УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №6.2

по предмету

Основы алгоритмизации и программирования

Вариант 1

Выполнил:

Бетеня К.С.

Проверила:

Данилова Г.В.

Группа 351005

Минск 2024

Задание:

Комбинаторика. Дана матрица a(m,n). Найти в ней путь от элемента a[i1,j1] до элемента a[i2,j2] с максимальной суммой. Ходить можно по горизонталям и вертикалям. Каждый элемент матрицы может входить в путь не более одного раза. Выделить этот путь в матрице!

Код программы на **Delphi**:

**Unit MainUnit;**

Interface

Uses

Winapi.Windows,

Winapi.Messages,

System.SysUtils,

System.Variants,

System.Classes,

Vcl.Graphics,

Vcl.Controls,

Vcl.Forms,

Vcl.Dialogs,

Vcl.Menus,

Vcl.StdCtrls,

Vcl.Mask,

Vcl.ExtCtrls,

Vcl.Buttons,

Vcl.Grids,

Clipbrd;

Type

TLabeledEdit = Class(Vcl.ExtCtrls.TLabeledEdit)

Public

Procedure WMPaste(Var Msg: TMessage); Message WM\_PASTE;

End;

TMainForm = Class(TForm)

MainMenu: TMainMenu;

FileBox: TMenuItem;

OpenFromFile: TMenuItem;

SaveInFile: TMenuItem;

Line: TMenuItem;

CloseButton: TMenuItem;

Instraction: TMenuItem;

AboutEditor: TMenuItem;

TaskLabel: TLabel;

MatrixSizeLabel: TLabel;

MRowsLEdit: TLabeledEdit;

NColsLEdit: TLabeledEdit;

MassiveStGrid: TStringGrid;

IStartPointLEdit: TLabeledEdit;

JStartPointLEdit: TLabeledEdit;

StartPointLabel: TLabel;

IEndPointLEdit: TLabeledEdit;

JEndPointLEdit: TLabeledEdit;

EndPointLabel: TLabel;

InputElemButton: TButton;

ResultSpButton: TButton;

ViewWayButton: TButton;

OpenDialog: TOpenDialog;

SaveDialog: TSaveDialog;

Procedure ResultSpButtonClick(Sender: TObject);

Procedure ViewWayButtonClick(Sender: TObject);

Procedure OpenFromFileClick(Sender: TObject);

Procedure SaveInFileClick(Sender: TObject);

Private

Procedure WMGetMinMaxInfo(Var Msg: TWMGetMinMaxInfo);

Public

{ Public declarations }

End;

Const

MIN\_SIZE: Integer = 1;

MAX\_SIZE: Integer = 6;

MIN\_INT: Integer = -1\_000\_000;

MAX\_INT: Integer = +1\_000\_000;

STRGRID\_CASE = ['0' .. '9', '-'];

NULL\_POINT: Char = #0;

ZERO\_KEY: Char = '0';

MINUS\_KEY: Char = '-';

Var

MainForm: TMainForm;

IfDataSavedInFile: Boolean = False;

Implementation

{$R \*.dfm}

Uses

FrontendUnit,

BackendUnit,

ResMatrixUnit;

Procedure TMainForm.InputElemButtonClick(Sender: TObject);

Begin

ChangeMassiveStGridSize(MassiveStGrid, StrToInt(MRowsLEdit.Text),

StrToInt(NColsLEdit.Text));

MassiveStGrid.ColCount := StrToInt(NColsLEdit.Text);

ClearElAndPoints();

ChangeHint(MRowsLEdit.Text, NColsLEdit.Text);

ChangeVisible(True);

End;

Procedure TMainForm.OpenFromFileClick(Sender: TObject);

Var

IsCorrect: Boolean;

Begin

Repeat

If OpenDialog.Execute() Then

Begin

IsCorrect := IsReadable(OpenDialog.FileName);

ReadFromFile(IsCorrect, OpenDialog.FileName);

If Not IsCorrect Then

MessageBox(0, 'Невозможен ввод из файла!', 'Ошибка', MB\_ICONERROR)

End

Else

IsCorrect := True;

Until IsCorrect;

End;

Procedure TMainForm.ResultSpButtonClick(Sender: TObject);

Var

I, J, I1, J1, I2, J2: Integer;

Begin

SetLength(Matrix, StrToInt(NColsLEdit.Text), StrToInt(MRowsLEdit.Text));

For I := 0 To StrToInt(MRowsLEdit.Text) - 1 Do

For J := 0 To StrToInt(NColsLEdit.Text) - 1 Do

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

J1 := StrToInt(IStartPointLEdit.Text) - 1;

I1 := StrToInt(JStartPointLEdit.Text) - 1;

J2 := StrToInt(IEndPointLEdit.Text) - 1;

I2 := StrToInt(JEndPointLEdit.Text) - 1;

SearchLongestWay(Matrix, I1, J1, I2, J2, ResWayCoords);

ViewWayButton.Visible := True;

End;

Procedure TMainForm.SaveInFileClick(Sender: TObject);

