УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №6.2

по предмету «Основы алгоритмизации и программирования»

Вариант 11

Выполнил:

Егоров А.С.

Гр. 351005

Проверил:

Данилова Г. В.

Минск 2024

**Задание:**

Комбинаторика. Построить магический квадрат нечётной степени

методом террас. Визуализировать!

**Код программы Delphi:**

Unit MainUnit;

Interface

Uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Grids, Vcl.Buttons,

Backend, AboutTheDeveloperUnit6\_2, ExitUnit6\_2, InstructionUnit6\_2,

Vcl.Menus;

Type

TPartOfSquare = (PSLeft, PSTop, PSRight, PSBottom, PSNone);

TuVCLMain = Class(TForm)

LbInfo: TLabel;

ESize: TEdit;

LbSizeInfo: TLabel;

StrGrSquare: TStringGrid;

BitBtnAccept: TBitBtn;

BitBtnNext: TBitBtn;

MainMenu1: TMainMenu;

BtFile: TMenuItem;

BtOpenFile: TMenuItem;

BtSaveFile: TMenuItem;

BtInstruction: TMenuItem;

BtAboutTheDeveloper: TMenuItem;

SaveDialog1: TSaveDialog;

OpenDialog1: TOpenDialog;

Procedure ESizeKeyPress(Sender: TObject; Var Key: Char);

Procedure ESizeChange(Sender: TObject);

Procedure FormCreate(Sender: TObject);

Procedure BitBtnAcceptClick(Sender: TObject);

Procedure BitBtnNextClick(Sender: TObject);

Procedure BtAboutTheDeveloperClick(Sender: TObject);

Procedure BtInstructionClick(Sender: TObject);

Procedure FormCloseQuery(Sender: TObject; Var CanClose: Boolean);

procedure btOpenFileClick(Sender: TObject);

procedure btSaveFileClick(Sender: TObject);

Private

BufferHandler: TBufferHandler;

WasChanges: Boolean;

Size: Integer;

HighSize: Integer;

PartOfSquare: TPartOfSquare;

Offset: Integer;

WasSaved: Boolean;

Procedure PrintLeftPartOfSquare();

Procedure PrintTopPartOfSquare();

Procedure PrintRightPartOfSquare();

Procedure PrintBottomPartOfSssquare();

Public

{ Public declarations }

End;

Var

UVCLMain: TuVCLMain;

Implementation

{$R \*.dfm}

Procedure TuVCLMain.BitBtnAcceptClick(Sender: TObject);

Var

I, J, Step1, Step2: Integer;

Number: Integer;

Begin

// выставление нужных нам параметров

Size := StrToInt(ESize.Text);

Offset := Size Div 2;

HighSize := Size \* 2 - 1;

// создание квадрата

StrGrSquare.ColCount := HighSize;

StrGrSquare.RowCount := HighSize;

StrGrSquare.Left := (UVCLMain.Width - StrGrSquare.DefaultColWidth \*

StrGrSquare.ColCount) Div 2;

StrGrSquare.Top := (UVCLMain.Height - StrGrSquare.DefaultRowHeight \*

StrGrSquare.RowCount) Div 2;

StrGrSquare.Height := (StrGrSquare.DefaultRowHeight + 2) \*

StrGrSquare.ColCount;

StrGrSquare.Width := (StrGrSquare.DefaultColWidth + 2) \*

StrGrSquare.RowCount;

// показываем

StrGrSquare.Visible := True;

StrGrSquare.Enabled := True;

J := 0;

Number := 1;

For Step1 := Size To HighSize Do

Begin

I := Step1 - 1;

For Step2 := 1 To Size Do

Begin

StrGrSquare.Cells[J, I] := IntToStr(Number);

Inc(Number);

Inc(J);

Dec(I);

End;

Dec(J, Size - 1);

End;

// и включаем кнопку следующий шаг

BitBtnNext.Visible := True;

BitBtnNext.Enabled := True;

End;

Procedure TuVCLMain.BitBtnNextClick(Sender: TObject);

Begin

Case PartOfSquare Of

PSLeft:

PrintLeftPartOfSquare();

PSTop:

PrintTopPartOfSquare();

PSRight:

PrintRightPartOfSquare();

PSBottom:

PrintBottomPartOfSssquare();

End;

If PartOfSquare < PSNone Then

PartOfSquare := Succ(PartOfSquare);

If PartOfSquare = PSNone Then

Begin

BitBtnNext.Visible := False;

BitBtnNext.Enabled := False;

BtSaveFile.Enabled := True;

End;

End;

procedure TuVCLMain.btOpenFileClick(Sender: TObject);

Var

FileReader : TFileReader;

begin

If OpenDialog1.Execute() Then

Begin

FileReader := TFileReader.Create();

FileReader.FileName := OpenDialog1.FileName;

FileReader.CheckFile();

// TODO статусы

If FileReader.Status = FsGood Then

Begin

Size := FileReader.ReadSize();

If (FileReader.Status = FsGood) Then

Begin

ESize.Text := IntToStr(Size);

End

Else

MessageBox(UVCLMain.Handle, ListOfMessages[FileReader.Status],

'Ой-йой', MB\_ICONERROR);

End

Else

MessageBox(UVCLMain.Handle, ListOfMessages[FileReader.Status],

'Ой-йой', MB\_ICONERROR);

End;

end;

procedure TuVCLMain.btSaveFileClick(Sender: TObject);

Var

FileWriter: TFileWriter;

Matrix : TMatrix;

I,J: Integer;

Begin

If SaveDialog1.Execute() Then

Begin

FileWriter := TFileWriter.Create();

FileWriter.FileName := SaveDialog1.FileName;

FileWriter.CheckFile();

If FileWriter.Status = FsGood Then

Begin

// reading matrix

SetLength(Matrix,Size,Size);

for I := 0 to High(Matrix) do

for J := 0 to High(Matrix[0]) do

Matrix[I][J] := StrToInt(StrGrSquare.Cells[J + offset,I + offset]);

// writing matrix

FileWriter.WriteList(Matrix);

If FileWriter.Status <> FsGood Then

Begin

MessageBox(UVCLMain.Handle, ListOfMessages[FileWriter.Status],

'Ой-йой', MB\_ICONERROR);

End

Else

WasSaved := True;

End

Else

MessageBox(UVCLMain.Handle, ListOfMessages[FileWriter.Status],

'Ой-йой', MB\_ICONERROR);

FileWriter.Destroy();

End;

end;

Procedure TuVCLMain.FormCreate(Sender: TObject);

Begin

BufferHandler := TBufferHandler.Create();

End;

Procedure TuVCLMain.PrintBottomPartOfSssquare;

Var

I, J, K: Integer;

Begin

K := Offset + 1;

While K < Offset + Size - 1 Do

Begin

I := HighSize - Offset;

J := K;

While (I < HighSize) And (StrGrSquare.Cells[J, I] <> '') Do

Begin

StrGrSquare.Cells[J, I - Size] := StrGrSquare.Cells[J, I];

StrGrSquare.Cells[J, I] := '';

Inc(J);

Inc(I);

End;

Inc(K, 2);

End;

End;

Procedure TuVCLMain.PrintLeftPartOfSquare;

Var

I, J, K: Integer;

Begin

K := Offset + 1;

While K < Offset + Size - 1 Do

Begin

I := K;

J := Offset - 1;

While (J > -1) And (StrGrSquare.Cells[J, I] <> '') Do

Begin

StrGrSquare.Cells[J + Size, I] := StrGrSquare.Cells[J, I];

StrGrSquare.Cells[J, I] := '';

Dec(J);

Inc(I);

End;

Inc(K, 2)

End;

End;

Procedure TuVCLMain.PrintRightPartOfSquare;

Var

I, J, K: Integer;

Begin

K := Offset + 1;

While K < Offset + Size - 1 Do

Begin

I := K;

J := HighSize - Offset;

While (J < HighSize) And (StrGrSquare.Cells[J, I] <> '') Do

Begin

StrGrSquare.Cells[J - Size, I] := StrGrSquare.Cells[J, I];

StrGrSquare.Cells[J, I] := '';

Inc(I);

Inc(J);

End;

Inc(K, 2);

End;

End;

Procedure TuVCLMain.PrintTopPartOfSquare;

Var

I, J, K: Integer;

Begin

K := Offset + 1;

While K < Offset + Size - 1 Do

Begin

I := Offset - 1;

J := K;

While (I > -1) And (StrGrSquare.Cells[J, I] <> '') Do

Begin

StrGrSquare.Cells[J, I + Size] := StrGrSquare.Cells[J, I];

StrGrSquare.Cells[J, I] := '';

Dec(I);

Inc(J);

End;

Inc(K, 2);

End;

End;

End.

