемую программу, вводя новые функции и процедуры. В рассмотренном примере задача свелась к введению строки объявления ***функционального типа*** functip1, процедуры Tabl в процедуре TForm1.Button1Click и новой подпрограммы функции *Fz*. Далее другие подобные функции вводятся аналогично путем расширения списка вводимых подпрограмм.

3.3. Пример приложения с созданием и подключением модуля

Система Delphi при начальной загрузке автоматически создает шаблон программы, имеющий в своем составе форму, модуль, файл проекта и др. В нашем случае требуется создать собственный модуль, который не имеет своей Формы. Поэтому, приступая к созданию модуля, следует сначала закрыть окно Формы и окно модуля Формы командой **File | Close All**.

Для создания модуля программиста следует выполнить команду ***File | New | Unit***. В результате будет создан файл с заголовком **Unit Unit1**. Этот файл следует сохранить с именем, совпадающим с именем заголовка создаваемого модуля. Подключается отдельно созданный модуль командой ***Project | Add to Project...*** После этого в разделе **Uses** соответствующей программы следует добавить имя подключаемого модуля. Далее в проекте могут использоваться подпрограммы, содержащиеся в подключенном модуле.

Задание. Постановка задачи подобна задаче, рассмотренной в предыдущем примере 3.2. В отличие от него требуется разместить подпрограммы вычисления функций *Fx*, *Fz* и *Sx* в отдельном модуле. Присвоить модулю имя Un3m и подключить его к основной программе.

Текст подключаемого модуля программы приведен в листинге 3.3.

Листинг 3.3

```
unit Un3m;
interface
{Функция гиперболического синуса Sh(x)}
Function Fz(x:extended):extended;
{Функция вычисления exp(x)}
Function Fx(x:extended):extended;
{ Процедура численного вычисления exp(x)}
Procedure Sx(x,eps:extended; var s:extended);
implementation
{Функция гиперболического синуса}
Function Fz(x:extended):extended;
 begin
  Fz:=(exp(x)-exp(-x))/2;
 end;
{Функция вычисления exp(x)}
```

```
Function Fx(x:extended):extended;
 begin
  Fx:=exp(x);
 end;
{Процедура численного вычисления exp(x)}
 Procedure Sx(x,eps:extended; var s:extended);
 var
   a: extended;
   k: integer;
 begin
  a:=1; s:=1; k:=0;
 while (abs(a) > eps) do
  begin
   k:=k+1;
   a:=a*x/k;
   s:=s+a;
  end;
 end;
end.
```

Текст основного модуля программы приведен в листинге 3.4.

Листинг 3.4

```
unit UnLR3m;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics,
```



```
Controls,
  Forms,  Dialogs, StdCtrls, Buttons, Spin, Un3m;
type
 TForm1 = class(TForm)
   Label1: TLabel;
   Label2: TLabel;
   Label3: TLabel;
   Edit1: TEdit;
   Edit2: TEdit;
   Button1: TButton;
   BitBtn1: TBitBtn;
   Memo1: TMemo;
   Label4: TLabel;
   Edit3: TEdit;
   Edit4: TEdit;
   procedure FormCreate(Sender: TObject);
   procedure Button1Click(Sender: TObject);
 private
   { Private declarations }
 public
   { Public declarations }
 end;

var
 Form1: TForm1;
implementation
```

```
{$R *.dfm}
{Начальная информация интерфейса}
procedure TForm1.FormCreate(Sender: TObject);
begin
  Edit1.Text:='-0,1';          // значение xn
  Edit2.Text:='1';             // значение xk
  Edit3.Text:='0,001';         // точность
  Edit4.Text:='4';             // количество n
  Memo1.Clear;
 {Вывод строки в многострочный редактор Memo1}
 Form1.Memo1.Lines.Add('Лаб. работа №3 ст. гр. 95 эи Чернышева Д,А.');
end;
{Процедура обработки события нажатия кнопки Выполнить}
procedure TForm1.Button1Click(Sender: TObject);
var
  xn,xk,eps: extended;
  i,n: integer;
  x,h,s,y,z:extended;
 begin
   xn:= StrToFloat(Edit1.Text);
   xk:= StrToFloat(Edit2.Text);
   eps:=StrToFloat(Edit3.Text);
   n:=  StrToInt(Edit4.Text);
   { Заголовок таблицы }
   Memo1.Lines.Add(#9+'  x'+#9+'  s'+#9+'  y'+#9+'  z');
```



```
   h:=(xk-xn)/(n-1);          // вычисление величины шага  h
   x:=xn;
  for i:=1 to n do
   Sx(x,eps,s);               // вызов процедуры Sx
   y:=Fx(x);                  // вызов функции Fx
   z:=Fz(x);                  // вызов функции Fz
   Form1.Memo1.Lines.Add(#9+FloatToStrF(x,ffFixed,6,4)+
                                                  // вывод x
  #9+FloatToStrF(s,ffFixed,6,4)+                  // вывод s
  #9+FloatToStrF(y,ffFixed,6,4)+                  // вывод y
  #9+FloatToStrF(z,ffFixed,6,4));                 // вывод z
    x:=x+h;
  end;
 end;
end.
```

Как видно из программы, подключение модуля Un3m выполняется введением этого имени в конце списка имен основного модуля, следующего после ключевого слова **uses**. Интерфейс приложения и результаты после выполнения приложения подобны примеру, рассмотренному на рисунке 3.3.

3.4. Индивидуальные задания

По указанию преподавателя выберите вариант индивидуального задания определенного уровня. Создайте приложение и протестируйте его работу.

Задания 1-го уровня

Составьте программу вычисления и вывода в таблицу на экране значений выбранной функции $y(x)$ для n-значений аргумента. На-

чальное x_n и конечное x_k значения аргумента функции x и величину n взять из таблицы 3.1. Кроме того, вывести в том же диапазоне значения функций sin(x) и tg(x), функции вычисления которых разместить в отдельном модуле программиста.

Таблица 3.1

Варианты заданий 1-го уровня

Номер варианта	x_n	x_k	n	$y(x)$
1	2	3	4	5
1	$\frac{\pi}{5}$	$\frac{9\pi}{5}$	18	$-\ln\left\|2\sin\left(\frac{x}{2}\right)\right\|$
2	0,1	1	14	$e^{\cos(x)}\cos(\sin(x))$
3	0,1	0,8	10	$-\frac{1}{2}\ln\left(1-2x\cos\frac{\pi}{3}+x^2\right)$
4	0,2	1	6	$\frac{1}{2}\ln(x)$
5	$\frac{\pi}{5}$	π	16	$\frac{1}{4}\left(x^2-\frac{\pi^2}{3}\right)$
6	0,1	0,8	20	$\frac{1}{2}-\frac{\pi}{4}\|\sin(x)\|$
7	0,1	1	18	$\left(1-\frac{x^2}{2}\right)\cos x-\frac{x}{2}\sin x$
8	0,1	0,8	12	$\frac{x(3-x)}{(1-x)^3}$
9	$\frac{\pi}{5}$	π	18	$\frac{\pi^2}{8}-\frac{\pi}{4}\|x\|$
10	0,1	1	10	$\frac{e^x+e^{-x}}{2}$
11	0,1	1	12	$e^{x\cos\frac{\pi}{4}}*\cos\left(x\cdot\sin\frac{\pi}{4}\right)$
12	0,1	1	8	$\cos(x)$
13	0,1	1	14	e^{-x^2}

Окончание таблицы 3.1

1	2	3	4	5
14	0,1	1	8	$\frac{e^x-e^{-x}}{2}$
15	0,1	1	10	e^{2x}
16	0,1	1	14	$\left(\frac{x^2}{4}+\frac{x}{2}+1\right)\cdot e^{\frac{x}{2}}$
17	0,1	0,5	15	$\mathrm{arctg}(x)$
18	0,1	2	8	$\left(1-\frac{x^2}{2}\right)\cos(x)-\frac{x}{2}\sin(x)$
19	-2	-0,1	16	$\ln\left(\frac{1}{2+2x+x^2}\right)$
20	-1	1	10	$\ln(1+x)$

Задания 2-го уровня

Составьте программу вычисления значений выбранной функции $y(x)$ для n-значений аргумента. Начальное x_n и конечное x_k значения аргумента функции x и величину n взять из таблицы 3.2. Кроме того, вывести в том же диапазоне значения функций $ch(x)$ и функции $z(x)$, которая принимает значение 0 при $x < 0$ и 1 при $x \geq 0$, функции вычисления которых разместить в отдельном модуле программиста.