Var

IsCorrect: Boolean;

Begin

Repeat

If SaveDialog.Execute Then

Begin

IsCorrect := IsWriteable(SaveDialog.FileName);

InputInFile(IsCorrect, SaveDialog.FileName);

If Not IsCorrect Then

MessageBox(0, 'Невозможна запись в файл!', 'Ошибка',

MB\_ICONERROR);

End

Else

IsCorrect := True;

Until IsCorrect;

End;

Procedure TLabeledEdit.WMPaste(Var Msg: TMessage);

Begin

If Clipboard.HasFormat(CF\_TEXT) Then

Begin

Try

If (MainForm.ActiveControl = MainForm.MRowsLEdit) And Not

IsCorrectClipboard(Clipboard.AsText, MainForm.MRowsLEdit, MAX\_INT,

MIN\_INT) Then

Raise Exception.Create('Некорректная цифра :(');

If (MainForm.ActiveControl = MainForm.NColsLEdit) And Not

IsCorrectClipboard(Clipboard.AsText, MainForm.NColsLEdit, MAX\_INT,

MIN\_INT) Then

Raise Exception.Create('Некорректная цифра :(');

If (MainForm.ActiveControl = MainForm.IStartPointLEdit) And Not

IsCorrectClipboard(Clipboard.AsText, MainForm.IStartPointLEdit,

1, StrToInt(MainForm.MRowsLEdit.Text)) Then

Raise Exception.Create('Некорректная цифра :(');

If (MainForm.ActiveControl = MainForm.JStartPointLEdit) And Not

IsCorrectClipboard(Clipboard.AsText, MainForm.JStartPointLEdit,

1, StrToInt(MainForm.NColsLEdit.Text)) Then

Raise Exception.Create('Некорректная цифра :(');

If (MainForm.ActiveControl = MainForm.IEndPointLEdit) And Not

IsCorrectClipboard(Clipboard.AsText, MainForm.IEndPointLEdit, 1,

StrToInt(MainForm.MRowsLEdit.Text)) Then

Raise Exception.Create('Некорректная цифра :(');

If (MainForm.ActiveControl = MainForm.JEndPointLEdit) And Not

IsCorrectClipboard(Clipboard.AsText, MainForm.JEndPointLEdit, 1,

StrToInt(MainForm.NColsLEdit.Text)) Then

Raise Exception.Create('Некорректная цифра :(');

Except

On E: Exception Do

Begin

MessageBox(0, PWideChar(E.Message), 'Ошибка', MB\_ICONERROR);

Exit;

End;

End;

End;

Inherited;

End;

End.

**Unit ResMatrixUnit;**

Interface

Uses

Winapi.Windows,

Winapi.Messages,

System.SysUtils,

System.Variants,

System.Classes,

Vcl.Graphics,

Vcl.Controls,

Vcl.Forms,

Vcl.Dialogs,

Vcl.StdCtrls,

Vcl.ExtCtrls;

Type

TForm1 = Class(TForm)

ResultLabel: TLabel;

ResWayPBox: TPaintBox;

Procedure ResWayPBoxPaint(Sender: TObject);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

Form1: TForm1;

Implementation

{$R \*.dfm}

Uses

BackendUnit;

Procedure TForm1.ResWayPBoxPaint(Sender: TObject);

Const

StartX1: Integer = 50;

StartY1: Integer = 50;

StartX2: Integer = 150;

StartY2: Integer = 150;

BorderLength: Integer = 100;

Var

I, J, X1, Y1, X2, Y2: Integer;

BitMap: TBitmap;

K: Integer;

Begin

BitMap := TBitmap.Create();

BitMap.Height := Form1.ClientHeight;

BitMap.Width := Form1.ClientWidth;

For I := Low(Matrix) To High(Matrix) Do

For J := Low(Matrix[I]) To High(Matrix[I]) Do

Begin

X1 := I \* BorderLength;

Y1 := J \* BorderLength;

X2 := I \* BorderLength;

Y2 := J \* BorderLength;

Canvas.Brush.Color := ClWhite;

For K := Low(ResWayCoords) To High(ResWayCoords) Do

If (ResWayCoords[K][0] = I) And (ResWayCoords[K][1] = J) Then

Canvas.Brush.Color := ClSkyBlue;

Canvas.Rectangle(StartX1 + X1, StartY1 + Y1, StartX2 + X2,

StartY2 + Y2);

Canvas.Font.Size := 14;

Canvas.TextOut(StartX1 + X1 + BorderLength Div 2 –

(Canvas.Font.Size Div 2) \* (Length(IntToStr(

Matrix[I][J])) - Ord(Not(Length(IntToStr(Matrix[I][J]))

= 1))), StartY1 + Y1 + BorderLength Div 2 –

Canvas.Font.Size, IntToStr(Matrix[I][J]));

End;

Canvas.Brush.Color := ClSkyBlue;

Canvas.Font.Size := 10;

