**Программирование на языке ПАСКАЛЬ**

**Содержание**

Введение\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3

1. Задание на курсовую работу \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3
2. Анализ задания \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3
3. Разработка структур данных \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4
4. Разработка алгоритмов\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_5
5. Текст программы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_9
6. Тестирование программы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20
7. Руководство пользователя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_22

Заключение\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_26

Список использованных источников\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_26

**Введение**

Система программирования Турбо Паскаль представляет собой единство двух в известной степени самостоятельных начал: компилятора с языка программирования Паскаль и некоторой инструментальной программной оболочки, способствующей повышению эффективности создания программ.

Процедуры и функции представляют собой важный инструмент Турбо Паскаля, позволяющий писать хорошо структурированные программы. В структурированных программах обычно легко прослеживается основной алгоритм, их нетрудно понять любому читателю, они проще в отладке и менее чувствительны к ошибкам программирования. Все эти свойства являются следствием важной особенности процедур (функций), каждая из которых представляет собой во многом самостоятельный фрагмент программы, связанный с основной программой лишь с помощью нескольких параметров.

Модули представляют собой прекрасный инструмент для разработки библиотек прикладных программ и мощное средство модульного программирования. В Турбо Паскале имеется 8 стандартных модулей, в которых содержится большое число разнообразных типов, констант, процедур и функций.

**1 Задание на курсовую работу**

В прямоугольной матрице выявить все подматрицы, по периметру которых расположены элементы, из которых можно образовать геометрическую прогрессию.

**2 Анализ задания**

Задачей курсового проектирования является разработка программного продукта, который позволил бы выявить в прямоугольной матрице все подматрицы, по периметру которых расположены элементы, из которых можно образовать геометрическую прогрессию.

Должен быть разработан пользовательский интерфейс, который бы обеспечил более удобную и наглядную работу с программой. Данная программа должна выполнять следующие действия:

* ввод данных с клавиатуры и из файла;
* обработку данных в соответствии с заданием;
* вывод результата на экран и в файл;
* интерфейс пользователя в текстовом режиме, содержащий меню (с подменю) и окна;
* обеспечивать различные способы ввода и вывода данных;
* использовать возможности модуля CRT;
* использовать структуры данных;
* программный код в виде процедур и функций;
* использовать программирование клавиатуры;
* использовать манипулятор «мышь»;
* использовать динамические структуры данных;
* собственный модуль.

**3 Разработка структур данных**

Основные переменные:

Mxx – исходная целочисленная матрица (максимум 10х10);

Per – целочисленные элементы периметра;

N, M – количество строк и столбцов матрицы;

L – количество элементов по периметру подматрицы;

mf – флаг ввода исходной матрицы – true, false – матрица не введена;

fileresu, filework – текстовые файлы с исходной матрицей и с результатом соответственно;

fil – способ вывода результата true – в файл, false – на экране в окне;

f – способ получения результата false – все подматрицы или true – по условию.

Стандартные модули: crt, app, objects, menus, drivers, views, dialogs, msgbox, memory, dos, stddlg, editors и собственный модуль: mod\_matr.

**4** **Разработка алгоритмов**

Опишем основные процедуры и функции программы.

Процедура GetRandomMatrix.

Процедура автоматического ввода матрицы со значениями.

Схема алгоритма данной процедуры представлена на рисунке 1.



Рисунок 1 – Схема алгоритма процедуры GetRandomMatrix.

3 блок – вводим количество строк n, количество столбцов m и максимальное pred значение элемента матрицы, 4-6 блоки – вводим случайные элемента матрица, не превышая pred, 7 блок – устанавливаем флаг в true, т. е. матрица введена, 8 блок – выводим матрицу в окне программы.

Процедура InputMatrix.

Процедура ввода матрицы со значениями вручную.

Схема алгоритма данной процедуры представлена на рисунке 2.



Рисунок 2 – Схема алгоритма процедуры InputMatrix.

3 блок – вводим количество строк n, количество столбцов m, 4-6 блоки – вводим элементы матрицы, 7 блок – устанавливаем флаг в true, т. е. матрица введена, 8 блок – выводим матрицу в окне программы.

Процедура Sort.

Процедура сортировки значений периметра подматрицы по возрастанию.

Схема алгоритма данной процедуры представлена на рисунке 3.



Рисунок 3 – Схема алгоритма процедуры Sort.

2, 3 блок – перебираем элементы периметра подматрицы, 4 блок – сравниваем текущее значение элемента матрицы меньше следующего, если да то переходим к 5 блоку, иначе к 3, 5 блок – меняем значения местами.

Процедура GetPerimetr.

Процедура определения элементов подматрицы по периметру.

Схема алгоритма данной процедуры представлена на рисунке 4.



Рисунок 4 – Схема алгоритма процедуры GetPerimetr.

2, 3 блок – обнуляем значения элементов периметра, 4 блок – количество элементов в перимерте = 1, 5, 6 блоки – перебираем элементы столбца периметра и заносим в массив per, 7, 8 блоки – перебираем элементы строки периметра и заносим в массив per, 9 блок – заносим в переменную l количество элементов в периметре подматрицы.

Функция GeomProg.

Функция определения геометрической прогрессии из элементов периметра подматрицы.

Схема алгоритма данной функции представлена на рисунке 5.



Рисунок 5 – Схема алгоритма процедуры GeomProg.

2 блок – вызов процедуры сортировки, 3 блок – флаг образования геометрической прогрессии, 4 блок – второе значение элемента массива периметра делим на первое, 5 блок – перебираем элементы массива периметра, 6 блок – следующее значение элемента массива делим на текущее и сравниваем с dv, 7 блок – не геометрическая прогрессия.

**5** **Текст программы**

program MTRKW; {*программа*}

uses

mod\_matr;

begin

Randomize;

matrix\_perim.init;

matrix\_perim.run;

matrix\_perim.done;

end.

Unit mod\_matr; {*модуль программы*}

interface

uses

crt,app,objects,menus,drivers,views,dialogs,msgbox,memory,dos,stddlg,editors;

const

cmwork=203; cmvvod=206; cmvval=207; cmvvse=204; cmvusl=205; cmvhel=257; cmisfl=208;

cmfilv=209; cmfilu=210;

wincoml:Tcommandset=[cmvvse,cmvusl,cmwork,cmfilv,cmfilu]; {*меню результат*}

wincom2:tcommandset=[cmvvod,cmvval,cmisfl]; {*меню ввод*}

type

pinputwindow=^tinputwindow;

tinputwindow=object(tdialog)

procedure handleevent(var event:tevent);virtual;

end;

pwind=^twin;

twin=object(twindow)

constructor init(bounds:trect);

end;

pinterior=^tinterior;

tinterior=object(tscroller)

constructor init(var bounds:trect; hs,vs:pscrollbar);

procedure handleevent(var event:tevent);virtual;

end;

tmatrix=object(tapplication)

procedure initstatusline; virtual;

procedure initmenubar; virtual;

procedure handleevent(var event:tevent); virtual;

procedure inputmatrix;

procedure getrandommatrix;

procedure main(f:boolean);

procedure PrintMatrix ;

procedure fileinput;

procedure fileoutputv(f:boolean);

end;

tdata=record

col:string[4];

row:string[4];

max:string[4];

end;

var

matrix\_perim:Tmatrix;

x,y:integer;

const

 MAX = 10; {*максимально матрица 10 на 10*}

type

Matrix = array [1..MAX,1..MAX] of integer;

Perim = array [1..MAX\*4] of integer;

var

wind:pwind;

Mxx:Matrix; {*исходная матрица*}

N,M:integer; {*количество строк и столбцов*}

l:integer; {*количество элементов по периметру подматрицы*}

per:Perim; {*элементы периметра*}

mf:boolean; {*флаг ввода исходной матрицы - true*}

data:tdata;

matr\_v:array[1..10] of string;

fileresu,filework:text; {*текстовые файлы с исходной матрицей и с результатом*}

fil:boolean; {*способ вывода результата true – в файл, false – на экране в окне*}

implementation

procedure readmas; *{очистка матрицы}*

var

 i,j:integer;

 s:string;

 sr:string[3];

begin

s:='';

for i:=1 to n do

begin

for j:=1 to m do

begin

str(mxx[i,j],sr);

s:=s+sr+' ';

end;

matr\_v[i]:=s;

s:='';

end;

end;

{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}

constructor tinterior.init; {*инициализация*}

begin

tscroller.init(bounds,hs,vs);

growmode:=gfgrowhix+gfgrowhiy;

setlimit(8000,7000);

end;

{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}

procedure tinterior.handleevent(var event:tevent); {*движение мыши*}

var

r:tpoint;

location:word;

begin

location:=0;

tscroller.handleevent(event);

case event.what of

evmousedown:

begin

makelocal(mousewhere,r);

location:=delta.y+r.y;

scrolldraw

end;

evkeydown:

case event.keycode of

kbgrayminus:if location>delta.y then

begin

dec(location); scrolldraw

end;

kbgrayplus:if location<delta.y+pred(size.y) then

begin

inc(location);

scrolldraw

end;

end;

end;

end;

{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}

constructor twin.init(bounds:trect); {*инициализация прокрутки*}

var

hs,vs:pscrollbar;

interior:pinterior;

begin

twindow.init(bounds,'',0);

getcliprect(bounds);

bounds.grow(-1,-1);

vs:=standardscrollbar(sbvertical+sbhandlekeyboard); hs:=standardscrollbar(sbhorizontal+sbhandlekeyboard);

interior:=new(pinterior,init(bounds,hs,vs));

insert(interior);

readmas;

end;

{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}

procedure tinputwindow.handleevent;

begin

inherited handleevent(event);

if event.what=evcommand then

endmodal(event.command);

end;

{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}

procedure tmatrix.PrintMatrix; {*Вывод рабочей матрицы в окне*}

var

i,j:integer; {*счётчики*}

sr:string;

r:trect;

begin

r.assign(0,0,80,23);

wind:=new(pwind,init(r));

desktop^.insert(wind);

with wind^ do

begin

r.assign(2,1,70,3);

insert(new(pstatictext,init(r,'Рабочая матрица:')));

for i:=1 to n do

begin

for j:=1 to m do

begin

str(mxx[i,j],sr);

r.assign(1+j\*4,1+i,4+j\*4,3+i);

insert(new(pstatictext,init(r,sr))); {*выводим матрицу*}

end;

end;

r.assign(2,i+2,70,4+i); insert(new(pstatictext,init(r,'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_')));

end;

end;

{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}

procedure tmatrix.initmenubar; {*Инициализация панели меню*}

var

r:trect;

begin

getextent(r);

r.b.y:=succ(r.a.y);

menubar:=new(pmenubar,init(r,newmenu(

newsubmenu('Ввод',hcnocontext,

newmenu(

newitem('Ручной','F3', kbf3,cmvvod,hcnocontext,

newitem('Автоматический','F4',kbf4,cmvval,hcnocontext,

newitem('Из файла','F7',kbf7,cmisfl,hcnocontext,

newline(

newitem('Выход','Alt+X',kbaltx,cmquit,hcnocontext,nil)))))),

newsubmenu('Результат',hcnocontext,

newmenu(

newitem('Все подматрицы','F5', kbf5,cmvvse,hcnocontext,

newitem('По условию','F6',kbf6,cmvusl,hcnocontext,

newline(

newitem('Записать все','F7', kbf7,cmfilv,hcnocontext,

newitem('Записать по условию','F8',kbf8,cmfilu,hcnocontext,

nil)))))),

newitem('О программе','F1', kbf1,cmvhel,hcnocontext,nil))) )))

end;

{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}

procedure tmatrix.initstatusline; {*Инициализация строки состояния*}

var

r:trect;

begin

getextent(r);

r.a.y:=pred(r.b.y);

statusline:=new(pstatusline,init(r,newstatusdef(0,$ffff,newstatuskey('Alt+X-Выход',kbAltX, cmQuit,nil),nil)));

disablecommands(wincoml);

enablecommands(wincom2)

end;

{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}

procedure tmatrix.GetRandomMatrix; {*Автоматический ввод матрицы*}

var

i,j:integer; {*счётчики*}

a,pred,s\_i,s\_j,s\_pred:integer;

r:trect; col,row,max: PInputLine;

inputwindow,inputw:pinputwindow;

begin

r.assign(15,5,65,16);

inputwindow:=new(pinputwindow, Init(r, 'Автоматический ввод'));

with inputwindow^ do

begin {*выводим модальное окно, в котором вводим количество строк, столбцов и мак. значение*}

r.Assign(37,2,45,3);

col:=New(PInputLine, Init(r,4));

Insert(col); r.Assign(2,2 , 35,3);

Insert(New(PLabel, Init(r, 'Количество строк матрицы', col)));

r.Assign(37,4,45,5);

row:=New(PInputLine, Init(r,4));

Insert(row); r.Assign(2,4,35,5);

Insert(New(PLabel, Init(r, 'Количество столбцов матрицы', row)));

r.Assign(37,6,45,7);

max:=New(PInputLine, Init(r,4)) ;

Insert(max); r.Assign(2,6,35,7);

Insert(New(PLabel, Init(r, 'Максимальное значение элемента', max)));

r.Assign(19,8,32,10);

Insert(New(PButton, Init(r, 'OK', cmOk, bfdefault)));

 end;

if desktop^.execview(inputwindow)=cmok then

begin {*вводим элементы в матрицу*}

inputwindow^.getdata(data);

val(data.col,n,s\_i);

val( data.row,m,s\_j) ;

val(data.max,pred,s\_pred) ;

for i:=1 to N do

for j:=1 to M do

begin

a:=random(pred)+1;

Mxx[i,j]:=a;

 end;

mf:=true; {*флаг ввода исходной матрицы*}

dispose(inputwindow,done);

enablecommands(wincoml);

tmatrix.printmatrix; {в*ыводим рабочую матрицу*}

end;

end;

{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}

procedure tmatrix.InputMatrix; {*Ручной ввод матрицы*}

var

i,j,s\_i,s\_j:integer; {*счётчики*}

a:integer;

r:trect;

col,row,c: PInputLine;

inputwindow,inputw:pinputwindow;

s,t:string;

begin

r.assign(15,5,65,16);

inputwindow:=new(pinputwindow, Init(r, 'Ручной ввод'));

with inputwindow^ do

begin {*выводим модальное окно, в котором вводим количество строк, столбцов*}

r.Assign(37,2,45,3);

col:=New(PInputLine, Init(r,4));

Insert(col);

r.Assign(2,2 , 35,3);

Insert(New(PLabel, Init(r, 'Количество строк матрицы', col)));

r.Assign(37,4,45,5);

row:=New(PInputLine, Init(r,4));

Insert(row);

r.Assign(2,4,35,5);

Insert(New(PLabel, Init(r, 'Количество столбцов матрицы', row)));

r.Assign(19,8,32,10);

Insert(New(PButton, Init(r, 'OK', cmOk, bfdefault)));

 end;

if desktop^.execview(inputwindow)=cmok then

begin {*водим элементы матрицы в окне*}

inputwindow^.getdata(data);

val(data.col,n,s\_i);

val( data.row,m,s\_j);

dispose(inputwindow,done);

for i:=1 to N do

for j:=1 to M do

begin

str(i,t);

str(j,s);

r.assign(15,5,65,16);

inputwindow:=new(pinputwindow, Init(r, 'Ввод элемента матрицы'));

with inputwindow^ do

begin

r.Assign(2,4,35,5);

Insert(New(PLabel, Init(r, 'Элемент матрицы'+'['+t+','+s+']', c)));

r.Assign(37,4,45,5);

Insert(New(PInputLine, Init(r,4)));

r.Assign(19,8,32,10);

Insert(New(PButton, Init(r, 'OK', cmOk, bfdefault)));

 end;

if desktop^.execview(inputwindow)=cmok then

begin {*заносим в матрицу Mxx значения*}

inputwindow^.getdata(data);

val(data.col,a,s\_i);

Mxx[i,j]:=a;

dispose(inputwindow,done);

end;

end;

mf:=true; {*флаг ввода исходной матрицы*}

enablecommands(wincoml);

tmatrix.printmatrix; {в*ыводим рабочую матрицу*}

end;

end;

{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}

procedure tmatrix.fileinput; {*ввод данных из файла*}

var

pf:pfiledialog;

s:pathstr;

x:char;

i,j:integer; {*счётчики*}

begin

new(pf,init('\*.txt','Выберите нужный файл:','Имя файла',fdopenbutton,0));

if desktop^.execview(pf)=stddlg.cmfileopen then

begin {*считывание матрицы из файла*}

pf^.getfilename(s);

assign(filework,s);

reset(filework); {*открываем файл для чтения*}

enablecommands(wincoml);

dispose(pf,done);

i:=1;

j:=1;

while not eof(filework ) do

begin

while not eoln(filework) do

begin

read(filework,mxx[i,j]); {*заносим в матрицу значения*}

read(filework,x);

j:=j+1;

n:=i;

m:=j-1;

end;

j:=1;

i:=i+1;

readln(filework);

end;

close(filework); {*закрываем файл*}

tmatrix.printmatrix; {*выводим рабочую матрицу*}

end;

end;

{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}

procedure Sort; {*сортируем элементы периметра по возрастанию*}

var

i,j:integer; {*счётчики*}

p:integer; {*вспомогательная переменная для обмена значениями*}

begin

for i:=2 to l do

for j:=l downto i do

if per[j-1]>per[j] then

begin {*меняем местами элементы*}

p:=per[j-1];

per[j-1]:=per[j];

per[j]:=p;

end;

end;

{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}

procedure GetPerimetr(n\_1,m\_1,n\_2,m\_2:integer); {*определение элементов по периметру*}

var

i:integer; {*счётчик*}

k:integer;

begin {*обнуляем массив*}

for i:=1 to 2\*N+2\*M do

per[i]:=0;

k:=1;

for i:=m\_1 to m\_2 do {*выбираем элементы столбца периметра*}

begin

per[k]:=Mxx[n\_1,i];

per[k+1]:=Mxx[n\_2,i];

inc(k,2)

end;

for i:=(n\_1+1) to (n\_2-1) do {*выбираем элементы строк периметра*}

begin

per[k]:=Mxx[i,m\_1];

per[k+1]:=Mxx[i,m\_2];

inc(k,2)

end;

dec(k,2);

l:=k;

inc(l);

end;

{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}

procedure PrintSubMatrix(n\_1,m\_1,n\_2,m\_2:integer); {*выводим в окно результирующую матрицу*}

var

i,j,k,l:integer;

r:trect;

s,sr:string;

begin

with wind^ do

begin

r.assign(2,n+4,70,n+20);

insert(new(pstatictext,init(r,' ')));

end;

for i:=n\_1 to n\_2 do

begin

for j:=m\_1 to m\_2 do

begin

str(mxx[i,j],sr);

with wind^ do

begin

r.assign(1+j\*4,1+i+n+6,4+j\*4,n+3+i+7);

insert(new(pstatictext,init(r,sr)));

end;

end;

end;

readkey;

end;

{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}

procedure PrintSubMatrixfile(n\_1,m\_1,n\_2,m\_2:integer); {*выводим в файл результ-щую матрицу*}

var

i,j,k,l:integer;

begin

for i:=n\_1 to n\_2 do

begin

for j:=m\_1 to m\_2 do

write(fileresu,mxx[i,j]:3,' ');

writeln(fileresu);

end;

writeln(fileresu);

writeln(fileresu);

end;

{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}

function GeomProg:boolean; {*функция вычисления геометрической прогрессии*}

var

i:integer;

dv:real;

begin

Sort; {*сортируем по возрастанию*}

GeomProg:=true; *{образует геометрическую прогрессию}*

dv:=per[2]/per[1];

for i:= 2 to l-1 do

if per[i+1]/per[i]<>dv then

begin

GeomProg:=false; *{не образует геометрическую прогрессию}*

break;

end;

end;

{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}

procedure tmatrix.Main(f:boolean); {*вывод результата*}

var

Sn,Sm,Snn,Smm:integer; {*угловые счётчики периметра*}

begin

Sn:=2;

Sm:=2;

Snn:=1;

Smm:=1;

while (Sn<>N) or (Sm<>M) do {*перебираем подматрицы*}

begin

GetPerimetr(Snn,Smm,(Snn+Sn)-1,(Smm+Sm)-1); {*определение эл-ов по периметру подматриц*}

if f then {*по условию*}

begin

if GeomProg then {*геометрическая прогрессия*}

if fil then {*в файл*}

PrintSubMatrixfile(Snn,Smm,(Snn+Sn)-1,(Smm+Sm)-1)

else {*в окно*}

PrintSubMatrix(Snn,Smm,(Snn+Sn)-1,(Smm+Sm)-1);

end

else {*все подматрицы*}

if fil then {*в файл*}

PrintSubMatrixfile(Snn,Smm,(Snn+Sn)-1,(Smm+Sm)-1)

else {*в окно*}

PrintSubMatrix(Snn,Smm,(Snn+Sn)-1,(Smm+Sm)-1);

if (Smm+Sm)<=M then

inc(Smm)

else

begin

if (Snn+Sn)<=N then

begin

inc(Snn);

Smm:=1;

end

else

begin

if Sm = M then

begin

Sm:=2;

inc(Sn);

snn:=1;

smm:=1;

end

else

begin

snn:=1;

smm:=1;

inc(sm)

end;

end;

end;

end;

GetPerimetr(1,1,N,M); {*определение элементов по периметру матрицы*}

if f then {*по условию*}

begin

if GeomProg then {*геометрическая прогрессия*}

if fil then {*в файл*}

PrintSubMatrixfile(1,1,n,m)

else {*в окно*}

PrintSubMatrix(1,1,N,M);

end

else {*все подматрицы*}

if fil then {*в файл*}

PrintSubMatrixfile(1,1,n,m)

Else {*в окно*}

begin

PrintSubMatrix(1,1,N,M);

readkey;

end;

end;

{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}

procedure tmatrix.fileoutputv(f:boolean); {*окно записи в файл результатов*}

var

pf:pfiledialog;

s:pathstr;

x:char;

i,j:integer;

begin

new(pf,init('\*.txt','Выберите нужный файл:','Имя файла',fdopenbutton,0));

if desktop^.execview(pf)=stddlg.cmfileopen then

begin

pf^.getfilename(s);

assign(fileresu,s);

rewrite(fileresu); {*открываем файл для записи*}

fil:=true;

if f then {*по условию*}

tmatrix.main(true)

else {*все подматрицы*}

tmatrix.main(false);

dispose(pf,done);

close (fileresu); {*закрываем файл*}

end;

fil:=false;

end;

{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}

procedure tmatrix.handleevent(var event:tevent);

var

r:trect;

i:integer;

begin

if event.what=evcommand then

case event.command of

cmisfl:tmatrix.fileinput;

cmfilu:tmatrix.fileoutputv(true);

cmfilv: tmatrix.fileoutputv(false);

cmvvod:tmatrix.inputmatrix;

cmvval:tmatrix.getrandommatrix;

cmvhel:messagebox(#3'Вывод подматриц,'#13+#3'периметр которых -'#13+#3'геометрическая прогрессия'#13+ #3'Алексей 2010',nil,mfinformation or mfokbutton);

cmvvse:

begin

with wind^ do

begin

r.assign(2,n+3,70,n+5);

insert(new(pstatictext,init(r,'Все подматрицы:')));

end;

tmatrix.main(false);

end;

cmvusl:

begin

with wind^ do

begin

r.assign(2,n+3,70,n+5 );

insert(new(pstatictext,init(r,'Подматрицы, у которых периметр - геометрическая прогрессия: ')));

end;

tmatrix.main(true);

end;

cmQuit: if messagebox(#3'Завершить работу?',nil,mfconfirmation or mfokcancel)=cmcancel then clearevent(event);

else

exit

end;

inherited handleevent(event);

end;

{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}

end.

**6 Тестирование программы**

Объектом испытаний является разработанная программа. Целью испытаний является проверка соответствия программного продукта поставленным требованиям.

Для проведения испытаний данные вводились с клавиатуры. Размерность тестируемой матрицы 3 – строки, 4 – столбца. Исходная матрица (рисунок 6):

Рисунок 6 – Исходная рабочая матрица

На рисунке 7 представлен результат работы программы по условию (рисунок 2).

Рисунок 7 – Результат работы программы в текстовом виде

Также, проводилось тестирование с различными примерами, которые здесь не приведены, но которые также показали верный результат.

Таким образом, тестирование прошло успешно.

**7 Руководство пользователя**

Запуск программы осуществляется нажатием Enter на имени файла: A:\ matr.exe.

После запуска программы появится пользовательское меню, которое показано на рисунке 8.

Рисунок 8 – Окно программы

Это меню содержит:

1. Ввод: «Ручной», «Автоматический», «Из файла» и «Выход» (рисунок 9).

Рисунок 9

1. Результат: «Все подматрицы», «По условию», «Записать все» и «Записать по условию» (рисунок 10).

Рисунок 10

1. О программе (рисунок 11).

Рисунок 11

Перемещение по кнопкам осуществляется клавишами клавиатуры, а именно стрелками вверх ↑ , вниз ↓, вправо →, влево ←, либо мышкой.

Ввод данных можно осуществлять вводом вручную, автоматически и из файла.

При нажатии клавиши Enter по кнопке с надписью «Ручной», появится окно (рисунок 12):

Рисунок 12 – Ручной ввод

Необходимо будет ввести количество строк и столбцов матрицы. Затем нажимаем клавишу «ОК», появляется окно (рисунок 13) и вводим элементы матрицы.

Рисунок 13 – Ввод элемента матрицы

По окончанию ввода элементов матрицы результат появится в окне (рисунок 14).

Рисунок 14 – Рабочая матрица

При нажатии клавиши Enter по кнопке с надписью «Автоматический», появится окно (рисунок 15):

Рисунок 15 – Автоматический ввод

Необходимо будет ввести количество строк и столбцов матрицы, а также максимальное значение элемента. Затем нажимаем клавишу «ОК» и результат автоматического заполнения элементов матрицы появится в окне (рисунок 14).

При нажатии клавиши Enter по кнопке с надписью «Из файла», появится окно (рисунок 16):

Рисунок 16 – Из файла

Необходимо будет выбрать текстовый файл с матрицей. Затем нажимаем клавишу «Open» и результат появится в окне (рисунок 14).

После ввода данных можно приступать к обработке.

При нажатии клавиши Enter по кнопке с надписью «Все подматрицы», будут появляться все подматрицы рабочей матрицы в окне, нажимая любую кнопку клавиатуры (рисунок 17).

Рисунок 17 – Все подматрицы

При нажатии клавиши Enter по кнопке с надписью «По условию», будут появляться все подматрицы, по периметру которых расположены элементы, из которых можно образовать геометрическую прогрессию, нажимая любую кнопку клавиатуры (рисунок 18).

Рисунок 18 –Геометрическая прогрессия

При нажатии клавиши Enter по кнопке с надписью «Записать все», появится окно (рисунок 19), где необходимо ввести имя файла и место его сохранения на диске. В результате все подматрицы запишутся в файл.

Рисунок 19 – Запись в файл

При нажатии клавиши Enter по кнопке с надписью «Записать по условию», появится окно (рисунок 19), где необходимо ввести имя файла и место его сохранения на диске. В результате все подматрицы по условию запишутся в файл.

При нажатии клавиши Enter по кнопке с надписью «Выход», вы выйдите из программы.

**Заключение**

В результате выполнения курсовой работы был разработан программный продукт, который позволит выявить в прямоугольной матрице все подматрицы, по периметру которых расположены элементы, из которых можно образовать геометрическую прогрессию.

В ходе выполнения работы разработано руководство пользователя и составлена пояснительная записка. Пояснительная записка оформлена в соответствии с действующим стандартом академии СТП.101.-2002.

Разработанный программный продукт отвечает требованиям задания, что подтверждается результатами тестирования.

Программа работает в любой операционной системе Windows и не приводит к зависанию компьютера и работает без сбоев.

**Список использованных источников**

1 Фаронов В.В. Турбо Паскаль. Начальный курс. Учебное пособие – М.: “Нолидж”, 2000.