Таблица 3.2

Варианты заданий 2-го уровня

Номер варианта	x_n	x_k	n	$y(x)$
1	2	3	4	5
1	0,1	1	16	$\frac{1}{4}\ln\frac{1+x}{1-x}+\frac{1}{2}\mathrm{arctg}(x)$
2	0,1	1	14	$e^{\cos(x)}\cos(\sin(x))$
3	0,1	1	8	$(1+2x^2)e^{x^2}$

Окончание таблицы 3.2

1	2	3	4	5
4	0,1	0,8	10	$-\frac{1}{2}\ln\left(1-2x\cos\frac{\pi}{3}+x^2\right)$
5	0,1	1	18	$\left(1-\frac{x^2}{2}\right)\cos x-\frac{x}{2}\sin x$
6	0,1	0,8	14	$\frac{x\cos\frac{\pi}{4}-x^2}{1-2x\cos\frac{\pi}{4}+x^2}$
7	0,1	1	12	$e^{x\cos\frac{\pi}{4}}\cdot\cos\left(x\cdot\sin\frac{\pi}{4}\right)$
8	0,1	1	12	$\frac{1+x^2}{2}\cdot\mathrm{arctg}(x)-\frac{x}{2}$
9	0,1	1	10	e^{2x}
10	0,1	1	14	$\left(\frac{x^2}{4}+\frac{x}{2}+1\right)\cdot e^{\frac{x}{2}}$
11	0,1	0,8	10	$x\cdot\mathrm{arctg}(x)-\ln\left(\sqrt{1+x^2}\right)$
12	0,1	2	8	$\left(1-\frac{x^2}{2}\right)\cos(x)-\frac{x}{2}\sin(x)$
13	0,2	0,8	12	$\frac{1}{4}\left(\frac{x+1}{\sqrt{x}}sh\left(\sqrt{x}\right)-ch\left(\sqrt{x}\right)\right)$

Задания 3-го уровня

Составьте программу вычисления значений выбранных функций $s(x)$ и $y(x)$ для n значений аргумента. Начальное x_n и конечное x_k значения аргумента функции x и величину n взять из таблицы 3.3. Кроме того, вывести в том же диапазоне значения функций $ch(x)$ и $\mathrm{tg}(x)$, функции вычисления которых разместить в отдельном модуле программиста. В качестве формальных параметров подпрограмм использовать переменные процедурного и функционального типов.

Таблица 3.3

Варианты заданий 3-го уровня

Номер варианта	x_n	x_k	$s(x)$	n	$y(x)$
1	0,1	1	$1+\frac{\ln 3}{1!}x+\frac{\ln^2 3}{2!}x^2+\ldots+\frac{\ln^n 3}{n!}x^n$	8	3^x
2	$\frac{\pi}{5}$	$\frac{4\pi}{5}$	$\sin(x)+\ldots+(-1)^{n+1}\frac{\sin(nx)}{n}$	6	$\frac{x}{2}$
3	0,2	1	$\frac{x-1}{x+1}+\ldots+\frac{1}{2n+1}\left(\frac{x-1}{x+1}\right)^{2n+1}$	6	$\frac{1}{2}\ln(x)$
4	0,1	1	$x-\frac{x^3}{3!}+\ldots+(-1)^n\frac{x^{2n+1}}{(2n+1)!}$	16	$\sin(x)$
5	0,1	1	$1+\frac{x^2}{2!}+\ldots+\frac{x^{2n}}{(2n)!}$	10	$\frac{e^x+e^{-x}}{2}$
6	0,1	1	$1-\frac{x^2}{2!}+\ldots+(-1)^n\frac{x^{2n}}{(2n)!}$	8	$\cos(x)$

3.5. Вопросы для самоконтроля

1. Назовите два типа подпрограмм.
2. Дайте определение фактических и формальных параметров в подпрограммах и опишите их соотношение.
3. Для чего в описании некоторых формальных параметров перед ними ставится ключевое слово **var**?
4. В чем отличие передачи параметров по ссылке и по значению? Каким образом в подпрограмме можно задать вид передачи параметров?
5. В каких случаях использование функций предпочтительнее использования процедур?

6. Переменным какого типа в качестве значений присваиваются имена соответствующих подпрограмм?

7. Следует ли для объявления типа процедурных типов в разделе **type** указывать в заголовке имя процедуры?

8. Являются ли модули автономно компилируемыми программными единицами?

9. Должно ли имя модуля совпадать с именем дискового файла этого модуля?

10. Под каким именем возвращает результат выполнения операции подпрограмма функция?

МАТЕРИАЛЫ К ЛАБОРАТОРНОЙ РАБОТЕ № 4

4. Программирование с использованием средств отображения графической информации

Цель работы – изучить возможности построения и отображения графиков и диаграмм с помощью компонента **TChart**.

Методика построения и отображения графиков и диаграмм с помощью компонента TChart

Для наглядности часто результаты расчетов представляются в виде графиков и диаграмм. Среда визуального программирования Delphi имеет пакет стандартных программ построения и вывода на экран графиков и диаграмм с помощью компонента ***TChart***.

Построение графика (диаграммы) производится после вычисления таблицы значений функции $y = f(x)$ на интервале $[x_{min}, \ldots, x_{max}]$ с заданным шагом. Полученная таблица передается в специальный двумерный массив ***Seriesk*** (k – номер графика) компонента ***TChart*** с помощью метода ***Add***. Компонент ***TChart*** осуществляет всю работу по созданию и отображению графиков, переданных в объект: строит и размечает оси, рисует координатную сетку, подписывает названия 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 ***Chart.BottomAxis*** содержит значение максимального предела нижней оси графика и при его изменении во время работы автоматически изменяется изображение графика (см. нижеприведенную программу).

4.1. Пример создания приложения

Задание. Составить программу построения и отображения графиков функций $\sin(x)$ и $z(x)$:

$$z(x)=\begin{cases}\dfrac{a+b}{4+\cos^2 x}, & при\ \ x<2{,}3\\ (a+b)/(x+1), & при\ \ 2{,}3\le x<5,\\ e^x+f(x), & при\ \ x\ge 5;\end{cases}$$

на интервале $[x_{\min} \ldots x_{\max}]$ при $f(x) = \cos(x)$.

4.1.1. Размещение компонентов на Форме

Панель диалога программы организуется в виде, представленном на рисунке 4.1. Для ввода исходных данных используются окна, сформированные компонентами ***Edit***. Выход из программы организуется посредством компонента ***BitBtn***. Компонент ***TChart*** может быть введен в форму путем нажатия пиктограммы  в меню компонентов на странице ***Additional***, либо командой ***View | Component List | Component | TChart.***

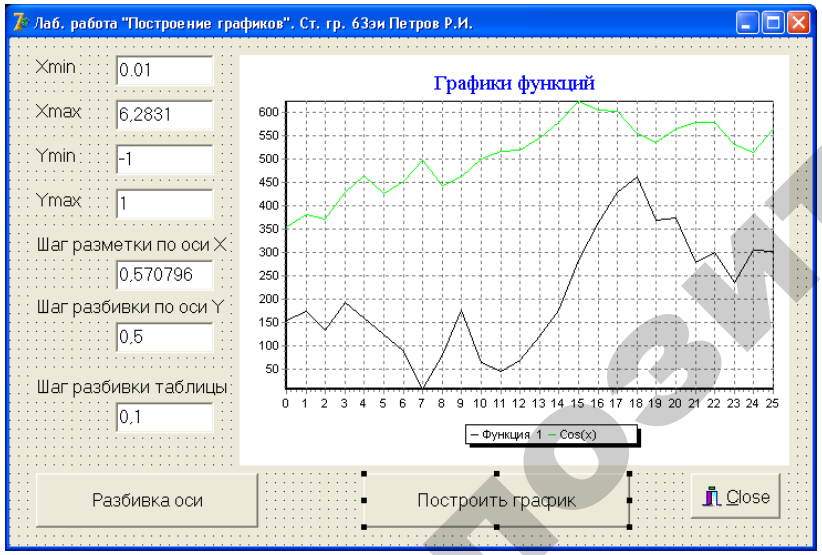


Рис. 4.1. Размещение компонентов на форме

4.1.2. Работа с компонентом TChart

Для изменения параметров компонента ***TChart*** необходимо дважды щелкнуть по нему клавишей мыши в окне формы. Появится окно редактирования ***EditingChart1*** (рисунок 4.2).