For I := Low(Matrix) To High(Matrix) Do

For J := Low(Matrix[I]) To High(Matrix[I]) Do

For K := Low(ResWayCoords) To High(ResWayCoords) - 1 Do

If (ResWayCoords[K][0] = I) And (ResWayCoords[K][1] = J) Then

Begin

X1 := I \* BorderLength;

Y1 := J \* BorderLength;

X2 := I \* BorderLength;

Y2 := J \* BorderLength;

If (ResWayCoords[K][0] - ResWayCoords[K + 1][0] = 1) Then

Canvas.TextOut(StartX1 + X1 - 6, StartY1 + Y1 + 10, '←');

If (ResWayCoords[K][0] - ResWayCoords[K + 1][0] = -1) Then

Canvas.TextOut(StartX2 + X2 - 6, StartY1 + Y1 + 10, '→');

If (ResWayCoords[K][1] - ResWayCoords[K + 1][1] = 1) Then

Canvas.TextOut(StartX2 + X2 - 20, StartY1 + Y1 - 12, '↑');

If (ResWayCoords[K][1] - ResWayCoords[K + 1][1] = -1) Then

Canvas.TextOut(StartX2 + X2 - 20, StartY2 + Y2 - 12, '↓');

End;

BitMap.Free();

End;

End.

**Unit BackendUnit;**

Interface

Uses

System.SysUtils;

Type

TMatrix = Array Of Array Of Integer;

TResCoords = Array Of Array [0 .. 1] Of Integer;

TSteps = Array [0 .. 3] Of Array [0 .. 1] Of Integer;

TUsed = Array Of Array Of Boolean;

Const

Steps: TSteps = ((0, 1), (1, 0), (-1, 0), (0, -1));

Var

ResWayCoords: TResCoords;

Matrix: TMatrix;

Ans: Integer = -2\_000\_000\_000;

Procedure SearchLongestWay(Matrix: TMatrix; I1, J1, I2, J2: Integer; Var ResWayCoords: TResCoords);

Function IsWriteable(FilePath: String): Boolean;

Procedure InputInFile(Var IsCorrect: Boolean; FilePath: String);

Function IsReadable(FilePath: String): Boolean;

Procedure ReadFromFile(Var IsCorrect: Boolean; FilePath: String);

Implementation

Uses

ResMatrixUnit,

MainUnit,

FrontendUnit;

Function IsWriteable(FilePath: String): Boolean;

Var

TestFile: TextFile;

Begin

Try

AssignFile(TestFile, FilePath);

Try

Rewrite(TestFile);

IsWriteable := True;

Finally

CloseFile(TestFile);

End;

Except

IsWriteable := False;

End;

End;

Procedure InputInFile(Var IsCorrect: Boolean; FilePath: String);

Var

MyFile: TextFile;

I, J, K: Integer;

OutputVal: String;

Begin

If IsCorrect Then

Begin

AssignFile(MyFile, FilePath, CP\_UTF8);

Try

ReWrite(MyFile);

Try

Write(MyFile, 'Ваша матрица :'#13#10);

For I := 0 To High(Matrix) Do

Begin

For J := 0 To High(Matrix[I]) Do

Write(MyFile, Matrix[I][J].ToString().PadLeft(6));

Write(MyFile, #13#10);

End;

Write(MyFile, #13#10'Ваш путь: '#13#10);

For I := 0 To High(Matrix) Do

Begin

For J := 0 To High(Matrix[I]) Do

Begin

OutputVal := '.';

For K := 0 To High(ResWayCoords) Do

Begin

If (ResWayCoords[K][0] = I) And

(ResWayCoords[K][1] = J) Then

OutputVal := IntToStr(K);

End;

Write(MyFile, OutputVal.PadLeft(6) + ' ');

End;

Write(MyFile, #13#10);

End;

Finally

Close(MyFile);

End;

IfDataSavedInFile := True;

Except

IsCorrect := False;

End;

End;

End;

Procedure CoordsRemove(Var Coords: TResCoords);

Var

NewCoords: TResCoords;

Counter: Integer;

Begin

SetLength(NewCoords, Length(Coords) - 1);

For Counter := Low(Coords) To High(Coords) - 1 Do

NewCoords[Counter] := Coords[Counter];

SetLength(Coords, Length(Coords) - 1);

Coords := Copy(NewCoords);

End;

Procedure CoordsAdd(I, J: Integer; Var Coords: TResCoords);

Var

NewCoords: TResCoords;

Counter: Integer;

Begin

SetLength(NewCoords, Length(Coords));

For Counter := Low(Coords) To High(Coords) Do

NewCoords[Counter] := Coords[Counter];

SetLength(Coords, Length(Coords) + 1);

For Counter := Low(NewCoords) To High(NewCoords) Do

Coords[Counter] := NewCoords[Counter];

Coords[Length(Coords) - 1][0] := I;

Coords[Length(Coords) - 1][1] := J;

End;

Procedure Rec(Sx, Sy, Fx, Fy, Sum: Integer; Matrix: TMatrix; Used: TUsed;

