УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №4.2

по предмету «Основы алгоритмизации и программирования»

Вариант 20

Выполнил:

Захвей И.В.

Гр. 351005

Проверил:

Данилова Г. В.

Минск 2024

**Задание:**

Вычислить определитель заданной матрицы, пользуясь формулой разложения по первой строке

**Код программы Delphi:**

**MainUnit.pas**

Unit ManeUnit;

Interface

Uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Classes, Vcl.Forms,

Vcl.ExtDlgs, Vcl.Menus, Vcl.StdCtrls, Vcl.Grids, Vcl.Dialogs, Vcl.Controls,

ManualUnit, DevInfUnit;

Type

TManeForm = Class(TForm)

MainMenu: TMainMenu;

FileButtonMenu: TMenuItem;

OpenButtonMenu: TMenuItem;

SaveButtonMenu: TMenuItem;

LineMenu: TMenuItem;

ExitButtonMenu: TMenuItem;

ManualButtonMenu: TMenuItem;

DeveloperButtonMenu: TMenuItem;

PopupMenu: TPopupMenu;

OpenTextFileDialog: TOpenTextFileDialog;

SaveTextFileDialog: TSaveTextFileDialog;

CheckButton: TButton;

InfLabel: TLabel;

SizeEdit: TEdit;

SizeLabel: TLabel;

MatrixLabel: TLabel;

MatrixGrid: TStringGrid;

DetLabel: TLabel;

DetEdit: TEdit;

Procedure ManualButtonMenuClick(Sender: TObject);

Procedure DeveloperButtonMenuClick(Sender: TObject);

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

Procedure ExitButtonMenuClick(Sender: TObject);

Procedure FormCreate(Sender: TObject);

Procedure OpenButtonMenuClick(Sender: TObject);

Procedure SaveButtonMenuClick(Sender: TObject);

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

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

Shift: TShiftState);

Procedure SizeEditChange(Sender: TObject);

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

Procedure CheckButtonClick(Sender: TObject);

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

Shift: TShiftState);

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

Shift: TShiftState);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Type

ERRORS\_CODE = (SUCCESS, INCORRECT\_DATA\_FILE, A\_LOT\_OF\_DATA\_FILE,

OUT\_OF\_BORDER, OUT\_OF\_BORDER\_SIZE);

TGridCracker = Class(TStringGrid);

IntArr = Array Of Integer;

Matrix = Array Of IntArr;

Const

INFTEXT = 'Вычислить определитель матрицы А';

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

VOID = #0;

BACKSPACE = #8;

MAX\_NUMB = 70;

MIN\_NUMB = -70;

MAX\_SIZE = 5;

MIN\_SIZE = 1;

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

'Данные в файле некорректные',

'В файле неверное количество элементов или стоит

лишний пробел',

'Числа должны быть в диапазоне [-70, 70]',

'Размер должен быть в диапазоне [1, 5]');

Var

ManeForm: TManeForm;

IsSaved: Boolean = True;

Implementation

{$R \*.dfm}

Procedure DeleteColRow(Var NewMatrix: Matrix; OldMatrix: Matrix;

ColInd: Integer; RowInd: Integer = 0);

Var

Size, I, J, CurRow, CurCol: Integer;

Begin

CurCol := 0;

CurRow := 0;

Size := Length(OldMatrix) - 1;

SetLength(NewMatrix, Size, Size);

For I := 0 To High(OldMatrix) Do

Begin

If I <> RowInd Then

Begin

For J := 0 To High(OldMatrix) Do

If J <> ColInd Then

Begin

NewMatrix[CurRow, CurCol] := OldMatrix[I, J];

Inc(CurCol);

End;

CurCol := 0;

Inc(CurRow);

End;

End;

End;

Function CalcDet(CurMatrix: Matrix): Integer;

Var

Det, I, Addition: Integer;

NewMatrix: Matrix;

Begin

Det := 0;

If Length(CurMatrix) = 1 Then

Det := CurMatrix[0, 0]

Else

For I := 0 To High(CurMatrix) Do

If CurMatrix[0, I] <> 0 Then

Begin

DeleteColRow(NewMatrix, CurMatrix, I);

Addition := CurMatrix[0, I] \* CalcDet(NewMatrix);

If I Mod 2 = 1 Then

Addition := -Addition;

Inc(Det, Addition);

End;

CalcDet := Det;

End;

Procedure FillGrid(Size: Integer; Grid: TStringGrid);