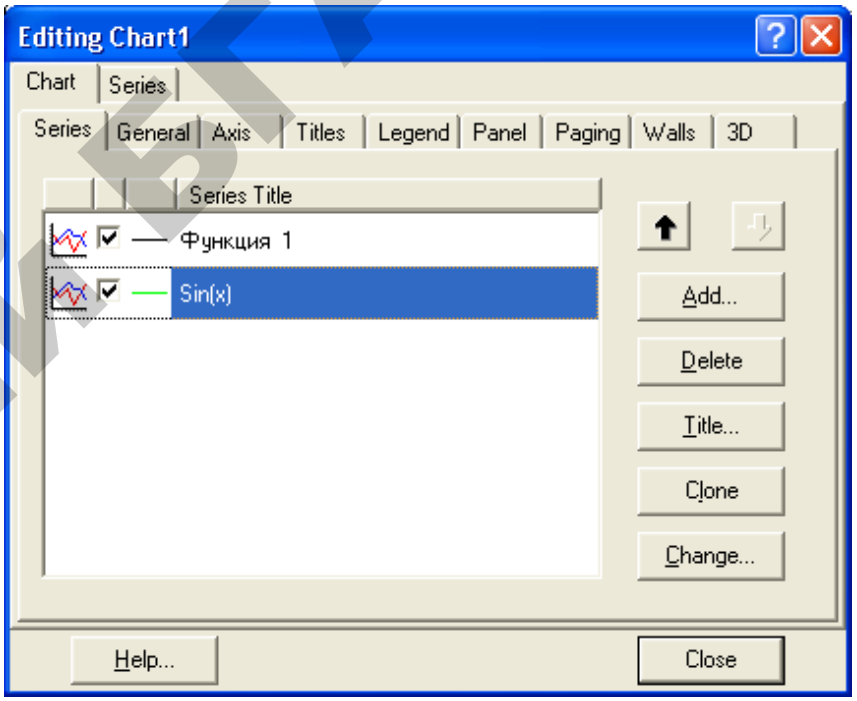


Рис. 4.2. Окно редактирования Editing Chart1

Для создания нового объекта ***Series1*** щелкнуть по кнопке ***Add*** на странице ***Series***. В появившимся диалоговом окне ***TeeChartGallery*** выбрать пиктограмму с надписью ***Line*** (график выводится в виде линий). Если нет необходимости представления графика в трехмерном виде, отключить независимый переключатель ***3D***. Этот переключатель позволяет определить внешний вид: сдвиг, наклон, толщину линии и т.д.

После нажатия на кнопку **ОК** появится новая серия с название ***Series1***. Для изменения названия нажать кнопку ***Title.***

В появившимся однострочном редакторе набрать имя отображаемой функции, например, «Функция 1». Аналогичным образом создать объект Series2, например, для функции sin(*x*).

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

Для разметки осей выбрать страницу ***Axis*** и установить параметры настройки осей. Нажимая различные кнопки меню, можно воспользоваться другими возможностями окна ***EditingChart***.

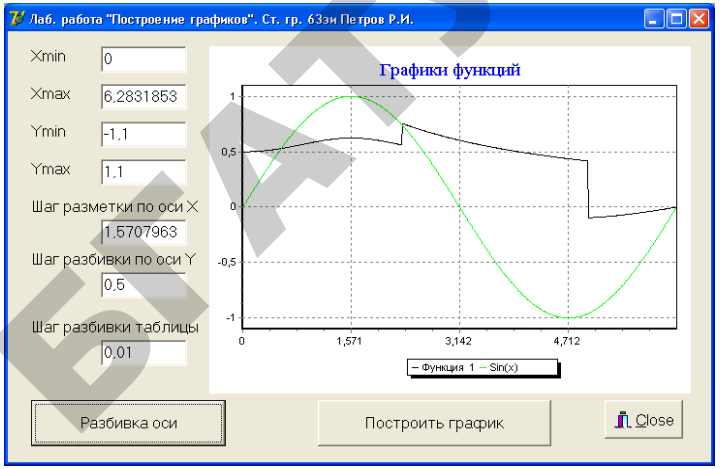
4.1.3. Разработка подпрограммы-обработчика события создания формы

При запуске программы, как указывалось, возникает событие *OnCreate*. Создадим подпрограмму-обработчик этого события (см. далее процедуру **procedure TForm1.FormCreate**(Sender: TObject)), которая обеспечивает установку начальных пределов и шага разметки координатных осей. Если ***Chart1.BottomAxis. Automatic*** имеет значение ***False***, то автоматическая установка параметров осей не работает.

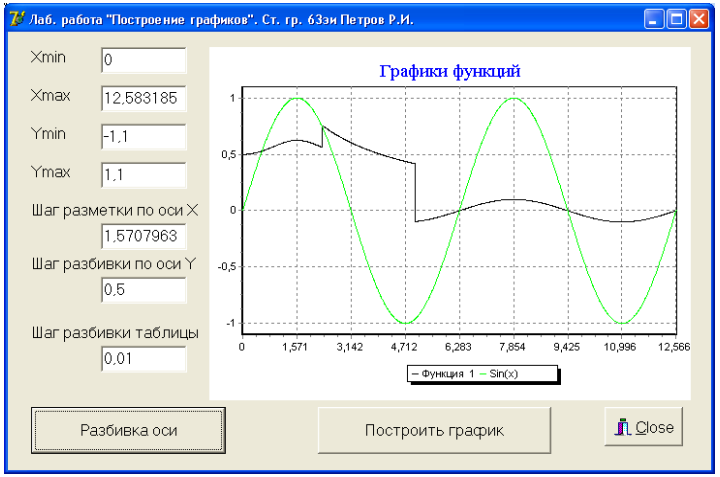
4.1.4. Разработка подпрограмм обработчика событий нажатия на кнопки

Процедура **TForm1.Button1Click** обрабатывает нажатие кнопки ***«Разбивка оси».*** Эта процедура обеспечивает изменение параметров таблицы путем введения новых значений параметров разбивки. Процедура **TForm1.Button2Click** обрабатывает нажатие кнопки ***«Построить графики»***. Для добавления координат точек (x, y) из таблицы значений в двумерный массив объекта ***Seriesk*** используется процедура **Series1.AddXY(Const AXValue, AYValue: Double; Const AXLabel: String; AColor: TColor): Longint;** где ***AXValue***, ***AYValue –*** координаты точки по осям *x* и *y*; Параметр**ALabel** необязательный, его можно задавать пустым: ''. ***AColor*** задает цвет линий (если равен ***clTeeColor***, то принимается цвет, определенный при проектировании формы).

После запуска программы и нажатия кнопки «Построить график» на экране отображаются графики рассматриваемых в примере функций, как показано на рисунке 4.3, *а*.