TempCoords: TResCoords; Var ResWayCoords: TResCoords);

Const

STEPS\_COUNT: Integer = 4;

Var

I, X, Y: Integer;

Begin

For I := 0 To STEPS\_COUNT - 1 Do

Begin

X := SX + Steps[I, 0];

Y := SY + Steps[I, 1];

If Not((X < 0) Or (Y < 0) Or (X >= Length(Matrix)) Or (Y >=

Length(Matrix[0])) Or (Used[X, Y])) Then

Begin

If (X = Fx) And (Y = Fy) And (Sum + Matrix[X][Y] > Ans) Then

Begin

Ans := Sum + Matrix[X][Y];

ResWayCoords := Nil;

ResWayCoords := Copy(TempCoords);

End;

CoordsAdd(X, Y, TempCoords);

Used[X, Y] := True;

Rec(X, Y, Fx, Fy, Sum + Matrix[X][Y], Matrix, Used, TempCoords,

ResWayCoords);

Used[X, Y] := False;

CoordsRemove(TempCoords);

End;

End;

End;

Procedure SearchLongestWay(Matrix: TMatrix; I1, J1, I2, J2: Integer; Var

ResWayCoords: TResCoords);

Var

Used: TUsed;

TempCoords: TResCoords;

Ans: Integer;

I: Integer;

J: Integer;

Begin

SetLength(Used, Length(Matrix));

For I := Low(Matrix) To High(Matrix) Do

Begin

SetLength(Used[I], Length(Matrix[I]));

For J := Low(Matrix[I]) To High(Matrix[I]) Do

Used[I][J] := False;

End;

CoordsAdd(I1, J1, TempCoords);

Used[I1, J1] := True;

Rec(I1, J1, I2, J2, Matrix[I1][J1], Matrix, Used, TempCoords, ResWayCoords);

CoordsAdd(I2, J2, ResWayCoords);

End;

Function TryReadNum(Var TestFile: TextFile; Var ReadStatus: Boolean; MAX\_NUM:

Integer): Integer;

Const

SPACE\_LIMIT: Integer = 4;

Var

EndOfNum: Boolean;

Character, BufChar: Char;

SpaceCounter, Num, MinCount: Integer;

Begin

Num := 0;

EndOfNum := False;

SpaceCounter := 0;

Character := NULL\_POINT;

BufChar := Character;

MinCount := 1;

While ReadStatus And Not(EndOfNum) And Not(EOF(TestFile)) Do

Begin

BufChar := Character;

Read(TestFile, Character);

ReadStatus := ReadStatus And Not((Character <> ' ') And Not((Character >

Pred('0')) And (Character < Succ('9'))) And

(Character <> #13) And (Character <> #10) And (Character <> '-'));

If (Character = ' ') Then

Inc(SpaceCounter)

Else

SpaceCounter := 0;

ReadStatus := Not(SpaceCounter = SPACE\_LIMIT);

If (Character > Pred('0')) And (Character < Succ('9')) Then

Num := Num \* 10 + Ord(Character) - 48;

If (Character = '-') Then

MinCount := -1;

ReadStatus := ReadStatus And Not((Character = '-') And (BufChar <> ' ')

And (BufChar <> #0));

ReadStatus := ReadStatus And Not((Character = '-') And (MinCount <> -1));

EndOfNum := ((Character = ' ') Or (Character = #13)) And ((BufChar >

Pred('0')) And (BufChar < Succ('9')));

ReadStatus := ReadStatus And Not((Num = 0) And (Character > Pred('0')) And

Character < Succ('9')));

ReadStatus := ReadStatus And Not(Num > MAX\_NUM);

End;

If ReadStatus Then

Num := MinCount \* Num;

TryReadNum := Num;

End;

Function CheckNum(Num, Max, Min, Count: Integer): Boolean;

Begin

CheckNum := Not((Num > MAX) Or (Num < MIN));

End;

Function TryRead(Var TestFile: TextFile): Boolean;

Var

M, N, ArrEl, Count, I, J: Integer;

I1, J1, I2, J2: Integer;

ReadStatus: Boolean;

Begin

Count := 0;

ReadStatus := True;

M := TryReadNum(TestFile, ReadStatus, MAX\_INT);

ReadStatus := CheckNum(M, MAX\_INT, MIN\_INT, 0);

N := TryReadNum(TestFile, ReadStatus, MAX\_INT);

ReadStatus := CheckNum(N, MAX\_INT, MIN\_INT, 0);

For I := 0 To M - 1 Do

For J := 0 To N - 1 Do

Begin

ArrEl := TryReadNum(TestFile, ReadStatus, MAX\_INT);

ReadStatus := CheckNum(ArrEl, MAX\_INT, MIN\_INT, Count);

Inc(Count);

End;

I1 := TryReadNum(TestFile, ReadStatus, M);

J1 := TryReadNum(TestFile, ReadStatus, N);

I2 := TryReadNum(TestFile, ReadStatus, M);

J2 := TryReadNum(TestFile, ReadStatus, N);