Var

I: Integer;

Begin

Grid.Visible := True;

Grid.ColWidths[0] := 50;

If Size > 5 Then

Begin

Grid.Width := (Grid.DefaultColWidth + 3) \* 5 + 25 + Grid.ColWidths[0];

Grid.Height := (Grid.DefaultRowHeight + 3) \* 6 + 25;

End

Else

Begin

Grid.Width := (Grid.DefaultColWidth + 4) \* Size + Grid.ColWidths[0];

Grid.Height := (Grid.DefaultRowHeight + 3) \* (Size + 1);

End;

Grid.ColCount := Size + 1;

Grid.RowCount := Size + 1;

Grid.Cells[0, 0] := '\';

For I := 1 To Size Do

Begin

Grid.Cells[I, 0] := IntToStr(I);

Grid.Cells[0, I] := IntToStr(I);

End;

End;

Procedure ClearGrid(Grid: TStringGrid);

Var

J, I: Integer;

Begin

Grid.Visible := False;

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

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

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

End;

Procedure TManeForm.CheckButtonClick(Sender: TObject);

Var

Arr: Matrix;

I, J: Integer;

Begin

SetLength(Arr, StrToInt(SizeEdit.Text), StrToInt(SizeEdit.Text));

For I := 0 To MatrixGrid.RowCount - 2 Do

For J := 0 To MatrixGrid.ColCount - 2 Do

Arr[J, I] := StrToInt(MatrixGrid.Cells[I + 1, J + 1]);

DetEdit.Text := IntToStr(CalcDet(Arr));

SaveButtonMenu.Enabled := True;

IsSaved := False;

End;

Procedure TManeForm.SizeEditChange(Sender: TObject);

Begin

DetEdit.Text := '';

CheckButton.Enabled := False;

SaveButtonMenu.Enabled := False;

If SizeEdit.Text = '' Then

Begin

MatrixLabel.Visible := False;

ClearGrid(MatrixGrid);

End

Else

Begin

MatrixLabel.Visible := True;

ClearGrid(MatrixGrid);

FillGrid(StrToInt(SizeEdit.Text), MatrixGrid);

End;

End;

Procedure TManeForm.SizeEditKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Begin

If ((SsShift In Shift) Or (SsCtrl In Shift)) And

Not((Key = VK\_RIGHT) Or (Key = VK\_LEFT)) Then

Key := 0;

If Key = VK\_DELETE Then

Key := 0;

If (Key = VK\_DOWN) And (MatrixGrid.Visible) Then

MatrixGrid.SetFocus;

End;

Function InsertKey(Index: Integer; SubStr: Char; SelLen: Integer;

Text: String): String;

Var

ResultText: String;

Begin

ResultText := Text;

If (SubStr = BACKSPACE) And (SelLen = 0) Then

Delete(ResultText, Index, 1)

Else

Begin

Delete(ResultText, Index + 1, SelLen);

If Substr <> BACKSPACE Then

ResultText.Insert(Index, String(SubStr));

End;

InsertKey := ResultText;

End;

Function CountOfSymbolInt(Num: Integer): Integer;

Var

NumLen: Integer;

Begin

NumLen := 0;

If Num < 0 Then

Inc(NumLen);

Repeat

Inc(NumLen);

Num := Num Div 10;

Until (Num = 0);

CountOfSymbolInt := NumLen;

End;

Procedure TotalKeyPress(Var Key: Char; SelStart, SelLength: Integer;

Const MIN, MAX: Integer; Text: String);

Var

ResultNum, RBorder, NumLen: Integer;

Buffer, Output: String;

Begin

Output := InsertKey(SelStart, Key, SelLength, Text);

If (Length(Output) <> 0) And (Output <> '-') And (Output <> '') Then

Begin

Try

ResultNum := StrToInt(Output);

Except

Key := VOID;

End;

If Key <> VOID Then

Begin

NumLen := CountOfSymbolInt(ResultNum);

If NumLen <> Length(Output) Then

Key := VOID;

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

Key := VOID;

End;

End;

End;

Procedure TManeForm.SizeEditKeyPress(Sender: TObject; Var Key: Char);

Var

ResultNum: Integer;

Begin

TotalKeyPress(Key, SizeEdit.SelStart, SizeEdit.SelLength, MIN\_SIZE,

MAX\_SIZE, SizeEdit.Text);

End;