а



б

Рис. 4.3. а – графики Функции1 и функции sin(*x*) при $x_{max} = 2\pi$;
б – графики Функции1 и функции sin(*x*) при $x_{max} = 4\pi$

Можно изменить параметры таблицы путем введения новых параметров разбивки и активизации компонента «Разбивка оси». Например, при $x_{max} = 4\pi$ график функции отображается на экране в форме, представленной на рисунке 4.3, *б*.

Текст программы приведен в листинге 4.1.

Листинг 4.1

```
unit UnProgrGraph;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls, TeeProcs, TeEngine,
Chart, Series;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Edit6: TEdit;
    Edit7: TEdit;
    Button1: TButton;
    Button2: TButton;
    BitBtn1: TBitBtn;
    Chart1: TChart;
    Series1: TLineSeries;
    Series2: TLineSeries;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private    { Private declarations }
  public     { Public declarations }
  end;
var
```



```
  Form1: TForm1;
  Xmin,Xmax,Ymin,Ymax,Hx,Hy,h:extended;
implementation
{$R *.dfm}
procedure TForm1.FormCreate(Sender: TObject);
begin
{Установка начальных значений параметров координатных осей}
  Xmin:=0.0;
  Xmax:=2*pi;
  Ymin:=-1;
  Ymax:=2;
  Hx:=pi/2;
  Hy:=0.5;
  h:=0.01;
  {Вывод данных в окно однострочных редакторов}
  Edit1.Text:=FloatToStr(Xmin);
  Edit2.Text:=FloatToStr(Xmax);
  Edit3.Text:=FloatToStr(Ymin);
  Edit4.Text:=FloatToStr(Ymax);
  Edit5.Text:=FloatToStr(Hx);
  Edit6.Text:=FloatToStr(Hy);
  Edit7.Text:=FloatToStr(h);
  Chart1.BottomAxis.Automatic:=False; // Отключение авто-
матического определения параметров нижней оси
   Chart1.BottomAxis.Minimum:=Xmin; // Установка левой
границы нижней оси
   Chart1.BottomAxis.Maximum:=Xmax; // Установка правой
границы нижней. оси
    Chart1.LeftAxis.Automatic:= False;    //Отключение авто-
матического определения параметров левой оси
    Chart1.LeftAxis.Minimum:= Ymin;   // Установка нижней
границы левой оси
   Chart1.LeftAxis.Maximum:= Ymax;  // Установка верхней
границы левой оси
   Chart1.BottomAxis.Increment:=Hx;   // Установка шага раз-
бивки по нижней оси
      Chart1.LeftAxis.Increment:= Hy;      // Установка шага
 разбивки по левой оси
```

```
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
{Чтение данных из окон однострочных редакторов}
 Xmin:=StrToFloat(Edit1.Text);
 Xmax:=StrToFloat(Edit2.Text);
 Ymin:=StrToFloat(Edit3.Text);
 Ymax:=StrToFloat(Edit4.Text);
 Hx:=StrToFloat(Edit5.Text);
 Hy:=StrToFloat(Edit6.Text);
 Chart1.BottomAxis.Minimum:=Xmin;    // Установка левой границы нижней оси
 Chart1.BottomAxis.Maximum:=Xmax; // Установка правой границы нижней оси
 Chart1.LeftAxis.Minimum:= Ymin;      // Установка нижней границы левой оси
 Chart1.LeftAxis.Maximum:= Ymax;      // Установка верхней границы левой оси
 Chart1.BottomAxis.Increment:=Hx; // Установка шага разбивки по нижней оси
 Chart1.LeftAxis.Increment:= Hy;      // Установка шага разбивки по левой оси
end;
procedure TForm1.Button2Click(Sender: TObject);
var  x,y1,y2:extended;
     a,b,fx:extended;
      k1,k2:extended;
begin
{Очистка графиков}
 Series1.Clear;
 Series2.Clear;
 Xmin:=StrToFloat(Edit1.Text);
 Xmax:=StrToFloat(Edit2.Text);
 h:=StrToFloat(Edit7.Text);
 x:=Xmin;
 {Варианты значений коэффициентов задаются преподавателем}
 a:=1.25;  b:=1.25; k1:=1;
```



```
 repeat
  fx:=cos(x);                  // Расчет Функции 1
  if x<2.3 then y1:=(a+b)/(4+sqr(cos(x)))
  else if (x>=2.3) and (x<5) then y1:=(a+b)/(x+1)
  else  y1:=k1*fx/10;
  Series1.AddXY(x,y1,'',clTeeColor); // Вывод точки на график
  y2:=sin(x);                  // Расчет функции sin(x)
 Series2.AddXY(x,y2,'',clTeeColor); // Вывод точки на график
  x:=x+h;                  // Увеличение x на величину h
 until(x>Xmax);
end;
end.
```

4.2. Индивидуальные задания

Задания 1-го уровня

Для выбранной функции $z(x)$, математическое выражение для вычисления которой представлено ниже, составьте программу построения и отображения графика этой функции. Функцию $f(x)$ принять в виде: $f(x)= \sin(x)$.

1. $$Z=\begin{cases}\dfrac{a+b}{4+\cos^2 x}, & при\ \ x<2{,}3\\ (a+b)/(x+1), & при\ \ 2{,}3\le x<5,\\ e^x+f(x), & при\ \ x\ge 5.\end{cases}$$

2. $$C=\begin{cases}ax^2+b-4, & если\ \ x<-2,\\ x^4-1-f(x), & если\ \ -2\le x<0,\\ x^3+bx, & если\ \ 0\le x<2.\\ 4f(x), & если\ \ x>2.\end{cases}$$

3. $$D=\begin{cases}|x|+a^2+4, & если\ \ a>2,\\ \dfrac{x^2+2ax}{3}, & если\ \ a=2,\\ 8x+f(x), & если\ \ a<2.\end{cases}$$

4. $D = \begin{cases} f(x), & при \quad x \le 0, \\ x, & при \quad 0 < x \le 1. \\ |a| + x^2, & при \quad x > 1. \end{cases}$

5. $Z = \begin{cases} \sqrt{3x^2 + 7\sin^2 x + 5}, & при \quad x < 0{,}1, \\ ax + b\sqrt{f^2(x) + 1}, & при \quad x = 0{,}1, \\ \sqrt{x^2 + 7\cos^2 x + x}, & при \quad x > 0{,}1. \end{cases}$

6. $C = \begin{cases} a^3 - b^2 + 2f(x), & если \quad x > d, \\ a^2 + b^2 - \mathrm{tg}x, & если \quad x = d, \\ a - 2b^4 + \cos x, & если \quad x < d. \end{cases}$

7. $Y = \begin{cases} e^{-bx} * \sin bx, & при \quad x < a, \\ \cos ax, & при \quad a \le x \le b, \\ e^{-ax} * f(x), & при \quad x > b. \end{cases}$

8. $F = \begin{cases} x^3 - 3x + 8, & при \quad x^2 - x < 1, \\ 4f(x), & при \quad x^2 - x = 1, \\ 1/(x^2 + 3x + 8), & при \quad x^2 - x > 1. \end{cases}$

9. $F = \begin{cases} x^3 + 3x + 8, & при \ x < 0, \\ 4 - f(x), & при \ 0 \le x < 1, \\ a/(x^3 + 3x + 8), & при \ x \ge 1. \end{cases}$

10. $Q = \begin{cases} \dfrac{1}{z^2 + 1} - 1, & если \ Z < 0, \\ 4/(f(x) + 2) + 4, & если \ Z = 0, \\ z^4 - x^3 + 1, & если \ Z > 0. \end{cases}$

11. $K = \begin{cases} f(x), & если \ a^2 + b - x > 0, \\ 2f(x), & если \ a^2 + b - x = 0, \\ 3f(x), & если \ a^2 + b - x < 0. \end{cases}$

12. $F = \begin{cases} f(x) - a, & при \ a > 0{,}3, \\ f^2(x) + a, & при \ a = 0{,}3, \\ \cos(x - a), & при \ a < 0{,}3. \end{cases}$

Задания 2-го уровня

Для назначенного преподавателем варианта из вышеуказанных составить программу построения и отображения графика функции $z(x)$. Кроме того, отобразить график функции $z_1(x) = exp(x)$.

