УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №4.1

по предмету

Основы алгоритмизации и программирования

Вариант 14

Выполнил:

Крутько А.А.

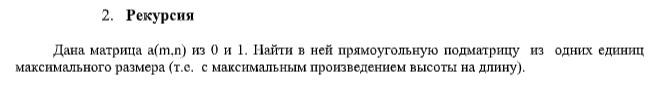
Проверила:

Данилова Г.В.

Группа 251004

Минск 2022

Задание:



Код программы на **Delphi**:

unit Unit1;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

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

Vcl.Menus;

type

TForm1 = class(TForm)

MainLabel: TLabel;

RowEdit: TEdit;

RowLabel: TLabel;

ColEdit: TEdit;

ColLabel: TLabel;

SetRowsAndColsButton: TBitBtn;

MatrixGrid: TStringGrid;

PopupMenu: TPopupMenu;

FindAnswerButton: TBitBtn;

AnswerLabel: TLabel;

NumberLabel: TLabel;

MainMenu1: TMainMenu;

N1: TMenuItem;

Help: TMenuItem;

Programmer: TMenuItem;

OpenFile: TMenuItem;

SaveFile: TMenuItem;

OpenDialog1: TOpenDialog;

SaveDialog1: TSaveDialog;

procedure FormCreate(Sender: TObject);

procedure EditChange(Sender: TObject);

procedure SetRowsAndColsButtonClick(Sender: TObject);

procedure EditKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

procedure EditKeyPress(Sender: TObject; var Key: Char);

procedure FindAnswerButtonClick(Sender: TObject);

procedure MatrixGridKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

procedure MatrixGridKeyUp(Sender: TObject; var Key: Word;

Shift: TShiftState);

procedure MatrixGridSetEditText(Sender: TObject; ACol, ARow: Integer;

const Value: string);

procedure MatrixGridKeyPress(Sender: TObject; var Key: Char);

procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);

procedure HelpClick(Sender: TObject);

procedure ProgrammerClick(Sender: TObject);

procedure OpenFileClick(Sender: TObject);

procedure SaveFileClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

Type

TArrOIOI = Array of Array of Integer;

var

Form1: TForm1;

IsEditing: Boolean;

Count: Integer;

implementation

{$R \*.dfm}

procedure TForm1.EditKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

TEdit(Sender).ReadOnly := (((Shift=[ssShift]) and (Key = VK\_INSERT)) or (Shift=[ssCtrl]) or (Shift=[ssAlt]));

end;

procedure TForm1.EditChange(Sender: TObject);

begin

SetRowsAndColsButton.Enabled := (RowEdit.GetTextLen > 0) and (ColEdit.GetTextLen > 0);

IsEditing := True;

Form1.Height := 150;

Form1.Width := 750;

NumberLabel.Visible := False;

AnswerLabel.Visible := False;

SaveFile.Enabled := False;

end;

Procedure NewPoint(Matrix: TArrOIOI; Var Row: Integer; Var Col: Integer);

Begin

Inc(Col);

If Col > High(Matrix[0]) Then

Begin

Col := Col Mod Length(Matrix[0]);

Inc(Row);

End;

End;

Function GetSumFromThisPointInWidth(Matrix: TArrOIOI; Row, Col: Integer): Integer;

Var

Sum: Integer;

NewRow, NewCol: Integer;

Temp: Integer;

Begin

NewRow := Row;

NewCol := Col;

Temp := Length(Matrix[0]);

Sum := 0;

While (NewRow < Length(Matrix))and (Matrix[NewRow, Col] = 1) do

Begin

NewCol := Col;

While (NewCol < Length(Matrix[0])) and (Matrix[NewRow, NewCol] = 1) do

Inc(NewCol);

If Temp > NewCol - Col Then

Temp := NewCol - Col;

Inc(NewRow);

Sum := (NewRow - Row) \* Temp;

If Sum > Count Then

Count := Sum;

End;

GetSumFromThisPointInWidth := Count;

End;

Function FindSubMatrix(Matrix: TArrOIOI; Var Row: Integer; Var Col: Integer): Integer;

Var

SumWidth: Integer;

Count: Integer;

Begin

Count := 0;

SumWidth := 0;

Repeat

If (Row = High(Matrix)) and (Col = High(Matrix[0])) Then

Begin

Count := Matrix[High(Matrix), High(Matrix[0])];

FindSubMatrix := Count;