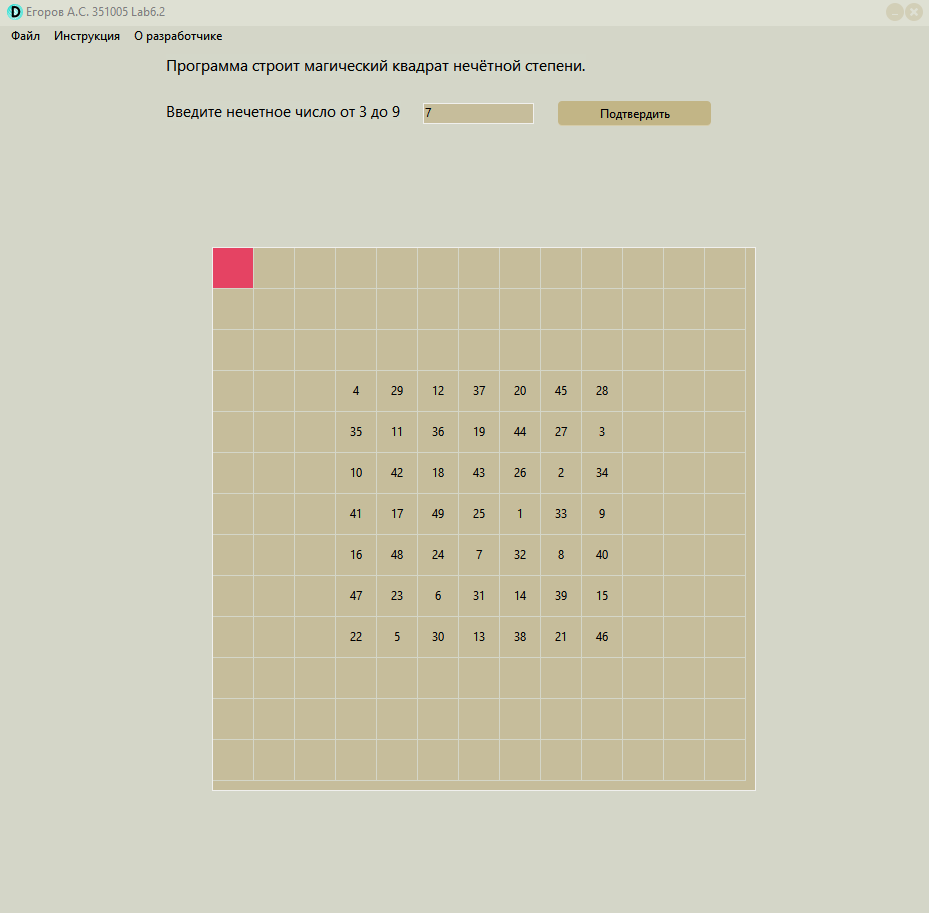
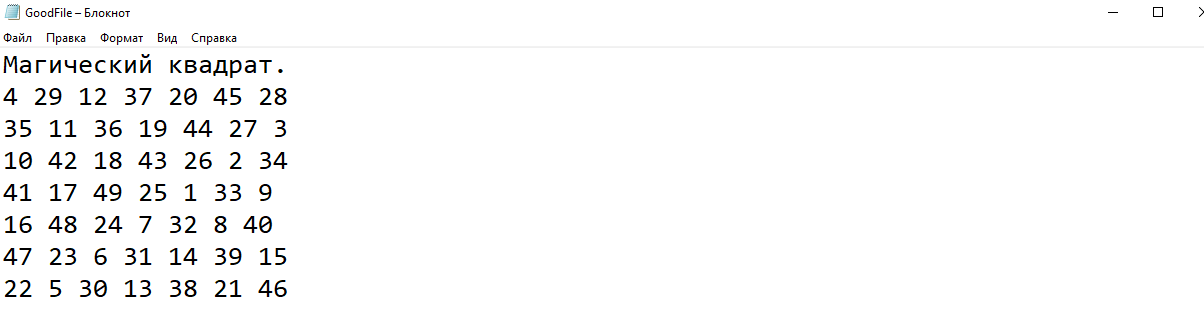
**Код программы CSharp:**

**MSquareBuilder.cs**

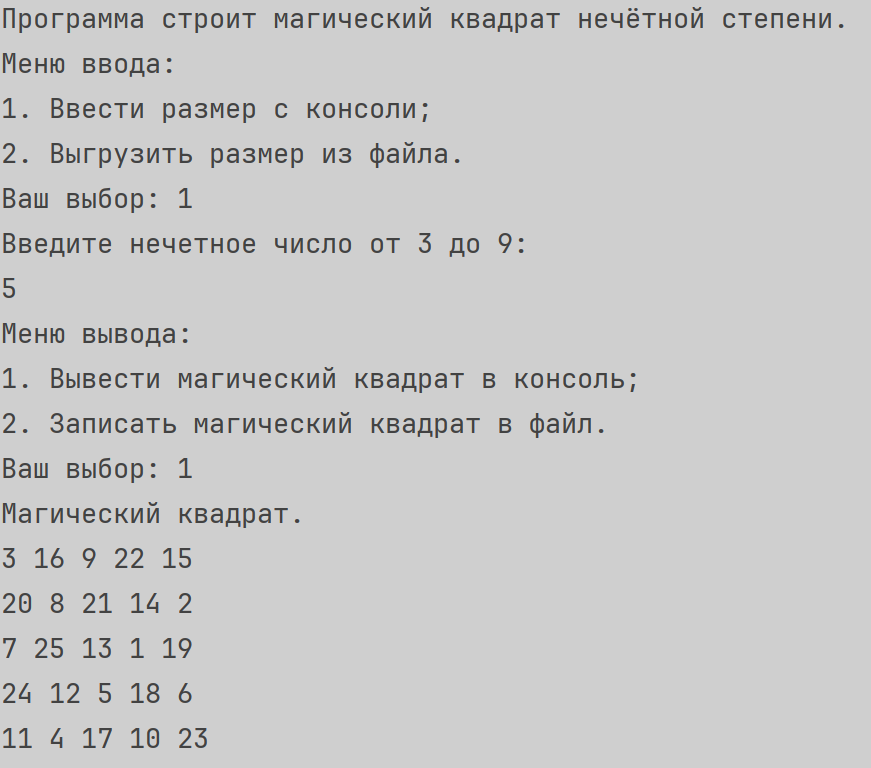
namespace Lab6\_2;  
  
public class MSquareBuilder  
{  
 private int offset;  
  
 private int highSize;  
   
 private int size;  
   
 private int[,] helpMatrix;  
  
 private int[,] matrix;  
   
 public int[,] Matrix  
 {  
 get  
 {  
 for (int i = 0; i < size; i++)  
 {  
 for (int j = 0; j < size; j++)  
 {  
 matrix[i, j] = helpMatrix[i + offset, j + offset];  
 }   
 }  
  
 return matrix;  
 }  
 }  
   
 public MSquareBuilder(int size)  
 {  
 this.highSize = size \* 2 - 1;  
 this.size = size;  
 this.offset = this.size / 2;  
 helpMatrix = new int[highSize,highSize];  
 matrix = new int[size, size];  
 }  
  
 public void BuildHugeMatrix()  
 {  
 int j = 0;  
 int i = 0;  
 offset = size / 2;  
   
 // fill matrix with zeros  
 for (i = 0; i < highSize; i++)  
 for (j = 0; j < highSize; j++)  
 helpMatrix[i, j] = 0;  
 // create matrix with diagonal elements  
 i = 0;  
 j = 0;  
 int number = 1;  
 for (int step1 = size - 1; step1 < highSize; step1++)  
 {  
 i = step1;  
 for (int step2 = 0; step2 < size; step2++)  
 helpMatrix[i--, j++] = number++;  
 j = j - size + 1;   
 }  
   
 }  
  
 public void MoveLeftPart()  
 {  
 int i, j, k;  
 k = offset + 1;  
 while (k < offset + size - 1)  
 {  
 i = k;  
 j = offset - 1;  
 while (j > -1 && helpMatrix[i,j] != 0)  
 {  
 helpMatrix[i,j + size] = helpMatrix[i, j];  
 helpMatrix[i++,j--] = 0;  
 }  
 k += 2;  
 }  
 }  
   
 public void MoveTopPart()  
 {  
 int i, j, k;-  
 k = offset + 1;  
 while (k < offset + size - 1)  
 {  
 i = offset - 1;  
 j = k;  
 while (i > -1 && helpMatrix[i,j] != 0)  
 {  
 helpMatrix[i + size, j] = helpMatrix[i, j];  
 helpMatrix[i--, j++] = 0;  
 }  
 k += 2;  
 }  
 }  
   
 public void MoveRightPart()  
 {  
 int i, j, k;  
 k = offset + 1;  
 while (k < offset + size - 1)  
 {  
 i = k;  
 j = highSize - offset;  
 while ((j < highSize) && helpMatrix[i,j] != 0)  
 {  
 helpMatrix[i, j - size] = helpMatrix[i, j];  
 helpMatrix[i++, j++] = 0;  
 }  
 k += 2;  
 }   
 }  
   
 public void MoveBottomPart()  
 {  
 int i, j, k;  
 k = offset + 1;  
 while (k < offset + size - 1)  
 {  
 i = highSize - offset;  
 j = k;  
 while (i < highSize && helpMatrix[i,j] != 0)  
 {  
 helpMatrix[i - size, j] = helpMatrix[i, j];  
 helpMatrix[i++, j++] = 0;  
 }  
 k += 2;  
 }   
 }  
   
}

**Скриншоты:**

**Delphi:**



**CSharp:**



**Блок Схема:**