Задания 3-го уровня

Для назначенного преподавателем варианта из вышеуказанных составить программу отображения диаграммы функции $z(x)$ в визуальной среде Delphi в трехмерном виде.

4.3. Вопросы для самоконтроля

1. Какой компонент удобнее всего использовать для построения графиков в Delphi и на какой странице Палитры компонент размещена его пиктограмма?
2. Как можно изменить параметры таблицы путем введения новых параметров разбивки?
3. С помощью какого встроенного редактора компонента ***TChart*** передаются данные о толщине, стиле и цвете линий, параметрах шрифта подписей, шагах разметки координатной сетки и другие настройки?
4. Какой метод компонента ***TChart*** обеспечивает передачу вычисленного двумерного массива исходных данных для построения графика?
5. Какой параметр процедуры **Series1.AddXY** позволяет задать цвет линий графика?

Примеры разноуровневых заданий для контроля знаний

Задания 1-го уровня

1. В юридической консультации для учета выполненных услуг формируется ведомость, содержащая следующую информацию: название услуги, фамилия клиента, фамилия консультанта, стоимость услуги. Вывести упомянутую ведомость в порядке убывания стоимости услуг и предусмотреть помещение списка в типизированный файл.

2. Создать приложение, которое обеспечивает вывод на экран таблицы значений функций $y(x) = Sh(x) = (e^x - e^{-x}) / 2$ и $z(x) = Ch(x) = (e^x + e^{-x}) / 2$. Начальное значение аргумента $x - x_n$, конечное значение – x_k , n – необходимое количество вычисляемых значений функций.

3. Составить программу построения и отображения графиков функций $Sh(x)$ и $\mathrm{Sin}(x)$. Изменить параметры таблицы путем введения новых параметров разбивки и активизации компонента «Разбивка оси».

Задания 2-го уровня

1. В почтовом отделении формируется список подписчиков на периодические издания, включающий информацию: фамилия подписчика, индекс издания, название издания, стоимость за год. Записать упомянутый список в типизированный файл. Вывести список в порядке убывания стоимости подписного издания на год, а так же рассчитать общую стоимость подписки на год по всему списку. Предусмотреть вывод информации в текстовый файл и просмотр текстового файла с помощью текстового редактора.

2. Создать приложение, которое обеспечивает вывод на экран таблицы значений функций $y(x) = e^{2x}$ и $s(x) = 1 + 2/x + \ldots + (2x)^n / n!$ заданной точностью *eps*, а также вычисления функции гиперболического синуса $z(x) = Ch(x) = (e^x + e^{-x}) / 2$. Вычисления функций $y(x)$ и $z(x)$ обеспечить в подпрограммах-функциях, а $s(x)$ – в подпрограмме-процедуре.

3. Составить программу в визуальной среде Delphi построения и отображения графика функций $f(x) = 1 + 3/x + \ldots + (3x)^n / n!$. Цвет линий – красный.

Задания 3-го уровня

1. В центре занятости для учета количества вакантных мест в учреждениях и организациях города по районам формируется таблица, содержащая следующую информацию: год, месяц, наименование района, наименование организации, наименование должности, количество вакантных мест**.** Вывести сформированный список в типизированный и в текстовый файлы. Рассчитать общее количество вакантных мест по каждому району, сформировать таблицу, содержащую поля: район, общее количество мест.

2. Создать приложение, которое обеспечивает вывод на экран таблицы значений функций $y(x) = \mathrm{arctg}(x)$ и $s(x) = x - x^3 + \ldots + (-1)^n x^{(2n+1)} / (2n + 1)$, а также вычисления функции гиперболического синуса $z(x) = Sh(x)$. Вычисления функций $y(x)$ и $z(x)$ обеспечить в подпрограммах-функциях, а $s(x)$ – в подпрограмме-процедуре. Для вычисления значений функций $y(x)$ и $z(x)$ использовать процедуру, обеспечивающую ввод подпрограмм в качестве параметров.

3. Составить программу в среде Delphi вычисления посевной площади участка прямоугольной формы длиной A м и шириной B м, на котором имеются три круглых резервуара для набора воды, диаметры которых соответственно D_1, D_2 и D_3 метров. Для передачи информации использовать подпрограммы в качестве параметров.

Варианты тестовых заданий по модулю 3

1. На какой странице палитры компонентов Delphi размещена пиктограмма компонента ***TChart***?

a) Additional;
b) Standard;
c) System;
d) Dialogs.

2. Каково назначение ключевого слова **var** в процедуре:

Procedure max(x,y: integer; **var** z: integer);
begin if x>y **then** z:=x **else** z:=y; **end**;

a) для обеспечения передачи значения параметра z по ссылке, т. е. передачи не самого значения параметра z, а адреса памяти, где раполагается соответствующий фактический параметр;

b) для обеспечения непосредственной передачи значения параметр z;

c) для обеспечения получения результата выполнения процедуры.

3. Какая процедура используется для открытия файлов?

a) AssignFile;
b) Reset;
c) Rewrite;
d) Read.

4. В языке Delphi предусмотрены следующие типы файлов, различающиеся по способу организации данных. Указать, в описании какого из типов файлов допущена ошибка:

1. ***TextFile*** – текстовый файл, представляющий собой набор символьных строк переменной длины;
2. ***File of <mun>*** – типизированный файл, представляющий собой набор данных указанного типа;
3. ***File*** – нетипизированный файл, представляющий собой набор неструктурированных данных.

a) в первом;
b) во втором;
c) в третьем;
d) все правильно.

5. Параметры-константы передаются в подпрограмму путем передачи адресов констант, и она работает:

a) непосредственно с фактическим параметром;

b) с копиями фактических параметров;
c) с адресами фактических параметров;
d) с формальными параметрами.

6. Между ***формальными*** и ***фактическими параметрами*** подпрограмм должно быть полное соответствие:

a) в количестве;
b) порядке следования;
c) типах;
d) системе счисления.

МОДУЛЬ R

АНАЛИЗ И ОБОБЩЕНИЕ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ. ЗНАКОМСТВО СО СТРУКТУРОЙ ТЕСТОВ

Визуальное проектирование и методология объектно-ориентированного событийного программирования

В основе идеологии Delphi лежит современная технология визуального проектирования и методология объектно-ориентирован ного событийного программирования.

Визуальный подход реализован практически во всех современных системах программирования. Этот процесс автоматизирован в *средах быстрого проектирования* (*R*apid *A*pplication *D*evelop-ment, *RAD*-среды). В таких средах все необходимые элементы оформления и управления создаются и обслуживаются с помощью готовых *визуальных компонентов*, которые мышью «перетаскиваются» в проектируемое окно. Их свойства и поведение затем настраиваются с помощью простых редакторов, визуально показывающих характеристики соответствующих элементов. При этом вспомогательный исходный текст программы, обеспечивающей создание и работу этих элементов, генерируется *RAD*-средой *автоматически*, что позволяет сосредоточиться только на логике решаемой задачи. В результате программирование во многом заменяется проектированием – такой подход называется *визуальным проектированием*. *Компонентный подход* к созданию программ позволяет *повторное использование* известных работ.

С активным распространением системы Windows и появлением *RAD*-сред широкую популярность приобрел событийный подход к созданию программ – *событийно-ориентированное программирование.* Любое событие: щелчок кнопки мыши, нажатие клавиши на клавиатуре, перемещение курсора на экране монитора и т. д. приводит к возникновению соответствующего *сообщения*, которое обрабатывается Windows.

Структура программы, созданной с помощью событийного программирования, следующая. Главная часть представляет собой один бесконечный цикл, который опрашивает Windows, следя за появлением нового сообщения. При его обнаружении вызывается подпрограмма, ответственная за *обработку* соответствующего события, причем обрабатываются не все события, а только нужные. Такой цикл опроса продолжается до тех пор, пока не будет получено сообщение «Завершить работу».