End

Else

If Matrix[Row,Col] = 0 Then

Begin

Newpoint(Matrix, Row,Col);

SumWidth := FindSubMatrix(Matrix,Row,Col);

End

Else

Begin

SumWidth := GetSumFromThisPointInWidth(Matrix, Row, Col);

NewPoint(Matrix, Row, Col);

End;

If SumWidth > Count Then

Count := SumWidth;

Until (Row = High(Matrix)) and (Col = High(Matrix[0]));

FindSubMatrix := Count;

End;

Function GetMatrix(): TArrOIOI;

var

Matrix: TArrOIOI;

I, J: Integer;

Begin

SetLength(Matrix, Form1.MatrixGrid.RowCount, Form1.MatrixGrid.ColCount);

For I := 0 to High(Matrix) do

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

Matrix[I,J] := StrToInt(Form1.MatrixGrid.Cells[J, I]);

GetMatrix := Matrix;

End;

Function IsRight: Boolean;

Var

I, J: Integer;

IsCorrect: Boolean;

Begin

IsCorrect := True;

I := 0;

While (I < Form1.MatrixGrid.RowCount) and IsCorrect do

Begin

J := 0;

While (J < Form1.MatrixGrid.ColCount) and IsCorrect do

Begin

Try

StrToInt(Form1.MatrixGrid.Cells[J, I]);

Except

IsCorrect := False;

End;

Inc(J);

End;

Inc(I);

End;

IsRight := IsCorrect;

End;

procedure TForm1.FindAnswerButtonClick(Sender: TObject);

Var

Matrix: TArrOIOI;

Row, Col: Integer;

begin

If IsRight Then

Begin

NumberLabel.Visible := True;

AnswerLabel.Visible := True;

Matrix := GetMatrix();

Row := 0;

Col := 0;

NumberLabel.Caption := IntToStr(FindSubMatrix(Matrix, Row, Col));

SaveFile.Enabled := True;

End

Else

MessageBox(Form1.Handle, PChar('Неправильные данные в матрице'), 'Ошибка',   
 MB\_ICONSTOP);

end;

procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose: Boolean);

begin

CanClose := MessageBox(Application.Handle,'Вы действительно хотите выйти из программы?','Подтверждение',MB\_YESNO)=ID\_YES;

end;

procedure TForm1.FormCreate(Sender: TObject);

begin

BorderIcons := BorderIcons - [biMaximize] + [biHelp];

BorderStyle := BsSingle;

Position := PoDesktopCenter;

MatrixGrid.Visible := False;

IsEditing := True;

Form1.Height := 150;

Form1.Width := 750;

Count := 0;

SaveFile.Enabled := False;

end;

procedure TForm1.MatrixGridKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

If (((Shift=[ssShift]) and (Key = VK\_INSERT)) or (Shift=[ssCtrl]) or   
 (Shift=[ssAlt])) Then

MatrixGrid.Options := MatrixGrid.Options - [goEditing];

end;

procedure TForm1.MatrixGridKeyUp(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

If (((Shift=[ssShift]) and (Key = VK\_INSERT)) or (Shift=[ssCtrl]) or (Shift=[ssAlt])) Then

MatrixGrid.Options := MatrixGrid.Options + [goEditing];

end;

procedure TForm1.MatrixGridSetEditText(Sender: TObject; ACol, ARow: Integer;

const Value: string);

Var

I, J: Integer;

IsCorrect: Boolean;

begin

I := 0;

IsCorrect := True;

While (I < MatrixGrid.RowCount) and IsCorrect do

Begin

J := 0;

While (J < MatrixGrid.ColCount) and IsCorrect do

Begin

If Length(MatrixGrid.Cells[J, I]) = 0 Then

IsCorrect := False;

Inc(J);

End;

Inc(I);

End;

SaveFile.Enabled := False;

AnswerLabel.Visible := False;

NumberLabel.Visible := False;

FindAnswerButton.Enabled := IsCorrect;

end;

procedure TForm1.HelpClick(Sender: TObject);

begin

ShowMessage('1. Размер Строк/Столбцов от 1 до 12'#13#10'2.В Матрице только   
 вводить нули и единицы');

end;

procedure TForm1.ProgrammerClick(Sender: TObject);

begin

ShowMessage('Группа 251004, Крутько Андрей Алексеевич');

end;

Function GetSize(Var FileInput: TextFile; Temp: String): String;

Const

MIN\_NUM = 1;