ReadStatus := ReadStatus And SeekEOF(TestFile);

TryRead := ReadStatus;

End;

Function IsReadable(FilePath: String): Boolean;

Var

TestFile: TextFile;

Begin

Try

AssignFile(TestFile, FilePath, CP\_UTF8);

Try

Reset(TestFile);

IsReadable := TryRead(TestFile);

Finally

Close(TestFile);

End;

Except

IsReadable := False;

End;

End;

Procedure ReadingProcess(Var IsCorrect: Boolean; Var MyFile: TextFile);

Var

M, N, I, J, El, I1, J1, I2, J2: Integer;

Sender: TObject;

Begin

Try

While Not(EOF(MyFile)) Do

Begin

Read(MyFile, M);

MainForm.MRowsLEdit.Text := IntToStr(M);

Read(MyFile, N);

MainForm.NColsLEdit.Text := IntToStr(N);

ChangeMassiveStGridSize(MainForm.MassiveStGrid, M, N);

MainForm.MassiveStGrid.ColCount := N;

ClearElAndPoints();

ChangeHint(IntToStr(M), IntToStr(N));

ChangeVisible(True);

For I := 0 To M - 1 Do

For J := 0 To N - 1 Do

Begin

Read(MyFile, El);

MainForm.MassiveStGrid.Cells[I, J] := IntToStr(El);

End;

Read(MyFile, I1);

MainForm.IStartPointLEdit.Text := IntToStr(I1);

Read(MyFile, J1);

MainForm.JStartPointLEdit.Text := IntToStr(J1);

Read(MyFile, I2);

MainForm.IEndPointLEdit.Text := IntToStr(I2);

Read(MyFile, J2);

MainForm.JEndPointLEdit.Text := IntToStr(J2);

End;

IsCorrect := True;

Except

IsCorrect := False;

End;

IsCorrect := IsCorrect And SeekEOF(MyFile);

End;

Procedure ReadFromFile(Var IsCorrect: Boolean; FilePath: String);

Var

MyFile: TextFile;

Sender: TObject;

Begin

If IsCorrect Then

Begin

AssignFile(MyFile, FilePath);

Try

Reset(MyFile);

Try

ReadingProcess(IsCorrect, MyFile);

MainForm.ResultSpButton.Enabled := True;

Finally

Close(MyFile);

End;

Except

IsCorrect := False;

End;

End;

End;

End.

Код программы на **C#**:

namespace Proj6\_2 {

internal enum IOChoose {

FILE = 1,

CONSOLE

}

class Combinator {

private const int FILE\_VALUE = (int)IOChoose.FILE;

private const int CONSOLE\_VALUE = (int)IOChoose.CONSOLE;

private const int MAX\_SIZE = 6;

private const int MIN\_SIZE = 1;

private const int MAX\_INT = +1\_000\_000;

private const int MIN\_INT = -1\_000\_000;

private const int STEPS\_COUNT = 4;

private const int SPACE\_LIMITS = 4;

private const int MIN\_FILE\_WAY\_SIZE = 4;

private static void conditionOutput() {

Console.WriteLine($"""

Combinatorics.

The user enters the matrix arr(m,n).

The program finds in it the path from the arr[i1,j1] element

to the arr[i2,j2]

element with the maximum amount. The algorithm moves

horizontally and vertically.

Each element of the matrix enters the path no more than once.

When displaying the result, the path in the matrix is

highlighted!

Limitations:

1. The dimensions of the matrix are within [{MIN\_SIZE};

{MAX\_SIZE}];

1. The elements of the matrix are within [{MIN\_INT};

{MAX\_INT}];

3. When entering/outputting through a file, only the extension

can be used.txt!

3.1 First, there are 2 numbers M and N in the file:

• M - row count;

• N - col count;

3.2 Then there are numbers, the number of which M x N;

3.3 Then there should be coordinates of 2 points: i1, j1,

i2, j2;

""");

}

static int inputNumberFromConsole(int MAX, int MIN) {

int val = 0;

bool isCorrect;

do {

try {

val = Convert.ToInt32(Console.ReadLine());

isCorrect = true;

} catch {

isCorrect = false;

}

isCorrect = isCorrect && !(val > MAX || val < MIN);

if (!isCorrect) Console.Write("Error! Try again: ");

} while (!isCorrect);

return val;

}

static int inputNumberFromFile(StreamReader inputReader, ref bool

isCorrectInput, int minNum, int maxNum) {

int num = 0, minCount = 1, spaceCounter = 0, character;

bool endOfNum = false;

while (isCorrectInput && !(endOfNum) && (character =

inputReader.Read()) != -1) {

int bufChar = character;

isCorrectInput = isCorrectInput && !((character != ' ') &&

!(character is > '/' and < ':') && (character !=

'\n') && (character != '\r') && (character != '-

'));

if (character == ' ') ++spaceCounter;

else spaceCounter = 0;

isCorrectInput = spaceCounter != SPACE\_LIMITS;

if (character is > '/' and < ':')

num = num \* 10 + character - 48;

if (character == '-') minCount = -1;

isCorrectInput = isCorrectInput && !((character == '-') &&

(minCount != -1));

endOfNum = (character == ' ' || character == '\n') &&

(bufChar > '/' || bufChar < ':');

isCorrectInput = isCorrectInput && !((bufChar == 0) &&

character is > '/' and < ':');

isCorrectInput = isCorrectInput && !(num > maxNum);