События могут быть:

пользовательскими, возникшими в результате действий пользователя;

системными, возникающими в операционной системе, например, сообщениями от таймера;

программными, генерируемыми самой программой, например, обнаружена ошибка, ее надо обработать.

Применение идей структурного и событийного программирования привело к существенному увеличению производительности труда программистов. Дальнейшее развитие технология программирования получила благодаря созданию и внедрению понятия *объекта*.

Реальные объекты окружающего нас мира обладают тремя базовыми характеристиками:

1) они имеют набор свойств;

2) способны разными методами изменять эти свойства;

3) могут реагировать на события, возникающие как в окружающем мире, так и внутри самого объекта.

Именно в таком виде в языках объектно-ориентированного программирования и реализовано понятие объекта как совокупности *свойств* (структур данных, характерных для этого объекта), *методов* их обработки (подпрограмм изменения свойств) и *событий*, на которые данный объект может реагировать и которые приводят, как правило, к изменению свойств объекта.

Объекты могут иметь идентичную структуру и отличаться только значениями свойств. В таких случаях в программе создается новый тип, основанный на единой структуре объекта, по аналогии с тем, как создаются новые типы для структур данных. Он называется *классом*, а каждый конкретный объект, имеющий структуру этого класса, называется *экземпляром класса*.

Описание нового класса похоже на описание новой структуры данных, только к полям (свойствам) добавляются методы – подпрограммы. Объединение данных с методами в одном типе (классе) называется *инкапсуляцией*.

Важнейшая характеристика класса – возможность создания на его основе новых классов с *наследованием* всех свойств и методов исходного класса и добавлением новых свойств и методов. Класс, не имеющий предшественника (родителя), называется *базовым*. Наследование по-

зволяет создавать новые классы, повторно используя уже готовый исходный код.

В большинстве случаев методы базового класса у классов-наследников приходится переопределять. Все переопределяемые методы по написанию (названию) будут совпадать с методами базового объекта, однако компилятор по типу объекта (его классу) распознает, какой конкретно метод надо использовать. Свойство объектов переопределять методы наследуемого класса называется *полиморфизмом*.

Технологии объектного, событийного и структурного программирования сегодня объединены в *RAD*-системах, которые содержат множество готовых классов, представленных в виде *визуальных компонентов*. Программисту надо только спроектировать внешний вид панелей интерфейса своего приложения и определить обработку основных событий, а именно: какие операторы будут выполняться при нажатии на кнопки, при выборе пунктов меню или щелчках клавишей мыши. Весь вспомогательный исходный код визуальная среда сгенерирует сама, позволяя программисту полностью сосредоточиться на реализации алгоритма.

Визуальное программирование в Delphi

Программирование в Delphi строится на тесном взаимодействии двух процессов: процесса конструирования визуального проявления программы (т. е. ее Windows-окна) и процесса написания кода, придающего элементам этого окна и программе в целом необходимую функциональность. Для написания кода используется окно редактора кода, для конструирования программы – остальные окна Delphi, и прежде всего – окно Форма.

Между содержимым окон Формы и кода существует неразрывная связь. Это означает, что размещение на Форме компонента приводит к автоматическому изменению кода программы и, наоборот, – удаление тех или иных автоматически вставленных фрагментов кода может привести к удалению соответствующих компонентов. Поэтому вначале конструируют форму, размещая на ней компоненты, а уже только после этого переходят к написанию фрагмента кода, обеспечивающего требуемое поведение компонента в работающей программе.

Окно Формы содержит проект Windows-окна программы. С самого начала работы над новой программой Delphi создает минимально необходимый код, обеспечивающий ее нормальное функционирование. Таким образом, простейшая программа готова сразу после выбора опции **File | New | Application** к ее запуску. Однако до этого *следует создать*

собственный рабочий каталог (папку) *и сохранить проект*. Как известно, проект – это совокупность файлов различного назначения. При сохранении проекта по запросу системы первоначально сохраняются модули программы, а затем файл основного проекта. Желательно, чтобы все файлы проекта размещались в единой папке.

После запуска на выполнение подготовленная Delphi-программа последовательно проходит три этапа – этапы компиляции, компоновки и исполнения. На этапе компиляции осуществляется преобразование подготовленного в окне кода текста программы на алгоритмическом языке в последовательность машинных инструкций. На этапе компоновки к ней подключаются необходимые вспомогательные подпрограммы, а на этапе исполнения готовая программа загружается в оперативную память и ей передается исполнение.

Каждый раз, когда создается новая форма (в программе может быть несколько форм и связанных с ними модулей), создается соответствующий модуль. При компиляции программы Delphi создает файлы с расширениями pas, dfm и dcu для каждого модуля: pas-файл содержит копию текста из окна кода программы, в файле с расширением dfm хранится описание содержимого окна Формы, а в dcu-файле – результат преобразования в машинные инструкции текста из обоих файлов. Файлы dcu создаются компилятором и дают необходимую базу для работы компоновщика, который преобразует их в единый загружаемый файл с расширением exe. Имеется возможность создавать модули программиста без форм.

Для размещения компонента на Форме сначала выбирается (щелчком по нему клавишей мыши) нужный в палитре компонентов, а затем выполняется шелчек клавишей мыши по точке рабочего пространства Формы, где должен располагаться левый верхний угол компонента. Возможны и другие варианты. Можно далее щелкнуть клавишей мыши внутри обрамляющих надпись черных прямоугольников и, не отпуская ее левую кнопку, сместить указатель так, чтобы он расположился в нужной области окна, после чего отпустить кнопку. Таким способом можно буксировать компонент по Форме, добиваясь нужного его положения, и изменять размеры компонента.

Реакция на те, или иные события – важнейший показатель функциональности программы. В связи с этим каждый компонент помимо свойств характеризуется также набором событий, на которые он может реагировать. Чтобы заставить программу реагировать на то или иное событие, например, нажатие кнопки, необходимо написать на языке Object Pascal фрагмент программы, называемый обработчиком события.

Этот фрагмент должен представлять собой последовательность текстовых строк, в которых программист указывает, что именно должна делать программа в ответ на нажатие кнопки. Фрагмент оформляется в виде специальной подпрограммы языка Object Pascal-процедуры.

Чтобы заставить Delphi самостоятельно сделать заготовку для процедуры-обработчика, например, события OnClick вновь вставленного компонента Button1, следует дважды подряд без паузы щелкнуть клавишей мыши по компоненту. В ответ Delphi активизирует окно кода, в нем отобразится текст заготовки процедуры:

```
procedure TForm1.ButtonlClick(Sender: TObject);
 begin
 end;
```

Слово procedure извещает компилятор о начале подпрограммы-процедуры. За ним следует имя процедуры TForm1.Button1Click. Это имя – составное, оно состоит из имени класса TForm1 и собственно имени процедуры Button1Click.

Классами в Delphi, как указывалось, называются функционально законченные фрагменты программ, служащие образцами для создания подобных себе экземпляров. Однажды создав класс, программист может включать его экземпляры (копии) в разные программы или в разные места одной и той же программы. В состав Delphi входит несколько сотен так называемых стандартных классов.

Каждый компонент принадлежит к строго определенному классу, а все конкретные экземпляры компонентов, вставляемые в форму, получают имя класса с добавленным числовым индексом. По используемому в Delphi соглашению все имена классов начинаются с буквы **T**. Таким образом, имя TForm1 означает имя класса, созданного по образцу стандартного класса TForm.

