УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №4.2

по предмету «Основы алгоритмизации и программирования»

Вариант 4

Выполнил:

Бражалович А. И.

Гр. 351004

Проверил:

Данилова Г. В.

Минск 2024

**Задание:**

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 двух раз.

**Код программы Delphi:**

Unit MainForm;

Interface

Uses

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

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Menus, Instruction, Developer,

Vcl.StdCtrls, Vcl.ExtDlgs, Vcl.Grids;

Type

TEStringGrid = Class(TStringGrid);

TMatrix = Array Of Array Of Integer;

ERRORS\_LIST = (CORRECT, RANGE\_ERR, NUM\_ERR, NOT\_READABLE, NOT\_WRITEABLE,

FILE\_EMPTY, LINE\_ERR, ORDER\_ERR);

TMainTaskForm = Class(TForm)

MainFormMenu: TMainMenu;

FileMenu: TMenuItem;

InstructionMenu: TMenuItem;

DeveloperMenu: TMenuItem;

OpenMenu: TMenuItem;

SaveMenu: TMenuItem;

N1: TMenuItem;

QuitMenu: TMenuItem;

TaskLabel: TLabel;

ResultButton: TButton;

OpenFile: TOpenDialog;

EnterMEdit: TEdit;

EnterNLabel: TLabel;

ArrLabel: TLabel;

SaveTextFile: TSaveTextFileDialog;

StringGrid: TStringGrid;

OutLabel: TLabel;

EnterNEdit: TEdit;

I1Edit: TEdit;

J1Edit: TEdit;

I2Edit: TEdit;

J2Edit: TEdit;

StartLabel: TLabel;

EndLabel: TLabel;

OutEdit: TEdit;

Procedure DeveloperMenuClick(Sender: TObject);

Procedure InstructionMenuClick(Sender: TObject);

Procedure FormCreate(Sender: TObject);

Procedure EnterMEditContextPopup(Sender: TObject; MousePos: TPoint;

Var Handled: Boolean);

Procedure IsEnterEditsEmpty(EditM, EditN: TEdit);

Procedure GetDataFromFile(Var F: TextFile; Sender: TObject);

Function FileReading(Var F: TextFile): ERRORS\_LIST;

Procedure ResultButtonClick(Sender: TObject);

Procedure EnterMEditChange(Sender: TObject);

Procedure QuitMenuClick(Sender: TObject);

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

Procedure EnterMEditKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

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

Procedure EnterMEditClick(Sender: TObject);

Procedure SaveMenuClick(Sender: TObject);

Procedure OpenMenuClick(Sender: TObject);

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

Procedure StringGridKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Procedure StringGridSetEditText(Sender: TObject; ACol, ARow: Integer;

Const Value: String);

Procedure StringGridMouseActivate(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y, HitTest: Integer;

Var MouseActivate: TMouseActivate);

Function FormHelp(Command: Word; Data: NativeInt;

Var CallHelp: Boolean): Boolean;

Procedure EnterNEditChange(Sender: TObject);

Procedure EnterNEditClick(Sender: TObject);

Procedure EnterNEditContextPopup(Sender: TObject; MousePos: TPoint;

Var Handled: Boolean);

Procedure EnterNEditKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

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

Procedure J1EditChange(Sender: TObject);

Procedure I1EditChange(Sender: TObject);

Procedure I2EditChange(Sender: TObject);

Procedure J2EditChange(Sender: TObject);

Procedure I1EditClick(Sender: TObject);

Procedure J1EditClick(Sender: TObject);

Procedure I2EditClick(Sender: TObject);

Procedure J2EditClick(Sender: TObject);

Procedure I1EditContextPopup(Sender: TObject; MousePos: TPoint;

Var Handled: Boolean);

Procedure J1EditContextPopup(Sender: TObject; MousePos: TPoint;

Var Handled: Boolean);

Procedure I2EditContextPopup(Sender: TObject; MousePos: TPoint;

Var Handled: Boolean);

Procedure J2EditContextPopup(Sender: TObject; MousePos: TPoint;

Var Handled: Boolean);

Procedure I1EditKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Procedure J1EditKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Procedure I2EditKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Procedure J2EditKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Procedure I1EditKeyPress(Sender: TObject; Var Key: Char);

Procedure J1EditKeyPress(Sender: TObject; Var Key: Char);

Procedure I2EditKeyPress(Sender: TObject; Var Key: Char);

Procedure J2EditKeyPress(Sender: TObject; Var Key: Char);

Procedure ButtonOn(I1, J1, I2, J2: TEdit);

Procedure MainFunc();

Procedure MakeUnVisibleAndClear(I1, J1, I2, J2: TEdit);

Procedure MakeVisible(I1, J1, I2, J2: TEdit);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Const

ERRORS: Array [ERRORS\_LIST] Of String = ('',

'Значение не попадает в диапазон!',

'Проверьте корректность ввода данных!', 'Файл закрыт для чтения!',

'Файл закрыт для записи!', 'Файл пуст!', 'Неверное число строк в файле',

'Неверный порядок матрицы!');

DIGITS = ['0' .. '9'];

NO\_ZERO\_DIGITS = ['1' .. '9'];

BACKSPACE = #8;

NONE = #0;

MIN\_N = 2;

MAX\_N = 4;

MIN\_X = -99;

MAX\_X = 100;

ALPHABET = ['A' .. 'Z', 'a' .. 'z'];

Var

MainTaskForm: TMainTaskForm;

Implementation

{$R \*.dfm}

Var

Saved: Boolean = True;

PerformCloseQuery: Boolean = True;

CtrlPressed: Boolean = False;

MaxSum: Integer = 0;

Steps: TMatrix;

Matrix: TMatrix;

Path: String;

PathLength: Integer;

StepCounter: Integer = 0;

Procedure TMainTaskForm.DeveloperMenuClick(Sender: TObject);

Var

DeveloperForm: TDeveloperForm;

Begin

DeveloperForm := TDeveloperForm.Create(Self);

DeveloperForm.ShowModal;

DeveloperForm.Free;

End;

Procedure TMainTaskForm.FormCreate(Sender: TObject);

Begin

MakeUnVisibleAndClear(I1Edit, J1Edit, I2Edit, J2Edit);

End;

Function TMainTaskForm.FormHelp(Command: Word; Data: NativeInt;

Var CallHelp: Boolean): Boolean;

Begin

CallHelp := False;

InstructionMenuClick(Self);

End;

Procedure TMainTaskForm.InstructionMenuClick(Sender: TObject);

Var

InstructionForm: TInstructionForm;

Begin

InstructionForm := TInstructionForm.Create(Self);

InstructionForm.ShowModal;

InstructionForm.Free;

End;

Function IsReadable(Var F: TextFile): ERRORS\_LIST;

Var

ERRORS: ERRORS\_LIST;

Begin

ERRORS := CORRECT;

Try

Try

Reset(F);

Finally

CloseFile(F);

End;

Except

ERRORS := NOT\_READABLE;