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

Begin

If SaveButtonMenu.Enabled And Not IsSaved Then

Case Application.MessageBox('Сохранить данные перед выходом?', 'Выход',

MB\_YESNOCANCEL + MB\_ICONQUESTION + MB\_DEFBUTTON3) Of

IDYES:

Begin

SaveButtonMenu.Click;

CanClose := True;

End;

IDNO:

CanClose := True;

IDCANCEL:

CanClose := False;

End

Else

Case Application.MessageBox('Вы точно хотите выйти?', 'Выход',

MB\_YESNO + MB\_ICONQUESTION + MB\_DEFBUTTON2) Of

IDYES:

CanClose := True;

IDNO:

CanClose := False;

End;

End;

Procedure TManeForm.FormCreate(Sender: TObject);

Begin

InfLabel.Caption := INFTEXT;

SizeEdit.Text := '';

End;

Function ReadOneFromFile(Var Numb: Integer; Var MyFile: TextFile;

IsElemRead: Boolean = True): ERRORS\_CODE;

Var

Err: ERRORS\_CODE;

NumbInt: Integer;

NumbStr: String;

Begin

Err := SUCCESS;

NumbInt := 0;

Try

Read(MyFile, NumbInt);

Except

Err := INCORRECT\_DATA\_FILE;

End;

If Err = SUCCESS Then

If IsElemRead Then

If (NumbInt > MAX\_NUMB) Or (NumbInt < MIN\_NUMB) Then

Err := OUT\_OF\_BORDER

Else

Numb := NumbInt

Else If (NumbInt > MAX\_SIZE) Or (NumbInt < MIN\_SIZE) Then

Err := OUT\_OF\_BORDER\_SIZE

Else

Numb := NumbInt;

ReadOneFromFile := Err;

End;

Procedure TManeForm.OpenButtonMenuClick(Sender: TObject);

Var

InfFile: TextFile;

Size, I, J: Integer;

Arr: Matrix;

Err: ERRORS\_CODE;

Begin

Err := SUCCESS;

I := 0;

J := 0;

If OpenTextFileDialog.Execute() Then

Begin

AssignFile(InfFile, OpenTextFileDialog.FileName);

Reset(InfFile);

Err := ReadOneFromFile(Size, InfFile, False);

If Err = SUCCESS Then

Begin

SetLength(Arr, Size, Size);

While (I < Size) And (Err = SUCCESS) Do

Begin

While (J < Size) And (Err = SUCCESS) Do

Begin

If Eof(InfFile) Then

Err := A\_LOT\_OF\_DATA\_FILE;

Err := ReadOneFromFile(Arr[I][J], InfFile);

Inc(J);

End;

J := 0;

Inc(I);

End;

End;

If Not EoF(InfFile) Then

Err := A\_LOT\_OF\_DATA\_FILE;

If Err = SUCCESS Then

Begin

SizeEdit.Text := IntToStr(Size);

FillGrid(Size, MatrixGrid);

For I := 0 To High(Arr) Do

For J := 0 To High(Arr) Do

MatrixGrid.Cells[J + 1, I + 1] := IntToStr(Arr[I][J]);

CheckButton.Enabled := True;

End

Else

Application.MessageBox(PChar(ERRORS[Err]), 'Ошибочка вышла',

MB\_OK + MB\_ICONERROR);

CloseFile(InfFile);

End;

End;

Procedure TManeForm.SaveButtonMenuClick(Sender: TObject);

Var

OutFile: TextFile;

I, J: Integer;

Begin

If SaveTextFileDialog.Execute() Then

Begin

AssignFile(OutFile, SaveTextFileDialog.FileName);

Rewrite(OutFile);

Writeln(OutFile, MatrixLabel.Caption);

With MatrixGrid Do

For I := 1 To RowCount - 1 Do

Begin

For J := 1 To ColCount - 1 Do

Write(OutFile, Cells[J, I] + ' ');

Write(OutFile, #13#10);

End;

Writeln(OutFile, DetLabel.Caption);

Write(OutFile, #13#10 + DetEdit.Text);

CloseFile(OutFile);

IsSaved := True;

End;

End;

Procedure TManeForm.ManualButtonMenuClick(Sender: TObject);

Var

Form2: TManualForm;

Begin

Form2 := TManualForm.Create(Self);

Form2.ShowModal;

Form2.Free;

End;

Procedure TManeForm.CheckButtonKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Begin