}

isCorrectInput = isCorrectInput && !(num > maxNum || num < minNum);

if (isCorrectInput) num = minCount \* num;

return num;

}

static void inputMatrixSize(out int m, out int n) {

Console.WriteLine("Input your matrix size (m x n)");

Console.Write("M: ");

m = inputNumberFromConsole(MAX\_SIZE, MIN\_SIZE);

Console.Write("N: ");

n = inputNumberFromConsole(MAX\_SIZE, MIN\_SIZE);

}

static void changeMatrixSize(out int[][] matrix, int m, int n) {

matrix = new int[m][];

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

matrix[i] = new int[n];

}

static void inputMatrixElements(ref int[][] matrix) {

Console.WriteLine("Input matrix elements.");

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

for (int j = 0; j < matrix[i].Length; j++) {

Console.Write($"Write arr[{i};{j}]: ");

matrix[i][j] = inputNumberFromConsole(MAX\_INT, MIN\_INT);

}

}

static void inputPoint(int maxm, int maxn, out int i, out int j,

char pointer) {

int min = 0;

Console.WriteLine($"Input i{pointer}, j{pointer}.");

Console.Write($"Input i{pointer}: ");

i = inputNumberFromConsole(maxm, min);

Console.Write($"Input j{pointer}: ");

j = inputNumberFromConsole(maxn, min);

}

static void inputFromConsole(out int[][] matrix, out int i1, out int j1,

out int i2, out int j2) {

Console.Clear();

inputMatrixSize(out int m, out int n);

changeMatrixSize(out matrix, m, n);

inputMatrixElements(ref matrix);

inputPoint(m - 1, n - 1, out i1, out j1, '1');

inputPoint(m - 1, n - 1, out i2, out j2, '2');

}

static void outputTextAboutIoSelection(string ioTextInfo) {

string outputString = $"""

Select how you will {ioTextInfo} data:

{IOChoose.FILE}: {FILE\_VALUE} {IOChoose.CONSOLE}:

{CONSOLE\_VALUE}

Your option:

""";

Console.Write(outputString);

}

/// <summary>

/// Here you can write a file for what purposes you are using

(input|output)

/// </summary>

/// <param name="ioTextInfo"></param>

/// <returns></returns>

static IOChoose chooseIoWay(string ioTextInfo) {

outputTextAboutIoSelection(ioTextInfo);

IOChoose result = 0;

int chosenPath = 0;

bool isCorrect;

do {

isCorrect = true;

try {

chosenPath = Convert.ToInt32(Console.ReadLine());

}

catch {

isCorrect = false;

}

switch (chosenPath) {

case FILE\_VALUE: result = IOChoose.FILE; break;

case CONSOLE\_VALUE: result = IOChoose.CONSOLE; break;

default: isCorrect = false; break;

}

if (!isCorrect) Console.Error.Write($"You should write one natural

number({FILE\_VALUE}|

{CONSOLE\_VALUE}): ");

else Console.WriteLine();

} while (!isCorrect);

return result;

}

static bool isWriteable(string filePath) {

try {

using StreamWriter writer = new StreamWriter(filePath);

writer.WriteLine(string.Empty);

writer.Close();

return true;

}

catch {

return false;

}

}

static bool isReadable(string filePath) {

try {

using StreamReader reader = new StreamReader(filePath);

reader.Read();

reader.Close();

return true;

}

catch {

return false;

}

}

static bool pathCondition(string filePath) {

if (filePath.Length < MIN\_FILE\_WAY\_SIZE) {

Console.Error.Write("The path is too short. Try again: ");

return false;

}

string buffer = filePath.Substring(filePath.Length –

MIN\_FILE\_WAY\_SIZE);

if (buffer.Equals(".txt")) return true;

Console.Error.Write("Write .txt file. Try again: ");

return false;

}

static string inputFilePath() {

string filePath = Console.ReadLine() ?? string.Empty;

while (!pathCondition(filePath)) filePath = Console.ReadLine() ??

string.Empty;

return filePath;

}

static bool accessModifierControl(string accessModifier, string filePath) {

bool resultModifier = true;

switch (accessModifier) {

case "input": resultModifier = isReadable(filePath); break;

case "output": resultModifier = isWriteable(filePath); break;

}

return resultModifier;

}

static bool isCanOpenFile(string filePath) {

FileInfo fileInfo = new FileInfo(filePath);

return fileInfo.Exists;

}

/// <summary>

/// Write "input" if you want to get the file path for input.

/// Write "output" if you want to get the path to the output file.