End;

IsReadable := ERRORS;

End;

Procedure TMainTaskForm.EnterMEditContextPopup(Sender: TObject;

MousePos: TPoint; Var Handled: Boolean);

Begin

Handled := True;

End;

Procedure GiveZeroOrNone(Edit: TEdit);

Var

Num: Double;

Begin

If TryStrToFloat(Edit.Text, Num) And (Num = 0) Then

Edit.Text := '0';

End;

Function IsCorrectRange(Value: Integer; Const MIN, MAX: Real): ERRORS\_LIST;

Var

ERRORS: ERRORS\_LIST;

Begin

ERRORS := CORRECT;

If ((Value < MIN) Or (Value > MAX)) Then

Begin

ERRORS := RANGE\_ERR;

End;

IsCorrectRange := ERRORS;

End;

Function CheckNumOfLines(Var F: TextFile; Const NUM\_OF\_LINES: Integer)

: ERRORS\_LIST;

Var

I: Integer;

Str: String;

Error: ERRORS\_LIST;

Begin

I := 0;

Str := '';

Error := CORRECT;

Reset(F);

While Not EOF(F) Do

Begin

Readln(F, Str);

Inc(I);

End;

CloseFile(F);

If I <> NUM\_OF\_LINES Then

Error := LINE\_ERR;

CheckNumOfLines := Error;

End;

Function CheckFileData(Var F: TextFile): ERRORS\_LIST;

Var

ERRORS: ERRORS\_LIST;

Value: Integer;

Num, I, J, Cols, Rows: Integer;

Str, StrNum, Mark: String;

Numbers: TArray<String>;

Begin

ERRORS := CORRECT;

J := 0;

I := 0;

Rows := 0;

Cols := 0;

Str := ' ';

Reset(F);

While (ERRORS = CORRECT) And Not EOLN(F) Do

Begin

Try

Read(F, Value);

Except

ERRORS := NUM\_ERR;

End;

If ERRORS = CORRECT Then

ERRORS := IsCorrectRange(Value, MIN\_N, MAX\_N);

End;

Cols := Value;

Readln(F);

While (ERRORS = CORRECT) And Not EOLN(F) Do

Begin

Try

Read(F, Value);

Except

ERRORS := NUM\_ERR;

End;

If ERRORS = CORRECT Then

ERRORS := IsCorrectRange(Value, MIN\_N, MAX\_N);

End;

Rows := Value;

Readln(F);

If ERRORS = CORRECT Then

Begin

For I := 1 To 2 Do

Begin

ReadLn(F, StrNum);

If TryStrToInt(StrNum, Value) Then

Begin

ERRORS := IsCorrectRange(Value, 1, Rows);

End

Else

ERRORS := NUM\_ERR;

End;

I := 0;

End;

If ERRORS = CORRECT Then

Begin

For I := 1 To 2 Do

Begin

Readln(F, StrNum);

If TryStrToInt(StrNum, Value) Then

Begin

ERRORS := IsCorrectRange(Value, 1, Cols);

End

Else

ERRORS := NUM\_ERR;

End;

I := 0;

End;

Str := ' ';

While (ERRORS = CORRECT) And Not EOF(F) And (Str <> '') Do

Begin

While (ERRORS = CORRECT) And Not EOLN(F) Do

Begin

ReadLn(F, StrNum);

StrNum := Trim(StrNum);

Numbers := StrNum.Split([' ']);

For Mark In Numbers Do

Begin

If TryStrToInt(Mark, Value) Then

Begin

ERRORS := IsCorrectRange(Value, MIN\_X, MAX\_X);

End

Else

ERRORS := NUM\_ERR;

Inc(I);

End;

If (I <> Cols) Then

ERRORS := ORDER\_ERR;

I := 0;

Inc(J);

End;

Read(F, Str);

End;

If ((I) <> Cols) And (ERRORS = CORRECT) And (J <> Rows) Then

ERRORS := ORDER\_ERR;

CloseFile(F);

CheckFileData := ERRORS;

End;

Procedure TMainTaskForm.GetDataFromFile(Var F: TextFile; Sender: TObject);

Var

NumStr, Mark: String;

Numbers: TArray<String>;

Num, I, J: Integer;

Begin

I := 0;

J := 0;

Reset(F);

Readln(F, NumStr);

EnterMEdit.Text := NumStr;

Readln(F, NumStr);

EnterNEdit.Text := NumStr;

Readln(F, NumStr);

I1Edit.Text := NumStr;

Readln(F, NumStr);

I2Edit.Text := NumStr;

Readln(F, NumStr);

J1Edit.Text := NumStr;

Readln(F, NumStr);

J2Edit.Text := NumStr;

While Not EOF(F) Do

Begin

While Not EOLN(F) Do

Begin

Read(F, NumStr);

NumStr := Trim(NumStr);

Numbers := NumStr.Split([' ']);

For Mark In Numbers Do

Begin

If TryStrToInt(Mark, Num) Then

Begin

StringGrid.Cells[I, J] := IntToStr(Num);

End;

Inc(I);

End;

End;

I := 0;

Readln(F);

Inc(J);

End;

CloseFile(F);

End;

Function TMainTaskForm.FileReading(Var F: TextFile): ERRORS\_LIST;

Var

ERRORS: ERRORS\_LIST;

Begin

ERRORS := CORRECT;

Reset(F);

If EOF(F) Then

ERRORS := FILE\_EMPTY;

CloseFile(F);

If (ERRORS = CORRECT) Then

ERRORS := CheckFileData(F);

If (ERRORS = CORRECT) Then

GetDataFromFile(F, Self);

If ERRORS = CORRECT Then

Begin

ResultButton.Enabled := True;

MakeVisible(I1Edit, J1Edit, I2Edit, J2Edit);

End;

FileReading := ERRORS;

End;

Procedure TMainTaskForm.OpenMenuClick(Sender: TObject);

Var

Error: ERRORS\_LIST;

F: TextFile;

FileName: String;

Begin

If OpenFile.Execute Then

Begin

FileName := OpenFile.FileName;

AssignFile(F, FileName);

Error := IsReadable(F);

If Error = CORRECT Then

Error := FileReading(F);

If Error <> CORRECT Then

Application.MessageBox(PWideChar(ERRORS[Error]), 'Ошибка',

MB\_OK Or MB\_ICONINFORMATION);

End;

End;

Function FillArrayFromStringGrid(MatrixGrid: TStringGrid): TMatrix;

Var

I: Integer;

Matrix: TMatrix;

J: Integer;

Begin

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

For J := 0 To MatrixGrid.ColCount - 1 Do

Begin

For I := 0 To MatrixGrid.RowCount - 1 Do

Begin

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

End;

End;

Result := Matrix;

End;

Procedure FillGrid(OrderM, OrderN: Integer; Grid: TStringGrid);

Var

I, J: Integer;

NumArr: TMatrix;

Begin

Grid.Width := (Grid.DefaultColWidth + 3) \* OrderM;