If (Key = VK\_UP) Then

MatrixGrid.SetFocus;

End;

Procedure TManeForm.DeveloperButtonMenuClick(Sender: TObject);

Var

Form3: TDeveloperForm;

Begin

Form3 := TDeveloperForm.Create(Self);

Form3.ShowModal;

Form3.Free;

End;

Procedure TManeForm.ExitButtonMenuClick(Sender: TObject);

Begin

Close();

End;

Function IsAllCellFill(Grid: TStringGrid; Key: Char;

CurCell: TInplaceEdit): Boolean;

Var

IsFilled: Boolean;

I, J: Integer;

Begin

IsFilled := True;

For I := 1 To Grid.ColCount - 1 Do

For J := 1 To Grid.RowCount - 1 Do

Begin

If (Grid.Col = I) And (Grid.Row = J) And Not(Key = VOID) Then

Begin

If (Grid.Cells[I, J] = '') And Not CharInSet(Key, DIGITS) Then

IsFilled := False;

With CurCell Do

If (Key = BACKSPACE) And

(InsertKey(SelStart, Key, SelLength, Text) = '') Then

IsFilled := False;

End

Else If (Grid.Cells[I, J] = '') Or (Grid.Cells[I, J] = '-') Then

IsFilled := False;

End;

IsAllCellFill := IsFilled;

End;

Procedure TManeForm.MatrixGridKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Begin

If ((SsShift In Shift) Or (SsCtrl In Shift)) And

Not((Key = VK\_RIGHT) Or (Key = VK\_LEFT)) Then

Key := 0;

If Key = VK\_DELETE Then

Key := 0;

If (Key = VK\_RETURN) And (CheckButton.Enabled) Then

CheckButton.Click;

If (Key = VK\_DOWN) And CheckButton.Enabled And

(MatrixGrid.Row = MatrixGrid.RowCount - 1) Then

CheckButton.SetFocus;

If (Key = VK\_UP) And (MatrixGrid.Row = 1) Then

SizeEdit.SetFocus;

End;

Procedure TManeForm.MatrixGridKeyPress(Sender: TObject; Var Key: Char);

Var

GridCel: TGridCracker;

EditingCell: TInplaceEdit;

Begin

GridCel := TGridCracker(Sender);

EditingCell := GridCel.InplaceEditor;

TotalKeyPress(Key, EditingCell.SelStart, EditingCell.SelLength, MIN\_NUMB,

MAX\_NUMB, EditingCell.Text);

If IsAllCellFill(GridCel, Key, EditingCell) Then

CheckButton.Enabled := True

Else

CheckButton.Enabled := False;

If Key <> VOID Then

Begin

DetEdit.Text := '';

SaveButtonMenu.Enabled := False;

End;

End;

End.

**DevInfUnit.pas**

Unit DevInfUnit;

Interface

Uses

Winapi.Windows, Winapi.Messages, System.Classes, Vcl.Forms,

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

Type

TDeveloperForm = Class(TForm)

InfLabel: TLabel;

Procedure FormCreate(Sender: TObject);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

DeveloperForm: TDeveloperForm;

Implementation

{$R \*.dfm}

Procedure TDeveloperForm.FormCreate(Sender: TObject);

Begin

InfLabel.Caption := 'Студент группы 351005, Захвей Иван'

End;

End.

**ManualUnit.pas**

Unit ManualUnit;

Interface

Uses

Winapi.Windows, Winapi.Messages, System.Classes, Vcl.Forms,

Vcl.StdCtrls, Vcl.Controls;

Type

TManualForm = Class(TForm)

ManualLabel: TLabel;

Procedure FormCreate(Sender: TObject);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Const

TextInf = '1. Элементы матрицы должны быть в диапазоне ' +

'от -70 до 70 включичельно' + #13#10 +

'2. Размер матрицы должен находиться' + ' в границах от 1 до 5' + #13#10 +

'3. В файле должен быть размер матрицы ' + 'и его элементы как в матрице';

Var

ManualForm: TManualForm;

Implementation

{$R \*.dfm}

Procedure TManualForm.FormCreate(Sender: TObject);

Begin

ManualLabel.Caption := TextInf;

End;

End.

**Код программы Java:**

import java.io.File;

import java.io.IOException;

import java.io.PrintWriter;

import java.nio.file.Paths;

import java.util.Scanner;

public class Main {

enum ErrCode {

SUCCESS,

INCORRECT\_DATA,

EMPTY\_LINE,

NOT\_TXT,

FILE\_NOT\_EXIST,

NOT\_ENOUGH\_DATA\_FILE,

A\_LOT\_OF\_DATA\_FILE,

IN\_OUT\_FILE\_EXCEPTION,

}

static final int MAX\_NUMB = 70,

MIN\_NUMB = -70,

MIN\_SIZE = 1,

MAX\_SIZE = 5,

MIN\_CHOICE = 1,

MAX\_CHOICE = 2;

static final String[] ERRORS = {"Successfully",

"Data is not correct, or number is too large (must be from %d

to %d)\n",

"Line is empty, please be careful",

"This is not a .txt file",

"This file is not exist",

"Data in file is not enough",

"A lot of elements in file",

"Exception with output/input from the file"};

static ErrCode readOneNum(Scanner input, int[] numberArr, final int MIN, final int MAX,

boolean isFile) {

int number = 0;

ErrCode err = ErrCode.SUCCESS;

try {

number = isFile ? Integer.parseInt(input.next()) :

Integer.parseInt(input.nextLine());

} catch (NumberFormatException e) {

err = ErrCode.INCORRECT\_DATA;

}

if ((err.equals(ErrCode.SUCCESS)) && (number < MIN || number > MAX))

err = ErrCode.INCORRECT\_DATA;

numberArr[0] = err == ErrCode.SUCCESS ? number : 0;

return err;

}

static int userChoice(Scanner input) {

int[] choiceArr = {0};

ErrCode err;

do {

System.out.println("Choose a way of input/output of data\n"

+ "1 -- Console\n"

+ "2 -- File");

err = readOneNum(input, choiceArr, 1, 2, false);

if (err != ErrCode.SUCCESS) {

System.err.printf(ERRORS[err.ordinal()], MIN\_CHOICE, MAX\_CHOICE);

}

} while (err != ErrCode.SUCCESS);

return choiceArr[0];

}

static int inputSizeConsole(Scanner input) {

int[] sizeArr = {0};

ErrCode err;

do {

err = readOneNum(input, sizeArr, MIN\_SIZE, MAX\_SIZE, false);

if (err != ErrCode.SUCCESS) {

System.err.printf(ERRORS[err.ordinal()], MIN\_SIZE, MAX\_SIZE);

System.out.println("Try again");

}

} while (err != ErrCode.SUCCESS);

return sizeArr[0];

}

static int[][] inputFromConsole(Scanner input) {

int size;

ErrCode err;

System.out.println("Enter size of matrix");

size = inputSizeConsole(input);

int[][] matrix = new int[size][size];

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

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

int[] numArr = {0};

do {

System.out.println("Enter element a[" + (i+1) + ", " + (j+1) + "]");

err = readOneNum(input, numArr, MIN\_NUMB, MAX\_NUMB, false);

if (err != ErrCode.SUCCESS)

System.err.printf(ERRORS[err.ordinal()], MIN\_NUMB, MAX\_NUMB);

} while (err != ErrCode.SUCCESS);

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

}

return matrix;

}

static ErrCode validateFileExistence(String fileName) {

File file = new File(fileName);

return file.exists() ? ErrCode.SUCCESS : ErrCode.FILE\_NOT\_EXIST;

}

static ErrCode validateFileExtension(String fileName) {

return fileName.endsWith(".txt") ? ErrCode.SUCCESS : ErrCode.NOT\_TXT;

}

static ErrCode enterFileName(String[] fileName, Scanner input) {

ErrCode err;

fileName[0] = input.nextLine();

if (fileName[0].isEmpty())

err = ErrCode.EMPTY\_LINE;

else {

err = validateFileExistence(fileName[0]);

if (err.equals(ErrCode.SUCCESS)) {

err = validateFileExtension(fileName[0]);

}

}

return err;

}

static String getFileName(Scanner input) {

String[] fileName = {""};

ErrCode err;

System.out.println("Enter full path to file");

do {

err = enterFileName(fileName, input);

if (err != ErrCode.SUCCESS) {

System.err.println(ERRORS[err.ordinal()]);

}

} while (err != ErrCode.SUCCESS);

return fileName[0];

}

static ErrCode readFile(int[][] matrix, Scanner file) {

ErrCode err = ErrCode.SUCCESS;

for (int i = 0; i < matrix.length && err == ErrCode.SUCCESS; i++)

for (int j = 0; j < matrix.length && err == ErrCode.SUCCESS; j++) {

int[] arrNum = new int[1];

if (file.hasNext()) {

err = readOneNum(file, arrNum, MIN\_NUMB, MAX\_NUMB, true);

if (err.equals(ErrCode.SUCCESS)) {

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

}

} else {

err = ErrCode.NOT\_ENOUGH\_DATA\_FILE;

}

}

if (file.hasNext()) {

err = ErrCode.A\_LOT\_OF\_DATA\_FILE;

}

return err;

}

static ErrCode readSize(int[] arrSize, Scanner file) {

ErrCode err;

if (file.hasNext())

err = readOneNum(file, arrSize, MIN\_SIZE, MAX\_SIZE, true);

else

err = ErrCode.NOT\_ENOUGH\_DATA\_FILE;

return err;

}

static int[][] inputFromFile(Scanner input) {

int[][] matrix = {};

int[] arrSize = new int[1];

ErrCode err;

String fileName;

do {

fileName = getFileName(input);

try(Scanner file = new Scanner(Paths.get(fileName))) {

err = readSize(arrSize, file);

if (err.equals(ErrCode.SUCCESS)) {

matrix = new int[arrSize[0]][arrSize[0]];

err = readFile(matrix, file);

}

} catch (IOException e) {

err = ErrCode.IN\_OUT\_FILE\_EXCEPTION;

}

if (err != ErrCode.SUCCESS) {

System.err.printf(ERRORS[err.ordinal()], MIN\_SIZE, MAX\_SIZE);

}

} while (err != ErrCode.SUCCESS);

return matrix;

}

static int[][] inputInf(Scanner input) {

int[][] matrix;

int choice;

choice = userChoice(input);

if (choice == 1) {

matrix = inputFromConsole(input);

} else {

matrix = inputFromFile(input);

}

return matrix;

}

static int[][] deleteColRow(int[][] oldMatrix, int colInd, int rowInd) {

int size = oldMatrix.length-1;

int[][] newMatrix = new int[size][size];

int curCol = 0;

int curRow = 0;

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

if (i != rowInd) {

for (int j = 0; j < oldMatrix.length; j++) {

if (j != colInd) {

newMatrix[curRow][curCol] = oldMatrix[i][j];

curCol++;

}

}

curCol = 0;

curRow++;

}

}

return newMatrix;

}

static int calculateDet(int[][] matrix) {

int det = 0;

if (matrix.length == 1)

det = matrix[0][0];

else {

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

if (matrix[0][i] != 0) {

int[][] newMatrix = deleteColRow(matrix, i, 0);

int addition = matrix[0][i] \* calculateDet(newMatrix);

addition = i % 2 == 0 ? addition : -addition;

det += addition;

}

}

}

return det;

}

static void outputToConsole(int det, int[][] matrix) {

for (int[] row : matrix) {

for (int element : row) {

System.out.print(element + "\t");

}

System.out.println();

}

System.out.println("Determinant of the matrix = " + det);

}

static void outputToFile(int det, int[][] matrix, Scanner input) {

ErrCode err;

do {

err = ErrCode.SUCCESS;

String fileName = getFileName(input);

try (PrintWriter file = new PrintWriter(fileName)) {

for (int[] row : matrix) {

for (int element : row) {

file.print(element + "\t");

}

file.println();

}

file.println("Determinant of the matrix = " + det);

} catch (IOException e) {

err = ErrCode.IN\_OUT\_FILE\_EXCEPTION;

System.err.println(ERRORS[err.ordinal()]);

}

} while (err != ErrCode.SUCCESS);

}

static void outputInf(int det, int[][] matrix, Scanner input) {

int choice = userChoice(input);

if (choice == 1) {

outputToConsole(det, matrix);

} else {

outputToFile(det, matrix, input);

}

}

static void printInf() {

System.out.println("Program calculates the determinant of a given matrix");

System.out.println("Size of matrix is from 1 to 5 and element from -70 to 70");

}

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int[][] matrix;

int det;

printInf();

matrix = inputInf(input);

det = calculateDet(matrix);

outputInf(det, matrix, input);

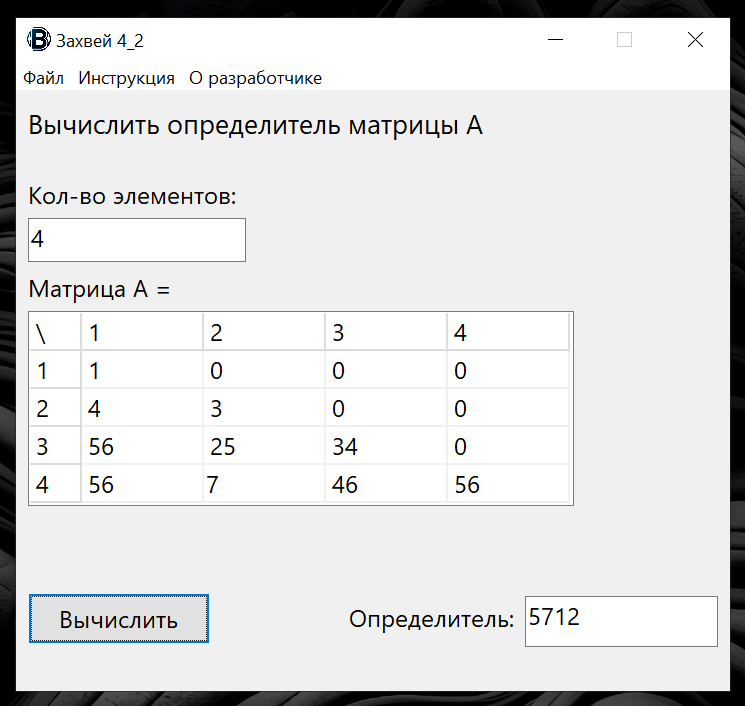
input.close();

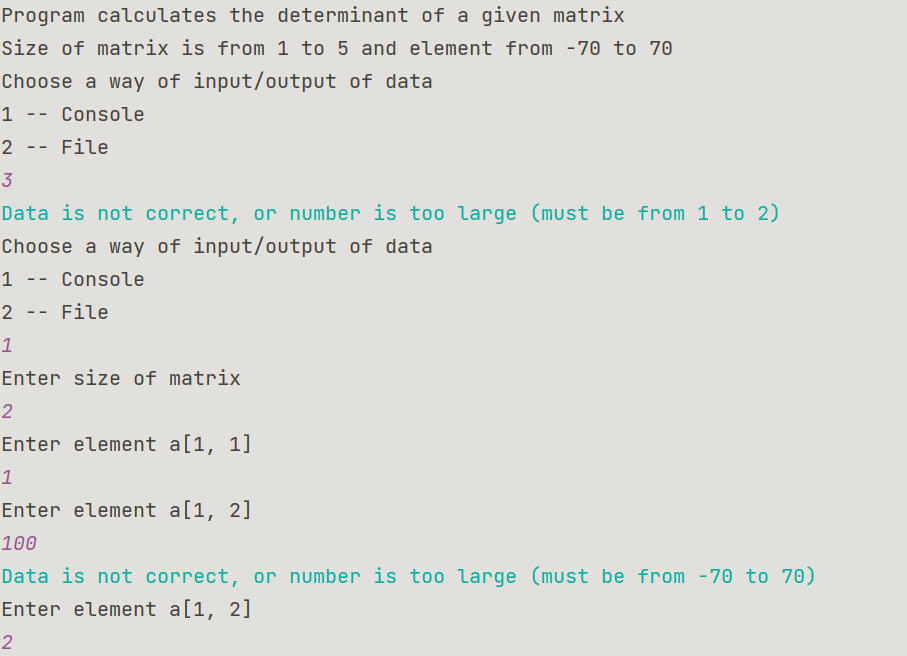
}

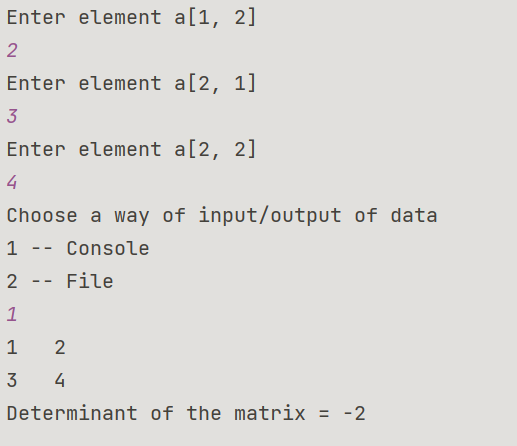
}

**Скриншоты:**

**Delphi:**

****

**Java:**

****

**Блок-схема:**