Например, в тексте начала кода программы, приведенной ниже:

```
type
 TFormi1= class(TForm)
 Button1: TButton;
 Label1: TLabel;
procedure TForm1.ButtonlClick(Sender: TObject);
 private
  { Private declarations }
public
 { Public declarations }
end;
var
```



```
 Form1: TForm1;
```

строка TForm1 = class(TForm) определяет новый класс TForm1, который порожден от (создан по образцу) стандартного класса TForm. Строка

```
Form1: TForm1;
```

создает экземпляр этого класса с именем Form1. Стандартный класс TForm описывает пустое Windows-окно, в то время как класс TForm1 описывает окно с уже вставленными в него компонентами «метка» и «кнопка». Описание этих компонентов содержат строки

```
Button1: TButton;
Label 1: TLabel;
```

Они указывают, что компонент Button1 (Кнопка 1) представляет собой экземпляр стандартного класса TButton, а компонент Label1 (Метка 1) – экземпляр класса TLabel.

За именем процедуры TForm1. Button1Click в круглых скобках следует описание параметра вызова Sender:TObject (параметр с именем Sender принадлежит классу TObject). Процедуры могут иметь не один, а несколько параметров вызова или не иметь их вовсе.

Строка

```
procedure TFormI.Button1Click(Sender: TObject);
```

называется заголовком процедуры. Ее завершает символ «;». Этот символ указывает компилятору на конец предложения языка. Из отдельных предложений составляется весь текст программы. В конце предложения ставится точка с запятой – это обязательное требование синтаксиса языка.

Следующие строки определяют тело процедуры:

```
 begin
 . . .
 end;
```

Слово begin сигнализирует компилятору о начале последовательности предложений, описывающих алгоритм работы процедуры, а слово end – о конце этой последовательности. Наполнить тело процедуры между этими словами нужными предложениями – задача программиста.

Например, если ввести в пустой строке между словами begin...end следующее предложение:

```
MessageBeep (MB_OK);
```

и запустить программу, предварительно включив звуковую систему компьютера, то в динамике компьютера будет раздаваться звуковой сигнал, т. к. вставленная строка реализует обращение к стандартной

процедуре, которая обеспечивает извлечение из динамика различных стандартных для Windows звуков.

Событие OnCreate возникает после создания Windows-окна, но до появления этого окна на экране. Чтобы создать обработчик этого события, следует раскрыть список компонентов в верхней части окна Инспектора объектов, выбрать компонент Form1 и дважды щелкнуть по свойству OnCreate на странице Events этого компонента (щелкать нужно по правой части строки onCreate). В ответ Delphi вновь активизирует окно кода и покажет вам заготовку для процедуры TForm1. FormCreate. После введения нужного оператора текст процедуры представляется следующим образом:

```
procedure TFormi.FormCreate(Sender: TObject);
begin
Button1.Caption  :=  'Выполнить';
end;
```

Единственная вставленная строка представляет собой так называемый ***оператор присваивания***. В левой части оператора указывается свойство Button1.Caption, а в правой части – значение 'Выполнить', которое требуется придать этому свойству. Связывает обе части комбинация символов «:=», которая читается как «присвоить значение». Символы «:=» пишутся слитно, без разделяющих пробелов, хотя перед двоеточием и после знака равенства можно для лучшей читаемости программы вставлять пробелы.

Присваиваемое свойству Caption значение является текстовой строкой. По правилам Object Pascal текстовая строка должна заключаться в обрамляющие апострофы. Внутри апострофов можно написать любое количество произвольных символов. Именно они (без обрамляющих апострофов) будут определять новую надпись на кнопке. После запуска программы на выполнение на кнопке отобразится измененная надпись. Любое свойство любого компонента можно изменять динамически, т. е. в ходе исполнения программы.

Для наглядности результаты расчетов по программе представляются в виде графиков и диаграмм. Среда визуального программирования Delphi имеет мощный пакет стандартных программ построения и вывода на экран графиков и диаграмм с помощью компонента ***TChart*** и обеспечивает передачу данных в MS Word и MS Excel.

Варианты тестовых заданий для контроля знаний

1. Какая страница окна Инспектора объектов предназначена для редактирования значений свойств компонентов?
   a) Properties;
   b) Events;
   c) Action.
2. Объявление переменных – это:
   a) процесс резервирования места в оперативной памяти для сохранения значений переменных;
   b) процесс резервирования места на жестком диске для сохранения значений переменных;
   c) процесс резервирования места в процессоре для сохранения значений переменных.
3. Константы:
   a) данные, значения которых не изменяются в процессе выполнения программы;
   b) данные, значения которых изменяются в процессе выполнения программы;
   c) числовые данные.
4. Переменные:
   a) данные, значения которых не изменяются в процессе выполнения программы;
   b) данные, значения которых могут изменяться в процессе выполнения программы;
   c) данные, у которых могут изменяться имена в процессе выполнения программы.
5. В результате выполнения фрагмента программы

```
If  x > 0 then y:=x*x+1
else  if  x=0 then y:=0
      else  y:=2*x;
```

y примет значение ноль, если значение *x*:
   a) положительное;
   b) равное нулю;
   c) отрицательное:
   d) неотрицательное.
6. Сколько раз выполнится оператор s:=s+i в результате выполнения фрагмента программы?

```
for i: = 1 to 10 do s:=s+i;
```

   a) 11;

b) 10;
c) 1;

7. После выполнения оператора цикла

for i: = 50 **downto** 25 **do** p:=p·i; переменная *i* будет иметь значение:

a) 25;
b) 26;
c) 24.

8. В тексте какой из приведенных ниже подпрограмм-функций допущена ошибка?

1.

```
function sec(x:extended):extended;
begin  rezult := 1/cos(x); end;
```

2.

```
function sec(x:extended):extended;
begin  sec := 1/cos(x); end;
```

a) в 1;
b) во 2;
c) в 1 и 2;
d) все правильно.

9. Свойства компонента на этапе выполнения программы

a) могут изменяться;
b) не могут изменяться;
c) могут изменяться, но не все.

10. Могут ли процедуры и функции использоваться в качестве формальных параметров подпрограмм?

a) да;
b) нет;
c) только процедуры.

11. Можно ли с помощью компонента TChart отобразить на экране график зависимости $y = f(x)$ в трехмерном виде?

a) нет;
b) да;
c) можно, если активизировать требуемый переключатель.

12. Для создания модуля программиста, который не имеет своей формы, следует сначала закрыть:

a) окно формы и окно модуля формы;
b) окно формы;
c) окно первоначального модуля системы.

ГЛОССАРИЙ

Ввод-вывод файлов – процесс чтения данных из файла (ввод), или записи данных в файл.

Вызов подпрограммы – активизация подпрограммы.

Вызывающая подпрограмма – подпрограмма, из которой вызывается другая подпрограмма.

График – изображение в виде линий изменения функции при изменении аргумента для математической, физической или другой зависимости.

Длинная строка – строка, размер которой ограничен только доступной памятью.

Доступ к файлу – чтение или запись данных в файл.

Закрыть файл – разорвать (отменить) связь между файлом и дескриптором файла.

Запись (record) – структура данных, представляющая собой объединенный общим именем набор фиксированного числа групп элементов различного типа.

Инициализация – присвоение переменной начального значения. Применительно к программам и подпрограммам инициализация означает присвоение их переменным и объектам начальных значений.

Класс – тип объекта, сложная структура, включающая, помимо описания данных, описания процедур и функций, которые могут быть выполнены над представителем класса – объектом.

Ключевое слово – зарезервированное слово или символ, распознаваемые транслятором как часть синтаксиса языка. Использование ключевых слов в качестве идентификаторов недопустимо.

Компонент – часть системы. В Delphi под компонентом чаще всего подразумевается Форма или элемент управления в среде разработки.

Константа – неизменное значение, хранящееся в памяти все время выполнения программы. Константы делятся на именные и неименные.

Корневой каталог – каталог, не расположенный ни в каком другом каталоге.

Локальная переменная – переменная, доступная только внутри подпрограммы, в которой она объявлена.

Маршрут – последовательность имен диска и подкаталогов, определяющая местонахождение файла или каталога.

Метод – подпрограмма, ассоциированная с классом объектов. Методы класса (процедуры и функции, объявление которых включено в описание класса) выполняют действия над объектом класса.