/// </summary>

/// <param name="accessModifier"></param>

/// <returns></returns>

static string inputPathToTheFile(string accessModifier) {

string filePath;

bool isCorrect;

do {

filePath = inputFilePath();

isCorrect = accessModifierControl(accessModifier, filePath) &&

isCanOpenFile(filePath);

if (!isCorrect) Console.Error.Write("Can't open a file. Try write

another way: ");

} while (!isCorrect);

return filePath;

}

static bool isProcessOfFileInputCorrect(string filePath, out int[][]

matrix, out int i1, out int j1,

out int i2, out int j2) {

bool isCorrectInput = true;

using StreamReader inputReader = new StreamReader(filePath);

int m = inputNumberFromFile(inputReader, ref isCorrectInput, MIN\_SIZE,

MAX\_SIZE);

int n = inputNumberFromFile(inputReader, ref isCorrectInput, MIN\_SIZE,

MAX\_SIZE);

changeMatrixSize(out matrix, m, n);

for (int i = 0; i < matrix.Length && isCorrectInput; ++i)

for (int j = 0; j < matrix[i].Length && isCorrectInput; ++j)

matrix[i][j] = inputNumberFromFile(inputReader, ref

isCorrectInput, MIN\_SIZE,

MAX\_SIZE);

i1 = inputNumberFromFile(inputReader, ref isCorrectInput, m, n);

j1 = inputNumberFromFile(inputReader, ref isCorrectInput, m, n);

i2 = inputNumberFromFile(inputReader, ref isCorrectInput, m, n);

j2 = inputNumberFromFile(inputReader, ref isCorrectInput, m, n);

isCorrectInput = isCorrectInput && inputReader.EndOfStream;

if (!isCorrectInput) Console.Error.WriteLine("Error in reading. Try

again.");

inputReader.Close();

return isCorrectInput;

}

static void inputFormFile(out int[][] matrix, out int i1, out int j1, out

int i2, out int j2) {

Console.Clear();

string filePath;

do {

Console.Write("Write way to your file (\*.txt): ");

filePath = inputPathToTheFile("input");

} while (!isProcessOfFileInputCorrect(filePath, out matrix, out i1,

out j1, out i2, out j2));

}

static void inputProcess(out int[][] matrix, out int i1, out int j1, out

int i2, out int j2) {

IOChoose path = chooseIoWay("input");

switch (path) {

case IOChoose.FILE: inputFormFile(out matrix, out i1, out j1, out

i2, out j2); break;

default: inputFromConsole(out matrix, out i1, out j1, out i2, out

j2); break;

}

}

static void searchLongestWay(int[][] matrix, int i1, int j1, int i2, int

j2, List<int[]> resWayCoords) {

int[,] st = {{0, 1}, {1, 0}, {-1, 0}, {0, -1}};

int ans = -int.MaxValue;

bool[,] used = new bool[matrix.Length, matrix[0].Length];

int[] start = { i1, j1 };

List<int[]> TempList = [];

TempList.Add(start);

used[i1, j1] = true;

rec(i1, j1, i2, j2, matrix[i1][j1]);

void rec(int sx, int sy, int fx, int fy, int sum) {

for (int i = 0; i < STEPS\_COUNT; i++) {

int x = sx + st[i, 0];

int y = sy + st[i, 1];

if (!(x < 0 || y < 0 || x >= matrix.Length || y >=

matrix[0].Length || used[x, y])) {

if (x == fx && y == fy && sum + matrix[x][y] > ans) {

ans = sum + matrix[x][y];

resWayCoords.Clear();

resWayCoords.AddRange(TempList);

}

int[] add = { x, y };

TempList.Add(add);

used[x, y] = true;

rec(x, y, fx, fy, sum + matrix[x][y]);

used[x, y] = false;

TempList.RemoveAt(TempList.Count - 1);

}

}

}

int[] end = { i2, j2 };

resWayCoords.Add(end);

}

static void outputFromConsole(string resultStr) {

Console.WriteLine(resultStr);

}

static bool isProcessOfFileOutputCorrect(string filePath, string

resultStr) {

try {

using (StreamWriter writerOutput = new StreamWriter(filePath))

writerOutput.WriteLine(resultStr);

Console.WriteLine("Data successfully written to file.");

return true;

}

catch {

Console.Error.WriteLine("Error in writing. Try again.");

return false;

}

}

static void outputFormFile(string resultStr) {

Console.Clear();

string filePath;

do {

Console.Write("Write way to your file (\*.txt): ");

filePath = inputPathToTheFile("output");

} while (!isProcessOfFileOutputCorrect(filePath, resultStr));

}

static void InputMatrixInRes(ref int[][] matrix, ref List<int[]>

resWayCoords, ref string resultStr) {

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

for (int j = 0; j < matrix[i].Length; j++) {

resultStr += matrix[i][j].ToString().PadLeft(6);

}

resultStr += "\n";