MAX\_NUM = 12;

Var

Size: Integer;

Str: String;

IsCorrect: Boolean;

Begin

If Not Eof(FileInput) Then

Begin

Try

Readln(FileInput, Str);

Size := StrToInt(Str);

Except

MessageBox(Form1.Handle, PChar('Неправильные данные в файле для количества ' + Temp), 'Ошибка', MB\_ICONSTOP);

Str := '';

IsCorrect := True;

End;

If Not(IsCorrect) and ((Size < MIN\_NUM) or (Size > MAX\_NUM)) Then

Begin

MessageBox(Form1.Handle, PChar('Выход за диапазон данных в файле для количества ' + Temp), 'Ошибка', MB\_ICONSTOP);

Str := '';

End;

End

Else

MessageBox(Form1.Handle, PChar('Нет данных в файле для количества ' + Temp), 'Ошибка', MB\_ICONSTOP);

GetSize := Str;

End;

Function TakeInformationIntoCell(Var FileInput: TextFile; Var IsCorrect: Boolean): String;

Var

Temp: Integer;

Str: String;

Begin

If Not Eof(FileInput) Then

Begin

Try

Read(FileInput, Temp);

Str := IntToStr(Temp);

Except

MessageBox(Form1.Handle, PChar('Неправильно введены значения'), 'Ошибка', MB\_ICONSTOP);

IsCorrect := False;

End;

If (Temp > 1) or (Temp < 0) Then

Begin

MessageBox(Form1.Handle, PChar('Неправильно введены значения'), 'Ошибка', MB\_ICONSTOP);

IsCorrect := False;

Str := '';

End;

End

Else

Begin

IsCorrect := False;

MessageBox(Form1.Handle, PChar('Файл считался не до конца'), 'Ошибка', MB\_ICONSTOP);

Str := '';

End;

TakeInformationIntoCell := Str;

End;

Procedure InputAmountsInMatrix(Var FileInput: TextFile);

Var

I,J: Integer;

IsCorrect: Boolean;

Begin

IsCorrect := True;

With Form1 do

Begin

I := 0;

While (I < MatrixGrid.RowCount) and IsCorrect do

Begin

J := 0;

While (J < MatrixGrid.ColCount) and IsCorrect do

Begin

MatrixGrid.Cells[J, I] := TakeInformationIntoCell(FileInput, IsCorrect);

Inc(J);

End;

Inc(I);

End;

FindAnswerButton.Enabled := IsCorrect;

End;

End;

procedure TForm1.OpenFileClick(Sender: TObject);

Var

FileInput: TextFile;

begin

If OpenDialog1.Execute Then

Begin

ShowMessage('File: ' + OpenDialog1.FileName);

AssignFile(FileInput, OpenDialog1.FileName);

Try

Try

Reset(FileInput);

RowEdit.Text := GetSize(FileInput, 'строк');

ColEdit.Text := GetSize(FileInput, 'столбцов');

If (RowEdit.Text <> '') and (ColEdit.Text <> '') Then

Begin

SetRowsAndColsButton.Click;

InputAmountsInMatrix(FileInput);

End;

Finally

CloseFile(FileInput);

End;

Except

MessageBox(Form1.Handle, PChar('Файл закрыт для чтения'), 'Ошибка', MB\_ICONSTOP);

End;

End;

end;

procedure TForm1.SaveFileClick(Sender: TObject);

Var

FileOutput: TextFile;

begin

If SaveDialog1.Execute Then

Begin

AssignFile(FileOutput, SaveDialog1.FileName);

Try

Try

ReWrite(FileOutput);

Writeln(FileOutput, AnswerLabel.Caption + NumberLabel.Caption);

ShowMessage('Результат успешно занесен в файл');

Finally

CloseFile(FileOutput);

End;

Except

MessageBox(Form1.Handle, PChar('Файл закрыт для Записи'), 'Ошибка', MB\_ICONSTOP);

End;

End;

end;

Procedure ClearMatrix();

var

I: Integer;

Begin

For I := 0 to Form1.MatrixGrid.RowCount - 1 do

Form1.MatrixGrid.Rows[I].Clear;

Form1.FindAnswerButton.Enabled := False;

End;

procedure TForm1.SetRowsAndColsButtonClick(Sender: TObject);

begin

ClearMatrix();

SetRowsAndColsButton.Enabled := False;

MatrixGrid.RowCount := StrToInt(RowEdit.Text);