Grid.Height := (Grid.DefaultRowHeight + 4) \* 2;

Grid.Enabled := True;

Grid.ColCount := OrderM;

Grid.RowCount := OrderN;

End;

Procedure ClearGrid(Grid: TStringGrid);

Var

J, I: Integer;

Begin

For I := 0 To Grid.ColCount Do

For J := 0 To Grid.RowCount Do

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

Grid.ColCount := 0;

Grid.Enabled := False;

End;

Procedure TMainTaskForm.SaveMenuClick(Sender: TObject);

Var

Error: ERRORS\_LIST;

F: TextFile;

FileName: String;

I, J: Integer;

Begin

If SaveTextFile.Execute Then

Begin

FileName := SaveTextFile.FileName;

FileName := ChangeFileExt(FileName, '.txt');

AssignFile(F, FileName);

If FileExists(FileName) Then

Begin

Error := IsReadable(F);

If Error = CORRECT Then

Begin

Append(F);

Write(F, OutEdit.Text);

CloseFile(F);

Saved := True;

End;

If Error <> CORRECT Then

Begin

Application.MessageBox(PWideChar(ERRORS[Error]), 'Ошибка',

MB\_OK Or MB\_ICONINFORMATION);

Saved := False;

End;

End

Else

Begin

Rewrite(F);

Write(F, OutEdit.Text);

CloseFile(F);

Saved := True;

End;

End;

End;

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

Var

Confirmation: Integer;

Begin

If PerformCloseQuery Then

Begin

If (Saved = False) Then

Begin

Confirmation := Application.MessageBox

('Вы не сохранили файл, хотите ли сохранить?', 'Выход',

MB\_YESNOCANCEl + MB\_ICONQUESTION + MB\_DEFBUTTON2);

Case Confirmation Of

MrYes:

Begin

SaveMenuClick(Sender);

If Saved = True Then

CanClose := True

Else

FormCloseQuery(Sender, CanClose);

End;

MrNo:

CanClose := True;

MrCancel:

CanClose := False;

End;

End

Else

Begin

Confirmation := Application.MessageBox

('Вы действительно хотите выйти?', 'Выход',

MB\_YESNO + MB\_ICONQUESTION + MB\_DEFBUTTON2);

CanClose := Confirmation = IDYES;

End;

End;

End;

Procedure TMainTaskForm.QuitMenuClick(Sender: TObject);

Var

Confirmation: Integer;

Begin

PerformCloseQuery := False;

If (Saved = False) Then

Begin

Confirmation := Application.MessageBox

('Вы не сохранили файл, хотите ли сохранить?', 'Выход',

MB\_YESNOCANCEl + MB\_ICONQUESTION + MB\_DEFBUTTON2);

Case Confirmation Of

MrYes:

Begin

SaveMenuClick(Sender);

If Saved = True Then

Close

Else

QuitMenuClick(Sender);

End;

MrNo:

Close;

End;

End

Else

Begin

Confirmation := Application.MessageBox('Вы действительно хотите выйти?',

'Выход', MB\_YESNO + MB\_ICONQUESTION + MB\_DEFBUTTON2);

If Confirmation = IDYES Then

Close;

End;

PerformCloseQuery := True;

End;

Procedure CheckInput(Text: String; Var Key: Char; Const MIN, MAX: Real);

Var

Value: Integer;

ERRORS: ERRORS\_LIST;

Begin

Value := 0;

ERRORS := CORRECT;

If TryStrToInt(Text + Key, Value) Then

Begin

ERRORS := IsCorrectRange(Value, MIN, MAX);

If ERRORS <> CORRECT Then

Begin

Key := #0;

End;

End;

End;

Function CreateZeroMatrix(StringGrid: TStringGrid): TMatrix;

Var

I, J, HighI, HighJ: Integer;

ZeroMatrix: TMatrix;

Begin

HighI := StringGrid.RowCount;

HighJ := StringGrid.ColCount;

SetLength(ZeroMatrix, HighI, HighJ);

For I := 0 To HighI - 1 Do

For J := 0 To HighJ - 1 Do

ZeroMatrix[I, J] := 0;

CreateZeroMatrix := ZeroMatrix;

End;

Procedure FindMaxSumPath(M, N, I1, J1, I2, J2, CurrentSum: Integer;

CurrentPath: String);

Begin

If Not((I1 < 0) Or (I1 > M - 1) Or (J1 < 0) Or (J1 > N - 1) Or

(Steps[I1, J1] >= 2)) Then

Begin

Inc(StepCounter);

CurrentPath := CurrentPath + Format('(%d). [%d,%d]; ',

[StepCounter, I1 + 1, J1 + 1]);

Inc(Steps[I1, J1]);

Inc(CurrentSum, Matrix[I1, J1]);

If (I1 = I2) And (J1 = J2) Then

Begin

If CurrentSum > MaxSum Then

Begin

MaxSum := CurrentSum;

Path := CurrentPath;

End;

End

Else

Begin

FindMaxSumPath(M, N, I1, J1 + 1, I2, J2, CurrentSum, CurrentPath);

FindMaxSumPath(M, N, I1 + 1, J1, I2, J2, CurrentSum, CurrentPath);

FindMaxSumPath(M, N, I1, J1 - 1, I2, J2, CurrentSum, CurrentPath);

FindMaxSumPath(M, N, I1 - 1, J1, I2, J2, CurrentSum, CurrentPath);

End;

Dec(Steps[I1, J1]);

Delete(CurrentPath, Length(CurrentPath) - 6, 7);

Dec(StepCounter);

End;

End;

Procedure TMainTaskForm.MainFunc();

Var

I1, J1, I2, J2: Integer;

Begin

Matrix := FillArrayFromStringGrid(StringGrid);

Steps := CreateZeroMatrix(StringGrid);

I1 := StrtoInt(I1Edit.Text) - 1;

J1 := StrtoInt(J1Edit.Text) - 1;

I2 := StrtoInt(I2Edit.Text) - 1;

J2 := StrtoInt(J2Edit.Text) - 1;

FindMaxSumPath(StringGrid.RowCount, StringGrid.ColCount, I1, J1, I2, J2,

MaxSum, Path);

If Path <> '' Then

OutEdit.Text := Path

Else

OutEdit.Text := 'Не удалось найти оптимальный путь';

Path := '';

End;

Procedure CheckComboButtons(Var Key: Char; Edit: TEdit);

Begin

If (Key = #22) Or ((Key = 'v') And (GetKeyState(VK\_CONTROL) < 0)) Then

Key := #0;

If Not CharInSet(Key, DIGITS) And (Key <> #8) Then

Key := #0;

End;

Procedure CheckShftAndArrows(Var Key: Word; Shift: TShiftState);

Begin

If (Key = VK\_INSERT) And (Shift = [SsShift]) Then

Key := 0;

If (Key = VK\_LEFT) Or (Key = VK\_UP) Then

Key := 0

Else If (Key = VK\_RIGHT) Or (Key = VK\_DOWN) Then

Key := 0;

End;

Procedure TMainTaskForm.EnterMEditKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Begin

CheckShftAndArrows(Key, Shift);

End;

Procedure TMainTaskForm.EnterMEditKeyPress(Sender: TObject; Var Key: Char);

Begin

CheckComboButtons(Key, EnterMEdit);

CheckInput(EnterMEdit.Text, Key, MIN\_N, MAX\_N);

End;

Procedure TMainTaskForm.I1EditKeyPress(Sender: TObject; Var Key: Char);

Var

MAX\_N\_EDIT: Integer;

Begin

CheckComboButtons(Key, I1Edit);

MAX\_N\_EDIT := StrtoInt(EnterNEdit.Text);

CheckInput(I1Edit.Text, Key, 1, MAX\_N\_EDIT);

End;

Procedure TMainTaskForm.J1EditKeyPress(Sender: TObject; Var Key: Char);

Var

MAX\_M\_EDIT: Integer;

Begin

CheckComboButtons(Key, J1Edit);

MAX\_M\_EDIT := StrtoInt(EnterMEdit.Text);

CheckInput(J1Edit.Text, Key, 1, MAX\_M\_EDIT);

End;

Procedure TMainTaskForm.I2EditKeyPress(Sender: TObject; Var Key: Char);

Var

MAX\_N\_EDIT: Integer;

Begin

CheckComboButtons(Key, I2Edit);

MAX\_N\_EDIT := StrtoInt(EnterNEdit.Text);

CheckInput(I2Edit.Text, Key, 1, MAX\_N\_EDIT);

End;

Procedure TMainTaskForm.J2EditKeyPress(Sender: TObject; Var Key: Char);

Var

MAX\_M\_EDIT: Integer;

Begin

CheckComboButtons(Key, J2Edit);

MAX\_M\_EDIT := StrtoInt(EnterMEdit.Text);

CheckInput(J2Edit.Text, Key, 1, MAX\_M\_EDIT);

End;

Procedure TMainTaskForm.IsEnterEditsEmpty(EditM, EditN: TEdit);

Begin

If (EditM.Text = '') Or (EditN.Text = '') Then

Begin

Saved := True;

SaveMenu.Enabled := False;

StringGrid.Visible := False;

ArrLabel.Visible := False;

ClearGrid(StringGrid);

ResultButton.Enabled := False;

End

Else

Begin

StringGrid.Visible := True;

ArrLabel.Visible := True;

ClearGrid(StringGrid);

FillGrid(StrtoInt(EditM.Text), StrtoInt(EditN.Text), StringGrid);

End;

End;

Procedure TMainTaskForm.ButtonOn(I1, J1, I2, J2: TEdit);

Begin

If (I1.Text = '') Or (J1.Text = '') Or (J2.Text = '') Or (I2.Text = '') Then

ResultButton.Enabled := False

Else

ResultButton.Enabled := True;

End;

Procedure TMainTaskForm.MakeUnVisibleAndClear(I1, J1, I2, J2: TEdit);

Begin

StartLabel.Visible := False;

EndLabel.Visible := False;

I1.Visible := False;

J1.Visible := False;

I2.Visible := False;

J2.Visible := False;

I1.Text := '';

J1.Text := '';

I2.Text := '';

J2.Text := '';

End;

Procedure TMainTaskForm.MakeVisible(I1, J1, I2, J2: TEdit);

Begin

StartLabel.Visible := True;

EndLabel.Visible := True;

I1.Visible := True;

J1.Visible := True;

I2.Visible := True;

J2.Visible := True;

End;

Procedure TMainTaskForm.J1EditChange(Sender: TObject);

Begin

ButtonOn(I1Edit, J1Edit, J2Edit, I2Edit);

End;

Procedure TMainTaskForm.J2EditChange(Sender: TObject);

Begin

ButtonOn(I1Edit, J1Edit, J2Edit, I2Edit);

End;

Procedure TMainTaskForm.EnterNEditChange(Sender: TObject);

Begin

IsEnterEditsEmpty(EnterMEdit, EnterNEdit);

OutEdit.Text := '';

MakeUnVisibleAndClear(I1Edit, J1Edit, I2Edit, J2Edit);

End;

Procedure TMainTaskForm.EnterMEditChange(Sender: TObject);

Begin

IsEnterEditsEmpty(EnterMEdit, EnterNEdit);

OutEdit.Text := '';

MakeUnVisibleAndClear(I1Edit, J1Edit, I2Edit, J2Edit);

End;

Procedure TMainTaskForm.I1EditChange(Sender: TObject);

Begin

ButtonOn(I1Edit, J1Edit, J2Edit, I2Edit);

End;

Procedure TMainTaskForm.I2EditChange(Sender: TObject);

Begin

ButtonOn(I1Edit, J1Edit, J2Edit, I2Edit);

End;

Procedure BlockClick(Edit: TEdit);

Begin

If Edit.SelStart <> Length(Edit.Text) Then

Edit.SelStart := Length(Edit.Text);

End;

Procedure TMainTaskForm.J2EditClick(Sender: TObject);

Begin

BlockClick(J2Edit)

End;

Procedure TMainTaskForm.J2EditContextPopup(Sender: TObject; MousePos: TPoint;

Var Handled: Boolean);

Begin

Handled := True;

End;

Procedure TMainTaskForm.J2EditKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Begin

CheckShftAndArrows(Key, Shift);

End;

Procedure TMainTaskForm.J1EditClick(Sender: TObject);

Begin

BlockClick(J1Edit)

End;

Procedure TMainTaskForm.J1EditContextPopup(Sender: TObject; MousePos: TPoint;

Var Handled: Boolean);

Begin

Handled := True;

End;

Procedure TMainTaskForm.J1EditKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Begin

CheckShftAndArrows(Key, Shift);

End;

Procedure TMainTaskForm.EnterMEditClick(Sender: TObject);

Begin

BlockClick(EnterNEdit);

End;

Procedure TMainTaskForm.I2EditClick(Sender: TObject);

Begin

BlockClick(I2Edit)

End;

Procedure TMainTaskForm.I2EditContextPopup(Sender: TObject; MousePos: TPoint;

Var Handled: Boolean);

Begin

Handled := True;

End;

Procedure TMainTaskForm.I2EditKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Begin

CheckShftAndArrows(Key, Shift);

End;

Procedure TMainTaskForm.EnterNEditClick(Sender: TObject);

Begin

BlockClick(EnterNEdit);

End;

Procedure TMainTaskForm.I1EditClick(Sender: TObject);

Begin

BlockClick(I1Edit)

End;

Procedure TMainTaskForm.I1EditContextPopup(Sender: TObject; MousePos: TPoint;

Var Handled: Boolean);

Begin

Handled := True;

End;

Procedure TMainTaskForm.I1EditKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Begin

CheckShftAndArrows(Key, Shift);

End;

Procedure TMainTaskForm.EnterNEditContextPopup(Sender: TObject;

MousePos: TPoint; Var Handled: Boolean);

Begin

Handled := True;

End;

Procedure TMainTaskForm.EnterNEditKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Begin

CheckShftAndArrows(Key, Shift);

End;

Procedure TMainTaskForm.EnterNEditKeyPress(Sender: TObject; Var Key: Char);

Begin

CheckComboButtons(Key, EnterNEdit);

CheckInput(EnterNEdit.Text, Key, MIN\_N, MAX\_N);

End;

Function IsGridFull(StringGrid: TStringGrid): Boolean;

Var

I, J, EmptyCount: Integer;

Begin

EmptyCount := 0;

For I := 0 To StringGrid.ColCount - 1 Do

Begin

For J := 0 To StringGrid.RowCount - 1 Do

Begin

If (StringGrid.Cells[I, J] = '') Or

(StringGrid.Cells[I, J] = '-') Then

Inc(EmptyCount);

End;

End;

Result := EmptyCount = 0;

End;

Procedure TMainTaskForm.ResultButtonClick(Sender: TObject);

Var

ResMatrix: TMatrix;

Begin

OutEdit.Text := '';

MainFunc();

Saved := False;

SaveMenu.Enabled := True;

If OutEdit.Text = '' Then

Begin

Saved := True;

SaveMenu.Enabled := False;

End

Else

Begin

Saved := False;

SaveMenu.Enabled := True;

End;

End;

Procedure TMainTaskForm.StringGridSetEditText(Sender: TObject;

ACol, ARow: Integer; Const Value: String);

Begin

If IsGridFull(StringGrid) And (EnterNEdit.Text <> '') And

(EnterMEdit.Text <> '') Then

Begin

MakeVisible(I1Edit, J1Edit, I2Edit, J2Edit);

End

Else

Begin

MakeUnVisibleAndClear(I1Edit, J1Edit, I2Edit, J2Edit);

OutEdit.Text := '';

Saved := True;

SaveMenu.Enabled := False;

End;

OutEdit.Text := '';

SaveMenu.Enabled := False;

End;

Procedure CheckComboButtonsGrid(Var Key: Char; StringGrid: TEStringGrid);

Begin

If (Key = #22) Or ((Key = 'v') And (GetKeyState(VK\_CONTROL) < 0)) Then

Key := NONE;

If Not CharInSet(Key, DIGITS) And (Key <> #8) And (Key <> '-') Then

Key := NONE;

If (Pos('-', StringGrid.Cells[StringGrid.Col, StringGrid.Row]) = 1) And

(StringGrid.InplaceEditor.SelStart = 0) Then

Key := NONE;

If (Key = '0') And (Length(StringGrid.Cells[StringGrid.Col, StringGrid.Row])

> 0) And (StringGrid.InplaceEditor.SelStart = 0) Then

Key := #0;

If (StringGrid.Cells[StringGrid.Col, StringGrid.Row] = '-') And

((Not CharInSet(Key, NO\_ZERO\_DIGITS)) Or (Key = '-')) And

(Key <> BACKSPACE) Then

Key := #0;

If (Key = '-') And (StringGrid.InplaceEditor.SelStart > 0) Then

Key := NONE;

If (Pos('-', StringGrid.Cells[StringGrid.Col, StringGrid.Row]) = 1) And

(StringGrid.InplaceEditor.SelStart = 0) And (Key = '0') Then

Key := NONE;

If (Length(StringGrid.Cells[StringGrid.Col, StringGrid.Row]) > 0) And

(StringGrid.Cells[StringGrid.Col, StringGrid.Row] = '0') And

(Key <> BACKSPACE) Then

Begin

Key := NONE;

End;

If (StringGrid.Cells[StringGrid.Col, StringGrid.Row] <> '') And

(StringGrid.InplaceEditor.SelStart = 0) And (Key = #8) Then

Key := #0;

End;

Procedure TMainTaskForm.StringGridKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Begin

CheckShftAndArrows(Key, Shift);

End;

Procedure TMainTaskForm.StringGridKeyPress(Sender: TObject; Var Key: Char);

Var

NumsGrid: TEStringGrid;

Begin

NumsGrid := TEStringGrid(Sender);

CheckComboButtonsGrid(Key, NumsGrid);

CheckInput(StringGrid.Cells[StringGrid.Col, StringGrid.Row], Key,

MIN\_X, MAX\_X);

End;

Procedure TMainTaskForm.StringGridMouseActivate(Sender: TObject;

Button: TMouseButton; Shift: TShiftState; X, Y, HitTest: Integer;

Var MouseActivate: TMouseActivate);

Begin

Clipboard.Clear;

End;

End.

Unit Developer;

Interface

Uses

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

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls;

Type

TDeveloperForm = Class(TForm)

DeveloperLabel: TLabel;

Procedure FormCreate(Sender: TObject);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

DeveloperForm: TDeveloperForm;

Implementation

{$R \*.dfm}

Procedure CenterDeveloperFormOnScreen(DeveloperForm: TDeveloperForm);

Begin

DeveloperForm.Left := (Screen.Width - DeveloperForm.Width) Div 2;

DeveloperForm.Top := (Screen.Height - DeveloperForm.Height) Div 2;

End;

Procedure TDeveloperForm.FormCreate(Sender: TObject);

Begin

CenterDeveloperFormOnScreen(Self);

DeveloperLabel.Caption := 'Разработчик: Бражалович Александр Иванович' +

#13#10 + 'Группа: 351005' + #13#10 + 'Tg: @Sunn4es';

DeveloperLabel.Update;

End;

End.

Unit Instruction;

Interface

Uses

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

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls;

Type

TInstructionForm = Class(TForm)

InstructionLabel: TLabel;

Procedure FormCreate(Sender: TObject);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

InstructionForm: TInstructionForm;

Implementation

{$R \*.dfm}

Procedure CenterFormOnScreen(InstructionForm: TInstructionForm);

Begin

InstructionForm.Left := (Screen.Width - InstructionForm.Width) Div 2;

InstructionForm.Top := (Screen.Height - InstructionForm.Height) Div 2;

End;

Procedure TInstructionForm.FormCreate(Sender: TObject);

Begin

CenterFormOnScreen(Self);

InstructionLabel.Caption := '1. Введите порядок матрицы N [2; 4].' + #13#10

+ '2. Введите элементы мтарицы [-99; 100]' + #13#10 +

'3. В "начальный элемент" нужно ввести координаты точки' + #13#10 +

' от которой вы хотите искать путь. В "конечный элемент"' + #13#10 +

' координаты на котором вы хотите закончить обход матрицы.' + #13#10 +

'4.1 В файле должно содержаться количество столбцов матрицы на первой строке.'

+ #13#10 + ' Во второй строке количество строк матрицы.' + #13#10 +

'4.2 На третей и четвёртой строках координаты I пути обхода' + #13#10 +

' (третья строка I1, четвёртая строка I2).' + #13#10 +

'4.3 На пятой и шестой строках координаты J пути обхода' + #13#10 +

' (пятая строка J1, шестая строка J2)' + #13#10 +

'4.4 С седьмой строки записываются элементы матрицы.' + #13#10 +

' Элементы строк матрцы записываются через пробел.' + #13#10 +

' Далее строки матрицы записываются с новой строки.';

End;

End.

**Код программы Java:**

import java.util.Scanner;

import java.io.File;

import java.io.FileWriter;

public class Main {

public static final int MIN\_MATRIX = 1;

public static final int MAX\_MATRIX = 4;

public static final int MIN\_ELEMENT = -100;

public static final int MAX\_ELEMENT = 100;

public static String path = "";

public static int stepCounter = 0;

public static int maxSum = 0;

public static int[][] coordArr = new int[2][2];

public enum ERRORS\_LIST {

CORRECT,

RANGE\_ERR,

NUM\_ERR,

NOT\_TXT,

PATH\_ERR,

NOT\_EXIST,

NOT\_READABLE,

NOT\_WRITEABLE,

CHOICE\_ERR,

FILE\_EMPTY,

EXTRA\_DATA

}

public static final String[]

ERRORS = {

"",

"Значение не попадает в диапазон!",

"Проверьте корректность ввода данных!",

"Расширение не txt!",

"Проверьте корректность ввода пути к файлу!",

"Файл закрыт для чтения!",

"Файл закрыт для записи!",

"Значения порядков не равны!",

"Проверьте корректность выбора!",

"Файл пуст!",

"Лишние данные!"

};

public static void printTask() {

System.out.println("Данная программа находит в матрице путь от элемента a[i1,j1] до элемента a[i2,j2] с максимальной суммой::\n\n");

}

public static void printError (ERRORS\_LIST error) {

System.out.println(ERRORS[error.ordinal()] + "\nПовторите попытку");

}

public static int chooseOption (Scanner inputScanner) {

ERRORS\_LIST error;

int option = 0;

String optionStr = "";

do {

error = ERRORS\_LIST.CORRECT;

try {

option = inputScanner.nextInt();

} catch (NumberFormatException e) {

error = ERRORS\_LIST.CHOICE\_ERR;

}

if ((error == ERRORS\_LIST.CORRECT) && (option != 1) && (option != 2)) {

error = ERRORS\_LIST.NUM\_ERR;

}

if ((error != ERRORS\_LIST.CORRECT) && (optionStr != "")) {

printError(error);

}

} while (error != ERRORS\_LIST.CORRECT);

return option;

}

public static String readPath (Scanner inputScanner) {

String pathTofile = "";

ERRORS\_LIST error;

do {

System.out.print("Введите путь к txt файлу: ");

pathTofile = inputScanner.nextLine();

if (pathTofile.equals("")) {

pathTofile = inputScanner.nextLine();

}

if (!pathTofile.endsWith(".txt")) {

error = ERRORS\_LIST.NOT\_TXT;

} else {

error = ERRORS\_LIST.CORRECT;

}

if (error != ERRORS\_LIST.CORRECT)

printError(error);

} while (error != ERRORS\_LIST.CORRECT);

return pathTofile;

}

public static File fileReading (Scanner inputScanner) {

ERRORS\_LIST error;

String pathToFile = "";

File file;

do {

error = ERRORS\_LIST.CORRECT;

pathToFile = readPath(inputScanner);

file = new File(pathToFile);

if (!file.exists())

error = ERRORS\_LIST.NOT\_EXIST;

if ((error == ERRORS\_LIST.CORRECT) && (!file.canRead()))

error = ERRORS\_LIST.NOT\_READABLE;

if ((error == ERRORS\_LIST.CORRECT) && (file.length() == 0))

error = ERRORS\_LIST.FILE\_EMPTY;

if (error != ERRORS\_LIST.CORRECT)

printError(error);

} while (error != ERRORS\_LIST.CORRECT);

return file;

}

public static File fileWriting(Scanner inputScanner) {

ERRORS\_LIST error;

File file;

String pathToFile = "";

do {

pathToFile = readPath(inputScanner);

file = new File(pathToFile);

error = ERRORS\_LIST.CORRECT;

if (!file.exists())

error = ERRORS\_LIST.NOT\_EXIST;

if ((error == ERRORS\_LIST.CORRECT) && !file.canWrite())

error = ERRORS\_LIST.NOT\_WRITEABLE;

if (error != ERRORS\_LIST.CORRECT)

printError(error);

} while (error != ERRORS\_LIST.CORRECT);

return file;

}

public static ERRORS\_LIST readOneNum(Scanner inputScanner, int[] numberArr, final int MIN, final int MAX, boolean isFile) {

int number = 0;

ERRORS\_LIST error;

error = ERRORS\_LIST.CORRECT;

try {

if (isFile)

number = (inputScanner.nextInt());

else

number = Integer.parseInt(inputScanner.nextLine());

} catch (NumberFormatException e) {

error = ERRORS\_LIST.NUM\_ERR;

}

if (error == ERRORS\_LIST.CORRECT && ((number < MIN) || (number > MAX)))

error = ERRORS\_LIST.RANGE\_ERR;

numberArr[0] = error == ERRORS\_LIST.CORRECT ? number : 0;

return error;

}

public static ERRORS\_LIST checkSpaceInFile(String bufStr) {

int i;

ERRORS\_LIST error;

i = 0;

error = ERRORS\_LIST.CORRECT;

while ((error == ERRORS\_LIST.CORRECT) && (i < bufStr.length())) {

if (bufStr.charAt(i) != ' ')

error = ERRORS\_LIST.EXTRA\_DATA;

i++;

}

return error;

}

public static int[][] readMatrix(Scanner inputScanner, int[] numberArr, int m, int n, int option) {

int i;

int j;

int[][] matrix;

String bufStr;

ERRORS\_LIST error;

error = ERRORS\_LIST.CORRECT;

i = 0;

matrix = new int[n][m];

while (i < n && error == ERRORS\_LIST.CORRECT) {

j = 0;

while (j < m && error == ERRORS\_LIST.CORRECT) {

if (option == 1) {

error = readOneNum(inputScanner, numberArr, MIN\_ELEMENT, MAX\_ELEMENT, true);

if (error == ERRORS\_LIST.CORRECT)

matrix[i][j] = numberArr[0];

} else {

do {

System.out.print("Введите в " + (i + 1) + " строке " + (j + 1) + " столбце элемент матрицы [" + MIN\_ELEMENT + "; " + MAX\_ELEMENT + "]: ");

error = readOneNum(inputScanner, numberArr, MIN\_ELEMENT, MAX\_ELEMENT, true);

if (error != ERRORS\_LIST.CORRECT)

System.out.println(ERRORS[error.ordinal()]);

else

matrix[i][j] = numberArr[0];

} while (error != ERRORS\_LIST.CORRECT);

}

j++;

}

if (option == 1 && error == ERRORS\_LIST.CORRECT) {

bufStr = inputScanner.nextLine();

error = checkSpaceInFile(bufStr);

}

i++;

}

if (error != ERRORS\_LIST.CORRECT) {

matrix = null;

printError(error);

}

return matrix;

}

public static int[][] readConsoleMatrix(Scanner inputScanner) {

int[] numberM = new int[1];

int[] numberN = new int[1];

int[] numberArr = new int[1];

int m;

int n;

int[][] matrix = {};

ERRORS\_LIST error;

do {

System.out.print("Введите порядок матрицы m [" + MIN\_MATRIX + "; " + MAX\_MATRIX + "]: ");

error = readOneNum(inputScanner, numberM, MIN\_MATRIX, MAX\_MATRIX, true);

if (error == ERRORS\_LIST.CORRECT) {

System.out.print("Введите порядок матрицы n [" + MIN\_MATRIX + "; " + MAX\_MATRIX + "]: ");

error = readOneNum(inputScanner, numberN, MIN\_MATRIX, MAX\_MATRIX, true);

}

if (error != ERRORS\_LIST.CORRECT)

printError(error);

else {

m = numberM[0];

n = numberN[0];

matrix = readMatrix(inputScanner, numberArr, m, n, 2);

}

} while (error != ERRORS\_LIST.CORRECT);

return matrix;

}

public static int[][] readFileMatrix(Scanner inputScanner) {

String bufStr;

int[] numberArr = new int[1];

int[] numberM = new int[1];

int[] numberN = new int[1];

int m;

int n;

int[][] matrix = {};

ERRORS\_LIST error;

File file;

do {

file = fileReading(inputScanner);

try(Scanner scanFile = new Scanner(file)) {

error = readOneNum(scanFile, numberM, MIN\_MATRIX, MAX\_MATRIX, true);

if (error == ERRORS\_LIST.CORRECT) {

bufStr = scanFile.nextLine();

error = checkSpaceInFile(bufStr);

}

error = readOneNum(scanFile, numberN, MIN\_MATRIX, MAX\_MATRIX, true);

if (error == ERRORS\_LIST.CORRECT) {

bufStr = scanFile.nextLine();

error = checkSpaceInFile(bufStr);

}

if (error != ERRORS\_LIST.CORRECT)

printError(error);

else {

m = numberM[0];

n = numberN[0];

matrix = readMatrix(scanFile, numberArr, m, n, 1);

if (matrix != null) {

coordArr = readFileCoord(scanFile, matrix);

}

}

} catch (Exception e) {

error = ERRORS\_LIST.NOT\_READABLE;

printError(error);

}

if (matrix == null)

error = ERRORS\_LIST.NUM\_ERR;

} while (error != ERRORS\_LIST.CORRECT);

return matrix;

}

public static int inputOption(Scanner inpuScanner) {

int option = 0;

File file;

System.out.println("\nВыберете способ ввода данных:");

System.out.println("Через файл - 1");

System.out.println("Через консоль - 2");

option = chooseOption(inpuScanner);

return option;

}

public static int[][] inputMatrix(Scanner inputScanner, int option) {

int[][] matrix;

if (option == 1) {

matrix = readFileMatrix(inputScanner);

} else {

matrix = readConsoleMatrix(inputScanner);

}

return matrix;

}

public static int[][] readConsoleCoord (Scanner inpScanner, int [][] matrix) {

ERRORS\_LIST error;

int [][] coordArr = new int[2][2];

int [] numberI1 = new int[1];

int [] numberJ1 = new int[1];

int [] numberI2 = new int[1];

int [] numberJ2 = new int[1];

do {

System.out.print("Введите координату стратового элемента I1 [" + 1 + "; " + (matrix.length) + "]: ");

error = readOneNum(inpScanner, numberI1, MIN\_MATRIX, (matrix.length), true);

if (error == ERRORS\_LIST.CORRECT) {

System.out.print("Введите координату стратового элемента J1 [" + 1 + "; " + (matrix[0].length) + "]: ");

error = readOneNum(inpScanner, numberJ1, MIN\_MATRIX, (matrix[0].length), true);

}

if (error == ERRORS\_LIST.CORRECT) {

System.out.print("Введите координату стратового элемента I2 [" + 1 + "; " + (matrix.length) + "]: ");

error = readOneNum(inpScanner, numberI2, MIN\_MATRIX, (matrix.length), true);

}

if (error == ERRORS\_LIST.CORRECT) {

System.out.print("Введите координату стратового элемента J2 [" + 1 + "; " + (matrix[0].length) + "]: ");

error = readOneNum(inpScanner, numberJ2, MIN\_MATRIX, (matrix[0].length), true);

}

if (error != ERRORS\_LIST.CORRECT) {

printError(error);

} else {

coordArr[0][0] = numberI1[0] - 1;

coordArr[1][0] = numberJ1[0] - 1;

coordArr[0][1] = numberI2[0] - 1;

coordArr[1][1] = numberJ2[0] - 1;

}

} while (error != ERRORS\_LIST.CORRECT);

return coordArr;

}

public static int[][] readFileCoord (Scanner inpScanner, int[][] matrix) {

ERRORS\_LIST errors;

int [] numberI1 = new int[1];

int [] numberJ1 = new int[1];

int [] numberI2 = new int[1];

int [] numberJ2 = new int[1];

int shift = matrix.length + 2;

do {

errors = readOneNum(inpScanner, numberI1, 1, matrix.length, true);

if (errors == ERRORS\_LIST.CORRECT) {

errors = readOneNum(inpScanner, numberJ1, 1, matrix[0].length, true);

}

if (errors == ERRORS\_LIST.CORRECT) {

errors = readOneNum(inpScanner, numberI2, 1, matrix.length, true);

}

if (errors == ERRORS\_LIST.CORRECT) {

errors = readOneNum(inpScanner, numberJ2, 1, matrix[0].length, true);

}

if (errors == ERRORS\_LIST.CORRECT) {

coordArr[0][0] = numberI1[0] - 1;

coordArr[1][0] = numberJ1[0] - 1;

coordArr[0][1] = numberI2[0] - 1;

coordArr[1][1] = numberJ2[0] - 1;

} else {

printError(errors);

}

} while (errors != ERRORS\_LIST.CORRECT);

return coordArr;

}

public static int[][] inputCoord (Scanner inputScanner, int option, int[][] matrix) {

if (option == 2) {

coordArr = readConsoleCoord(inputScanner,matrix);

}

return coordArr;

}

public static void findpath (int[][] matrix, int i1, int j1, int i2, int j2, int currentSum, String currentPath, int [][] steps) {

if (!((i1 < 0) || (i1 > matrix.length - 1) || (j1 < 0) || (j1 > matrix[0].length - 1) || (steps[i1][j1] >= 2))) {

stepCounter++;

currentPath += String.format("(%d). [%d,%d]; ", stepCounter, i1 + 1, j1 + 1);

steps[i1][j1]++;

currentSum += matrix[i1][j1];

if ((i1 == i2) && (j1 == j2)) {

if (currentSum > maxSum) {

maxSum = currentSum;

path = currentPath;

}

} else {

findpath(matrix, i1, j1 + 1, i2, j2, currentSum, currentPath, steps);

findpath(matrix,i1 + 1, j1, i2, j2, currentSum, currentPath,steps);

findpath(matrix, i1 , j1 - 1, i2, j2, currentSum, currentPath, steps);

findpath(matrix, i1 - 1, j1, i2, j2, currentSum, currentPath, steps);

}

steps[i1][j1]--;

stepCounter--;

currentPath = currentPath.substring(0, currentPath.length() - 7);

}

}

public static int[][] createZeroMatrix (int [][] matrix) {

int [][] zeroMatrix = new int[matrix.length ][matrix[0].length ];

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

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

zeroMatrix[i][j] = 0;

}

}

return zeroMatrix;

}

public static void printResult(Scanner inputScanner,String path) {

ERRORS\_LIST error;

File file;

int option = 0;

System.out.println("\nВыберете способ вывода результата:");

System.out.println("Через файл - 1");

System.out.println("Через консоль - 2");

option = chooseOption(inputScanner);

if (option == 1)

{

file = fileWriting(inputScanner);

try(FileWriter writer = new FileWriter(file, true)) {

writer.write("\nПуть обхода матрицы с максимальной суммой: " + path);

} catch (Exception e) {

error = ERRORS\_LIST.NOT\_WRITEABLE;

System.out.println(ERRORS[error.ordinal()]);

}

} else {

System.out.printf("\nПуть обхода матрицы с максимальной суммой: " + path);

}

}

public static void main(String[] args) {

Scanner inpScanner = new Scanner(System.in);

int[][] steps;

int [][] matrix;

int sum = 0;

int option;

printTask();

option = inputOption(inpScanner);

matrix = inputMatrix(inpScanner, option);

steps = createZeroMatrix(matrix);

coordArr = inputCoord(inpScanner, option, matrix);

findpath(matrix, coordArr[0][0],coordArr[1][0], coordArr[0][1], coordArr[1][1], sum, path, steps);

printResult(inpScanner, path);

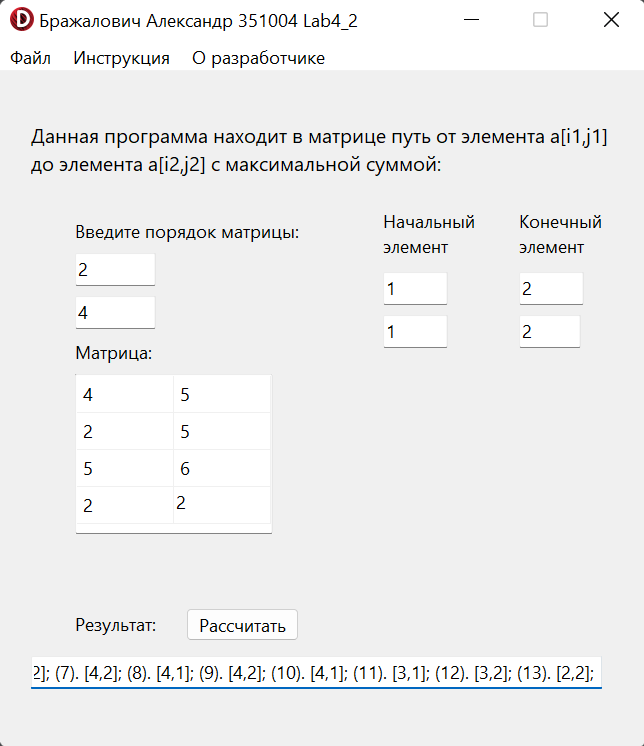
inpScanner.close();

}

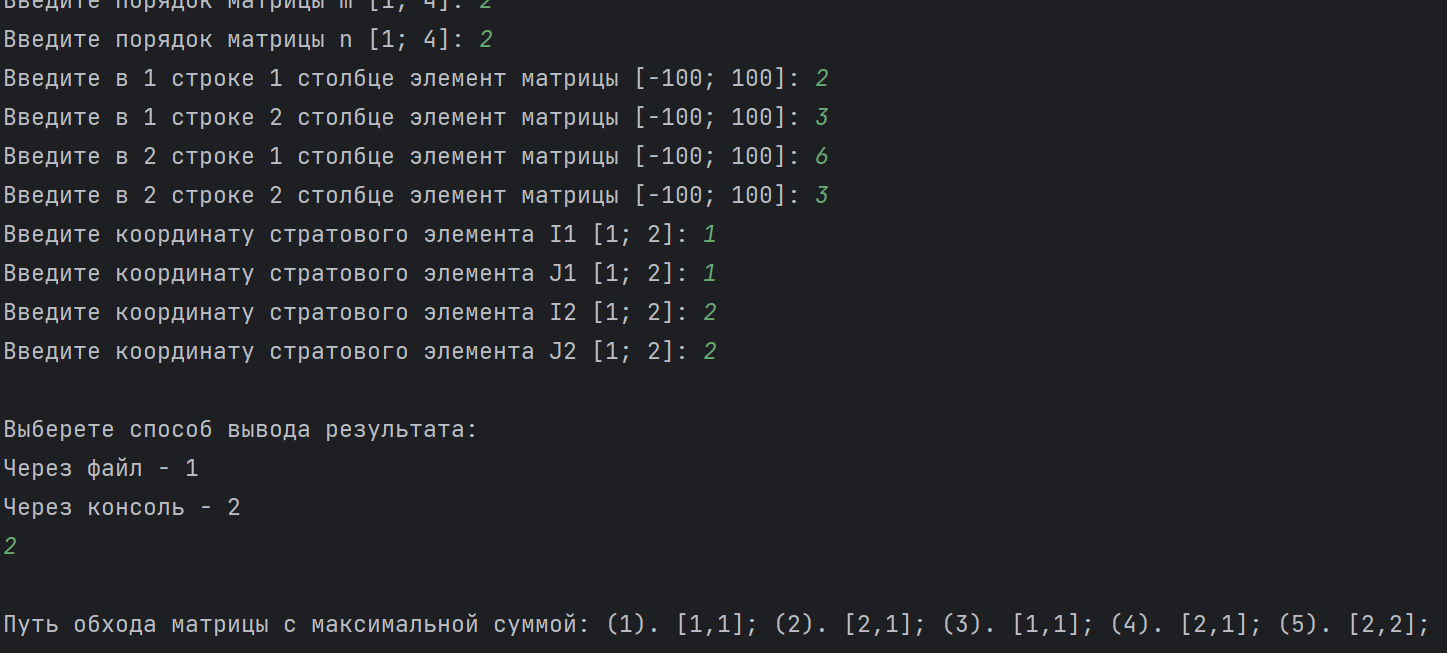
}

**Скриншоты:**

**Delphi:**

****

**Java:**

****

**Блок-схема:**