Модуль – часть программы, расположенная в отдельном файле. В Object Pascal модулем обычно называется файл исходного кода, с которым ассоциирован файл формы.

Модульное программирование – разработка программ на основе отдельных модулей.

Открытие файла – подготовка файла данных к доступу и связывание с файловой переменной.

Обработчик события – подпрограмма, автоматически вызываемая операционной системой при наступлении события.

Передавать информацию в подпрограмму – присваивать формальным параметрам подпрограммы значения фактических параметров.

Передача параметров по значению – вид передачи параметров, при котором в подпрограмму передаются значения (копии) фактических параметров. Это предотвращает изменение в подпрограмме передаваемых объектов. По умолчанию в Object Pascal параметры передаются по значению.

Передача параметров по ссылке – вид передачи параметров, при котором в подпрограмму передаются не значения фактических параметров (т.е. копии параметров), а их адреса. Это позволяет подпрограмме изменять передаваемые объекты.

Повторное использование кода – важнейший принцип модульного программирования, предполагающий использование в разрабатываемой программе исходных кодов модулей уже разработанных программ.

Подпрограмма – часть программы, решающая определенную вычислительную задачу. Подпрограммы делятся на процедуры и функции.

Пользовательская подпрограмма – подпрограмма, разработанная программистом, т.е. пользователем Delphi (в отличие от встроенных подпрограмм, входящих в комплект поставки Delphi).

Поля записи – отдельные группы данных записи одинакового типа.

Последовательный файл – файл, доступ к элементам которого выполняется только последовательно – от начала файла до его конца.

Приватная подпрограмма – подпрограмма, область видимости которой ограничена модулем, в котором она определена.

Процедура – подпрограмма, не возвращающая значение в вызывающую подпрограмму. Обычно процедуры предназначены для выполнения какой-либо задачи, хотя они и могут передавать информацию в вызывающую подпрограмму посредством списка параметров.

Публичная подпрограмма – подпрограмма, область видимости которой может распространяться на все модули программы.

Разработка сверху вниз – методология программирования на основе пошагового решения задачи.

Символы-разделители – символы, разделяющие в текстовом файле отдельные порции данных. К ним относятся пробелы, символы окончания строки и символы табуляции.

Событие – действие пользователя или операция программы, обнаруженные операционной системой (например, нажатие клавиши или активизация Формы).

Структурированная программа – программа, разбитая на отдельные подпрограммы в целях упрощения сложной задачи. Чаще всего структурированная программа состоит из множества модулей, содержащих подпрограммы.

Тестовая программа – программа, разработанная для проверки правильности работы некоторой подпрограммы.

Управляющие символы – невидимые символы текстового файла, выполняющие специальные функции, например возврат каретки или подача новой строки.

Файл (англ. file – папка) – совокупность логически взаимосвязанной информации, занимающая поименованную область внешней памяти.

Файл данных – файл, содержащий данные (в отличие от файлов, содержащих программу).

Файл модуля – файл, содержащий исходный код модуля.

Файл проекта – файл, играющий роль главной подпрограммы проекта. Из файла проекта вызываются файлы модулей и форм, составляющих проект.

Файл формы – файл, в котором перечислены объекты, расположенные на форме, а также их свойства.

Форма – объект, на визуальном изображении которого размещаются компоненты Delphi. При выполнении программы форма выводится на экран в виде окна.

Фактический параметр – значение, передаваемое в подпрограмму.

Фокус – свойство объекта, состоящее в том, что он может принимать входную информацию. Об объекте говорят, что он «имеет, получил или потерял фокус».

Формальный параметр – переменная, в которой хранится передаваемая в подпрограмму информация.

Функция – подпрограмма, возвращающая в вызывающую подпрограмму одно значение. Обычно функции применяются, когда подпрограмму удобнее использовать в качестве операнда какого-либо выражения.

ЛИТЕРАТУРА

Основная

1. *Архангельский, И. М.* Программирование в Delphi 7: учебник / И. М. Архангельский. – Москва: Бином-Пресс, 2008. – 1078 с.
2. *Культин, Н. В.* Основы программирования в Delphi 2010 / Н. В. Культин. – Санкт-Петербург : БХВ, 2010. – 448 с.
3. *Культин, Н. В.* Основы программирования в Delphi 7. / Н. В. Культин. – СПб: БХВ-Петербург, 2006. – 608 с.
4. Программирование в среде Delphi : лабораторный практикум для студентов всех специальностей / А. Б. Закалюкин [и др.]; под общ. ред. А. К. Синицына. – Минск : БГУИР, 1998. – 94 с.
5. Программирование в визуальной среде Delphi : методические указания: в 2 ч. Ч. 1 / Р. И. Фурунжиев [и др.]. – Минск : БГАТУ, 2004. – 64 с.
6. Программирование в визуальной среде Delphi : методические указания: в 2 ч. Ч. 2 / Р. И. Фурунжиев [и др.]. – Минск : БГАТУ, 2005. – 118 с.
7. Программирование в среде Delphi : лабораторный практикум для студентов всех специальностей / под ред. А. К. Синицына. – Минск : БГУИР, 2001. – 98 с.
8. *Прищепов, М. А.* Программирование на языках Basic, Pascal и Object Pascal в среде Delphi : учебное пособие / М. А. Прищепов, Е.В. Севернева, А.И. Шакирин; под общ. ред. М. А. Прищепова. – Минск : ТетраСистемс, 2006. – 320 с.
9. *Фаронов, В. В.* Delphi. Программирование на языке высокого уровня : учебник / В. В. Фаронов. – Санкт-Петербург : Питер, 2010. – 640 с.
10. *Шакирин, А. И.* Основы программирования на алгоритмическом языке Object Pascal в среде Delphi : конспект лекций / А. И. Шакирин. – Минск : БГАТУ, 2005. – 142 с.

Дополнительная

11. *Севернева, Е. В.* Основы алгоритмизации и программирования : учебно-методический комплекс / Е. В. Севернева, Н. М. Желобкевич. – Минск : БГАТУ, 2009. – 112 с.
12. Программирование на языке Object Pascal в визуальной среде Delphi : методические указания к выполнению курсовой работы / сост.: Р. И. Фурунжиев, Т. В. Ероховец, О. М.Львова. – Минск : БГАТУ, 2004. – 46 с.
13. *Фурунжиев, Р. И.* Программирование в Delphi : методические указания к выполнению курсовой работы / сост. Р. И. Фурунжиев. – Минск : Ротапринт БАТУ, 2003. – 39 с.
14. Программирование на языке Object Pascal в визуальной среде Delphi : методические указания: в 2 ч. Ч. 1 / сост. А. И. Шакирин. – Минск: Ротапринт БАТУ, 2001. – 74 с.
15. Программирование на языке Object Pascal в визуальной среде Delphi : методические указания: в 2 ч. Ч. 2 / сост. А. И. Шакирин. – Минск : Ротапринт БАТУ, 2001. – 44 с.

СОДЕРЖАНИЕ

ДЛЯ ЗАМЕТОК

ДЛЯ ЗАМЕТОК

Учебное издание

ОСНОВЫ ПРОГРАММИРОВАНИЯ
(в визуальной среде Delphi)

Учебно-методический комплекс

В 2 частях

Часть 2

Составители:
Фурунжиев Риза Ибраимович,
Исаченко Елена Михайловна,
Ероховец Тамара Викторовна

Ответственный за выпуск *О. Л. Сапун*
Редактор *Н. А. Антипович*
Компьютерная верстка *А. И. Стебуля*

Подписано в печать 09.11.2010 г. Формат $60\times84^{1}/_{16}$.
Бумага офсетная. Ризография.
Усл. печ. л. 5,81. Уч.-изд. л. 4,54. Тираж 150 экз. Заказ 1033.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный аграрный технический университет».
ЛИ № 02330/0552984 от 14.04.2010.
ЛП № 02330/0552743 от 02.02.2010.
Пр. Независимости, 99–2, 220023, Минск.