}

}

static void inputLongestWayInRes(ref int[][] matrix, ref List<int[]>

resWayCoords, ref string resultStr) {

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

for (int j = 0; j < matrix[i].Length; j++) {

string outputVal = ".".PadLeft(6);

for (int k = 0; k < resWayCoords.Count; k++)

if (resWayCoords[k][0] == i && resWayCoords[k][1] == j)

outputVal = k.ToString().PadLeft(6);

resultStr += outputVal + " ";

}

resultStr += "\n";

}

}

static void outputMatrix(ref int[][] matrix, ref List<int[]> resWayCoords,

ref string resultStr) {

resultStr += "Input Matrix: \n";

InputMatrixInRes(ref matrix, ref resWayCoords, ref resultStr);

resultStr += "Longest Way: \n";

inputLongestWayInRes(ref matrix, ref resWayCoords, ref resultStr);

}

static void outputResult(ref int[][] matrix, ref List<int[]>

resWayCoords) {

Console.Clear();

string resultStr = "Result longest way: \n";

outputMatrix(ref matrix, ref resWayCoords, ref resultStr);

IOChoose path = chooseIoWay("output");

switch (path) {

case IOChoose.FILE: outputFormFile(resultStr); break;

case IOChoose.CONSOLE: outputFromConsole(resultStr); break;

}

}

public static void Main(string[] args) {

List<int[]> resWayCoords = [];

conditionOutput();

inputProcess(out int[][] matrix, out int i1, out int j1, out int i2,

out int j2);

searchLongestWay(matrix, i1, j1, i2, j2, resWayCoords);

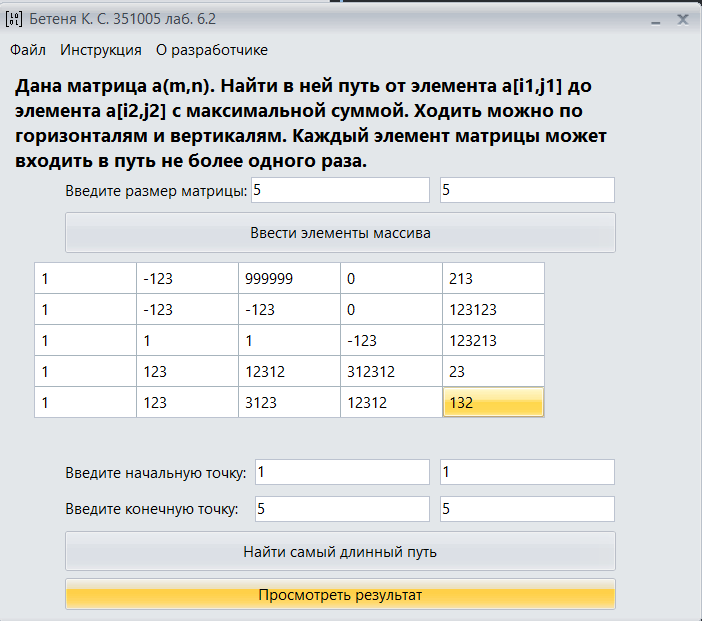
outputResult(ref matrix, ref resWayCoords);

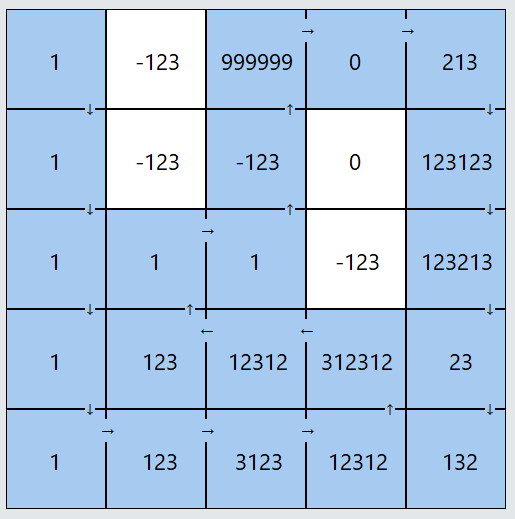
}

}

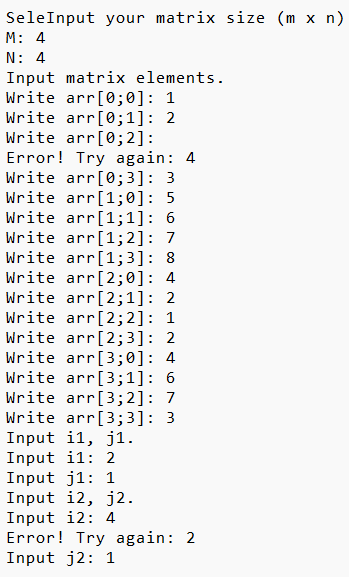
}

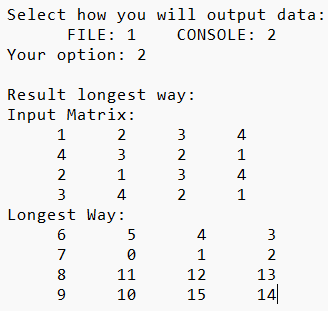
Результат в **Delphi**:





Результат в **C#**:





Блок-схема:

