ФГБОУ ВПО
ДАГЕСТАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

А.М. МАГОМЕДОВ

ПРАКТИКА ПРОГРАММИРОВАНИЯ ВТОРОЙ СЕМЕСТР

УЧЕБНОЕ ПОСОБИЕ

МАХАЧКАЛА – 2012

ФГБОУ ВПО
ДАГЕСТАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

А.М. МАГОМЕДОВ

ПРАКТИКА ПРОГРАММИРОВАНИЯ ВТОРОЙ СЕМЕСТР

УЧЕБНОЕ ПОСОБИЕ

для студентов высших учебных заведений, обучающихся по направлениям подготовки высшего профессионального образования:
010100 (*математика*),
010200 (*математика и компьютерные науки*),
010300 (*фундаментальная информатика и информационные технологии*),
010400 (*прикладная математика и информатика*)

МАХАЧКАЛА – 2012

УДК 681.3.068+800.92Delphi

Магомедов А. М.
Практика программирования. Второй семестр. – Махачкала: Издательство ДГУ, 2012. – 104 с.

Брошюра представляет собой сборник заданий и программ, разработанных в среде Delphi и рассчитаннах на студентов, освоивших семестровый курс по основам программирования. Структура рабочей программы соответствующей дисциплины на учебный семестр учтена как по временны́м параметрам, так и по содержанию.

ВВЕДЕНИЕ

Каждое из 18 заданий рассчитано на одно практическое занятие учебного семестра. Задания сгруппированы в 4 главы. Первые две главы суть компьютерное сопровождение математических дисциплин (алгебра, анализ, теория графов и комбинаторика), третья относится к основам программирования, заключительная глава посвящена элементам компьютерной графики.

Задания снабжены краткими пояснениями, ссылками на литературу и на методы, встроенные в специальные программы (например, Mathematica 8.0 или MS Excel 2010). Даны оценки сложности, отмечены особенности алгоритмов, приведены полные листинги программ и скриншоты окон приложений. Хотя решение каждого задания представлено в замкнутом виде и может быть рассмотрено вполне самостоятельно, последовательное изучение заданий представляется более предпочтительным, во всяком случае, — в пределах каждой главы.

В первом задании (вычисление определителя) определение искомой величины приведено в рекуррентной форме. Если с точки зрения профессионального программирования наиболее органичным представляется применение рекурсии, то в целях обучения имеет смысл продемонстрировать и другие подходы; в программе реализован метод приведения к треугольному виду, для представления исходной матрицы выбрана таблица Delphi с интерактивным изменением размеров.

В решении второго задания (решение системы л.а.у.) получили развитие как алгоритмическая составляющая первого решения, так и его интерфейс: приведение к треугольному виду применено здесь для решения системы л.а.у., для представления данных использованы две таблицы Delphi — для расширенной матрицы системы и неизвестных величин соответственно.

В решении третьего задания (построение интерполяционного многочлена Лагранжа) наряду с построением искомого многочлена обсуждается техника вывода графика. В целях облегчения восприятия основной идеи изложение ограничено непосредственным выводом на холст компоненты.

В четвертом задании (метод минимальных квадратов) решение системы л.а.у. является не самоцелью, а средством вычисления коэффициентов некоторого многочлена; при этом система л.а.у. сформирована условиями достижения минимума некоторой функцией многих переменных. Для вывода графика многочлена использован изложенный в третьем параграфе метод, который претерпел дальнейшее развитие: построение рисунка выполняется в памяти, а вывод на холст осуществляется лишь после завершения постро-

ения.

Вычислительные задачи анализа (например, вычисление суммы ряда Тейлора со сколь угодно большой точностью или вычисление определителя порядка 1000), неминуемо приводят к многоразрядной арифметике. Первую главу завершают арифметические действия с побайтовым выделением памяти для цифр десятичного представления.

При выполнении заданий первой главы предполагается, что алгоритмы решений изучены ранее в рамках соответствующих математических дисциплин. И основным здесь является создание интерфейса, кодирование алгоритма на Delphi, отладка и тестирование — нередко рутинный, но всегда необходимый труд, расширяющий с каждым шагом кругозор начинающего программиста, углубляющий его знания. В известных традициях предисловий отечественных учебников напомним, что господство первобытного человека над природой началось с труда и развития пальцев: «Обивая клинок своего каменного топора, он в то же время оттачивал лезвие своих способностей» [20, с. 56]. Господство человека над компьютерными проблемами если и начинается с освоения клавиатуры и мыши, то отнюдь не достигается одним только кодированием известных алгоритмов. В решениях заданий второй главы, в частности, уделено внимание оценкам сложности алгоритмов. Первые три из заданий главы разрешимы за полиномиальное время, а для решения NP-полной задачи о разбиении предложен псевдополиномиальный алгоритм. Подчеркнем, что в тандеме «компьютер — алгоритм» наблюдается неуклонное превалирование второй компоненты. Этим вящим торжеством мысли над материей и продиктовано название главы.

Если в решениях первой главы преобладают вычислительные аспекты, то в программах для теоретико-графовых задач второй главы нельзя обойти вниманием и вопросы наглядности. В свою очередь, вопросы визуальной перестройки графов, требующие немало строк кода, способны оттеснить алгоритмические аспекты. Закрепление алгоритмических аспектов на первом плане достигнуто ценой неизбежных компромиссов, «распределенных» по всей второй главе.

В решении первого задания (построение минимального остовного дерева) бóльшая часть кода посвящена графическому интерфейсу — в полной уверенности, что самобытный алгоритм решения, признанный одним из самых элегантных алгоритмов теории графов, не будет обделен вниманием читателя. Заимствование разработаннного здесь интерфейса в решение второго задания (вычисление максимального потока) в ущерб подробному изложению алгоритма представляется неоправданным. Вряд ли в теории графов найдется алгоритм, способный соперничать с алгоритмом Форда и Фалкерсона по

применимости к многоразличным прикладным задачам.

Дальше. В решении третьего задания (поиск кратчайших путей) целелесообразно освободить программу от той части кода, которая отвечает за уже освоенную технику вывода графов.

В решении четвертого задания, посвященного знаменитой задаче разбиения, алгоритм выступает в качестве единственной ценности решения; все изобразительные элементы отброшены. Несмотря на непритязательный вид, данный алгоритм открывает для программиста принципиально новые горизонты. Осмысление его сакральной псевдополиномиальной сущности благотворно скажется на алгоритмической культуре студента при условии пристального интереса к святая святых современной математики — проблеме «$NP \neq P$».

В отличие от первых двух глав, где программирование играет вспомогательную роль в освоении важных понятий и методов сопутствующих математических дисциплин, третья глава (примеры базовых элементов программирования) посвящена изучению нетривиальных средств Delphi: генерация объектов программным путем, создание процессов, построение регионов, эмуляция устройств ввода.

Наконец, в заключительной главе (элементы компьютерной графики) проработка многочисленных вопросов Delphi осуществляется на материале, носящем подготовительный характер для изучения дисциплины «Компьютерная графика»: столкновение плоских фигур, трехмерное моделирование, построение и деформация поверхностей.

Глава 1.

Алгебра и начала анализа

§ 1.1. Определитель матрицы

Литература: [6, с. 105]. Определитель 1×1-матрицы (a_{11}) равен числу a_{11}. Пусть определители матриц порядков $1, 2, \ldots, n-1$ уже введены. Назовем определителем матрицы $A = (a_{ij})$ величину

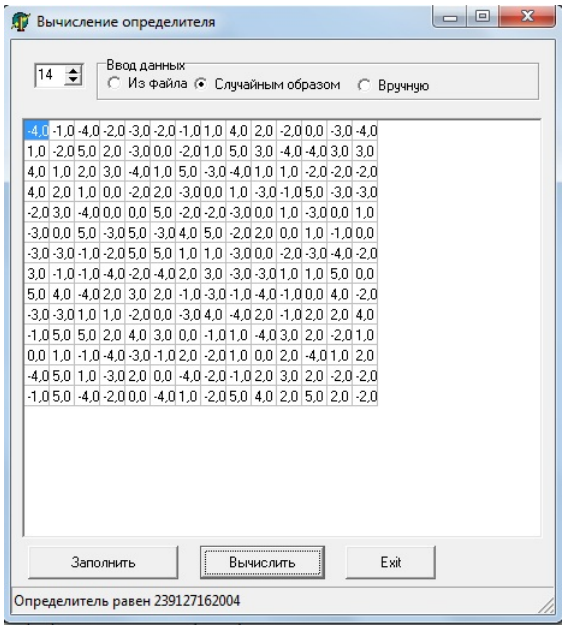
$$D = a_{11}D_1 - a_{21}D_2 + \cdots + (-1)^{n-1}a_{n1}D_n,$$

где D_k — определитель матрицы порядка $n-1$:

$$\begin{pmatrix} a_{12} & \ldots & a_{1n} \\ \ldots\ldots\ldots & & \\ a_{k-1,2} & \ldots & a_{k-1,n} \\ a_{k+1,2} & \ldots & a_{k+1,n} \\ \ldots\ldots\ldots & & \\ a_{n2} & \ldots & a_{nn} \end{pmatrix},$$

получающейся из A вычеркиванием первого столбца и k-й строки.

Методы вычисления определителя встроены в многочисленные пакеты прикладных программ: например, функция МОПРЕД в MS Excel 2010, функция *Det* в системе компьютерной математики Mathematica 8.0 и т.д. В следующей программе для вычисления определителя применен метод приведения к треугольному виду. Актуальный размер матрицы (на рисунке — 14) указан в компоненте класса TSpinEdit и может быть изменен в режиме диалога с автоматической перестройкой матрицы — компоненты класса TStringGrid.




```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils,
  Variants,  Classes, Graphics,
  Controls, Forms,  Dialogs,
  StdCtrls, ComCtrls, ExtCtrls,
  Spin, Grids;
type
  TForm1 = class(TForm)
    OpenDialog1: TOpenDialog;
    StatusBar1: TStatusBar;
    SpinEdit1: TSpinEdit;
    GroupBox1: TGroupBox;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    RadioButton3: TRadioButton;
    StringGrid1: TStringGrid;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure SpinEdit1Change(Sender: TObject);
    procedure StringGrid1KeyUp(Sender: TObject; var Key: Word; Shift: TShiftState);
  end;
Const
  maxN=100;
Type
  Arr = array [1..maxN, 1..maxN] of extended;
var
  Form1: TForm1;
  Determinant: extended;
  n: integer;
  a: Arr;
  InputOK: boolean=false;
implementation
{$R *.dfm}
```

```
{Следующая функция предназначена для вычисления определителя
матрицы a[1:n, 1:n] методом сведения к треугольному виду}
function det (n:integer; a: Arr): extended;
var
  temp, koef: extended;
  i, j, k, m: integer;
  z: integer;
  find: boolean;
begin
  z:=1;
  for i:=1 to n do
  begin
    if a[i,i]=0.0 then
    begin
      Find:=false;
      for k:=i+1 to n do
        if a[k,i]<> 0.0 then
        begin {Переставить i-ю строку с такой из последующих строк,
        которая содержит в i-й позиции ненулевое значение}
        {0} find:=true;
        {1} for j:=1 to n do
            begin
              temp:=a[i,j]; a[i,j]:=a[k,j]; a[k,j]:=temp
            end; {for j}
        {2} z:=-z;
         { Пользуясь тем, что определитель меняет знак при перестановке любых двух
         строк, запомним в z то значение (-1 или +1), которое при умножении на
         определитель полученной матрицы даст определитель исходной матрицы }
        end; {if a[k,i]<> 0.0}
    if not Find Then begin
                       det:=0.0;
                       exit;
                     end;
    end; {if a[i,i]=0.0}
  {Коль скоро оператор exit не выполнен, a[i,i] отлично от нуля. Приступим к
  обнулению всех элементов матрицы a, лежащих в столбце i ниже элемента a[i,i].
  Для этого воспользуемся тем свойством, что определитель не меняется, если  к
  элементам какой-либо строки "m" прибавить элементы  другой строки "i",
  умноженных на какое-либо число (в следующем цикле - на koef=-a[m,i]/a[i,i])}
  for m:=i+1 to n do
  begin
    koef:= -a[m,i]/a[i,i];
```

```
    for j:=i to n do
        a[m,j]:= a[m,j]+ koef*a[i,j];
  end;
  end;
  {Матрица приведена к треугольному виду. Определитель треугольной матрицы
  равен произведению диагональных элементов}
  temp:=z;
  For i:=1 to n do temp:= temp* a[i,i];
  det:=temp;
end;
//---------------------------------------------
procedure TForm1.Button2Click(Sender: TObject);
begin
    Determinant:=det (n, a);
    StatusBar1.SimpleText:= 'Определитель равен ' +FloatToStr (Determinant);
 end;
//---------------------------------------------
procedure TForm1.Button3Click(Sender: TObject);
begin
  close
end;
//---------------------------------------------
procedure TForm1.FormActivate(Sender: TObject);
var
  i, j: integer;
begin
  with StringGrid1 do begin
    ColCount:=3;
    RowCount:=ColCount;
    n:=ColCount;              //порядок матрицы
    SpinEdit1.Value:=n;
    DefaultColWidth:=18;
    DefaultRowHeight:=18;
    FixedCols:=0;
    FixedRows:=0;
    Options:=[goEditing,      //разрешение на редактирование
    goVertLine,goHorzLine];  //отображение разделительных линий
    Font.Size:=8;
    for i:=0 to n-1 do
      for j:=0 to n-1 do Cells[i,j]:='0';
  end;
end;
```

```
//-------------------------------------------
procedure TForm1.SpinEdit1Change(Sender: TObject);
var
  i, j: integer;
begin
  n:= SpinEdit1.Value;
  with StringGrid1 do
  begin
    RowCount:=n;
    ColCount:=n;
    for i:=0 to n-1 do
    for j:=0 to n-1 do Cells[i,j]:='0';
  end;
end;
//-------------------------------------------
procedure TForm1.Button1Click(Sender: TObject);
//Ввод данных. 3 способа
var
  f: textFile;
  i, j: integer;
begin
  InputOK:=false;
  if Radiobutton1.Checked then
  begin
    OpenDialog1.InitialDir:=GetCurrentDir;
    if OpenDialog1.execute then
    begin
      assignFile (f, OpenDialog1.FileName);
      reset (f);
      readln (f ,n);
      SpinEdit1.Value:=n;
      try
       For i:=1 to n do
         For j:=1 to n do
         begin
           read (f, a[i,j]); StringGrid1.Cells [j-1,i-1]:=Format ('%f',[a[i,j]]);
         end;
      except
        CloseFile (f);
        showMessage ('Ошибка при вводе значения a['+ IntToStr (i)+','+
         IntToStr (j)+'] '+ 'из файла '+OpenDialog1.FileName);
        exit;
```

```
      end; //try
      Button2.Enabled:=true;
      StatusBar1.SimpleText:='n= '+ IntToStr (n)+ '. Матрица выбрана из файла '+
      OpenDialog1.FileName;
    end; //if OpenDialog1.execute
    end; //if RadioButton1.checked
    //----------------------------------------
    if RadioButton2.Checked then
    begin
      Randomize;
      with StringGrid1 do
        for i:=0 to n-1 do
          for j:=0 to n-1 do begin
                              a[i+1,j+1]:=5-random (10);
                              Cells[j,i]:=Format ('%2.1f',[a[i+1,j+1]]);
                            end;
    end;//if RadioButton2.Checked
    //----------------------------------------
    if RadioButton3.Checked then
    begin
       For i:=1 to n do
         For j:=1 to n do begin
           try
             a[i,j]:= StrToFloat(StringGrid1.Cells [j-1,i-1]);
           except
             showMessage ('Ошибка при вводе a['+ IntToStr (i)+',' + IntToStr (j)+']');
             exit;
           end; //try
         end; //for j
    end; //if RadioButton3.Checked
    InputOK:=true;
    Button2.Enabled:=true;
end;
//-----------------------------------------
procedure TForm1.StringGrid1KeyUp(Sender: TObject; var Key: Word;  Shift: TShiftState);
begin
  RadioButton1.Checked:=false;
  RadioButton2.Checked:=false;
  RadioButton3.Checked:=true;
end;
end.
```

§ 1.2. Решение системы л.а.у. методом Гаусса

Литература: [2]. Решению системы линейных алгебраических уравнений (л.а.у.) уделяется внимание в любом пакете компьютерной математики; в частности, в Mathematica 8.0 предусмотрена функция *LinearSolve*.

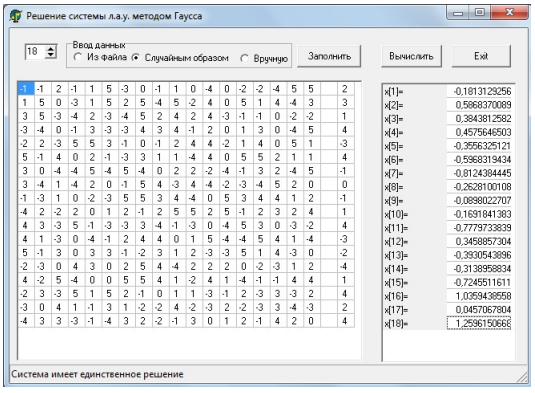
В MS Excel 2010 для решения системы л.а.у. $ax = b$ сначала с помощью функции МОБР(диапазон матрицы a) вычисляется обратная матрица a^{-1}, затем для вычисления x применяется функция МУМНОЖ (диапазон a^{-1}; диапазон b). Отметим, что выполнение второй операции требует известной аккуратности (несколько избыточной, по нашему мнению):

- выделить диапазон для результата;
- выполнить операцию вставки функции МУМНОЖ;
- указать диапазоны: сначала для a^{-1}, затем для b, и подтвердить нажатием кнопки ОК; в первой ячейке диапазона, выделенного для результата, появится значение первого неизвестного;
- нажать клавишу $F2$ (в первой ячейке отобразится формула), затем — аккорд клавиш $Ctrl + Shift + Enter$.

Метод Гаусса (или метод последовательных исключений) решения системы

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \quad i = 1, 2, \ldots, n,$$

состоит из двух этапов. На первом этапе (прямой ход) путем последовательного исключения неизвестных система приводится к треугольному виду. В случае единственности решения программа последовательно вычисляет значения неизвестных (обратный ход).



Как видно из рисунка, предусмотрены различные способы ввода данных; компонента класса TSpinEdit в верхнем-левом углу рисунка предназначена для изменения порядка системы л.а.у. и, соответственно, размеров таблицы StringGrid1 для расширенной матрицы системы (слева) и таблицы StringGrid2 для неизвестных (справа).

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls, ExtCtrls, Spin, Grids;
type
  TForm1 = class(TForm)
    OpenDialog1: TOpenDialog;
    StatusBar1: TStatusBar;
    SpinEdit1: TSpinEdit;
    GroupBox1: TGroupBox;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    RadioButton3: TRadioButton;
    StringGrid1: TStringGrid;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    StringGrid2: TStringGrid;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure SpinEdit1Change(Sender: TObject);
    procedure StringGrid1KeyUp(Sender: TObject; var Key: Word;
      Shift: TShiftState);
  end;
Const
  maxN=100;
Type
  Arr = array [1..maxN, 1..maxN] of extended;
  ar  = array [1..maxN] of extended;
var
  Form1: TForm1;
  w: integer;
  Determinant: extended;
  n: integer;
```

```
  a: Arr;
  x, b: ar;
  InputOK: boolean=false;
implementation
{$R *.dfm}
//---------------------------------------
Function Gauss(n: integer; a: Arr; b: ar; var x: ar): boolean;
var
  i, j ,k, L: integer;
  NonZero: Boolean;
  temp, koef: extended;
begin
  Gauss:=false;
  {решение системы: прямой ход}
  for i:=1 to n do
  begin
    if a[i,i]=0 then
    begin
      NonZero:=false; k:=i;
      while k<n do begin
       inc (k);
       if a[k,i]<>0 then begin
                           NonZero:=true;
                           For j:=i to n do
                           begin
                             temp:=a[k,j]; a[k,j]:=a[i,j]; a[i,j]:=temp;
                           end;
                           break
                         end;
      end; //while
      if NonZero=false then exit;
    end; //if a[i,i]=0

    for j:=i+1 to n do
    begin
      koef:=a[j,i]/a[i,i];
      for k:=i to n do
        A[j,k]:=A[j,k]-koef*A[i,k];

      B[j]:=B[j]-koef*B[i];
    end; //for j
    end; //for i
```

```
    Gauss:=true;
    {решение системы: обратный ход}
    for i:=n downto 1 do
    begin
      for k:=i+1 to n do b[i]:=b[i]-a[i,k]*x[k];
      x[i]:=b[i]/a[i,i];
    end;
end;
//---------------------------------------------
procedure TForm1.Button2Click(Sender: TObject);
var
  i: integer;
begin
  if Gauss(n, a, b, x) then
  begin
    for i:=1 to n do
         StringGrid2.Cells [1,i-1]:=Format ('%15.10f ', [x[i]]);
    StatusBar1.SimpleText:= 'Система имеет единственное решение';
  end
  else
  StatusBar1.SimpleText:= 'Система не имеет решения,'+
     ' либо имеет бесконечно много решений.';
end;
//---------------------------------------------
procedure TForm1.Button3Click(Sender: TObject);
begin
  close
end;
//---------------------------------------------
procedure TForm1.FormActivate(Sender: TObject);
var
  i, j: integer;
begin
  with StringGrid1 do begin
    ColCount:=3;
    RowCount:=ColCount;
    n:=ColCount;            //размер матрицы
    SpinEdit1.Value:=n;
    w:=20;
    DefaultColWidth:=w;
    DefaultRowHeight:=16;
    FixedCols:=0;
```

```
    FixedRows:=0;
    Options:=[goEditing,    //разрешение на редактирование
    goVertLine,goHorzLine]; //отображение разделительных линий
    Font.Size:=8;
    for i:=0 to n-1 do
      for j:=0 to n-1 do Cells[i,j]:='0';
  end;

   with StringGrid2 do begin
    ColCount:=2;
    RowCount:=n;
    DefaultRowHeight:=16;
    FixedCols:=1;
    FixedRows:=0;
    Options:=[goEditing,goHorzLine];
    Font.Size:=8;
    for i:=0 to n-1 do
      Cells[0,i]:=Format('x[%d]=', [i+1]);
  end; //with
end;
//-----------------------------------------------
procedure TForm1.SpinEdit1Change(Sender: TObject);
var
  i, j: integer;
begin
  n:= SpinEdit1.Value;
  with StringGrid1 do
  begin
    RowCount:=n;
    ColCount:=n+2;
    for i:=0 to n+1 do
      for j:=0 to n-1 do Cells[i,j]:='0';
    for i:=0 to n-1 do Cells[n+1,i]:='b['+intToStr(i+1)+']';
    for i:=0 to n-1 do Cells[n,i]:='';
    for i:=0 to n+1 do Cells[i,n]:='';
  end;
  with StringGrid2 do
  begin
    RowCount:=n;
    for i:=0 to n-1 do
      begin
        Cells[0,i]:=Format('x[%d]=', [i+1]);
```

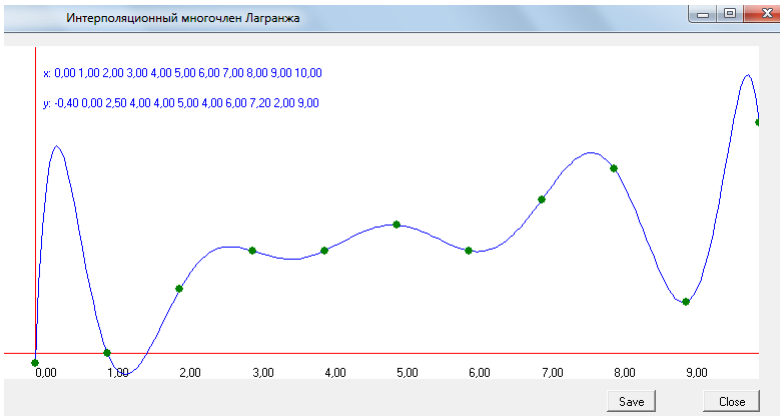
```
        Cells[1,i]:=' ';
      end;
  end;
end;
//-----------------------------------------
procedure TForm1.Button1Click(Sender: TObject);
var
  f: textFile;
  i,j: integer;
begin
  InputOK:=false;
  if Radiobutton1.Checked then
  begin
    OpenDialog1.InitialDir:=GetCurrentDir;
    if OpenDialog1.execute then begin
      assignFile (f, OpenDialog1.FileName);
      reset (f);
      readln (f ,n);
      SpinEdit1.Value:=n;
      try
       For i:=1 to n do
       begin
         For j:=1 to n do
         begin
           read (f, a[i,j]);
           StringGrid1.Cells [j-1,i-1]:=Format ('%f',[a[i,j]]);
         end;
         read (f, b[i]);
         StringGrid1.Cells [n+1,i-1]:=Format ('%f',[b[i]]);
       end;
      except
        CloseFile (f);
        showMessage ('Ошибка при вводе значения a['+ IntToStr (i)+','+
         IntToStr (j)+'] '+ 'из файла '+OpenDialog1.FileName);
        exit;
      end; //try
      Button2.Enabled:=true;
      StatusBar1.SimpleText:='n= '+ IntToStr (n)+
                             '. Матрица выбрана из файла '+ OpenDialog1.FileName;
    end; //if OpenDialog1.execute
    end; //if RadioButton1.checked
    //----------------------------------------
```

```
    if RadioButton2.Checked then
    begin
      Randomize;
      with StringGrid1 do
        for i:=0 to n-1 do
        begin
          b[i+1]:= 5-random (10);                  //случайные числа из диапазона -4..5
          Cells[n+1,i]:=Format ('%2.0f',[b[i+1]]);
          for j:=0 to n-1 do begin
                                a[i+1,j+1]:=5-random (10);
                                Cells[j,i]:=Format ('%2.0f',[a[i+1,j+1]]);
                              end;
        end;
    end;//if RadioButton2.Checked
    //----------------------------------------
    if RadioButton3.Checked then
    begin
       For i:=1 to n do
       begin
         b[i]:= StrToFloat(StringGrid1.Cells [n+1,i-1]);
         For j:=1 to n do
         try
           a[i,j]:= StrToFloat(StringGrid1.Cells [j-1,i-1]);
         except
           showMessage ('Ошибка при вводе значения a['+
                          IntToStr (i)+',' +IntToStr (j)+']');
           exit;
         end; //try
       end; //for i
    end;//if RadioButton3.Checked
    InputOK:=true;
    Button2.Enabled:=true;
end;
//-----------------------------------------
procedure TForm1.StringGrid1KeyUp(Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  RadioButton1.Checked:=false;
  RadioButton2.Checked:=false;
  RadioButton3.Checked:=true;
end;
end.
```

§ 1.3. Интерполяционный многочлен Лагранжа

Литература: [2], [14]. Нарисовать график многочлена степени n, проходящий через заданные точки (x_i, y_i), $i = 0, \ldots, n$:

$$f(x) = \sum_{i=0}^{n} y_i \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}.$$



Более полное изложение см. в [14, с. 150–154], где предусмотрены дополнительные возможности: перетаскивание вершин и др.

Для построения интерполяционного многочлена в системе Mathematica 8.0 используется функция *InterpolatingPolynomial*, после чего для вывода графика можно применить функцию *Plot*, совмещенную с функцией *ListPlot* для отображения исходных точек. В MS Excel 2010 для построения графика можно использовать мастер диаграмм и графиков.

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, Buttons, ExtDlgs;
type
  TForm1 = class(TForm)
    Image1: TImage;
    SpeedButton1: TSpeedButton;
    SpeedButton2: TSpeedButton;
    SavePictureDialog1: TSavePictureDialog;
    procedure SpeedButton1Click(Sender: TObject);
```

```
    procedure FormActivate(Sender: TObject);
    procedure SpeedButton2Click(Sender: TObject);
  private
    Procedure DrawLagr;
  end;
var
  Form1: TForm1;
  W,  H: integer;
const
  n=10;
  MidN=200;    {число точек между узловыми точками}
  x1=-1; y1=-1; x2=10;  y2=12; {окно <<на бумаге>>}
type
  Arr=array[0..n] of extended;
var
 {координаты узловых точек}
  xt: Arr=(0,1,2,3,4,5,6,7,8,9,10);
  yt: Arr=(-0.4,0,2.5, 4, 4, 5, 4, 6, 7.2, 2, 9);
  deltaX:        {шаг между узловыми точками}
  extended;
implementation
{$R *.dfm}
function MyLagr (n: integer; xt: extended; x,y: Arr): extended;
var
  i, j: integer;
  S, P: extended;
begin
  S:=0;
  for i:=0 to n do
  begin
     P:=1;
     for j:=0 to n do
       if i=j then continue else
        P:=P*(xt-x[j])/(x[i]-x[j]);
     S:=S+y[i]*P;
  end;
  MyLagr:=S;
end;
//----------------------------------------
Function ScaleX (x: extended): integer;
begin
  ScaleX:= trunc (W*(x-x1)/(x2-x1));
```

```
end;
//---------------------------------------------
Function ScaleY (y: extended): integer;
begin
  ScaleY:= trunc (H*(y-y2)/(y1-y2));
end;
//---------------------------------------------
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
  close;
end;
//---------------------------------------------
Procedure TForm1.DrawLagr;
var
  i: integer;
  x0, y0: integer;
  u,v: extended;
  s: string;
Begin
  With Image1.Canvas do
  begin
    Pen.Color:=clRed;
    //Построение осей координат
    MoveTo (ScaleX(0), ScaleY (y1)); LineTo (ScaleX(0), ScaleY(y2));
    MoveTo (ScaleX(x1), ScaleY(0));  LineTo (ScaleX(x2), ScaleY(0));
    //Построение графика
    deltaX:=(xt[n]-xt[0])/MidN; u:=xt[0]; v:=MyLagr(n, u, xt, yt);
    for i:=0 to n do
    begin
      textOut (ScaleX (xt[i]), ScaleY (-0.5), Format ('%f', [xt[i]]));
    end;
    MoveTo (ScaleX(u), ScaleY(v));
    Pen.Color:=clBlue;
    For i:=1 to MidN do
    begin
      u:=u+deltaX; v:=MyLagr (n,u,xt, yt);
      LineTo (ScaleX(u), ScaleY(v));
    end; //for i

    for i:=0 to n do begin
      x0:=ScaleX(xt[i]); y0:=ScaleY(yt[i]);
      brush.Color:=clGreen; pen.color:=clGreen;
```

```
      Ellipse (x0-4, y0-4, x0+4, y0+4);
    end;
    end; //With
    With image1.canvas do
    begin
      font.color:= clBlue; brush.Color:=clWhite;
      s:='x: ';
      For i:=0 to n do s:=s+ Format ('%f ', [xt[i]]);
      TextOut (80, 20, s);
      s:='y: ';
      For i:=0 to n do s:=s+ Format ('%f ', [yt[i]]);
      TextOut (80, 50, s);
    end;
end;
//---------------------------------------------
procedure TForm1.FormActivate(Sender: TObject);
begin
  W:=Image1.Width; H:=Image1.Height;
  DrawLagr;
end;
//---------------------------------------------
procedure TForm1.SpeedButton2Click(Sender: TObject);
begin
  SavePictureDialog1.InitialDir:=getCurrentDir;
  if SavePictureDialog1.Execute then
    Image1.Picture.SaveToFile(SavePictureDialog1.Filename);
end;
end.
```

§ 1.4. Метод наименьших квадратов

Литература: [2]. Метод наименьших квадратов берет начало с работ Гаусса и Лежандра конца 18-го и начала 19-го веков и в узком смысле (рассматриваемом здесь) заключается в применении элементов теории вероятностей к решению системы переопределенных уравнений: после подстановки в исходную систему уравнений значений неизвестных величин, вычисленных данным способом, в правых частях получаются величины, «малые» в том смысле, что сумма квадратов подобных же «остатков» после подстановки каких бы то ни было других значений неизвестных будут не меньше; в результате создается ценная возможность судить о вероятности ошибок неизвестных.

Для заданных «экспериментальных точек» $(x_0, y_0), \cdots, (x_m, y_m)$ требуется найти многочлен заданной степени n (многочлен необязательно принимает значение y_k при значении аргумента x_k): $c_0x^0 + \cdots + c_nx^n$,

такой, что сумма квадратов отклонений: $g(c_0, \cdots, c_n) =$

$$\sum_{k=0}^{m} \left(c_0 {x_k}^0 + \cdots + c_n {x_k}^n - y_k\right)^2$$

достигает наименьшее значение. Необходимое условие экстремума:

$$\frac{\partial g}{\partial c_i} = 0, \quad i = 0, \cdots, n,$$

в нашем случае есть условие минимума и равносильно следующей системе л.а.у. относительно неизвестных $c_0, \cdots, c_n$:

$$\sum_{j=0}^{n} a_{ij} c_j = b_i,$$

где $a_{ij} = \sum\limits_{k=0}^{m} {x_k}^{i+j}, \quad b_i = \sum\limits_{k=0}^{m} y_k {x_k}^i; \quad i = 0, \cdots, n.$

Из контекста понятно, что в данном случае система л.а.у. имеет единственное решение.

В системе компьютерной математики Mathematica 8.0 метод наименьших квадратов реализован функцией *LeastSquares*, в MS Excel 2010 — при помощи диаграммы и линии тренда.

```
unit Unit1;
interface
uses  Math, Windows, Messages, SysUtils, Variants, Classes, Graphics,
      Controls, Forms,  Dialogs, ExtCtrls, StdCtrls;
type
  TForm1 = class(TForm)
    Image1: TImage;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  end;
var
  Form1: TForm1;
```

```
const
  n=30; //n+1 - число узловых точек
  m=10; //степень искомого многочлена
type
  arrN = array   [0..n] of double; //тип для векторов x и y из n+1 узлов
  arM = array [0..m] of double;    //тип для искомого вектора коэффициентов
                                   //многочлена степени m
  arrM = array[0..m, 0..m] of double;
  //матрица коэффициентов системы л.а.у. для вычисления коэффициентов многочлена
const
  p=50;                      //число точек между узлами
  a1=-1; a2=8; b1=-1; b2=8; //воображаемое окно для рисования графика на бумаге
var
  i: Byte;
  W, H: integer; //окно на экране
  c: arM;
  x, y: arrN;
implementation
{$R *.dfm}
//ScaleX и ScaleY пересчитывают координаты в окно (a1,b1)-(a2,b2)
Function ScaleX(x: double): integer;
begin
  ScaleX:=Round (W*(x-a1)/(a2-a1));
end;
Function ScaleY(y: double): integer;
begin
  ScaleY:=Round (H*(y-b2)/(b1-b2));
end;
//---------------------------------------------
procedure SimpleGauss(n,m: Byte; x,y: arrN; var  c: arM);
var
  A: arrM;      //матрица коэффициентов системы л.а.у.
  B: arM;       //свободные члены системы л.а.у.
  i,j,k: Byte;
begin
  for j:=0 to m do
  for k:=j to m do
  begin
    A[j,k]:=0;
    for i:=0  to n do
      A[j,k]:=A[j,k]+Power (x[i],k+j);
    A[k,j]:=A[j,k];
```

```
  end;
    for k:=0 to m do
    begin
      B[k]:=0;
      for i:=0 to n do
        B[k]:=B[k]+y[i]*Power(x[i],k);
     end;
    {решение системы: прямой ход}
    for i:=0 to m-1 do
      for j:=i+1 to m do
    begin
      for k:=i+1 to m do
         A[k,j]:=A[k,j]-A[i,j]*A[k,i]/A[i,i];
      B[j]:=B[j]-B[i]*A[i,j]/A[i,i];
    end;
    {решение системы: обратный ход}
    for j:=m downto 0 do
    begin
      c[j]:=B[j];
      for k:=j+1 to m do c[j]:=c[j]-A[k,j]*c[k];
      c[j]:=c[j]/A[j,j];
    end;
end;
//---------------------------------------------------------
procedure TForm1.Button1Click(Sender: TObject);
//Вывод графика в Image1
const
  r=2;
var
  i: integer;
  x1, y1, m1, h1: double;
begin
  randomize;   m1:=0;
  for i:=0 to n do begin
                     x[i]:= i/8;
                     y[i]:=6*random;
                     if y[i]>m1 then m1:=y[i];
                    end;
  SimpleGauss(n,m,x,y,c);
  W:=Image1.Width; H:=Image1.Height;
  with Image1.Canvas do
  begin
```

```
    MoveTo(ScaleX(a1),ScaleY(0));    LineTo(ScaleX(a2),ScaleY(0));  //ось OX
    MoveTo(ScaleX(0) ,ScaleY(b1));   LineTo (ScaleX(0),ScaleY(b2)); //ось OY
    brush.Color:= clGreen;
    pen.color:=clGreen;
    pen.Width:=1;
    for i:=0 to n do   //нарисуем n+1  заданных точек
    Ellipse (ScaleX(x[i])-r, ScaleY(y[i])-r, ScaleX(x[i])+r, ScaleY(y[i])+r);
    pen.color:=clBlue;
    brush.Color:= clWhite;
    for i:=1 to n do
      if i mod 4=0 then TextOut (ScaleX(x[i]), ScaleY(-0.25), Format ('%f',[x[i]]));
      for i:=1 to round (m1) do TextOut (ScaleX(-0.125), ScaleY(i), Format ('%d',[i]));
      h1:=(x[n]-x[0])/p;
//Найдем самую левую точку графика и установим курсор
      x1:=x[0];
      y1:=Poly (x1, c);
      MoveTo (ScaleX(x1), ScaleY(y1));
      for  i:=1  to p do begin
//Из текущей точки графика проведем отрезок к следующей точке
        x1:=x1+h1;
        y1:=Poly (x1, c);
        LineTo(ScaleX(x1),ScaleY(y1));
      end;
  end;
end;
end.
```

§ 1.5. Многоразрядные арифметические операции

Пусть над двумя заданными многоразрядными целыми положительными числами требуется выполнить выбранную арифметическую операцию; если выбрана операция деления, дополнительно запросить требуемую точность.

В системах компьютерной математики предусмотрены действия над числами практически неограниченной разрядности. Например, в Mathematica 8.0 команда 2012! «мгновенно» вычислит результат, а команда

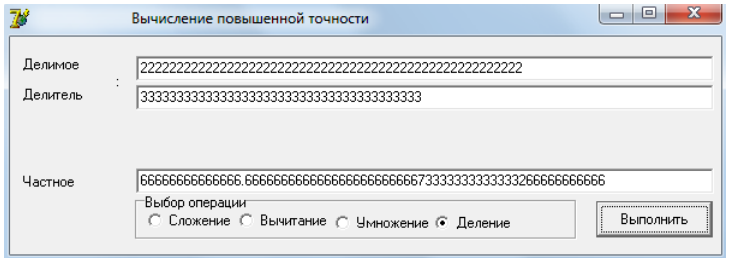
Floor[N[Log[10, 2012!], 10]]+1

позволит узнать, что десятичная запись результата содержит 5776 цифр. Другой пример: команда N[Pi, 2000000] выведет число π с 2000000 верными знаками.

Для последнего вычисления компьютеру автора с характеристиками

2.2 GHz, 3.00 Гб и т. п. понадобилось около 15 сек. Упомянутая частота (но не другие характеристики компьютера) сравнима с аналогичной характеристикой процессора персонального компьютера, позволившего Фабрису Беллару установить в 2010 г. мировой рекорд вычисления π с точностью до 2,7 трлн знаков после запятой. Если склеить в длину листы формата A4 в количестве, обеспечивающем десятичную запись данного результата, то протяженность «папируса» превысит расстояние от Земли до Солнца (впрочем, бумаги не хватит, даже если вырубить все леса на планете).

Известно немало библиотек, разработанных для действий с многоразрядными числами: коммерческие (IMSL), бесплатные для некоммерческого использования (LiDIA, MIRACL) и бесплатные (GMP, NTL, CLN, MPI, Imath). Отметим здесь же, что в языке Phyton для действий над целыми числами неограниченной длины предусмотрен специальный тип long (см., например, [13]).




```
unit Unit1;
interface
uses
   math, Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
   Forms,  Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Button1: TButton;
    Edit4: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    GroupBox1: TGroupBox;
    RadioButton1: TRadioButton;
```

```
    RadioButton2: TRadioButton;
    RadioButton3: TRadioButton;
    RadioButton4: TRadioButton;
    Label5: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure RadioButton2Click(Sender: TObject);
    procedure RadioButton3Click(Sender: TObject);
    procedure RadioButton4Click(Sender: TObject);
    procedure RadioButton1Click(Sender: TObject);
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
function dt(c: char): byte;
//по заданному символу находим его цифровое представление
begin
  dt:=ord(c)-ord('0');
End;
//---------------------------------------------
function Sums(a,b: string): string;
{По двум заданным строковым представлениям a и b натуральных чисел большой
разрядности вычисляется их сумма, и ее строковое представление возвращается
в виде значения функции. Применяется при делении "уголком"}
var
  c: string;
  l,w,i,t: integer;
  la,lb: integer;
Begin
   la:=length(a);
   lb:=length(b);
   if la<lb then  for i:=la+1 to lb do a:='0'+a
   else  for i:=lb+1 to la do b:='0'+b;
//Строки выровнены добавлением лидирующих нулей
  l:=length(a);
  c:=''; // Инициализация пустой строкой
  w:=0;
  for i:=l downto 1 do
  Begin t:=dt(a[i])+dt(b[i])+w; if t>=10 then
  //здесь dt(a[i]) - цифра, соответствующая символу a[i]
  begin
```

```
    c:=chr(t-10+48)+c;
    w:=1
  end
  else  begin
          c:=chr(t+48)+c;
          w:=0
        end
  End;
  c:=chr(w+48)+c;
  Sums:=c
End;
//-----------------------------------------------
//Вычитание <<столбиком>> из большего числа меньшее
function Subs(a,b: string): string;
var
  c: string;
  i,j,k,l: integer;
  la,lb: integer;
Begin
  la:=length(a); lb:=length(b);
   if la<lb
    then  for i:=la+1 to lb do a:='0'+a
    else  for i:=lb+1 to la do b:='0'+b;
    l:=length(a);
    c:='';
    for i:=l downto 1 do
      if dt(a[i])>=dt(b[i]) then
        c:=chr(dt(a[i]) - dt(b[i])+48)+c
      else begin
            j:=i;
            repeat
              dec(j)
            until a[j]<>'0';
            for k:=j+1 to i-1 do
            a[k]:='9';
            c:=chr(dt(a[i])+10-dt(b[i])+48)+c;
            a[j]:=chr(dt(a[j])-1+48)
          end;
    Subs:=c
End;
//-----------------------------------------------
Function Mult(a: string;b: char):string;
```

```
//Умножение строкового числа a на стороковую цифру b
var
  digit, L, w, i, t: integer;
  c: string;
Begin
  c:='';
  L:=length(a);
  w:=0; // "в уме" - нулевое значение
  digit:=dt(b);
  for i:=L downto 1 do
   Begin
    t:=digit*dt(a[i])+w; // t mod 10 - остаток от деления на 10
    c:=chr(( t mod 10)+48)+c;
    w:=t div 10;
    // это 0, если число меньше 10 (т.е. <<в уме ничего нет>>),
    // либо 1, если число больше 10 (тогда в уме - 1).
   End;
   c:=chr(w+48)+c;
   Mult:=c
End;
//Сравнение двух чисел, представленных в виде строк: что больше?
//-------------------------------------------
Function Comp(a,b: string): boolean;
var
  la, lb, i: integer;
Begin
  la:=length(a);
  lb:=length(b);
  if la<lb then
    for i:=la+1 to lb do a:='0'+a
  else
    for i:=lb+1 to la do b:='0'+b;
  Comp:= a<b
End;
//-------------------------------------------
Function Compt(a: string;b:string): boolean;
var
  la, lb, i: integer;
Begin
  la:=length(a);
  lb:=length(b);
  if la<lb then
```

```
    for i:=la+1 to lb do a:='0'+a
  else
    for i:=lb+1 to la do b:='0'+b;
  Compt:= a<=b
End;
//---------------------------------------------
function Mult2(sa, sb: string): string;
//Умножение
var
  i, k, n: integer;
  a, c: array of byte;
  sc, sw, s: string;
begin
  n:=max(length (sa), length(sb));
  setLength (a, n+1);
  SetLength (c, n+1);
  while Length (sa)<n do sa:='0'+sa;

  while Length (sb)<n do sb:='0'+sb;
  for i:=1 to n do a[i]:= StrToInt(sa[i]);
  sc:='0';
  for i:=Length(sb) downto 1 do
  begin
    sw:=Mult (sa, sb[i]);
    for k:=1 to n-i do sw:=sw+'0';
    s:=Sums(sc, sw);
    sc:=s;
  end;
  while sc[1]='0' do sc:=copy (sc,2,1000);
  result:=sc;
end;
//---------------------------------------------
function Dels(a: string; d:string; q: integer):string;
label 1, metka1;
var
  d1, d2, d3, a0, a3, s: string;
  k, i, L: integer;
  digit, i0:char;
  delta: integer;
  la, ld: integer;
Begin
  la:=length(a);
```

```
  ld:=length(d);
  Delta:=0;
  if la<ld
    then  for i:=la+1 to ld do begin a:=a+'0'; inc (Delta) end
  else  for i:=ld+1 to la do d:='0'+d;
  d3:='';
  L:=length (a);
  for i:=1 to q do a:=a+'0';
  L:=L+q+Delta;
//В ответе следует перенести знак запятой влево на q+delta позиций
  d2:=d; k:=0;
  repeat
    inc(k)
  until Compt(d2,copy(a,1,k));
  d1:=copy(a,1,k-1);
  While k<=L do begin
    d1:=d1+copy(a,k,1);
  for digit:='0'  to '9' do
  begin
    a3:=Mult(d2, digit);
    if Comp(d1,a3) then goto 1
    else
      begin  i0:=digit; a0:=a3; end
    end;
    1: digit:=i0;
    d3:=d3+digit;
    d1:= Subs(d1, a0);
    inc(k);
  END;
  if Length (d3)-delta-q=0 then s:='0' else
  s:=copy (d3,1, Length (d3)-delta-q);
  d3:=s+'.'+copy(d3,Length (d3)-delta-q+1, 1000);
  result:=d3;
End;
//---------------------------------------------
procedure TForm1.Button1Click(Sender: TObject);
var
  a, b: string;
  q: integer;
begin
  label3.Hide;
  a:=Edit1.Text;
```

```
  b:=Edit2.Text;
  Edit3.Hide;
  if radiobutton1.Checked then Edit4.text:=Sums (a,b);
  if radiobutton2.Checked then Edit4.text:=Subs (a,b);
  if radiobutton3.Checked then Edit4.text:=Mult2 (a,b);
  if radiobutton4.Checked then begin
                                 q:=strToInt (Edit3.Text);
                                 Edit4.text:=dels (a,b,q);
                               end;
end;
//---------------------------------------------
procedure TForm1.FormActivate(Sender: TObject);
begin
  label1.Caption:='Слагаемое 1';
  label2.Caption:='Слагаемое 2';
  label3.Caption:='Число знаков после запятой';
  label4.Caption:='Сумма';
  label5.Caption:='+';
  Edit3.Hide; Label3.Hide;
end;
//---------------------------------------------
procedure TForm1.RadioButton2Click(Sender: TObject);
begin
  edit1.Text:=''; edit2.Text:=''; edit3.Text:=''; edit4.Text:='';
  label3.Hide;
  Edit3.Hide;
  if radiobutton2.Checked then begin
                                 label1.caption:='Уменьшаемое';
                                 label2.caption:='Вычитаемое';
                                 label4.caption:='Разность';
                                 label5.caption:='-';
                               end;
end;
//---------------------------------------------
procedure TForm1.RadioButton3Click(Sender: TObject);
begin
  edit1.Text:=''; edit2.Text:=''; edit3.Text:=''; edit4.Text:='';
  label3.Hide; Edit3.Hide;
  if radiobutton3.Checked then begin
                                 label1.caption:='Множимое';
                                 label2.caption:='Множитель';
                                 label4.caption:='Произведение';
```

```
                                label5.caption:='x';
                               end;
end;
//---------------------------------------------
procedure TForm1.RadioButton4Click(Sender: TObject);
begin
  edit1.Text:=''; edit2.Text:=''; edit3.Text:=''; edit4.Text:='';
  label3.Show; Edit3.Show;
  if radiobutton4.Checked then begin
                                label1.caption:='Делимое';
                                label2.caption:='Делитель';
                                label4.caption:='Частное';
                                label5.caption:=':';
                               end;
end;
//---------------------------------------------
procedure TForm1.RadioButton1Click(Sender: TObject);
begin
  edit1.Text:=''; edit2.Text:=''; edit3.Text:=''; edit4.Text:='';
  label3.Hide; Edit3.Hide;
  if radiobutton1.Checked then begin
                                label1.caption:='Слагаемое 1';
                                label2.caption:='Слагаемое 2';
                                label4.caption:='Сумма';
                                label5.caption:='+';
                               end;
end;
end.
```

Глава 2.

Эффективные алгоритмы

§ 2.1. Минимальное остовное дерево

Литература: [9]. Под остовным деревом связного неориентированного графа понимают дерево, образованное всеми вершинами графа и некоторыми его ребрами. Наша цель — поиск и построение остовного дерева с минимальной суммой весов ребер. Алгоритм поиска минимального остовного дерева просматривает ребра исходного графа, упорядоченные по неубыванию весов: если ребро не образует цикл в совокупности с ребрами, уже включенными в дерево, то данное ребро включается в дерево (окрашивается в синий цвет), в противном случае ребро не включается в дерево (окрашивается в красный цвет).

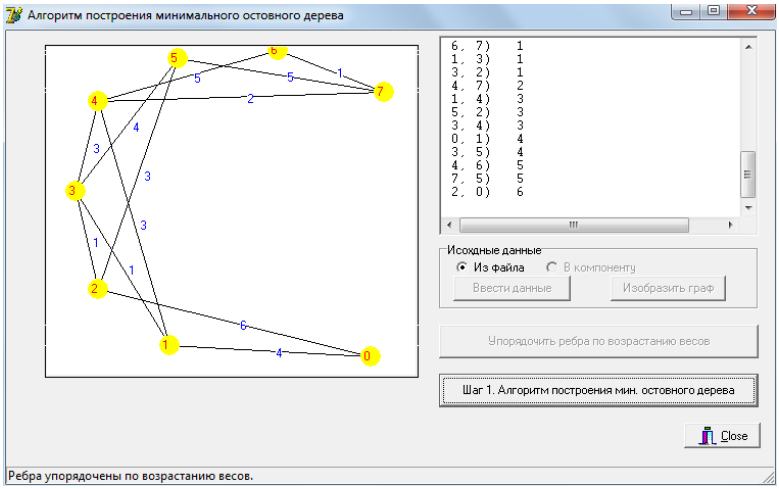
Проверка того, образует ли просматриваемое ребро цикл в совокупности с ребрами, уже включенными в дерево (синими ребрами), выполняется так. Ребра, включенные в дерево, составляют граф, имеющий одну или несколько связных компонентов. Вершины, принадлежащие отдельной связной компоненте, объединяются в совокупность, которую назовем «букетом». Некоторое ребро образует цикл с ребрами, уже включенными в дерево, если обе его концевые вершины принадлежат одному и тому же букету.

Работа алгоритма по построению минимального остовного дерева заканчивается, когда количество ребер, окрашенных в синий цвет, становится равным числу, на единицу меньшему числа вершин исходного графа, или, что то же самое, когда все вершины графа оказываются в одном букете. Если же граф не содержит остовное дерево, т. е. является несвязным, алгоритм заканчивается после раскраски всех ребер графа; при этом число ребер, окрашенных в синий цвет, оказывается недостаточным для формирования остовного дерева.

Минимальное остовное дерево в связном графе с множеством ребер E ребрами можно найти за время $O(|E| \cdot log|E|)$ (только не надо думать, что автор забыл указать основание логарифма).

При вводе данных из файла программа предоставляет возможность ука-

зывать точки размещения вершин на холсте компоненты; после ввода и отображения графа вершины можно перетаскивать для улучшения зрительного восприятия. Предусмотрена визуализация результатов промежуточных шагов алгоритма. На рисунке показан результат после упорядочения ребер по неубыванию весов.




```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, ComCtrls, ExtCtrls, Buttons;
type
  TForm1 = class(TForm)
    Memo1: TMemo;
    StatusBar1: TStatusBar;
    OpenDialog1: TOpenDialog;
    PaintBox1: TPaintBox;
    GroupBox1: TGroupBox;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    BitBtn1: TBitBtn;
    procedure Button1Click(Sender: TObject);
```

```
    procedure Button2Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure PaintBox1MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure PaintBox1MouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure PaintBox1MouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
  private
    procedure reDraw;
  end;
const
   nMax=50;   r=10;
type
  TEdgeWeightColor= Record x,y,w: byte; C: Tcolor  end;
var
  Form1: TForm1;
  SubStep2: integer=0; //сколько раз выполняется итерация шага 2.
  ClickCounter: integer=0;
  Tree:set of byte;
  pressed: boolean=false;
  //Проверка, нажата ли кнопка мыши.
  i0: integer=0;
  //Выбор перемещаемой вершины.
  E: array of TEdgeWeightColor;
  //Динамический массив, каждый элемент -
  //запись: номер двух вершин, составляющих концы очередного ребра, и вес ребра
  //количество элементов массива (число ребер), считывается из файла
  n, //число вершин,
  NE://число ребер
  integer;
  VertexPosition: array of TPoint; //координаты концов вершины
  BlossemCount: integer=0; //количество букетов, т.е. числа связных компонент
  Blossem: array of Set of byte;//Каждый букет - набор вершин связной компоненты графа
  EnterTime: boolean = true;
{Перемещение мыши по компоненте PaintBox1 в период ввода данных не влечет
 действий при EnterTime=true и имеет целью добраться до позиций вершин графа.
 После завершения ввода данных (EnterTime уже будет присвоено False)
  применяется для перетаскивания вершин в новые позиции}
```

```
  ShowResult: boolean = false;
 implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
var
  i, j, x1, y1, x2, y2: integer;
  f: TextFile;
  s: string;
begin
  if RadioButton1.Checked then //выбор файла  с данными
  begin
    OpenDialog1.InitialDir:=GetCurrentDir; //начать поиск с текущей папки
    if OpenDialog1.Execute then assignFile (f, OpenDialog1.FileName);
    reset (f);
    read (f, n, NE);
    ShowMessage (Format ('n= %2d, NE=%2d', [n, NE]));
    SetLength (E, NE); //Определили границы массива ребер
    SetLength (VertexPosition, n); //Определили границы массива координат вершин
    SetLength (Blossem, n);
//Границы массива букетов определили с запасом:
//количество букетов не превышает количества вершин
    StatusBar1.Panels[0].text:=Format ('n= %d, NE=%d', [n, NE]);
    Memo1.Lines.Add (Format ('n= %d, NE=%d', [n, NE]));
    Memo1.Lines.Add ('(N Ребро Вес)');
    Memo1.Lines.Add('');
    for i:=1 to NE do
    begin
      read (f, E[i].x, E[i].y, E[i].w);
      E[i].C:= clBlack;
      s:=Format ('%2d,%2d) %3d', [E[i].x, E[i].y, E[i].w]);
      Memo1.Lines.Add(s); //Вывели ребро и его вес
    end;
    end; //if RadioButton1.checked
  Button1.Enabled:=false;
//Если компонента находится в фокусе ввода, то
//ее можно выбрать не только щелчком мыши, но и нажатием клавиши ENTER
  x1:=PaintBox1.Left-2;
  x2:=PaintBox1.Left+PaintBox1.Width+2;
  y1:=PaintBox1.Top-2;
  Y2:=PaintBox1.Top+PaintBox1.Height+2;
  Canvas.PolyLine ([Point(x1, y1), Point(x1,y2), Point(x2,y2),
   Point(x2,y1), Point(x1,y1)]);
```

```
  Application.ProcessMessages;
  ShowMessage ('Ввод данных завершен. Пожалуйста, '+ intToStr(n) +
  ' щелчками укажите слева удобные Вам позиции ' +#13#10+
  'размещения вершин в пределах прямоугольника, изображенного слева.'+
  'Вершины потом можно будет перетаскивать  и вручную.');
  StatusBar1.SimpleText:=
                'Укажите места для вершин щелчками внутри прямоугольника.';
end;
//-----------------------------------------------
procedure TForm1.Button2Click(Sender: TObject);
begin
  redraw;
end;
//-----------------------------------------------
procedure TForm1.reDraw;
var
  i,j: integer; x1, y1, x2, y2: integer; a, b: integer; s: string;
begin
  Button2.Enabled:=false;
  Button3.Enabled:=true;
  Button3.SetFocus;
with PaintBox1.canvas do
begin
  brush.Style:=bsSolid;
  brush.Color:=clWhite;
  pen.Color:=clWhite;
  rectangle (-1,-1, width, height);
  pen.Color:=clBlack;
  for i:=1 to NE do
  begin
     with E[i] do
     begin
       Brush.Style:=bsSolid;
       a:=E[i].x; b:=E[i].y;
       s:= Format ('i=%d, (%d,%d)', [i, a, b]);
       x1:=VertexPosition[a].x;
       y1:=VertexPosition[a].y;
       s:=s+Format ('Координаты (%d,%d)',[x1,y1]);
       MoveTo(x1,y1);
       //Курсор разместили в позицию одной из вершин ребра E[i]=(a,b).
       x2:=VertexPosition[b].x;
       y2:=VertexPosition[b].y;
```

```
       s:=s+ Format ('Координаты (%d,%d)',[x2,y2]);
       LineTo (x2,y2);
       font.Color:=clBlue;
       TextOut((x1+x2) div 2+r-4, (y1+y2) div 2-4, intToStr(E[i].w));
     end;
  end; //for i
     for i:=1 to n do  begin //переберем все вершины i
         a:=E[i].x; b:=E[i].y;
         x1:=VertexPosition[a].x;
         y1:=VertexPosition[a].y;
         Brush.Color:=clYellow;
         Pen.Color:=clYellow;
         ellipse(x1-r, y1-r, x1+r, y1+r);
         Font.Color:=clRed;
         TextOut(x1-r div 2 -2, y1 -r div 2 -2, intToStr(a));
     if ShowResult then
     begin
        pen.color:=clLime; Brush.Color:=clLime;
        rectangle(x1+r+1, y1-r, x1+2*r, y1+r);
        font.Color:=clRed;
      end; //if ShowResult
     end; //for i
  end; //with PaintBox1.Cavas
  StatusBar1.SimpleText:='Можно мышью перетащить вершины в новые позиции.'
end;
//-----------------------------------------------
procedure TForm1.FormCreate(Sender: TObject);
begin
  Memo1.Font.Name:='Courier';
//Данный шрифт обеспечивает одинаковую ширину всех символов,
//что способствует выравниванию столбцов в компоненте Tmemo.
end;
//-----------------------------------------------
procedure TForm1.FormShow(Sender: TObject);
var
  x1, y1, x2, y2: integer;
begin
   Button1.SetFocus; //Если компонента находится в фокусе ввода, то
//ее можно выбрать не только щелчком мыши, но и нажатием клавиши ENTER
//Найдем координаты угловых вершин компоненты PaintBox
  x1:=PaintBox1.Left-2;
  x2:=PaintBox1.Left+PaintBox1.Width+2;
```

```
  y1:=PaintBox1.Top-2;
  y2:=PaintBox1.Top+PaintBox1.Height+2;
//И соединим их, очерчивая область, где можно щелкать для размещения вершин графа
  Canvas.PolyLine ([Point(x1, y1), Point(x1,y2), Point(x2,y2), Point(x2,y1)]);
  With PaintBox1.Canvas do
  begin
    brush.color:=clWhite;
    rectangle (-1,-1, width, Height);
  end;
  Application.ProcessMessages;
end;
//-----------------------------------------------
procedure TForm1.PaintBox1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var
  i: integer;
begin
  Case EnterTime of
  True:    //в период ввода данных, более точно, при рисовании графа
  BEGIN
    inc (ClickCounter);
    StatusBar1.SimpleText:=Format ('ClickCounter=%4d, x=%4d, y=%4d',
    [ClickCounter,x,y]);
    VertexPosition[ClickCounter].X:=x;
    VertexPosition[ClickCounter].Y:=y;
  if ClickCounter=n then
  begin
    MessageDlg ('Ввод завершен. ' +
    'Для рисования нажмите кнопку ИЗОБРАЗИТЬ ГРАФ.'+
    ' Если изображение неудачное, мышью перетащите вершины в новые позиции.',
     mtInformation, [mbOK], 0);
    StatusBar1.SimpleText:='Для изображения графа нажмите кнопку';
    EnterTime:=false; //признак завершения ввода данных
    Button2.Enabled:=true;
    Button2.SetFocus;
    //теперь в фокусе ввода находится кнопка для изображения графа
  end;
  END;
  False:
  //уже после завершения рисования, в период перетаскивания
  //вершин мышью для улучшения вида
  BEGIN
```

```
    for i:=0 to n-1 do
    if sqr(x-VertexPosition[i].x)+sqr(y-VertexPosition[i].y)< sqr(r) then
     begin  Pressed:=true; i0:=i; end;
      //если нажатие мыши произошло на вершине i, либо очень близко от нее,
      //то номер этой вершины запоминается в i0,
      //чтобы при перемещении мыши скорректировать месторасположение вершины i0
      //и подправить изображения ее связей со смежными вершинами
  END;
 end; //case
end;//TForm1.PaintBox1MouseDown
//-----------------------------------------------
procedure TForm1.PaintBox1MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
   Pressed:=false;
{  В следующей процедуре - обработчике перемещения мыши
   действия по запоминанию новых координат выполняются,
   если только это перемещение   происходит при нажатой кнопке мыши,
   т.е. если переменная Pressed имеет значение TRUE. }
end;
//-----------------------------------------------
procedure TForm1.PaintBox1MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
  if Pressed then //если перемещение мыши происходит при нажатой кнопки мыши
  begin
    VertexPosition[i0].x:=x; VertexPosition[i0].y:=y; redraw;
  end;
  //Выбранную вершину i0 разместим в том месте, где мы отжали кнопку мыши и
  //вызовом ReDraw перерисуем граф
end;
//-----------------------------------------------
procedure TForm1.Button3Click(Sender: TObject);
//Упорядочить ребра (более точно, элементы массива E) по возрастанию
var
  i: integer;
  t: boolean;
  temp: TEdgeWeightColor;
  s: string;
begin
  Button3.Enabled:=false;
  Button4.Enabled:=true;
```

```
  Button4.SetFocus; Application.ProcessMessages;
  t:=true; //чтобы следующий цикл выполнился хотя бы один раз
  While t do
  begin
    t:=false; //узелок на память не завязан, потому что нечего пока запоминать
    for i:=1 to NE-1 do
    //выполним проход по ребрам и, если весь i-го ребра больше веса (i+1)-го ребра:
    //   E[i].w>E[i+1].w,
    //то 1) <<завяжем узелок>> на память, 2) переставим эти два элемента массива E
    if E[i].w>E[i+1].w then
    begin
       temp:=E[i]; E[i]:=E[i+1]; E[i+1]:=temp; t:=true;
    end;
  end; //while t
  //Ребра отсортированы по неубыванию весов
  //Выведем их в Memo1.
  Memo1.Lines.add ('Выведем ребра в порядке неубывания весов:');
  for i:=1 to NE do
    begin
      s:=Format ('%2d,%2d) %3d', [E[i].x, E[i].y, E[i].w]);
      Memo1.Lines.Add(s); //Вывели ребро и его вес
    end;
  ShowResult:=true;
  StatusBar1.SimpleText:='Ребра упорядочены по возрастанию весов.';
end;
//-----------------------------------------------
Procedure Taint (i, a,b: integer; Color: Tcolor);
//закрасить ребро (a,b) в цвет Color
var
  x1, y1, x2, y2, j: integer;
  s: string;
Begin
  x1:=VertexPosition[a].x; y1:=VertexPosition[a].y;
  s:=s+Format ('(%d,%d)',[x1,y1]);
  x2:=VertexPosition[b].x; y2:=VertexPosition[b].y;
  s:=s+ Format (' -- (%d,%d)',[x2,y2]);
  s:=s+ Format (' -- (%d,%d)',[x2,y2]); Form1.Memo1.Lines.add(s);
  E[i].C:= Color;
  with Form1.PaintBox1.Canvas do
       BEGIN
         pen.Color:=Color;
         pen.width:=2;
```

```
          For j:=1 to 10 do begin
            Pen.Style:=psClear; MoveTo(x1,y1);   LineTo (x2,y2);
            sleep (10);
            Pen.Style:=psSolid; MoveTo(x1,y1);   LineTo (x2,y2);
            sleep (10);
          end;
        font.Color:=clBlue;
        TextOut((x1+x2) div 2+r-4, (y1+y2) div 2-4, intToStr(E[i].w));
      END; //with PaintBox1.Canvas
 end;
//-----------------------------------------------
procedure TForm1.Button4Click(Sender: TObject);
Label Step1, Step2, Step3;
var
  i, j, a, b, x1, y1, x2, y2: integer;
  s: string;
  bc: integer; //для перебора букетов
  t: boolean;
  bc_a, bc_b: integer;
begin
  case Copy (Button4.Caption, 1, 5)='Шаг 1' of
  TRUE:
  begin
    Button4.Caption:='Шаг 2';
    StatusBar1.Panels[0].Text:='Шаг 1. Выбор первого ребра мин. ост. дерева.';
Step1:  Memo1.Text:=
    'Шаг 1. Выбрать ребро с наименьшим весом, не являющееся петлей.'+
    ' Окрасить это ребро в голубой цвет и сформировать букет,'+
    ' включив в него концевые вершины окрашенного ребра.';
     i:=0;
     repeat
       inc (i);
       with E[i] do
     begin
       Brush.Style:=bsSolid;
       a:=E[i].x; b:=E[i].y; if a=b then continue;
       //если выбранное ребро - петля, то поиск продолжается
       //Ребро найдено. Окрасим ее
       E[i].C:=clBlue; //это для использования в алгоритме.
       BlossemCount:=1;
       Blossem [BlossemCount]:=[a,b];
       //Сформировали букет из вершин -- концов выбранного ребра.
```

```
      //Все дальнейшие действия процедуры -- окраска ребра в графе
      // и вывод информации в Memo1
      Taint (i, a, b, clBlue);

      break; //прерываем цикл.
    end; //with E[i] do
     until i=NE;
     if i=NE then ShowMessage
       ('Не найдено ребро, не являющееся меткой. Это ошибка. Проверьте решение.');
   END; //true
   FALSE:
Step2:
    Begin
     Memo1.Lines.Add('Шаг 2.');
     StatusBar1.Panels[1].Text:='Шаг 2.';
     inc (SubStep2);
     Button4.enabled:=false;
       Button4.Caption:='Выполнение шага 2. Подождите.';
    //Выбрать любое неокрашенное ребро, которое не является петлей.
    //Если в графе такого ребра не найдется, закончить процедуру:
    //исходный граф не содержит остовного дерева.
    s:='';
    i:=0;
    While i<=NE do
    begin
      inc (i);
      //Если выбранное ребро - петля или окрашено, то поиск продолжается.
      a:=E[i].x; b:=E[i].y;
      s:= s+Format ('NE=%d i=%d a=%d b=%d w=%d C=%d черный=%D',
       [NE, i,a,b,E[i].w,E[i].C, clBlack]);
      if (a<>b) and (E[i].C=clBlack) then
      begin
        s:=s+'подходит';
        break;
      end;
      //Найдено неокрашенное ребро с разными концами (не петля)
     end;
    if i=NE+1 then begin
    ShowMessage ('Поскольку не найдено неокрашенное ребро, отличное от петли, '+
    ' то данный граф не обладает остовным графом. Завершите алгоритм.');
    Exit;
    end; //if i=NE
```

```
{После указанного выбора возможны четыре различных случая.
a) Обе концевые вершины a, b выбранного ребра принадлежат
    одному и тому же букету.
b) Одна из концевых вершин выбранного ребра принадлежит некоторому букету, а
    другая концевая вершина не принадлежит ни одному из уже
    сформированных букетов.
c) Ни одна из концевых вершин не принадлежит ни одному из сформированных
    букетов.
d) Концевые вершины выбранного ребра принадлежат различным букетам.}
    bc_a:=0; bc_b:=0;
    for bc:=1 to BlossemCount do
    if a in Blossem [bc] then begin bc_a:=bc; break end;
//Если вершина a принадлежит какому-либо букету, то номер букета равен bc_a
//Если bc_a=0, то вершина a не принадлежит ни одному из букетов
    for bc:=1 to BlossemCount do
    if b in Blossem [bc] then begin bc_b:=bc; break end;
//Если вершина b принадлежит какому-либо букету, то номер букета равен bc_b
//Если bc_b=0, то вершина b не принадлежит ни одному из букетов
//---------------------------------------------------------------------------
   if (bc_a=bc_b) and (bc_a<>0) then //это случай (a)
// Если имеет место случай (a), окрасить выбранное ребро в оранжевый цвет
//(оно не включается в дерево) и вернуться к началу шага 2.
   begin
     Taint (i, a, b, clRed);
     goto Step2;
   end; //случай (a)
//---------------------------------------------------------------------------
     if (bc_a<>0) and (bc_b=0) then
{это случай (b)
 Если имеет место случай (b), окрасить выбранное ребро в голубой цвет
(оно включается в дерево) и включить его концевую вершину, не принадлежавшую
  ни одному букету, в тот же букет, которому принадлежит другая концевая
 вершина рассматриваемого ребра.}
   begin
     Taint (i, a, b, clBlue);
     s:=Format ('%2d,%2d) %3d', [E[i].x, E[i].y, E[i].w]);  Memo1.Lines.Add(s);
     include (Tree, i);
//ребро с новым (после упорядочения) номером i включается в дерево
     include (Blossem [bc_a], b);
    end; //случай (b)
    if (bc_a=0) and (bc_b<>0) then //другой вариант случая (b)
   begin
```

```
     Taint (i, a, b, clBlue);
     s:=Format ('%2d,%2d) %3d', [E[i].x, E[i].y, E[i].w]);  Memo1.Lines.Add(s);
     include (Tree, i);
//ребро с новым (после упорядочения) номером i включается в дерево
     include (Blossem [bc_b], a);
   end; //случай (b)
//---------------------------------------------------------------------------
   if (bc_a=0) and (bc_b=0) then //это случай (c)
//Если имеет место случай (c), окрасить рассматриваемое ребро в голубой цвет
// и сформировать новый букет из его концевых вершин.
   begin
     Taint (i, a, b, clBlue);
     s:=Format ('%2d,%2d) %3d', [E[i].x, E[i].y, E[i].w]);  Memo1.Lines.Add(s);
     inc (BlossemCount);
     Blossem [BlossemCount]:=[a,b];
   end; //случай (c)
//---------------------------------------------------------------------------
   if (bc_a<>0) and (bc_b<>0) and (bc_a<>bc_b) then //это случай (d)
   begin
//если имеет место случай (d), окрасить рассматриваемое ребро в голубой цвет,
// а оба букета, которым принадлежат его концевые вершины, слить в один новый букет.
//По завершении шага 2 перейти к шагу 3.
     Taint (i, a, b, clBlue);
     s:=Format ('%2d,%2d) %3d', [E[i].x, E[i].y, E[i].w]);  Memo1.Lines.Add(s);
     Blossem[bc_a]:=Blossem[bc_a] + Blossem[bc_b];
     Blossem[bc_b]:=[];
   //  Goto Step2;
   end;
Step3: // Если все вершины графа вошли в один букет, то завершить алгоритм
//в противном случае вернуться к шагу 2.
    Memo1.Lines.Add('Шаг 3.');
    StatusBar1.Panels[1].Text:='Шаг 3.';
    t:=false;
    For i:=1 to BlossemCount do
    if Blossem[i]=[1..n] then
    begin
      t:=true;
//нашли букет, включающий все вершины
//это набор ребер с цветом clBlue, надо вывести их вершины
      s:='';
      for j:=1 to NE do
        if E[j].C=clBlue then s:=s+Format ('(%d,%d)', [E[j].x, E[j].y]);
```

```
      ShowMessage ('Найдено минимальное остовное дерево: '+ s);
      Memo1.Lines.Add(s);
     end;
Sleep (100);
     if t=false then goto Step2;
   END; //FALSE
  end;  //case
  StatusBar1.SimpleText:= 'Минимальное остовное дерево: '+s;
end;
end.
```

§ 2.2. Максимальный поток в сети

Литература: [11, с. 404–418], [12, с. 335–345]. Под транспортной сетью N понимают связный ориентированный граф, в котором имеется лишь одна вершина s типа «источник» и лишь одна вершина t типа «сток», а каждой дуге $e = (i, j)$ сопоставлено вещественное число $c(e) = c(i, j) \geqslant 0$ («пропускная способность»). Потоком f в транспортной сети N называется заданная на множестве дуг функция, сопоставляющая каждой дуге $e = (i, j)$ вещественное $f(e) = f(i, j)$ так, что удовлетворяются условия:

- $f(e) \leqslant c(e)$ (ограничение по пропускной способности);
- для всех $i \neq s, t$: $\sum\limits_j f(i, j) = \sum\limits_j f(j, i)$ (условие сохранения).

Величиной потока f называется $val(f) = \sum\limits_j f(s, j)$. Поток наибольшей величины называется максимальным. Пусть в сети N задан некоторый поток f. Если дуга e_i пути

$$P: \quad s, ...e_i, ..., t$$

ориентирована от s к t, то обозначим $\varepsilon(e_i) = c(e_i) - f(e_i)$; если дуга e_i ориентирована от t к s, то $\varepsilon(e_i) = f(e_i)$. Наименьшее из $\varepsilon(e_i)$ обозначим $\varepsilon(P)$. Если $\varepsilon(P) > 0$, то путь P назовем f-увеличивающим.

Следующий алгоритм вычисления максимального потока принадлежит Форду и Фалкерсону (*модификация Эдмондса и Карпа*). Его сложность для сети с n вершинами и множеством дуг E равна $O(n \cdot |E|^2)$. Алгоритмы сложности $O(n^3)$ см., например, в [1].

Будем различать три состояния вершины: «не помечено», «помечено и не просмотрено», «помечено и просмотрено». Метка вершины v состоит из трех частей вида $pm(v), p(v), d(v)$, где $pm(v) =$«+» или «-»; $p(v)$ — вершина, предшествующая v в (известном из контекста) f-увеличивающем пути из s

до v; $d(v)$ — величина, на которую можно увеличить имеющийся поток из s до v:

1°. $+p(v)$: поток допускает увеличение на $d(v)$ вдоль дуги $p(v) \to v$;

2°. $-p(v)$: поток допускает уменьшение на $d(v)$ вдоль дуги $v \leftarrow p(v)$.

В начале алгоритма помечивания величина потока f равна нулю.

Фаза1: Помечивание

Состояние каждой вершины установим в значение «не помечено». Вершине s присваивается метка $(+0, \varnothing)$, после чего состояние источника — «помечено и не просмотрено».

Итерация помечивания

Из помеченных и просмотренных вершин x выберем *помеченную ранее*. Для каждой смежной с x непомеченной вершины y выполнить:

- если существует дуга (x, y) и $f(x, y) < c(x, y)$, то вершине y присвоить метку $(+x, \quad min\{d(x), \quad c(x, y) - f(x, y)\})$ и, в случае $y = t$, перейти к Фазе 2;
- если существует дуга (y, x) и $f(y, x) > 0$, то вершине y присвоить метку $(-x,\ min\{d(x),\ f(y, x)\})$ и при $y = t$ перейти к Фазе 2.

После помечивания вершины y ее состояние — «помечено и не просмотрено»; вершина x после обработки всех ее смежных вершин помечена и просмотрена.

Если удалось пометить новую вершину, то следует повторить итерацию помечивания, в противном случае полученный поток f является максимальным и алгоритм завершается.

Фаза 2: увеличение величины потока на значение $d(t)$

$v := t$;

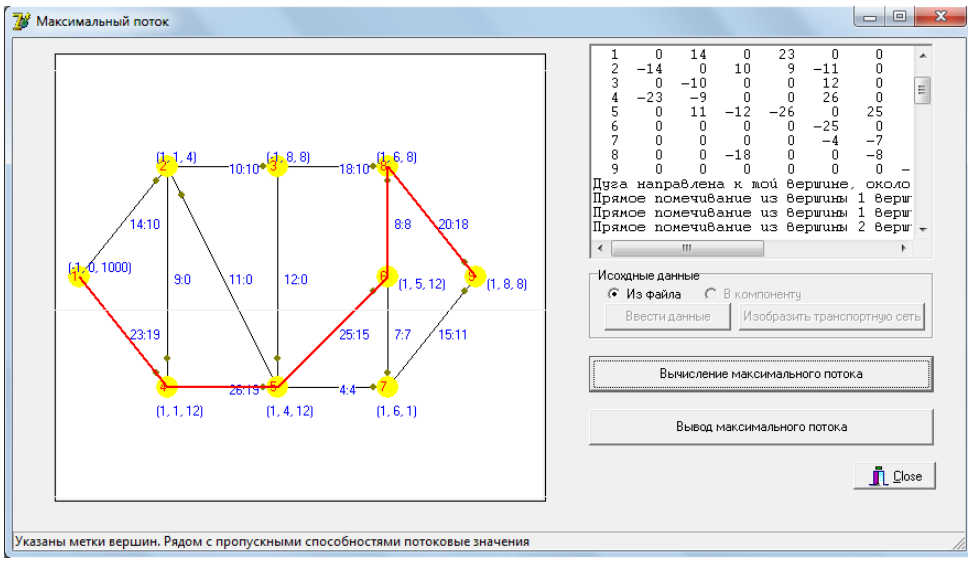
Повторять действия:

- $u := p(v)$;
- если $pm(v) = +1$, то $f(x, y) := f(x, y) + d(t)$;
- если $pm(v) = -1$, то $f(y, x) := f(y, x) - d(t)$;
- $v := u$

пока не выполнено условие $v = s$.

Стереть все метки (сохранив при этом вычисленные потоковые значения) и вернуться к Фазе 1.

В программе предусмотрены действия по выбору способа ввода, визуализация сети, пошаговый вывод с показом f-увеличивающего пути. На рисунке заметны крупные точки; точка на дуге символизирует стрелку, направленную к той из двух концевых вершин, которая ближе к данной точке.




```
unit Unit1;
interface
uses  Math,
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls, ExtCtrls, Buttons;
type
  TForm1 = class(TForm)
    Memo1: TMemo;
    StatusBar1: TStatusBar;
    OpenDialog1: TOpenDialog;
    PaintBox1: TPaintBox;
    GroupBox1: TGroupBox;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    BitBtn1: TBitBtn;
```

```
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure PaintBox1MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure PaintBox1MouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure PaintBox1MouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure FormKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
  private
    procedure reDraw;
  end;
const
   n=9;
   None =0; //(not marked)
   MN=1; // (marked but not viewed);
   MV=2; // (marked and viewed);
   t=n;
   s=1;
var
  Pressed: boolean=false;
  maxFlow: boolean=false;
  i0: integer = 0;
  Status: array [1..n] of byte;
  VertexMark: array [1..n] of record Direct, From, Delta: integer end;
const
  dx=100; dy=100; r=10; x0=20; y0=200;

   MyInfinity=1000;
   //вместо "бесконечности", в нашем примере сумма стоимостей всех дуг < 1000.
   //В общем случае можно взять любое значение, превышающее сумму стоимостей всех дуг
var
  Form1: TForm1;
  ClickCounter: integer=-1;
  EndIteration: Boolean=false;
  ArrayPath: array [1..n] of integer; // для прорисовки f-увеличивающего пути
  CountArrayPath: integer; //количество пунктов вдоль f-увеличивающего пути
```

```
  C, f: array[1..n, 1..n] of integer;  //Данные к построению транспортной сетии
  VertexPosition: array[1..n] of TPoint=(
  (x:x0; y:y0),              (x:dx; y:y0-dy),
  (x:+2*dx;y:y0-dy),         (x:+dx;y:y0+dy),
  (x:+2*dx;y:y0+dy),         (x:+3*dx;y:y0),
  (x:+3*dx;y:y0+dy),         (x:+3*dx;y:y0-dy),
  (x:+4*dx-x0;y:y0)
   );
   VP: array[1..n] of TPoint=(
  (x:x0-10; y:y0-15),         (x:dx-10; y:y0-dy-15),
  (x:+2*dx-10;y:y0-dy-15),    (x:+dx-10;y:y0+dy+15),
  (x:+2*dx-10;y:y0+dy+15),    (x:+3*dx+10;y:y0+0),
  (x:+3*dx-10;y:y0+dy+15),    (x:+3*dx-10;y:y0-dy-15),
  (x:+4*dx-x0+10;y:y0+0)
       );
  EnterTime: boolean = true;
  ShowResult: boolean = false;
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
var
  i, j, x1, y1, x2, y2: integer; f: TextFile;
  s: string;
begin
  if RadioButton1.Checked then
  begin
    OpenDialog1.InitialDir:=GetCurrentDir;
    if OpenDialog1.Execute then
      assignFile (f, OpenDialog1.FileName);
    reset (f);
    Memo1.Lines.Add(Format ('n= %d', [n]));
    s:='   ';  for i:=1 to n do s:=s+Format ('%5d',[i]);
    Memo1.Lines.Add(s);
    //Для удобства сверки данных удобно вверху изображать номера вершин
    Memo1.Lines.Add('');
    //Пропустим пару строк
    Memo1.Lines.Add('');
    for i:=1 to n do
    begin
      s:=Format ('%3d', [i]);
      //В начале  каждой строки удобно выводить номер соответствующей вершины
      for j:=1 to n do begin
```

```
                        read (f, c[i,j]);  s:=s+Format ('%5d',[c[i,j]])
                       end;
       Memo1.Lines.Add(s);
    end;
    end; //if RadioButton1.checked
  Button1.Enabled:=false;
  Button2.Enabled:=true;
//Если компонента находится в фокусе ввода, то
//ее можно выбрать не только щелчком мыши, но и нажатием клавиши ENTER
  x1:=PaintBox1.Left-2; x2:=PaintBox1.Left+PaintBox1.Width+2;
  y1:=PaintBox1.Top-2; Y2:=PaintBox1.Top+PaintBox1.Height+2;
  Canvas.PolyLine ([Point(x1, y1), Point(x1,y2), Point(x2,y2),
                                          Point(x2,y1), Point(x1,y1)]);
  Application.ProcessMessages;
end;
//-----------------------------------------------------------
procedure TForm1.Button2Click(Sender: TObject);
begin
  redraw;
  Memo1.Lines.Add('Дуга направлена к той вершине, около которой на дуге размещен знак');
end;
//-----------------------------------------------------------
procedure TForm1.reDraw;
var
  i, j: integer; x1,y1,x2,y2,xi, yi: integer;
begin
Button2.Enabled:=false;
Button3.Enabled:=true;
Button3.SetFocus;
//  StatusBar1.SimpleText:='Рядом с каждой дугой изображен поток';
with PaintBox1.canvas do
begin
  brush.Style:=bsSolid;
  brush.Color:=clWhite; pen.Color:=clWhite;
  rectangle (-1,-1, width, height);
  pen.Color:=clBlack;
  for i:=1 to n do
  begin
     with VertexPosition[i] do
     begin
       Brush.Style:=bsSolid;
       Pen.Width:=1;
```

```
      MoveTo(x,y);//Курсор разместили в позиции i-й вершины
      for j:=1 to n do
      //переберем все вершины j, опуская те, которые не связаны с i
      if (c[i,j] >0) then
      begin
        x1:=VertexPosition[j].x; y1:=VertexPosition[j].y;
        MoveTo (x,y);
        LineTo(x1,y1);
        font.Color:=clBlue;
        if not ShowResult then
          TextOut((x+x1) div 2+r-4, (y+y1) div 2-4, intToStr(c[i,j]))
        else
          TextOut((x+x1) div 2+r-4, (y+y1) div 2-4, intToStr(c[i,j])+ ':'+intToStr (f[i,j]));
        x2:=round (x+0.87*(x1-x)); y2:=round (y+0.87*(y1-y));
        pen.Color:=clOlive;    Brush.Color:=clOlive;
        ellipse (x2-3,y2-3, x2+3,y2+3);
        pen.Color:=clBlack;    Brush.Color:=clWhite;
      end;
    end; //with VertexPosition[i] do
  end; //for i;
  pen.Color:=clBlack;  Brush.Color:=clWhite;
  for i:=1 to n do
  begin
    xi:=VertexPosition[i].x; yi:=VertexPosition[i].y;
    Brush.Color:=clYellow;
    Pen.Color:=clYellow;
    ellipse(xi-r, yi-r, xi+r, yi+r);
    Font.Color:=clRed; //brush.Color:=Clred;
    TextOut(xi-r div 2 -2, yi-r div 2 -2, intToStr(i));
    if ShowResult then
    if maxFlow=false then
    begin
      Font.Color:=clBlue;
      Brush.Style:=bsClear;
      TextOut (VP[i].x, VP[i].Y,
      Format ('(%d, %d, %d)',
           [VertexMark[i].Direct, VertexMark[i].From, VertexMark[i].Delta]));
      Font.Color:=clBlack;
    end;
  end; //i
end; //with PaintBox1.Cavas
StatusBar1.SimpleText:=
```

```
        'Маркеры на дуге означают направленность дуги к ближайшей вершине'
end;
//-------------------------------------------------------
procedure TForm1.FormCreate(Sender: TObject);
begin
  Memo1.Font.Name:='Courier';   //Данный шрифт обеспечивает одинаковую
//ширину всех символов. Это полезно для выравнивания столбцов в MEMO
end;
//-------------------------------------------------------
procedure TForm1.FormShow(Sender: TObject);
var
  x1, y1, x2, y2: integer;
begin
   Button1.SetFocus; //Если компонента находится в фокусе ввода, то
//ее можно выбрать не только щелчком мыши, но и нажатием клавиши ENTER
//Найдем координаты угловых вершин компоненты PaintBox
  x1:=PaintBox1.Left-2; x2:=PaintBox1.Left+PaintBox1.Width+2;
  y1:=PaintBox1.Top-2; Y2:=PaintBox1.Top+PaintBox1.Height+2;
//И соединим их, очерчивая область, где можно щелкать для размещения вершин графа
  Canvas.PolyLine ([Point(x1, y1), Point(x1,y2), Point(x2,y2), Point(x2,y1)]);
  With PaintBox1.Canvas do
  begin
    brush.color:=clWhite;
    rectangle (-1,-1, width, Height);
  end;
  Application.ProcessMessages;
end;
//-------------------------------------------------------
procedure TForm1.PaintBox1MouseDown(Sender: TObject; Button: TMouseButton;
                                                  Shift: TShiftState; X, Y: Integer);
var
  i: integer;
begin
  if EnterTime then
 //в период ввода данных, более точно, при рисовании графа
  BEGIN
  inc (ClickCounter);
  StatusBar1.SimpleText:=Format ('ClickCounter=%4d, x=%4d, y=%4d',
    [ClickCounter,x,y]);
  VertexPosition[ClickCounter].X:=x;
  VertexPosition[ClickCounter].Y:=y;
  if ClickCounter=n-1 then begin
```

```
      MessageDlg ('Ввод завершен, благодарю Вас. ' +
      'Для рисования нажмите кнопку ИЗОБРАЗИТЬ ГРАФ.'+
      ' Если изображение неудачное, мышью перетащите вершины в новые позиции.',
       mtInformation, [mbOK], 0);
      StatusBar1.SimpleText:='Для изображения графа нажмите кнопку';
      EnterTime:=false; //признак завершения ввода данных
      Button2.Enabled:=true;
      Button2.SetFocus;
//теперь в фокусе ввода находится клавиша для изображения графа
                           end;
  END;
//уже после завершения рисования, в период перетаскивания вершин мышью
//для улучшения вида

end;//TForm1.PaintBox1MouseDown
//-------------------------------------------------------
procedure TForm1.PaintBox1MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
   Pressed:=false;
//В следующей процедуре - обработчике перемещения мыши
//действия по запоминанию новых координат выполняются, если только это перемещение
//происходит при нажатой кнопке мыши, т.е. если переменная Pressed имеет значение TRUE.
end;
//-------------------------------------------------------
procedure TForm1.PaintBox1MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
 if Pressed then //если перемещение мыши происходит при нажатой кнопки мыши
  begin
    VertexPosition[i0].x:=x; VertexPosition[i0].y:=y; redraw;
  end;
//Выбранную вершину i0 разместим в том месте, где мы отжали кнопку мыши и
//вызовом ReDraw перерисуем граф
end;
//-------------------------------------------------------
procedure TForm1.Button3Click(Sender: TObject);
Label Step10, Step11, Step12, Step2;
var
  Minimum: integer;
  deltaT: integer;
//после достижения стока в процессе помечивания
```

```
//поток можно увеличить на величину deltaT
  YesMark: boolean;
//получает True, если в текущей итерации удалось пометить хотя бы одну новую вершину
  i, j, x, y, x1, y1: integer;
  VisualPath: String;
begin
  Button3.Enabled:=false;
  Button4.Enabled:=true;
  Button4.SetFocus;
  StatusBar1.SimpleText:=
  'Указаны метки вершин. Рядом с пропускными способностями потоковые значения';
//Вначале все вершины не имеют пометок
Step10:
       with VertexMark[1]  do begin Direct:=-1; From:=0; Delta:=MyInfinity; end;
       for i:=1 to n do for j:=1 to n do f[i,j]:=0;
       //вначале поток положим равным нулю
Step11:
       for i:=1 to n do Status[i]:=None;
       Status[1]:=MN;                    //Вершина 1 помечена, но не просмотрена
//Хотя в первый раз нам уже известна помеченная и не просмотренная вершина
//(назовем ее x), в дальнейшем надо ее искать.
Step12:     if EndIteration then exit; Application.ProcessMessages;
             YesMark:=False;  x:=1;
             Repeat
               if Status[x]=MN then break;
               inc (x);
             until x=n+1;
if x=n+1 then
begin
  ShowMessage
  ('Для вывода максимального потока нажмите на кнопку "Вывод максимального потока"');
  exit;
end;
//---Алгоритм нахождения максимального потока может завершиться только здесь----
//    В данном контексте x - найденная помеченная и непросмотренная вершина.
      for y:=1 to n do begin
        if Status[y]<> None then continue;
        //Если смежная x вершина y уже помечена, ее не помечиваем
        if c[x,y]>0 then //прямое помечивание
         if c[x,y]>f[x,y] then
           begin
             YesMark:=true; //запомним, что удалось пометить хоть одну вершину
```

```
            Minimum:= min (VertexMark[x].Delta, c[x,y]-f[x,y]);
            with VertexMark[y] do begin Direct:=+1; From:=x; Delta:=Minimum end;
            //теперь вершина y имеет статус "Помечена, но не просмотрена"
            Status[y]:=MN;
            Memo1.Lines.Add(Format
            ('Прямое помечивание из вершины %d вершины %d. Метка: (%d, %d, %d).',
            [x,y, VertexMark[y].Direct,  VertexMark[y].From, VertexMark[y].Delta]));
            if y=t then goto Step2;
            //Это мы достигли стока, т.е. получили увеличивающий путь
          end;
        if c[x,y]<0 then //обратное помечивание
         if f[y,x]>0 then
          begin
            YesMark:=true; //запомним, что удалось пометить хоть одну вершину
            Minimum:= min (VertexMark[x].Delta, f[y,x]);
            with VertexMark[y] do begin Direct:=-1; From:=x; Delta:=Minimum end;
            //теперь вершина y имеет статус "Помечена, но не просмотрена"
            Status[y]:=MN;
            Memo1.Lines.Add(Format
            ('Прямое помечивание из вершины %d вершины %d. Метка: (%d, %d, %d).',
            [x,y, VertexMark[y].Direct,  VertexMark[y].From, VertexMark[y].Delta]));
            if y=t then goto Step2;
            //Это мы достигли стока, т.е. получили увеличивающий путь
          end;
       end; //for y
//После того, как из помеченной, но не просмотренной вершины x выполнили
//перебор всех смежных вершин, вершина x не только помечена, но уже и просмотрена.
       Status[x]:=MV;
       Goto Step12;
STEP2: //Сюда переходим лишь после достижения стока. Тогда надобно увеличить поток
//по некоторому пути на значение deltaT=VertexMark[t].Delta
       ShowResult:=true;
       deltaT:=VertexMark[t].Delta;
       i:=t;
       VisualPath:=Format ('%d', [i]); ArrayPath[1]:=t; CountArrayPath:=1;
       repeat
          if EndIteration then exit; Application.ProcessMessages;
          j:=VertexMark[i].From;
                          VisualPath:=Format ('%d--', [j])+VisualPath;

          if VertexMark[i].Direct = +1 then f[j,i]:=f[j,i]+deltaT;
          if VertexMark[i].Direct = -1 then f[i,j]:=f[i,j]-deltaT;
```

```
          i:=j;            inc(CountArrayPath);
                           ArrayPath[CountArrayPath]:=i;
        until i=s;
        Memo1.Lines.Add ('На сколько увеличивается величина потока:'+
        intToStr (deltaT));
        Memo1.Lines.Add('Очередной f-увеличивающий путь:'+VisualPath);
        ShowResult:=true;
        ReDraw;
//Выведем f-увеличивающий путь красным цветом
      with PaintBox1.Canvas do
      begin
        Pen.Width:=2;
        j:=ArrayPath[1];
        x1:=VertexPosition[j].x;  y1:=VertexPosition[j].y;
        MoveTo(x1,y1);
        pen.Color:=clRed;
        for i:=2 to CountArrayPath do
        begin
            j:=ArrayPath[i];
            x1:= VertexPosition[j].x;  y1:=VertexPosition[j].y;
            LineTo(x1,y1);
            Application.ProcessMessages;
            sleep (200);
         end;
      end;
      ShowMessagePos('f-увеличивающий путь изображен красным цветом.'+#13#10+
                     ' Нажмите для перехода к следующей итерации', 100, 100);
      //завершили изображение пути.
      StatusBar1.SimpleText:=
      'Указаны метки вершин. Рядом с пропускными способностями потоковые значения';
      goto Step11;
end;
//-----------------------------------------------------------
procedure TForm1.Button4Click(Sender: TObject);
var
  i, j: integer;
  s: string;
begin
  maxFlow:=true;
  Button3.Enabled:=false; Button4.Enabled:=false;
  Memo1.Lines.Add('');
  Memo1.Lines.Add
```

```
   ('ЗНАЧЕНИЕ МАКСИМАЛЬНОГО ПОТОКА ПО ДУГЕ УКАЗАНО ПОСЛЕ ПРОПУСКНОЙ СПОСОБНОСТИ');
  s:='   ';  for i:=1 to n do s:=s+Format ('%6d',[i]);
  Memo1.Lines.Add(s);
//Для удобства сверки данных удобно вверху изображать номера вершин
   Memo1.Lines.Add('');
   for i:=1 to n do
   begin
      s:=Format ('%2d) ', [i]);
//В начале  каждой строки удобно выводить номер соответствующей вершины
      for j:=1 to n do begin
                          s:=s+Format ('%3d:%2d',[c[i,j], f[i,j]])
                         end;
       Memo1.Lines.Add(s);
  end;
  Redraw;
  StatusBar1.SimpleText:='Вычислен максимальный поток';
end;
//---------------------------------------------------------
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Key=VK_ESCAPE then begin EndIteration:=true end;
end;
end.
```

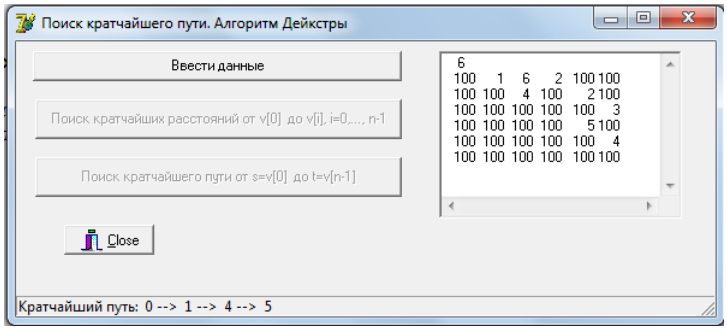
§ 2.3. Поиск кратчайших путей

Литература: [11, с. 361–366]. Алгоритм, построенный нидерландским математиком Э. Дейкстрой в 1959 году (Эдскар Вибе Дейкстра, 1930–2002 гг.), находит кратчайшее расстояние от одной из вершин графа до всех остальных. Хотя обычно алгоритм применяется для графов без ребер отрицательного веса, в действительности достаточно более слабого условия — выполнения неравенства треугольника. См. также алгоритм Фллойда [11, с. 366–368].

Сложность алгоритма Дейкстры равна $O(n^2)$, где n — число вершин графа. Интересно, что задача вычисления самого длинного пути в графе является NP-полной, даже если веса всех ребер равны единице [4, с. 268–269]. Открытой остается задача о существовании алгоритма нахождения кратчайшего пути между двумя заданными вершинами графа за время $O(|E|)$, где E — множество ребер.

Алгоритм Дейкстры широко применяется в программировании и технологиях, см. [12, с. 316–335]; например, его использует протокол OSPF для

устранения кольцевых маршрутов.




```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, ComCtrls, ExtCtrls, Buttons;
type
  TForm1 = class(TForm)
    StatusBar1: TStatusBar;
    OpenDialog1: TOpenDialog;
    Button3: TButton;
    Button4: TButton;
    BitBtn1: TBitBtn;
    Button1: TButton;
    Memo1: TMemo;
    procedure Button1Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
  end;
const
   nMax=50;
   MyInfinity=100;
var
  Form1: TForm1;
  ClickCounter: integer=-1;
  i0: integer=-1;
  C: array[0..nMax, 0..nMax] of integer;
  n: integer;
  L, From: array[0..nMax] of integer;
  EnterTime: boolean = true;
```

```
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
var
  i, j: integer;
  f: TextFile;
begin
  OpenDialog1.InitialDir:=GetCurrentDir;
  if OpenDialog1.Execute then
      assignFile (f, OpenDialog1.FileName);
  Memo1.Lines.LoadFromFile (OpenDialog1.FileName);
  reset (f);
  read (f, n);
  for i:=0 to n-1 do
       for j:=0 to n-1 do read (f, c[i,j]);
  button3.Enabled:=true;
  button3.SetFocus;
end;
//------------------------------
procedure TForm1.Button3Click(Sender: TObject);
//Алгоритм Дейкстры
var
  MN: integer;
  i: integer;
  S: set of Byte;
  p: byte;
begin
  Button3.Enabled:=false;
  Button4.Enabled:=true;
  Button4.SetFocus;
  Application.ProcessMessages;
  L[0]:=0;   L[0]:=0;
  for i:=1 to n-1 do begin L[i]:= C[0,i]; From[i]:=0 end;
  S:=[0];
  p:=0;
  While S<>[0..n-1] do
BEGIN
  MN:=MyInfinity;
  for i:=0 to n-1 do
  if not (i in S)
   then if L[i]<MN then begin p:=i; MN:=L[i] end;
 include (S, p);
```

```
 for i:=0 to n-1 do begin
                     if i in S then continue;
                     if C[p,i]=MyInfinity then continue;
                     if L[p]+C[p,i]<L[i] then begin
                                               L[i]:= L[p]+C[p,i];
                                               From [i]:=p;
                                             end;
                   end; //for i
 END;
 StatusBar1.SimpleText:=
 'Рядом с каждой вершиной v[i] изображены две метки (L[i]:From[i])';
end;
//------------------------------
procedure TForm1.Button4Click(Sender: TObject);
var
  i: integer;
  s: string;
begin
  Button3.Enabled:=false;
  Button4.Enabled:=false;
  i:=n-1;
  s:=Format ('%2d', [i]);
  repeat
    i:=From[i];
    s:=Format ('%2d', [i])+' --> '+s;
  until i=0;
  StatusBar1.SimpleText:= 'Кратчайший путь: '+s;
end;
end.
```

§ 2.4. Разбиение множества

Литература: [4, с. 81–84, с. 117–118]. Заданное множество $A = \{a_1, \ldots, a_n\}$ натуральных элементов требуется разбить на два подмножества с равными суммами элементов.

Пусть сумма элементов четна:

$$\sum_{i=1}^{n} a_i = 2B,$$

где B целое положительное число; понятно, что в противном случае искомое разбиение не существут (как, скажем, и в том случае, когда один из элементов

множества превышает B).

Для решения данной NP-полной задачи используется следующий псевдополиномиальный алгоритм. Сначала выполняется построение логической таблицы T со строками $1, \ldots, n$ и столбцами $0, 1, \ldots, B$, где значение T_{ij} равно значению истинности утверждения: «В $\{a_1, a_2, \ldots, a_i\}$ найдется подмножество, сумма элементов которого равна j». Если каким-либо образом удалось заполнить таблицу, то значение T_{nB} соответствует ответу на вопрос о существовании требуемого разбиения. Покажем, как заполняется первая строка таблицы и как по значениям строки $i-1$ вычисляются элементы строки i.

1) В первой строке таблицы истинны значения T_{10} и T_{1,a_1} и только они: ведь одноэлементное множество $\{a_1\}$ обладает лишь несобственными подмножествами.

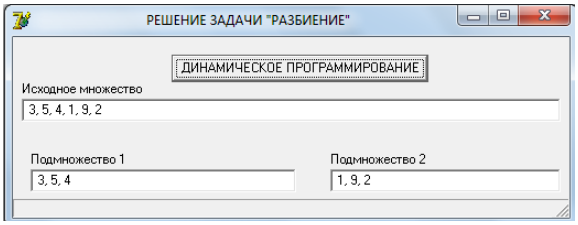
2) Пусть при некотором i, $i > 1$, вычислены значения $T_{i-1,0}, \ldots, T_{i-1,B}$. Тогда T_{ij} вычисляется по формуле:

$$T_{ij} = T_{i-1,j} \ \vee \ T_{i-1,j-a_i}$$

со следующим уточнением: если $j < a_i$, то $T_{ij} = T_{i-1,j}$.

Несмотря на кажущуюся простоту изложенного алгоритма, его нельзя отнести к разряду полиномиальных. В данной брошюре были бы неуместны ни анализ причин этого феномена (см., например, [4, с. 118]), ни даже определение термина *псевдополиномиальный алгоритм*. Отметим только, что для целей компьютерных вычислений псевдополиномиальные алгоритмы «достаточно эффективны».

Процесс генерации разбиения исходного множества на искомые подмножества (при $T_{nB} = true$) несложен и понятен из приведенного ниже листинга программы.




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

```
  TForm1 = class(TForm)
    Button1: TButton;
    OpenDialog1: TOpenDialog;
    StatusBar1: TStatusBar;
    LabeledEdit1: TLabeledEdit;
    LabeledEdit2: TLabeledEdit;
    LabeledEdit3: TLabeledEdit;
    procedure Button1Click(Sender: TObject);
  end;
const
  n1=100;
var
  Form1: TForm1;
  a: array [1..n1] of byte;
  T: array [1..n1, 0..100] of boolean;
  M: set of byte;
  M1, M2: string;
implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
var
  s: string;
  f: TextFile;
  i, j, k, B, B2, n: integer;
  TNB: string;
begin
  OpenDialog1.InitialDir:=GetCurrentDir;
  if not openDialog1.execute then exit;
   s:= openDialog1.FileName;
   assignFile (f, s);
   reset (f);
   i:=0; B2:=0;
   while not eof (f) do
   begin
     inc (i);
     read (f, a[i]);
     B2:=B2+a[i]
   end;
   CloseFile (f);
   n:=i; //количество элементов
```

```
   M1:='';
   For i:= 1 to n do
   M1:= M1+', '+intToStr (a[i]);  M1:= copy (M1, 2, 100);
   LabeledEdit3.Text:= M1;
   //Если сумма нечетная, то решение отсутствует
   if odd (B2) then begin
     statusBar1.simpleText:='Нет решения, так как сумма нечетная';
     Application.Terminate;
   end;
   B:= B2 div 2;
   //Если имеется элемент со значением, превышающим половину суммы,
   //задача также не имеет решения
   for i:=1 to n do
     if a[i]> B then begin
     statusBar1.simpleText:='Нет решения, так как элемент a['+intToSTr (i)+'] = '+
      intToStr (a[i]) +' имеет значение, превышающее половину суммы ('+
      intToSTr (B);
     Application.Terminate;
   end;
// Заполним массив сначала значениями FALSE
   for i:=1 to n do
   for j:=0 to B do T[i,j]:=false;
// Заполним элементы первой строки массива T
   T[1, 0]:= TRUE;  T [1, a[1]]:= TRUE;
// Если T[i-1,j] = TRUE, то T[i,j] = TRUE;
// Если T[i-1,j]=FALSE, то T[i,j] = TRUE тогда и только тогда, когда
// выполнено условие: T[i-1, j-a[i]] = TRUE

// Последовательно заполним все строки, начиная со второй строки, используя
// при этом лишь предыдущую строку (первая строка у нас уже заполнена)
   for i:=2 to n do
     for j:=0 to B do
     if not T[i-1,j] and (j-a[i]<0) then T[i,j]:=false
     else T[i,j]:= T[i-1,j] or T[i-1, j-a[i]];
// Таблица полностью заполнена. Остается исследовать значение последней
// ячейки - T[n, B];
   if T[n,B] then TNB:='TRUE' else TNB:='FALSE';

   if TNB = 'FALSE' then
   begin
     statusBar1.simpleText:='Нет решения, так как значение в последней ячейке '+
     'T['+intToSTr (n)+','+ intToStr (B)+'] = '+ TNB;
```

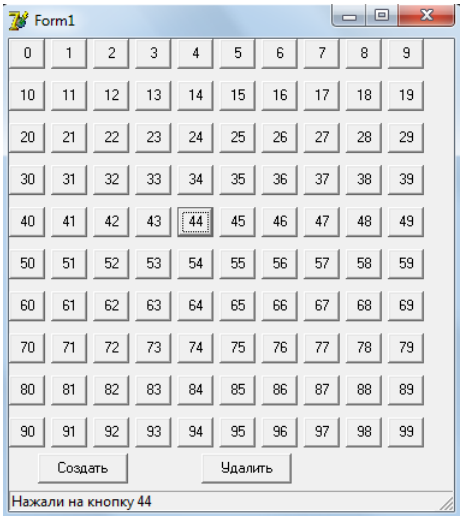
```
     Application.Terminate;
   end;
// При выполнении следующего фрагмента T[n, B] = TRUE и, следовательно,
// задача имеет решение. Необходимо его найти.
   M:=[];
   j:=B; i:=n;
 while true do
 begin
// В столбце j находим самое первое значение T[i,j] = TRUE. Элемент a[i]
// включается в множество M.
   for k:=1 to i do
    if T[k, j] then break;
   i:=k;
   include (M, a[i]);
   if i=1 then break;
   j:= j-a[i];
   i:=i-1;
 end;

 M1:='';
 M2:='';
 For i:= 1 to n do
 if a[i] in M then M1:= M1+', '+intToStr (a[i]) else
 M2:= M2+', '+intToStr (a[i]);

 M1:= copy (M1, 2, 100); M2:= copy (M2, 2, 100);
 LabeledEdit1.Text:= M1;  LabeledEdit2.Text:= M2;
 end;
end.
```

Глава 3.

Примеры базовых элементов программирования

§ 3.1. Создание кнопок

Создать 100 кнопок и с надписями «00», «01», ..., «99» разместить по 10 кнопок в каждом ряду. При щелчке на любую кнопку отобразить информацию о ней, например, ее надпись. Предусмотреть удаление кнопок.




```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants,
  Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls;
type
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    StatusBar1: TStatusBar;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    Procedure A (sender: TObject);
  end;

const
  n=100;
var
  Form1: TForm1;
  v: array[0..n-1] of TButton;
implementation
{$R *.dfm}
procedure TForm1.A(sender: TObject);
begin
  StatusBar1.simpletext:= 'Нажали на кнопку '+
```

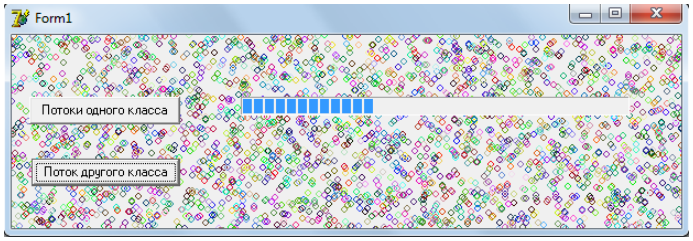
```
  (sender as TButton).caption;
end;
//---------------------------------
procedure TForm1.Button1Click(Sender: TObject);
const
  r=10; w=30;
var
  i: integer;
begin
  for i:=0 to n-1 do
  begin
     v[i]:=TButton.Create(self);
     with v[i] do begin
       left:=(i mod r)*(w+5);
       top:=(i div r)*(w+5);
       width:=w;
       caption:=intToStr(i);
       name:='b'+caption;
       parent:=Form1;
       onclick:=A;
     end;
  end;
end;
//---------------------------------
procedure TForm1.Button2Click(Sender: TObject);
var
  i: integer;
begin
  for i:=0 to n-1 do
  begin
    v[i].Free; sleep (10);
  end;
end;
end.
```

§ 3.2. Визуализация потоков

Литература: [3, с. 448–451], [5, гл. 29]. Пусть при нажатии на одну из двух кнопок требуется запустить два потока одного класса, состязающиеся за протяженность цветной полосы в компоненте ProgressBar, а при нажатии на другую кнопку — запустить поток другого класса, выводящий на форму

разноцветные окружности малого радиуса.

Вызов методов Delphi 7.0 одновременно из разных потоков может привести к конфликтам, которые чаще всего объясняются проблемами быстрого чередования обработки вывода на холст с использованием классов TBrush и TPen. Поэтому прибегают к синхронизации потоков.




```
unit Unit1;
interface
uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics,
 Controls, Forms,   Dialogs, StdCtrls, ComCtrls;
type
  TForm1 = class(TForm)
    Button1: TButton;
    ProgressBar1: TProgressBar;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    procedure wr1;
    procedure wr2;
    procedure Pr;
  end;
//---------------------------------
MT0 = class(TThread)
      public
      procedure Execute; override;
end;
MT1 = class(TThread)
  public
    w: integer;
    procedure Execute; override;
end;
```

```
var
  Form1: TForm1;
  GLOBVAR: integer=0;
implementation
{$R *.dfm}
//---------------------------------
procedure MT1.Execute;
begin
 while not terminated do
 case w of
 1: synchronize (form1.wr1);
 2: synchronize (form1.wr2);
end;
end;
procedure TForm1.wr1;
begin
  inc (GLOBVAR); application.ProcessMessages;
  ProgressBar1.Position:=20+round(GLOBVAR/200);
end;
//---------------------------------
procedure TForm1.wr2;
begin
  dec (GLOBVAR); application.ProcessMessages;
  ProgressBar1.Position:=20+round(GLOBVAR/200);
end;
//---------------------------------
procedure TForm1.Button1Click(Sender: TObject);
var
 t1, t2: MT1;
begin
  t1:=MT1.create(true);   t1.w:=1;  t1.Priority:=tpLower;
  t2:=MT1.create(true);   t2.w:=2;  t2.Priority:=tpLower;
  t1.resume;    t2.resume;
end;
//---------------------------------
procedure TForm1.pr;
var
  x,y: integer;
begin
  x:=random(width); y:=random (height);
  with canvas do
  begin
```

```
    brush.style:=bsClear;
    Pen.Color:=RGB (random(256), random(256), random(256));
    ellipse (x-3,y-3,x+3,y+3);
  end;
  application.ProcessMessages;
end;
//---------------------------------
procedure MT0.Execute;
begin
 while not terminated do synchronize (Form1.pr);
end;
//---------------------------------
procedure TForm1.Button2Click(Sender: TObject);
var
  t0: MT0;
begin
  t0:=MT0.create(true); t0.resume;
end;
end.
```

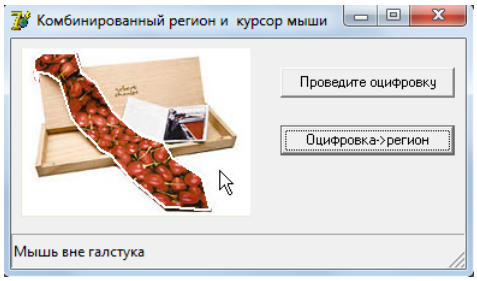
§ 3.3. Распознавание регионов

Литература: [18, с. 20–37]. Регионы могут иметь форму многоугольника, эллипса (или комбинации этих и полученных фигур). Поскольку при комбинировании в качестве прямоугольника можно рассматривать даже отдельно взятую точку, то в результате можно получить регион *любых* очертаний. Регионы служат не только целям разнообразия элементов оконного интерфейса и улучшения их привлекательности, но позволяют также ответить на вопрос о взаимном расположении региона и заданной точки (логическая функция $PtInRegion(h, x, y)$). Они полезны и для привязки действий к отдельным частям рисунка, имеющих «неправильную форму» (например, к контурам отдельных персон на фотографии толпы). Особенно интересно использование регионов для создания окна замысловатой конфигурации (например, окна приложения, созданного на основе изображения взлетающей птицы).

```
Пусть объявлены переменные h, h1 и т.д. типа HRGN.
Для  создания регионов можно использовать функцию
a1) h:=CreateEllipticRgn (x1, y1, x2, y2);
// параметры - координаты описывающего эллипс прямоугольника
a2) h:=CreateRectRgn(i1, j1, i2, j2);
```

```
a3) h:= CreatePolygonRgn
   (массив элементов типа tpoint,
   количество задействованных начальных точек массива,
   какой способ закрашивания используется для определения вхождения
   точек в регион:  WINDING / ALTERNATE / POLYFILL_LAST);

Для установки региона вызывается функция
   SetWindowRgn (handle, h, true или false)
   где handle - идентификатор формы или другого объекта,
    h - идентификатор региона,
    значение третьего параметра определяет, будет ли окно перерисовано после
    установки региона.

Наибольший интерес составляют комбинированные регионы. Создаются они так:
 CombineRgn (Hrezult, h1, h2, operation), где
           Hresult (результат) - идентификатор существующего (!) региона,
           h1, h2 - идентификаторы регионов-операндов,
           operation - одна из операций:
             RGN_AND (пересечение), RGN_COPY (копия первого операнда),
             RGN_DIFF (разность), RGN_OR (объединение).
```

Составим программу, которая выводит сообщение о расположении курсора мыши относительно заданной «неправильной» фигуры. В качестве такой фигуры на рисунке выбрано изображение галстука.




```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants,
  Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, jpeg, ExtCtrls,
  ComCtrls;
type
  TForm1 = class(TForm)
    Image1: TImage;

    Button1: TButton;
    SaveDialog1: TSaveDialog;
    Button2: TButton;
    StatusBar1: TStatusBar;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
```

```
    procedure Image1MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Button2Click(Sender: TObject);
    procedure Image1MouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
  end;
const
  MaxN=1000;
var
  Form1: TForm1;
  f: textFile;
  n: integer;
  digit: boolean = true;
  M: array [0..MaxN-1] of TPoint;
  h1: HRGN;
  PointView: Boolean=false;
implementation
{$R *.dfm}
procedure TForm1.FormCreate(Sender: TObject);
begin
  saveDialog1.InitialDir:=GetCurrentDir;
  if fileExists ('rgn1.txt') then button1.hint:= 'Оцифровка уже проведена'
  else
  begin
    button1.hint:= 'Оцифровка еще не проведена';
    n:=0;
  end;
  Button1.ShowHint:= true;
end;
//-------------------------------------------------------
procedure TForm1.Button1Click(Sender: TObject);
var
  i: integer;
  s: string;
begin
  case Button1.caption [1] of
  '1':  begin
          if fileExists ('rgn1.txt') then
             digit:= MessageDlg ('Файл оцифровки уже существует,'+
             ' провести повторную оцифровку?',
                    mtConfirmation, [mbOK, mbCancel], 0) = mrOK;
          if digit then begin
                        if saveDialog1.Execute then
```

```
                          begin
                            assignFile (f, saveDialog1.FileName);
                            rewrite (f);
                          end;
                        Button1.caption:= 'Проведите оцифровку'
           end; // if digit
         end; //'1'
  '2':  begin
          Button1.caption:= '3Запись в файл проведена';
          writeln (f, '//',n);
          button1.Enabled:=false;
          for i:=0 to n-1 do writeln (f, M[i].x, ' ', M[i].y);
          closeFile(f);
         end; //'2'
   end; //case
end;
//-------------------------------------------------------
procedure TForm1.Image1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  if digit then
  begin
    Button1.caption:= '2Запись в файл';
    inc (n);  M[n-1].x:=x; M[n-1].y:=y;
  end;
end;
//-------------------------------------------------------
procedure TForm1.Button2Click(Sender: TObject);
var
  s: string;
  i: integer;
begin
  assignFile (f, 'rgn1.txt');
  reset (f);
  readln (f,s);
  n:= strToInt (copy(s,3,5));
  for i:=0 to n-1 do
    readln (f, M[i].x, M[i].y);
  CloseFile (f);
//-------------------------------------------------------
 if MessageDlg ('Региональный подход с рисунком (OK) или без (CANCEL)?',
                    mtConfirmation, [mbOK, mbCancel], 0) = mrOK
```

```
 then
  with image1.Canvas do
  begin
    Pen.color:=clWhite;
    Pen.Width:=2;
    moveTo (M[0].x, M[0].Y);
    for i:=1 to n-1 do
    begin
      LineTo (M[i].X, M[i].y);
      sleep (100);
      application.ProcessMessages;
    end;
//-----------------------------------------------------
  end;
  h1:=CreatePolygonRgn (M, n, WINDING);
  PointView:=true;
end;
//-----------------------------------------------------
procedure TForm1.Image1MouseMove(Sender: TObject; Shift: TShiftState;
 X, Y: Integer);
begin
  if not PointView then exit;
  if PtInRegion(h1,x,y) then
  statusBar1.SimpleText:= 'Мышь на галстуке'
  else
    statusBar1.SimpleText:= 'Мышь вне галстука'
end;
end.
```

§ 3.4. Эмуляция клавиатуры и мыши

Литература: [5, с. 675–678], [16, с. 132–135], [17, с. 33–39], [18, с. 106–110].

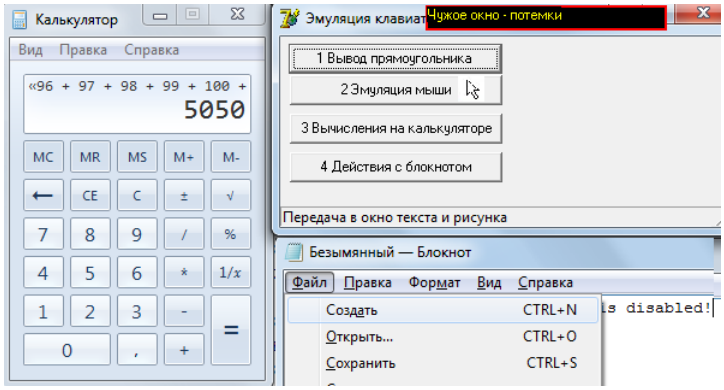
При щелчке по кнопке 1, 2, 3 или 4 требуется выполнить соответствующее действие из следующего списка:

1) в заголовке формы нарисовать прямоугольник с текстом;
2) запустить бесконечный цикл, каждая итерация которого состоит из двух действий:
   - установить курсор в точку (0,0);
   - установить курсор на вторую кнопку формы и выполнить щелчок левой кнопкой мыши;

3) запустить программу «Калькулятор» и выполнить действия по вычислению суммы 1+2+...+100;

4) запустить программу «Блокнот», ввести текст и сохранить в файле.

Для передачи кодов нажатых клавиш в окно другой программы необходимо подключить модуль *sndkey*32, созданный К. Хендерсоном [17, с. 33–39]. Для эмуляции кнопок мыши используется API-функция `mouse_event`. Примеры с применением этой функции и аналогичной функции `keybd_event` для эмуляции нажатия клавиш см., например, в [19, Гл. 6].




```
unit Unit1;
interface
uses sndKey32, Windows, Messages, SysUtils, Variants, Classes, Graphics,
  Controls, Forms, Dialogs, StdCtrls, ExtCtrls, ComCtrls;
type
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    StatusBar1: TStatusBar;
    Button3: TButton;
    Button4: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button2MouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
  end;
var
  Form1: TForm1;
```

```
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
//В заголовке формы нарисовать прямоугольник с текстом
var
  x, y: integer;
begin
  with TCanvas.Create do
  begin
    Handle:=GetDC (0);
    pen.width:=2;
    pen.Color:=clRed;
    Brush.Color:=clBlack;
    x:=Form1.left+Form1.Width div 3;
    y:=form1.top+5;
    rectangle (x,y, x+200, y+20);
    Font.Color:=clYellow;
    textout(x,y, 'Чужое окно - потемки');
   end;
end;
//-----------------------------------------------------
procedure TForm1.Button2Click(Sender: TObject);
//При щелчке на кнопку Button2 запустить бесконечный цикл,
//в каждой итерации которого выполняются два действия:
// 1) установить курсор в точку (0,0),
// 2) установить курсор на кнопку Button2, нажать и отпустить левую кнопку мыши.
const
  x=0; y=0;
begin
  SetCursorPos (x,y);  sleep (1000);
  SetCursorPos (Form1.Left+Button2.left+Button2.Width div 2,
   Form1.top+(Form1.Height-Form1.ClientHeight {высота заголовка})
     + Button2.Top);
  sleep (1000);
  mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0,0,GetMessageExtraInfo);
  application.ProcessMessages;
  sleep (1000);
  mouse_event(MOUSEEVENTF_LEFTUP, 0, 0,0,GetMessageExtraInfo);
  application.ProcessMessages;
  sleep (1000);
end;
//---------------------------------------
```

```
procedure TForm1.Button2MouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
var
  s: string;
  c: char;
begin
  c:=(Sender as TButton).Caption[1];
  case c of
  '1': s:='Передача в окно текста и рисунка';
  '2': s:='Эмуляция мыши: перемещение с нажатием левой кнопки';
  '3': s:='Запуск программы "Калькулятор"';
  '4': s:='Действия с программой "Блокнот"';
  end;
    StatusBar1.SimpleText:=s;
end;
//---------------------------------------
procedure TForm1.Button3Click(Sender: TObject);
//Запустить калькулятор и вычислить сумму 1+2+...+100
var
  i: integer;
Begin
  WinExec ('calc', SW_ShowNormal);            //Запуск программы "калькулятор"
  sendKeys('0', true); sleep (1000);          //Нелишняя инициализация!
  for i:=1 to 100 do sendKeys(PChar(intToStr (i)+'{+}'), true);
end;
//---------------------------------------
procedure TForm1.Button4Click(Sender: TObject);
//Запустить программу "Блокнот", ввести текст и сохранить в файле
begin
  WinExec ('Notepad', SW_ShowNormal); sleep (1000);
//   if appActivate ('Безымянный - Блокнот') then
  sendKeys('Any text (the cyrillic font is disabled!)',true); sleep (500);
  //Некириллический текст передан в окно редактора
  sendkeys('{F10}{Enter}', true); sleep (500);    //перешли в меню
  sleep (10000);
  sendkeys('{Down 3}{Enter}', true); sleep (500); //в меню выбрали пункт "Сохранить как"
  sendkeys('11.txt{ENTER}', true); sleep (500);   //сохранили в файле с именем '11.txt'
  sendKeys('{f10}{Enter}',true); sleep (500);     //перешли в меню
  sendKeys('{down 6}{Enter}', true);              // и выбрали пункт "Выйти"
end;
end.
```

Глава 4.

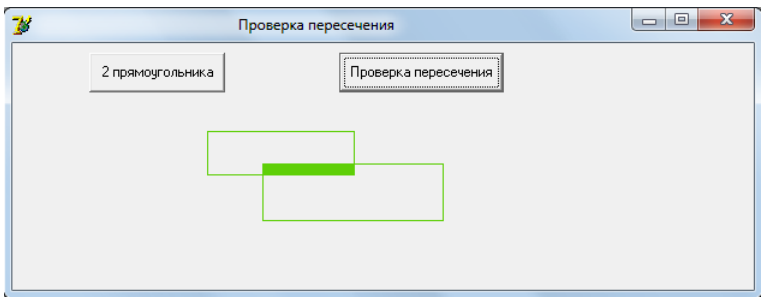
Элементы компьютерной графики

§ 4.1. Пересечение прямоугольников

Пусть (x'_1, y'_1) и (x''_1, y''_1) — соответственно левый-нижний и правый-верхний углы первого прямоугольника, обозначения (x'_2, y'_2) и (x''_2, y''_2) имеют аналогичный смысл для второго прямоугольника. Прямоугольники имеют непустое пересечение тогда и только тогда, когда

$$\max\{x'_1, x'_2\} \le \min\{x''_1, x''_2\} \quad \text{и} \quad \max\{y'_1, y'_2\} \le \min\{y''_1, y''_2\}.$$

Трудно ожидать, что студенты математических специальностей немедленно согласятся с этим утверждением, поэтому напомним, что рассматриваются прямоугольники со сторонами, параллельными сторонам окна приложения.




```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
   Forms, Dialogs, Math, StdCtrls;
type
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
```

```
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  end;
type
  arr2= array[1..2] of integer;
var
  Form1: TForm1;
  IsCross: Boolean;
  x1, y1, x2, y2: arr2;
  Xmin, Xmax, Ymin, Ymax: integer;
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
begin
  randomize;
  with canvas do
  begin
    pen.color:= RGB (random (256), random (256), random (256));
    brush.Style:=bsClear;
    x1[1]:= 100+random (200); x2[1]:=random (200) +10+ x1[1];
    y1[1]:= 100+random (200); y2[1]:=random (200) +15+ y1[1];
    Rectangle  (x1[1], y1[1], x2[1], y2[1]);
    x1[2]:= 100+random (200); x2[2]:=random (200) +10+ x1[2];
    y1[2]:= 100+random (200); y2[2]:=random (200) +15+ y1[2];
    canvas.Rectangle  (x1[2], y1[2], x2[2], y2[2]);
  end;
end;
//-----------------------------------------
 function Cross (x1, x2, y1, y2: arr2; var Xmin, Xmax, Ymin, Ymax: integer): boolean;
  begin
    result:=true;
    Xmin:=max(x1[1], x1[2]);
    Xmax:=min(x2[1], x2[2]);
    if Xmin>Xmax then begin Result:=false; exit end;

    Ymin:=max(Y1[1], Y1[2]);
    Ymax:=min(Y2[1], Y2[2]);
    if Ymin>Ymax then Result:=false;
end;
//-----------------------------------------
procedure TForm1.Button2Click(Sender: TObject);
begin
```

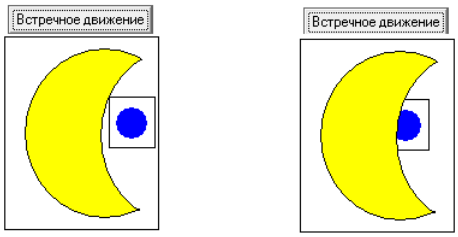
```
   if Cross (x1, x2, y1, y2, Xmin, Xmax, Ymin, Ymax)
   then
   with canvas do
   begin
     brush.Color:= pen.Color;
     rectangle (Xmin, Ymin, Xmax, Ymax);
   end
   else ShowMessage ('Не пересекаются');
end;
end.
```

§ 4.2. Соприкосновение плоских фигур

Литература: [7, с. 156]. Пусть рассматриваются две плоские фигуры (необязательно выпуклые), перемещающиеся по плоскости при каждом тике таймера. Требуется распознать момент их соприкосновения.

Такая задача часто встречается при проектировании игр. Идея изложенного ниже алгоритма встречается в [7] на с. 156.

Вначале сохраним резервные копии рисунков T_1 и T_2. Пусть цвета фонов фигур равны p_1 и p_2 соответственно. При каждом тике проверяется наличие у прямоугольников непустого пересечения (см. §4.1). Если в предыдущий момент времени пересечение было пусто, а в текущий момент времени прямоугольники имеют непустое пересечение (некоторый прямоугольник T), то при малом значении свойства Interval таймера можно считать, что размеры прямоугольника T малы. Поэтому выполнение перебора точек прямоугольника T не займет много времени; если найдется точка, такая, что цвета соответствующих точек в T_1 и T_2 отличны от p_1 и p_2 соответственно (и только тогда), то фигуры имеют общую точку, и из-за малости прямоугольника T эта точка будет близка к точке соприкосновения.



На правом рисунке видно, что фигуры при встречном движении прошли незначительное расстояние (всего несколько точек) после соприкосновения.

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
   Forms, Dialogs, Math, StdCtrls;
type
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  end;
type
  arr2= array[1..2] of integer;
var
  Form1: TForm1;
  IsCross: Boolean;
  x1, y1, x2, y2: arr2;
  Xmin, Xmax, Ymin, Ymax: integer;
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
begin
  randomize;
  with canvas do
  begin
    pen.color:= RGB (random (256), random (256), random (256));
    brush.Style:=bsClear;
    x1[1]:= 100+random (200);   x2[1]:=random (200) +10+ x1[1];
    y1[1]:= 30+random (100);    y2[1]:=random (40) +15+ y1[1];
    Rectangle  (x1[1], y1[1], x2[1], y2[1]);
    x1[2]:= 100+random (200);   x2[2]:=random (200) +10+ x1[2];
    y1[2]:= 30+random (100);    y2[2]:=random (100) +15+ y1[2];
    canvas.Rectangle  (x1[2], y1[2], x2[2], y2[2]);
  end;
end;
//-----------------------------------------
 function Cross (x1, x2, y1, y2: arr2;
                  var Xmin, Xmax, Ymin, Ymax: integer): boolean;
  begin
    result:=true;
    Xmin:=max(x1[1], x1[2]);
```

```
    Xmax:=min(x2[1], x2[2]);
    if Xmin>Xmax then begin Result:=false; exit end;

    Ymin:=max(Y1[1], Y1[2]);
    Ymax:=min(Y2[1], Y2[2]);
    if Ymin>Ymax then Result:=false;
end;
//-----------------------------------------
procedure TForm1.Button2Click(Sender: TObject);
begin
   if Cross (x1, x2, y1, y2, Xmin, Xmax, Ymin, Ymax) then
   with canvas do
   begin
     brush.Color:= pen.Color;
     rectangle (Xmin, Ymin, Xmax, Ymax);
   end
   else ShowMessage ('Не пересекаются');
end;
end.
```

§ 4.3. Схема трехмерного вывода. Сфера

Литература: [10]. Требуется вывести изображение сферы. Выбор радиокнопки определяет каркасный или закрашенный варианты вывода; вращение сферы начинается (или возобновляется) нажатием кнопки START, для приостановки используется кнопка STOP.

Поскольку визуализации трехмерных фигур посвящены несколько параграфов, на примере сферы уместно изложить упрощенную схему визуализации для общего случая.

Начало схемы.

1. Определение начальных данных

- В целях краткости примем несколько упрощений: центр сфера размещается в начале системы координат, количества выводимых меридиан и параллелей равны (в программе $n = 120$) и др.; $\Delta\alpha = \frac{2\pi}{n}$ и $\Delta\beta = \frac{2\pi}{n}$ — приращения по долготе и широте соответственно.

- Пусть «на бумаге» воображаемый прямоугольник вывода проекции сферы задан в виде прямоугольника, ограниченного прямыми $x = x_1 \equiv -6$, $x = x_2 \equiv 6$, $y = y_1 \equiv -6$, $y = y_2 \equiv 6$.

- Радиус сферы зададим с учетом размеров этого прямоугольника, например, $R = 4$.
- Присвоим ненулевые значения углам поворота сферы β_0 и γ_0 относительно осей OY и OZ.
- Для вывода вывода проекции сферы можно выбрать компоненту класса $TImage$.
- Целесообразно объявить в программе массив для проекций угловых точек текущей ячейки сетки: `M: array[1..4] of Tpoint.`

2. Организация циклов вывода параллелей, меридианов и соответствующих ячеек образованной ими сетки (проекций на экран)
Для $i = 0, 1, \ldots, n$ выполнить:
begin
$\alpha = \Delta\alpha \cdot i$; //долгота
Для $j = 0, 1, \ldots, n$ выполнить:
begin
$\beta = \Delta\beta \cdot j$; //широта
...

2.1. Определение на поверхности сферы радиуса R точки $P(\alpha, \beta)$, соответствующей значениям α и β текущей итерации

Для вычисления координат точки $P(\alpha, \beta)$:

$$x[i,j] = R\sin\beta \cdot \cos\alpha,\ y[i,j] = R\sin\beta \cdot \sin\alpha,\ z[i,j] = R\cos\beta$$

достаточно заметить, что длина проекции радиуса-вектора $\overline{OP}$ на плоскость XOY равна $R \cdot \sin\beta$.

2.2. Выполнение поворотов вокруг осей OY и OZ

Выполним для каждой точки $x[i, j]$ повороты на угол β_0 и угол γ_0 вокруг осей OY и OZ соответственно:

```
t:=x[i,j];
x[i,j]:=t*cos(b0)-z[i,j]*sin(b0);
z[i,j]:=z[i,j]*cos(b0)+t*sin(b0);
t:=x[i,j];
x[i,j]:=t*cos(g0)-y[i,j]*sin(g0);
y[i,j]:=y[i,j]*cos(g0)+t*sin(g0).
```

2.3. Масштабирование

Преобразуем размеры «бумажного» рисунка к размерам компоненты:

```
x[i,j]:=Ширина * (x[i,j]-x1)/(x2-x1);
y[i,j]:=Высота * (y[i,j]-y1)/(y2-y1).
```

2.4. Вычисление проекций угловых точек $M[1], M[2], M[3]$ и $M[4]$ пространственного четырехугольника (ячейки сетки)

```
i1:=i+1; j1:=j+1;
if i1=n+1 then i1:=0;
if j1=n+1 then j1:=0;
M[1].x:= round(x[i,j]);     M[1].y:= round(y[i,j]);
M[2].x:= round(x[i,j1]);    M[2].y:= round(y[i,j1]);
M[3].x:= round(x[i1,j1]);   M[3].y:= round(y[i1,j1]);
M[4].x:= round(x[i1,j]);    M[4].y:= round(y[i1,j]).
```

Отступление. Избыточность вычислений, незримо присутствующая и в других местах (многочисленные перевычисления синусов и косинусов для одних и тех же значений аргументов и др.), здесь особенно бросается в глаза: ведь вычисления, связанные с вершиной, выполняются для всех четырех ячеек, для которых вершина является угловой. Но в данной брошюре вопросы оптимизации не рассматриваются.

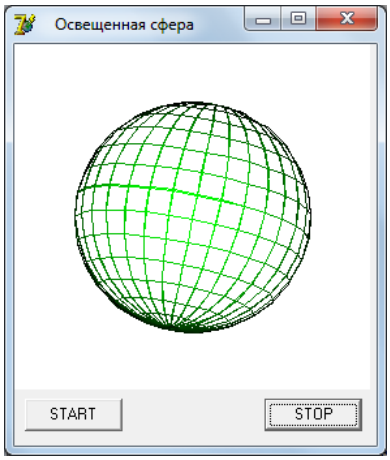
2.5. Вывод освещенной ячейки поверхности сферы

Ввиду малости ячейки сетки, при ее освещении достаточно принять во внимание лишь одну угловую точку $(x[i,j], y[i,j], z[i,j])$. При наших предположениях, координаты точки на сфере совпадают с координами нормали, проведенной в этой точке. Интенсивность освещения примем пропорциональной косинусу угла между направлением на источник освещения с координатами (0,0,R) и нормалью в точке:

```
z1:=z[i,j];
if z1>0 then //видимая часть, выступающая от плоскости
begin        //проекции (экрана) к наблюдателю
 koef:=round (255*z1/R)*$010101;
 Pen.Color:=koef;
 Brush.Color:=koef;
 PolyLine([M[1], M[2], M[3], M[4]]); //каркасный вариант
 PolyGon([M[1], M[2], M[3], M[4]]);  //закрашенный вариант
end;
```

end//завершение цикла по j (см. начало п. 2)

end//завершение цикла по i (см. начало п. 2)




```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants,
  Classes, Graphics, Controls, Forms,
  Dialogs,  StdCtrls, ExtCtrls;
type
  TForm1 = class(TForm)
    Button1: TButton;
    Image1: TImage;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    procedure SDraw(b0, g0: double);
  end;
const
  n=30;
  R=4; x1=-6; x2=6; y1=-6; y2=6;
var
  Form1: TForm1;
  S: Tbitmap;
  H, W: integer;                //размеры компоненты
  xi, yi, zi, x, y, z: array[0..n, 0..n] of double;
  M: array[1..4] of Tpoint;

  b0: double=0; //угол поворота сферы вокруг оси OY
  g0: double=0; //угол поворота сферы вокруг оси OZ
  STOP: boolean=false;
  k: integer=0;
implementation
{$R *.dfm}
Procedure TForm1.SDraw (b0, g0: double);
//Вывод сферы на холст
var
  i, j: integer;   //номера параллелей и меридианов
  i1, j1: integer; //для i+1 и j+1
  koef: integer;   //интенсивность освещения
  t: double;       //вспомогательная переменная
```

```
  z1: double;      //распознавание видимости точки от наблюдателя
begin
 with S.canvas do
 BEGIN
   Pen.color:=clWhite;
   Brush.Color:=clWhite;
   Rectangle (-1,-1, W, H);             //Очистили холст
   for  j:=0 to n do
   for i:=0 to n do
   begin
     //Выполним повороты:
      t:=xi[i,j];                       //на угол b0 вокруг оси OY
      x[i,j]:=t*cos(b0)-zi[i,j]*sin(b0);
      y[i,j]:=yi[i,j];
      z[i,j]:=zi[i,j]*cos(b0)+t*sin(b0);

      t:=x[i,j];                        //на угол g0 вокруг оси OZ
      x[i,j]:=t*cos(g0)-y[i,j]*sin(g0);
      y[i,j]:=y[i,j]*cos(g0)+t*sin(g0);

      x[i,j]:=W * (x[i,j]-x1)/(x2-x1);  //масштабирование
      y[i,j]:=H * (y[i,j]-y1)/(y2-y1);

      i1:=i+1; j1:=j+1;
      if i1=n+1 then i1:=0;
      if j1=n+1 then j1:=0;
       z1:=z[i,j];
       M[1].x:= round(x[i,j]);        M[1].y:= round(y[i,j]);
       M[2].x:= round(x[i,j1]);       M[2].y:= round(y[i,j1]);
       M[3].x:= round(x[i1,j1]);      M[3].y:= round(y[i1,j1]);
       M[4].x:= round(x[i1,j]);       M[4].y:= round(y[i1,j]);
      if z1>0 then  //точка находится на видимой стороне сферы
      begin
       koef:=round (255*z1/R)*$000100;
       Pen.Color:=koef;                    //освещенный...
       PolyLine([M[1], M[2], M[3], M[4]]); //...каркас
       //для закрашивания снять следующие комментарии:
       //Brush.Color:=koef;
       //PolyGon([M[1], M[2], M[3], M[4]]);
      end;
  end; //for j
  END; //with S.canvas
```

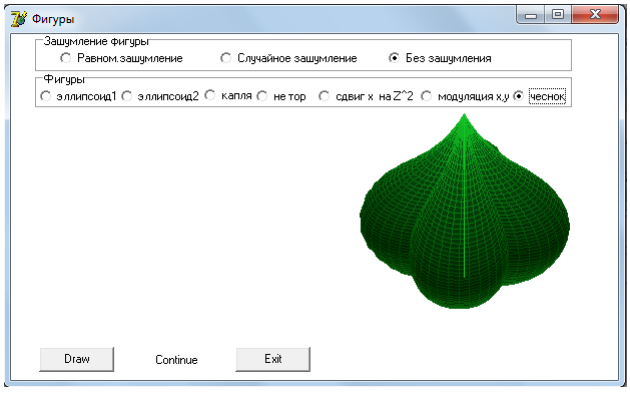
```
  k:=k+1;
  if k>1 then
     Image1.Canvas.Draw(0,0,S);
  application.ProcessMessages;
end;
//---------------------------------------
procedure TForm1.Button1Click(Sender: TObject);
//Обращение к процедуре рисования Sdraw после нового поворота
begin
 STOP:=FALSE;
 repeat
   b0:=b0+0.01;
   g0:=g0+0.01;
   Sdraw(b0, g0);
 until STOP;
end;
//---------------------------------------
procedure TForm1.Button2Click(Sender: TObject);
begin
  STOP:=TRUE;
end;
//---------------------------------------
procedure TForm1.FormCreate(Sender: TObject);
const
  da=2*PI/n; db=da;
var
  i, j: integer;
  a, b: double; //широта и долгота
begin
//Создадим S:
  H:=Image1.Height;
  W:=Image1.Width;
  S:=Tbitmap.Create;
  S.Width:=W;
  S.Height:=H;
//Вычислим координаты угловых точек сетки меридианов и параллелей:
  for  j:=0 to n do
  begin
    b:=j*db;                          //Широта
    for i:=0 to n do
    begin
      a:=da*i;                        //Долгота
```

```
      xi[i,j]:= R*cos(b)*cos(a);      //Текущая точка сетки...
      yi[i,j]:= R*cos(b)*sin(a);      //...из меридианов...
      zi[i,j]:= R*sin(b);             //...и широт
    end;
  end;
end;
end.
```

§ 4.4. Деформация поверхностей

Литература: [10]. В зависимости от выбранной кнопки программа выводит одну из нескольких пространственных фигур, для каждой из которых уточняется характер зашумления. Если ни одна из радиокнопок нижнего ряда (см. рисунок) не выбрана, то выводится сфера, зашумление которой определяется выбором кнопки верхнего ряда. Характер поверхности фигуры задается формулами зависимости произвольной точки, лежащей на поверхности, а также зависит от количества «параллелей» и «меридианов». Формулы могут быть произвольными. С целью сравнения результатов программы с рисунками из [10], в листинге программы после строки «Выбор фигуры» выбраны формулы, приведенные в [10] на с. 220–222.

Для вывода используется схема, изложенная в предыдущем параграфе. Новым здесь является зашумление поверхности путем малых возмущений длин радиусов-векторов некоторых точек поверхности. Действия по сглаживанию поверхности путем усреднения координат точки сетки по ближайшим девяти точкам см. в листинге после строки «2. Сглаживание поверхности»; количество итераций сглаживания, равное трем, выбрано экспериментальным путем.



Фигуры перемещаются по горизонтали, «отталкиваясь» от краев ком-

поненты класса TImage. Нажатие на вторую кнопку (см. рисунок), предназначенную для приостановки движения (и/или для возобновления прерванного движения) приводит к изменению не только надписи кнопки («Stop» – «Continue»), но и значения одноименной глобальной логической переменной. Подчеркнем, что обработчик события щелчка по первой кнопке (на рисунке — кнопка с надписью «Draw»), содержащий в качестве условия завершения цикла равенство Stop=False (см. листинг), не обратил бы внимания на изменение значения переменной Stop, если бы не вызов метода Application.ProcessMessages, настаивающего на предварительной обработке всех сообщений из очереди к программе, прежде чем продолжить работу.

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Spin, ExtCtrls, Buttons;
type
  TForm1 = class(TForm)
    Button1: TButton;
    GroupBox1: TGroupBox;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    RadioButton3: TRadioButton;
    SpeedButton1: TSpeedButton;
    GroupBox2: TGroupBox;
    RadioButton4: TRadioButton;
    RadioButton5: TRadioButton;
    RadioButton6: TRadioButton;
    RadioButton7: TRadioButton;
    RadioButton8: TRadioButton;
    RadioButton9: TRadioButton;
    RadioButton10: TRadioButton;
    Image1: TImage;
    Button2: TButton;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
  Procedure Prepation;
  procedure Draw;
  procedure Word_To_Pro (var x0, y0: integer; alfa, beta: extended;
```

```
  var xw,yw,zw: extended;   var xp1, yp1, zp1: integer);
  end;
const
  R=70; N=60;
var
  Form1: TForm1;
  stop: boolean = false;
  alfa, beta: extended;
  b: tBitmap;
  xx, yy, zz: array [0..N, 0..N] of extended;
  xp, yp, zp: array [0..N, 0..N] of integer;
  x0, y0: integer;
  iRandom, jRandom: set of byte;  //выбранные для зашумления точки
  noise: boolean=false;           //флаг зашумления
  Nnoise: integer = 2;
implementation
{$R *.dfm}
//повороты на углы alpha и beta
procedure Tform1.Word_To_Pro (var x0, y0: integer; alfa, beta: extended;
  var xw,yw,zw: extended;
  var xp1, yp1, zp1: integer);
begin
  xp1:= round (xw*cos(alfa) - yw*sin(alfa));
  yp1:= round (xw*sin(alfa)*cos(Beta) + yw*cos(alfa)*cos(Beta)-zw*sin(Beta));
  zp1:= round (xw*sin(alfa)*sin(Beta) + yw*cos(alfa)*sin(Beta)+zw*cos(Beta));
  xp1:=xp1+x0;  yp1:=yp1+y0;
end;
//-----------------------------------
Procedure TForm1.Prepation;
var
  R1,L1, B1, dL, dB, BP: extended;
  i, j, k, iM, iP, jM, jP: integer;
  ch: integer;
begin
  ch:=3;
  if RadioButton4.checked then ch:=4;  if RadioButton5.checked then ch:=5;
  if RadioButton6.checked then ch:=6;  if RadioButton7.checked then ch:=7;
  if RadioButton8.checked then ch:=8;  if RadioButton9.checked then ch:=9;
  if RadioButton10.checked then ch:=10;
//1. Подготовка. Формирование точек поверхности объекта
  noise:= false;
  Nnoise:=2;
```

```
if RadioButton1.Checked then begin Nnoise:=0; noise:=true end;
if RadioButton2.Checked then begin Nnoise:=1; noise:=true end;
dL:=2*Pi/N;
dB:=Pi/N;
for i:=0 to N do begin
  B1:= -0.5*Pi+i*dB;
for j:= 0 to N do
begin
  L1:=0+j*dL;
  R1:= R;
  Case Nnoise of
  0:  if (j<>0) and (i<>0) and  (j mod 8 =0) and  (i mod 8 =0) then
                                      R1:= R+ (0.5-random)*R*2.5;
  1:  if (j<>0) and (i<>0) and (i in iRandom) and (j in jRandom) then
                             R1:= R+ (0.5-random)*R*2;
  end;
  //Если радиокнопки 4-10 выключены, то - сфера (зашумленная или нет)

  xx[i,j]:= R1*cos (B1)*sin (L1);
  yy[i,j]:= R1*cos (B1)*cos (L1);
  zz[i,j]:= R1*sin(B1);
  //------------------- ВЫБОР ФИГУРЫ -----------------------
  BP:=B1/Pi*2;
  case ch of
  4:  zz[i,j]:= 0.5*R1*sin(B1);
  5:  zz[i,j]:= 1.4*R1*sin(B1);
  6:  if B1>0 then zz[i,j]:= zz[i,j]+R1*BP*BP*BP*BP;

  7: zz[i,j]:=  R1*sin(B1) - R1*BP*BP*BP;
  8:  begin
        xx[i,j]:=  xx[i,j]+zz[i,j]*zz[i,j]/R;
        zz[i,j]:=2*zz[i,j];
       end;
  9:  begin
        xx[i,j]:=  xx[i,j]*(1+0.5*abs(sin(2*L1)));
        yy[i,j]:=yy[i,j]*(1+0.5*abs(sin(2*L1)));
        zz[i,j]:=zz[i,j];
      end;

  10: begin
         xx[i,j]:=  xx[i,j]*(1+0.5*abs(sin(2*L1)));
         yy[i,j]:=yy[i,j]*(1+0.5*abs(sin(2*L1)));
```

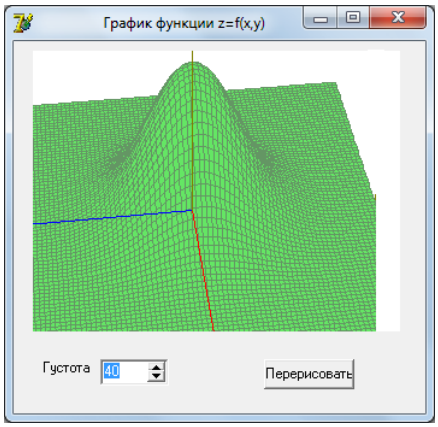
```
          if B1>0 then zz[i,j]:=zz[i,j]+R*BP*BP*BP*BP*BP;
        end;
    end; //case
   End;
  end;
//2. Сглаживание поверхности
if noise then
for k:=1 to 3 do  //три итерации сглаживания поверхности
  for i:= 0 to n do
  for j:= 0 to n do
  begin
    iM:=i-1; iP:=i+1; jM:=j-1; jP:=j+1;
     if iM=-1 then iM:=n; if jM=-1 then jM:=n;
     if iP=n+1 then iP:=0; if jP=n+1 then jP:=0;
//кольцевая индексация: 0, 1, ..., n-1, n=-1, n+1=0, 1, ..., n-1, n=-1, ...
//Усреднение по девяти ближайшим точкам: по три в каждом ряду и каждом столбцу
     xx[i,j]:= ((xx[iM,jM]+ xx[iM, j]+ xx[iM, jP]+
                     xx[i,jM]+ xx[i, j]+ xx[i, jP]+
                     xx[iP,jM]+ xx[iP, j]+ xx[iP, jP])/9);

     yy[i,j]:= ((yy[iM,jM]+ yy[iM, j]+ yy[iM, jP]+
                     yy[i,jM]+ yy[i, j]+ yy[i, jP]+
                     yy[iP,jM]+ yy[iP, j]+ yy[iP, jP])/9);
     zz[i,j]:= ((zz[iM,jM]+ zz[iM, j]+ zz[iM, jP]+
                     zz[i,jM]+ zz[i, j]+ zz[i, jP]+
                     zz[iP,jM]+ zz[iP, j]+ zz[iP, jP])/9);
  end;//ij
end;
//---------------------------------------
procedure Tform1.Draw;
var
  i, j, i1, j1: integer;
begin
 with b.canvas do
 for i:=0 to N do begin
    for j:= 0 to N do
    begin
      Pen.Color:= RGB (round(10*abs(zz[i,j])/R),
                  round(70*(abs(zz[i,j])+60)/R),      round(20*abs(zz[i,j])/R));

      Brush.Color:= RGB (round(5*abs(zz[i,j])/R),
                  round(35*(abs(zz[i,j])+60)/R),      round(10*abs(zz[i,j])/R));
```

```
      if zp[i,j]>=0 then
      begin
        i1:=i+1; j1:=j+1; if i1=N+1 then i1:=0; if j1=N+1 then j1:=0;
        PolyLine ([Point (xp[i,j], yp[i,j]),Point (xp[i1,j], yp[i1,j]),
                   Point (xp[i1,j1], yp[i1,j1]),  Point (xp[i,j1], yp[i,j1])]);

        PolyGon ([Point (xp[i,j], yp[i,j]),Point (xp[i1,j], yp[i1,j]),
                   Point (xp[i1,j1], yp[i1,j1]),  Point (xp[i,j1], yp[i,j1])]);
        end;
    end;//j
   end;//i
  Image1.Canvas.Draw(0, 0, b);
End;
//-------------------------------------------------
procedure TForm1.FormCreate(Sender: TObject);
var
  i, j: byte;
begin
//в дальнейшем все графические построения будут выполнены в
//памяти на холсте переменной b; после завершения построений
//рисунок будет отображен на image1
  b:= tBitmap.Create;
  b.Width:= Image1.width;
  b.Height:=Image1.Height;
  x0:= b.width div 2;   y0:= b.Height  div 2;
  iRandom:=[];
  for i:= 1 to 20 do include (iRandom, random (N));
  jRandom:=[];
  for j:= 1 to 20 do include (jRandom, random (N));
end;
//-----------------------------------
procedure TForm1.Button1Click(Sender: TObject);
var
  i, j, x1: integer;
  deltaX: shortint;
begin
  Prepation;
  deltaX:=1;
  if RadioButton1.Checked then begin Nnoise:=0; noise:=true end;
  if RadioButton2.Checked then begin Nnoise:=1; noise:=true end;
  alfa:=0.0;
  while not stop do
```

```
  begin
      {Плавное изменение углов поворота}
      alfa:=alfa+0.01; beta:=alfa;
      {Стирание всего b}
      with b.canvas do
      begin
        brush.color:= clWhite;                      //очищаем...
        rectangle (-1, -1, b.width+1, b.height+1); //... холст
      end;
     {Смещение абсциссы точки вывода на холст b - перемещение по горизонтали}
      x0:= x0 + deltaX;
      if (x0+R=b.width) or (x0=0+R) then  deltaX:= -deltaX;
      {отталкивание от левого и правого краев}
      for i:=0 to N do
      for j:=0 to N do
         Word_To_Pro (x0, y0, alfa, beta,  xx[i,j], yy[i,j], zz[i,j],
         xp[i,j], yp[i,j], zp[i,j]);

      draw; {Вывод содержимого  b на холст формы}
      application.ProcessMessages;
   end;
end;
//-----------------------------------
procedure TForm1.SpeedButton1Click(Sender: TObject);
//Остановка движения -- продолжение движения
begin
   stop:=not stop; //участвует в цикле вывода изображения
   if SpeedButton1.caption = 'Stop' then SpeedButton1.caption:='Continue'
   else
   if SpeedButton1.caption = 'Continue' then SpeedButton1.caption:='Stop';
   button1.click; //Нажатие на кнопку Draw
end;
//-----------------------------------
procedure TForm1.Button2Click(Sender: TObject);
begin
  close
end;
end.
```

§ 4.5. График функции двух переменных

Требуется построить график функции $z = f(x, y)$. В программе выбрано $f(x, y) = e^{-8(x^2+y^2)}$. Предусмотрены повороты рисунка мышкой.




```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants,
  Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, StdCtrls, Spin;
type
  TForm1 = class(TForm)
    SpinEdit1: TSpinEdit;
    Label1: TLabel;
    Image1: TImage;
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
     procedure Image1MouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Image1MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Image1MouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure Button1Click(Sender: TObject);
  private
      b: Tbitmap;
      N: integer;
      i1, j1, i2, j2: integer;
      MustDraw: boolean;
      x,y: array [0..3] of integer;
      xmin, xmax, ymin, ymax: double;
      alpha, beta, A: double;
      procedure Cnv (x, y, z: double; var xp, yp: integer);
  procedure D3 (h: double);
  end;
var
  Form1: TForm1;
  h: double;
implementation
{$R *.dfm}
```

```
procedure Tform1.Cnv (x, y, z: double; var xp, yp: integer);
var
  x1, y1, z1, x2, y2, z2: double;
begin
  //Поворот вокруг оси 0Z на угол alpha
  x1:= x*cos (alpha) + y*sin (alpha);
  y1:= x*sin (alpha) - y*cos (alpha);
  z1:=z;
  //Поворот вокруг оси 0X на угол beta
  x2:= x1;
  y2:= y1*cos (beta) + z1*sin(beta);
  z2:=y1*sin (beta) - z1*cos (beta);

  x2:= x2/(z2/A+1); y2:=y2/(z2/A+1);
  //масштабирование
  with b.canvas do
  begin
    xp:= round (b.width *(x2-xmin)/(xmax-xmin));
    yp:= round (b.height*(y2-ymin)/(ymax-ymin));
  end;
end;
//---------------------------------
Function F (x,y: double): double;
begin  F:= exp(-8*x*x-8*y*y); end;
//---------------------------------
procedure Tform1.D3 (h: double);
const
  h0=0.0;
var
  i, j: integer;
begin
  with b.canvas do
begin
//сотрем прежний рисунок
  brush.color:=clWhite;
  rectangle (-1, -1, b.width, b.height);

  pen.color:=RGB(100, 150, 100);
  brush.color:= RGB(100, 230, 100);
//Сетка с поворотом каждой ячейки
  for j:=-N to N do
  for i:=-N to N do
```

```
  begin
    cnv (h0+h*i,   h0+h*j,     F(h0+h*i, h0+h*j),    x[0], y[0]);
    cnv (h0+h*i,   h+h0+h*j,   F(h0+h*i, h+h0+h*j),  x[1], y[1]);
    cnv (h+h0+h*i, h+ h0+h*j,  F(h+h0+h*i, h+h0+h*j),x[2], y[2]);
    cnv (h+h0+h*i, h0+h*j,     F(h+h0+h*i, h0+h*j),  x[3], y[3]);

    polygon ([Point(x[0],y[0]), Point(x[1],y[1]),
                                        Point(x[2],y[2]), Point(x[3],y[3])]);

    polyLine ([Point(x[0],y[0]), Point(x[1],y[1]),
                                        Point(x[2],y[2]), Point(x[3],y[3])]);
  end; //for i
  //нарисуем оси координат:
  brush.color:=clWhite;
  pen.Color:=clRed; font.Color:=clRed;
  moveto (i1, j1);
  Lineto (i2, j2);
  textOut (i2+3, j2, 'X');
  pen.Color:=clBlue; font.Color:=clBlue;
  cnv (0, 0, 0, i1, j1);
  cnv (0, 1.2, 0, i2, j2);
  moveto (i1, j1);
  Lineto (i2, j2);
  textOut (i2+3, j2, 'Y');
  pen.Color:=clOlive; font.Color:=clOlive;
  cnv (0, 0, 0, i1, j1);
  cnv (0, 0, 1.2, i2, j2);
  moveto (i1, j1);
  Lineto (i2, j2);
  textOut (i2+3, j2-3, 'Z');
end; //with
Image1.Canvas.Draw(0,0,b);
end;
//---------------------------------
procedure TForm1.FormCreate(Sender: TObject);
begin
  b:=Tbitmap.create;
  b.width:=Image1.width;
  b.Height:=b.Width;
  N:=spinEdit1.value; h:=1.0/N;
  A:= -6.5;
  alpha:=1; beta:=-0.8;
```

```
  xmin:=-1; xmax:=1; ymin:=-1; ymax:=1;
  D3 (h); //Вывод начального рисунка
end;
//---------------------------------
procedure TForm1.Image1MouseUp(Sender: TObject; Button: TMouseButton;
                                    Shift: TShiftState; X, Y: Integer);
begin
  MustDraw:=false;
end;
//---------------------------------
procedure TForm1.Image1MouseDown(Sender: TObject; Button: TMouseButton;
                                    Shift: TShiftState; X, Y: Integer);
begin
  MustDraw:=true;
end;
//Движение мыши определяет углы поворота alpha и beta
procedure TForm1.Image1MouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
var
  a1, b1: double;
begin
  if mustDraw then begin
    a1:=x-b.width div 2; b1:=y-b.height div 2;
    if a1<>0 then alpha:=arctan (b1/a1) else alpha:=Pi/2;
    beta:=sqrt (sqr (a1/N) + sqr (b1/N));
    D3 (h);
  end;
end;
//Перерисовка с измененной густотой линий
procedure TForm1.Button1Click(Sender: TObject);
begin
  h:=1.0/spinEdit1.value;
  D3(h);
end;
end.
```

Литература

1. **Malhotra, V.M.** An $O(V^3)$ algorithm for maximum flows in networks [Текст] / V.M. Malhotra, M.P. Kumar, and S.N. Maheswari / Information Processing Lett., 7, p. 277–278, 1978.

2. **Бахвалов, Н.С.** Численные методы [Текст] / Н.С. Бахвалов, Н.П. Жидков, Г.М. Кобельков. – 7-е изд. – М.: БИНОМ. Лаборатория знаний, 2012. – 636 с.

3. **Бобровский, С.И.** Delphi 7. Учебный курс [Текст] / С.И. Бобровский. – СПб.: Питер, 2004. – 736 с.

4. **Гэри, М.** Вычислительные машины и труднорешаемые задачи [Текст] / М. Гэрри, Д. Джонсон. – М.: Мир, 1982. – 416 с.

5. **Дарахвелидзе, П.Г.** Программирование в Delphi 7 [Текст] / П.Г. Дарахвелидзе, Е.П. Марков. – СПб.: БХВ-Петербург, 2003. – 784 с.

6. **Кострикин, А.И.** Введение в алгебру [Текст] / А.И. Кострикин. – М.: Наука, 1977. – 495 с.

7. **Краснов, М.В.** DirectX. Графика в проектах Delphi [Текст] / М.В. Краснов. – СПб.: БХВ-Петербург, 2001. – 416 с.

8. **Культин, Н.Б.** Delphi в задачах и примерах [Текст] / Н.Б. Культин. – СПб.: БХВ-Петербург, 2003. – 288 с.

9. **Майника, Э.** Алгоритмы оптимизации на сетях и графах [Текст] / Э. Майника. – М.: Мир, 1985. – 323 с.

10. **Порев, В.Н.** Компьютерная графика [Текст] / В.Н. Порев. – СПб.: БХВ-Петербург, 2004. – 432 с.

11. **Свами, М.** Графы, сети и алгоритмы [Текст] / М. Свами, К. Тхуласираман. – М.: Мир, 1984. – 455 с.

12. **Стивенс, Р.** Delphi. Готовые алгоритмы [Текст] / Р. Стивенс; пер. с англ. Мерещука П.А. – М.: ДМК Пресс; СПб.: Питер, 2004. – 386 с.

13. **Сузи, Р. А.** Язык программирования Python [Текст] / Р.А. Сузи. – М.: Интернет-университет информационных технологий, Бином. Лаборатория, 2007. – 206 с.

14. **Тюкачёв, Н.А.** Программирование в Delphi для начинающих [Текст] / Н.А. Тюкачёв, К.С. Рыбак, Е.Е. Михайлова. – СПб.: БХВ-Петербург, 2007. – 672 с.

15. **Тюкачёв, Н.А.** Delphi 5. Создание мультимедийных приложений. Учебный курс [Текст] / Н.А. Тюкачёв, Ю.Свидиров. – СПб.: Питер, 2001. – 400 с.

16. **Флёнов, М.Е.** Delphi в шутку и всерьез: что умеют хакеры [Текст] / М.Е. Флёнов. – СПб.: Питер, 2005. – 271 с.

17. **Хендерсон, К.** Руководство разработчика баз данных в Delphi 2 [Текст] / К. Хендерсон. – К.: Диалектика, 1996. – 544 с.

18. **Чиртик, А.А.** Delphi. Трюки и эффекты [Текст] / А.А. Чиртик, В.В. Борисок, Ю.И. Корвель. – СПб.: Питер, 2007. – 400 с.

19. **Шкрыль, А.А.** Delphi. Народные советы [Текст] / А.А. Шкрыль. – СПб.: БХВ-Петербург, 2007. – 400 с.

20. **Энгельс, Ф.** Диалектика природы [Текст] / Ф. Энгельс. – М.: Партиздат, 1932.

Содержание

Абдулкарим Магомедович Магомедов

ПРАКТИКА ПРОГРАММИРОВАНИЯ
ВТОРОЙ СЕМЕСТР

Учебное пособие

Редактор Хуршилова М.Б.
Корректор Цахаева Э.И.
Компьютерная верстка Губарева С.В.

Подписано в печать 10.09.12	Формат 30×42 1/8
Печать офсетная. Усл. п. л. 13	Уч.-изд. л. 6,4
Тираж 60 экз.	Заказ

Издательство ДГУ
г. Махачкала, ул. М. Ярагского, 59^e