MatrixGrid.ColCount := StrToInt(ColEdit.Text);

MatrixGrid.Height := (MatrixGrid.DefaultRowHeight + 3) \* MatrixGrid.RowCount;

MatrixGrid.Width := (MatrixGrid.DefaultColWidth + 3) \* MatrixGrid.ColCount;

If IsEditing Then

Begin

If MatrixGrid.ColCount > 9 Then

Form1.Width := Form1.Width + (MatrixGrid.DefaultColWidth + 3) \*   
 (MatrixGrid.ColCount - 9) + 5;

Form1.Height := Form1.Height + (MatrixGrid.DefaultRowHeight + 5) \*   
 MatrixGrid.RowCount;

If MatrixGrid.RowCount = 1 Then

Form1.Height := Form1.Height + 26;

FindAnswerButton.Left := MatrixGrid.Left + MatrixGrid.ColCount \*   
 (MatrixGrid.DefaultColWidth + 3);

FindAnswerButton.Top := MatrixGrid.Top;

AnswerLabel.Left := MatrixGrid.Left + MatrixGrid.ColCount \*   
 (MatrixGrid.DefaultColWidth + 3);

AnswerLabel.Top := MatrixGrid.Top + 38;

NumberLabel.Left := MatrixGrid.Left + MatrixGrid.ColCount \*   
 (MatrixGrid.DefaultColWidth + 3) + 55;

NumberLabel.Top := MatrixGrid.Top + 38;

End;

MatrixGrid.Visible := True;

IsEditing := False;

end;

end.

Код программы на **C#**:

void MainInfo()  
{  
 Console.WriteLine("Программа находит матрицу максимальной площади   
 квадратичной/прямоугольной формы, в прямоугольной матрице, состоящей из   
 нулей и единиц");  
 Console.WriteLine("Выберите способ ввода: \"C\" для консоли, \"F\" для файла");  
}  
  
sbyte GetSize()  
{  
 sbyte size = 0;  
 bool isInCorrect;  
 do  
 {  
 isInCorrect = false;  
 try  
 {  
 size = Convert.ToSByte(Console.ReadLine()); //обернуть в checked  
 }  
 catch (Exception)  
 {  
 Console.Error.WriteLine("Некорректный ввод");  
 isInCorrect = true;  
 }  
 if (!isInCorrect && (size < 1 || size > 15))  
 {  
 isInCorrect = true;  
 Console.WriteLine("Неправильный диапазон данных");  
 }   
 } while (isInCorrect);  
 return size;  
}  
  
sbyte[,] GetMatrixFromConsole()  
{  
 Console.Write("Введите размер от 1 до 15 по строкам: ");  
 sbyte row = GetSize();  
 Console.Write("Введите размер от 1 до 15 по столбцам: ");  
 sbyte col = GetSize();  
 sbyte[,] matrix = new sbyte[row, col];  
 for (int i = 0; i < matrix.GetLength(0); i++)  
 {  
 Console.Write($"Введите {i + 1} строку из нулей и единиц: ");  
 bool isCorrect;  
 do  
 {  
 isCorrect = false;  
 string? str = Console.ReadLine();  
 string[] strArr = str.Split(' ');  
 if (strArr.Length != matrix.GetLength(1))  
 {  
 isCorrect = true;  
 Console.WriteLine("Неправильный размер символов для строк   
 матрицы");  
 }  
 for (int j = 0; j < strArr.Length && !isCorrect; j++)  
 {  
 try  
 {  
 if (strArr[j].Equals("0") || strArr[j].Equals("1"))  
 matrix[i, j] = Convert.ToSByte(strArr[j]);  
 else  
 isCorrect = true;  
 }  
 catch (Exception e)  
 {  
 isCorrect = true;  
 Console.WriteLine(e.Message);  
 }  
 }  
 } while (isCorrect);  
 }  
  
 return matrix;  
}  
  
ConsoleKeyInfo GetTheKey()  
{  
 ConsoleKeyInfo key;  
 do  
 {  
 key = Console.ReadKey();  
 } while (key.Key != ConsoleKey.C && key.Key != ConsoleKey.F);  
 return key;  
}  
  
void NewPoint(ref sbyte row, ref sbyte col, in sbyte[,] matrix)  
{  
 col++;  
 if (col > matrix.GetUpperBound(1))  
 {  
 col = Convert.ToSByte(col % matrix.GetLength(1));  
 row++;  
 }  
}  
  
int GetSumFromThisPointInWidth(sbyte row, sbyte col, sbyte[,] matrix)  
{  
 int sum = 0, newRow = row, temp = matrix.GetLength(1), count = 0;  
 while (newRow < matrix.GetLength(0) && matrix[newRow, col] == 1)  
 {  
 int newCol = col;  
 while (newCol < matrix.GetLength(1) && matrix[newRow, newCol] == 1)  
 newCol++;  
 if (temp > newCol - col)  
 temp = newCol - col;  
 newRow++;  
 sum = (newRow - row) \* temp;  
 if (sum > count)  
 count = sum;  
 }  
 return count;  
}  
  
int FindSubMatrix(ref sbyte row, ref sbyte col, in sbyte[,] matrix)  
{  
 int sumWidth = 0, count = 0;  
 do  
 {  
 if ((row == matrix.GetUpperBound(0)) && (col ==   
 matrix.GetUpperBound(1)))  
 return matrix[matrix.GetUpperBound(0), matrix.GetUpperBound(1)];  
 else if (matrix[row, col] == 0)  
 {  
 NewPoint(ref row, ref col, matrix);  
 sumWidth = FindSubMatrix(ref row, ref col, matrix);  
 }  
 else  
 {  
 sumWidth = GetSumFromThisPointInWidth(row, col, matrix);  
 NewPoint(ref row, ref col, matrix);  
 }  
  
 if (sumWidth > count)  
 count = sumWidth;  
 } while (row != matrix.GetUpperBound(0) || col !=   
 matrix.GetUpperBound(1));  
 return count;  
}  
  
void WriteMatrixInConsole(in sbyte[,] matrix)  
{  
 sbyte row = 0, col = 0;  
 Console.WriteLine($"Максимальная площадь = {FindSubMatrix(ref row, ref   
 col, matrix)}");  
}  
  
int GetSizeFile(StreamReader st, ref bool isCorrect)  
{  
 string? str;  
 int size = 0;  
 if (!isCorrect && (str = st.ReadLine()) != null)  
 {  
 if (!int.TryParse(str, out size) && (size > 15 || size < 1))  
 isCorrect = true;  
 }  
 else  
 isCorrect = true;  
 return size;  
}  
sbyte[,] GetMatrixFromFile()  
{  
 var matrix = new sbyte[0,0];  
 bool isCorrect;  
 string? str;  
 int i = 0;  
 do  
 {  
 isCorrect = false;  
 try  
 {  
 using (StreamReader st = new StreamReader(TakeFilePath()))  
 {  
 int row = GetSizeFile(st, ref isCorrect);  
 int col = GetSizeFile(st, ref isCorrect);  
 if (!isCorrect)  
 matrix = new sbyte[row, col];  
 while (!isCorrect && i < matrix.GetLength(0) && (str =   
 st.ReadLine()) != null)  
 {  
 string[] strTemp = str.Split(' ');  
 if (strTemp.Length != matrix.GetLength(1))  
 {  
 isCorrect = true;  
 Console.WriteLine($"Проверьте ваши данные в {i + 1}   
 строке");  
 }  
 for (int j = 0; j < strTemp.Length && !isCorrect; j++)  
 {  
 try  
 {  
 if (strTemp[j].Equals("0") ||   
 strTemp[j].Equals("1"))  
 matrix[i, j] = Convert.ToSByte(strTemp[j]);  
 else  
 isCorrect = true;  
 }  
 catch (Exception e)  
 {  
 isCorrect = true;  
 Console.WriteLine(e.Message);  
 }  
 }  
 i++;  
 }  
 }  
 }  
 catch (Exception e)  
 {  
 isCorrect = true;  
 Console.WriteLine(e.Message);  
 }  
 } while (isCorrect);  
 return matrix;  
}  
  
void WriteMatrixInFile(in sbyte[,] matrix)  
{  
 sbyte row = 0, col = 0;  
 try  
 {  
 using (StreamWriter sw = new StreamWriter(TakeFilePath(), false))  
 sw.WriteLine($"Максимальная площадь = {FindSubMatrix(ref row, ref   
 col, matrix)}");  
 }  
 catch (Exception e)  
 {  
 Console.WriteLine(e.Message);  
 }  
}  
  
string? TakeFilePath()  
{  
 string? path;  
 bool isIncorrect;  
 do {  
 isIncorrect = false;  
 Console.WriteLine("Введите путь к файлу");  
 path = Console.ReadLine();  
 if (!File.Exists(path) || !path.EndsWith(".txt"))  
 {  
 isIncorrect = true;  
 Console.WriteLine("Проверьте параметры файла");  
 }  
 } while (isIncorrect);  
 return path;  
}

void ShowMatrix(in sbyte[,] matrix)

{

for (int i = 0; i < matrix.GetLength(0); i++)

{

for(int j = 0; j < matrix.GetLength(1); j++)

Console.Write(matrix[i,j] + "\t");

Console.WriteLine();

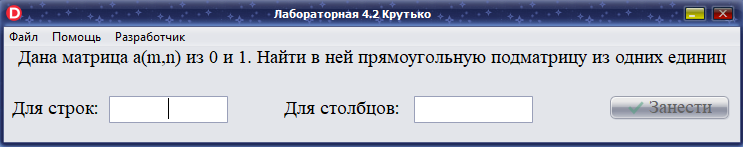
}

}

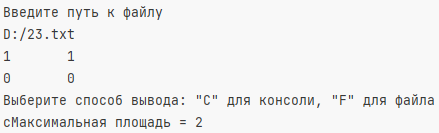
MainInfo();  
ConsoleKeyInfo key = GetTheKey();  
Console.Clear();  
sbyte[,] matrix = (key.Key == ConsoleKey.C) ? GetMatrixFromConsole() :   
 GetMatrixFromFile

ShowMatrix(matrix);  
Console.WriteLine("Выберите способ вывода: \"C\" для консоли, \"F\" для   
 файла");  
key = GetTheKey();  
if (key.Key == ConsoleKey.C)  
 WriteMatrixInConsole(matrix);  
else  
 WriteMatrixInFile(matrix);

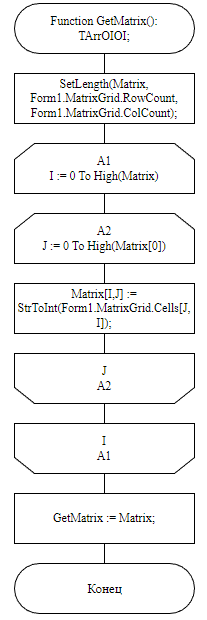
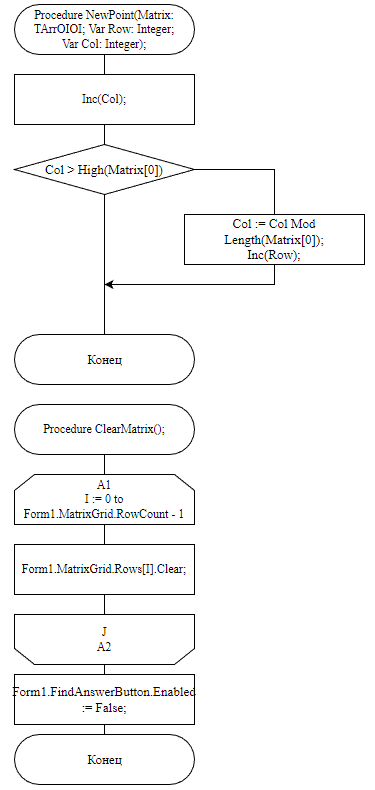
Результат на **Delphi**:

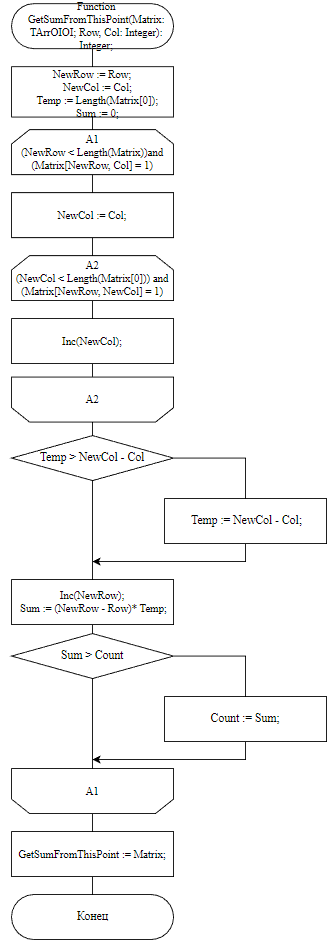


Результат на **C#**:



**Блок-cхема**:

****

****

