УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №7.2

по предмету «Основы алгоритмизации и программирования»

Вариант 3

Выполнил:

Бражалович А.И.

Гр. 351004

Проверил:

Данилова Г. В.

Минск 2024

**Задание:**

Преобразовать матрицу смежности в списки инцидентности.

**Код программы Delphi:**

Unit MainUnit;

Interface

Uses

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

System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs,

Vcl.Menus, InstructionUnit, DeveloperUnit, Vcl.StdCtrls,

Vcl.Grids, Vcl.Imaging.Pngimage, Vcl.ExtCtrls;

Type

ERRORS\_CODE = (CORRECT, INCORRECT\_RANGE, EXTRA\_DATA, IS\_NOT\_READABLE,

IS\_NOT\_WRITEABLE, INCORRECT\_DATA\_IN\_FILE, INCORRECT\_NUMS\_AMOUNT);

TStringGridEx = Class(TStringGrid);

TMatrix = Array Of Array Of Integer;

TArr = Array Of Integer;

PVertex = ^TVertex;

TVertex = Record

Value: Char;

Next: PVertex;

End;

Type

TMainForm = Class(TForm)

MainMenu1: TMainMenu;

FileButton: TMenuItem;

OpenFileButton: TMenuItem;

SaveFileButton: TMenuItem;

LineSeparator: TMenuItem;

ExitButton: TMenuItem;

InstructionButton: TMenuItem;

DeveloperButton: TMenuItem;

OpenFile: TOpenDialog;

SaveFile: TSaveDialog;

OrderEdit: TEdit;

MatrixGrid: TStringGrid;

ResultButton: TButton;

OrderLabel: TLabel;

MatrixGridLabel: TLabel;

Task: TLabel;

ResultGrid: TStringGrid;

ResultLabel: TLabel;

Function ReadFileData(Var F: TextFile; Sender: TObject): ERRORS\_CODE;

Function FormHelp(Command: Word; Data: NativeInt; Var CallHelp: Boolean): Boolean;

Procedure InstructionButtonClick(Sender: TObject);

Procedure SaveOnClick(Sender: TObject);

Procedure DeveloperOnClick(Sender: TObject);

Procedure ExitOnClick(Sender: TObject);

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

Procedure OpenOnClick(Sender: TObject);

Procedure OrderEditChange(Sender: TObject);

Procedure OrderEditContextPopup(Sender: TObject; MousePos: TPoint; Var Handled: Boolean);

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

Procedure OrderEditKeyDown(Sender: TObject; Var Key: Word; Shift: TShiftState);

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

Procedure MatrixGridKeyDown(Sender: TObject; Var Key: Word; Shift: TShiftState);

Procedure MatrixGridSetEditText(Sender: TObject; ACol, ARow: Integer; Const Value: String);

Procedure ResultButtonClick(Sender: TObject);

Procedure ResultEditChange(Sender: TObject);

Procedure ResultGridDrawCell(Sender: TObject; ACol, ARow: Integer; Rect: TRect; State:

TGridDrawState);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

MainForm: TMainForm;

Implementation

{$R \*.dfm}

Var

IsSaved: Boolean = True;

AdjacencyArr: Array Of PVertex;

Const

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

BACKSPACE = #8;

NONE = #0;

ERRORS: Array [ERRORS\_CODE] Of String = ('', 'Значение не попадает в диапазон!',

'Лишние данные в файле!',

'Файл закрыт для чтения!', 'Файл закрыт для записи!',

'Некорректный тип данных в файле!',

'Неправильное количество чисел в файле!');

MIN\_N = 1;

MAX\_N = 9;

MIN\_NUM = 0;

MAX\_NUM = 1;

Function IsValidRange(Text: String; MIN, MAX: Integer): Boolean;

Var

IsValidInput: Boolean;

Num: Real;

Begin

IsValidInput := True;

Num := StrToFloat(Text);

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

IsValidInput := False;

IsValidRange := IsValidInput;

End;

Procedure TMainForm.DeveloperOnClick(Sender: TObject);

Var

DeveloperForm: TDeveloperForm;

Begin

DeveloperForm := TDeveloperForm.Create(Self);

DeveloperForm.ShowModal;

DeveloperForm.Free;

End;

Procedure TMainForm.InstructionButtonClick(Sender: TObject);

Var

InstructionForm: TInstructionForm;

Begin

InstructionForm := TInstructionForm.Create(Self);

InstructionForm.ShowModal;

InstructionForm.Free;

End;

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

Var

I: Integer;

Begin

Grid.ColCount := RowNum + 1;

Grid.RowCount := RowNum + 1;

If RowNum < 5 Then

Begin

Grid.Width := (Grid.DefaultColWidth + 4 - Grid.GridLineWidth) \*

Grid.ColCount;

Grid.Height := (Grid.DefaultRowHeight + 4 - Grid.GridLineWidth) \*

Grid.RowCount;

End

Else

Begin

Grid.Width := (Grid.DefaultColWidth + 4 - Grid.GridLineWidth) \* 5 + 32;

Grid.Height := (Grid.DefaultRowHeight + 4 - Grid.GridLineWidth) \* 5 + 32;

End;

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

For I := 1 To RowNum Do

Begin

Grid.Cells[0, I] := Chr(Ord('A') + I - 1);

Grid.Cells[I, 0] := Chr(Ord('A') + I - 1);

End;

End;

Procedure ClearGrid(Grid: TStringGrid);

Var

J, I: Integer;

Begin

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

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

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

End;

Procedure CheckRightKeyPress(Var Key: Char; SelLength: Integer; Text: String; Const MIN, MAX: Integer);

Var

IsValidInput: Boolean;

Begin

IsValidInput := (CharInSet(Key, DIGITS) Or (Key = BACKSPACE)) And

(Key <> '0');

If (SelLength > 0) And (SelLength < Length(Text)) Then

Key := NONE;

If (IsValidInput) And (Key <> BACKSPACE) Then

IsValidInput := IsValidRange(Text + Key, MIN, MAX);

If Not IsValidInput Then

Key := NONE;

End;

Procedure DeleteLists();

Var

CurrentNode: PVertex;

I: Integer;

Begin

For I := 0 To Length(AdjacencyArr) - 1 Do

Begin

While AdjacencyArr[I].Next <> Nil Do

Begin

CurrentNode := AdjacencyArr[I].Next;

AdjacencyArr[I].Next := CurrentNode.Next;

Dispose(CurrentNode);

End;

Dispose(AdjacencyArr[I]);

End;

SetLength(AdjacencyArr, 0);

End;

Procedure TMainForm.OrderEditChange(Sender: TObject);

Var

Num: Integer;

Begin

If (OrderEdit.Text = '') Or Not TryStrToInt(OrderEdit.Text, Num) Then

Begin

MatrixGridLabel.Visible := False;

MatrixGrid.Visible := False;

ResultGrid.Visible := False;

ResultLabel.Visible := False;

ClearGrid(MatrixGrid);

End

Else

Begin

MatrixGridLabel.Visible := True;

MatrixGrid.Visible := True;

ClearGrid(MatrixGrid);

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

End;

If Length(AdjacencyArr) > 0 Then

DeleteLists;

SaveFileButton.Enabled := False;

IsSaved := True;

ResultButton.Enabled := False;

End;

Procedure TMainForm.OrderEditContextPopup(Sender: TObject; MousePos: TPoint; Var Handled: Boolean);

Begin

Handled := True;

End;

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

Begin

If (Key = VK\_DELETE) Then

Key := 0;

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

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

Key := 0;

End;

Procedure TMainForm.OrderEditKeyDown(Sender: TObject; Var Key: Word; Shift: TShiftState);

Begin

CheckRightKeyDown(Key, Shift);

End;

Procedure TMainForm.OrderEditKeyPress(Sender: TObject; Var Key: Char);

Var

SelLength: Integer;

Text: String;

Begin

SelLength := OrderEdit.SelLength;

Text := OrderEdit.Text;

CheckRightKeyPress(Key, SelLength, Text, MIN\_N, MAX\_N);

End;

Procedure TMainForm.MatrixGridKeyDown(Sender: TObject; Var Key: Word; Shift: TShiftState);

Begin

CheckRightKeyDown(Key, Shift);

End;

Function IsAllCellFill(Grid: TStringGrid): Boolean;

Var

IsFilled: Boolean;

Col, Row, Num: Integer;

Begin

IsFilled := MainForm.OrderEdit.Text <> '';

Row := 1;

While IsFilled And (Row < Grid.RowCount) Do

Begin

Col := 1;

While IsFilled And (Col < Grid.ColCount) Do

Begin

If (MainForm.MatrixGrid.Cells[Col, Row] = '') Or

Not TryStrToInt(MainForm.MatrixGrid.Cells[Col, Row], Num) Then

IsFilled := False;

Inc(Col);

End;

Inc(Row);

End;

IsAllCellFill := IsFilled;

End;

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

Var

MatrixGrid: TStringGridEx;

Text: String;

Begin

MatrixGrid := TStringGridEx(Sender);

If Assigned(MatrixGrid.InplaceEditor) Then

Begin

Text := MatrixGrid.InplaceEditor.Text;

If (Length(Text) > 0) And (Key <> BACKSPACE) Then

Key := NONE

Else If (Key <> '1') And (Key <> '0') And (Key <> BACKSPACE) Then

Key := NONE;

End;

End;

Procedure TMainForm.MatrixGridSetEditText(Sender: TObject; ACol, ARow: Integer; Const Value: String);

Begin

ResultButton.Enabled := IsAllCellFill(MatrixGrid);

ClearGrid(ResultGrid);

ResultGrid.Visible := False;

ResultLabel.Visible := False;

SaveFileButton.Enabled := False;

IsSaved := True;

If Length(AdjacencyArr) > 0 Then

DeleteLists;

End;

Procedure TMainForm.ExitOnClick(Sender: TObject);

Begin

Close;

End;

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

Var

Confirmation: Integer;

Begin

If (IsSaved = 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

SaveOnClick(Sender);

If IsSaved = 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;

Function TMainForm.FormHelp(Command: Word; Data: NativeInt; Var CallHelp: Boolean): Boolean;

Begin

CallHelp := False;

End;

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

Var

Error: ERRORS\_CODE;

Begin

Error := CORRECT;

Try

Try

Reset(F);

Finally

CloseFile(F);

End;

Except

Error := IS\_NOT\_READABLE;

End;

IsReadable := Error;

End;

Function ReadFileNum(Var F: TextFile; Var Num: Integer; Const MIN, MAX: Integer) : ERRORS\_CODE;

Var

Error: ERRORS\_CODE;

Begin

Error := CORRECT;

Try

Read(F, Num);

Except

Error := INCORRECT\_DATA\_IN\_FILE;

End;

If (Error = CORRECT) And Not(IsValidRange(IntToStr(Num), MIN, MAX)) Then

Error := INCORRECT\_RANGE;

ReadFileNum := Error;

End;

Function CheckSpaceInFile(BufStr: String): ERRORS\_CODE;

Var

I: Integer;

Error: ERRORS\_CODE;

Begin

I := 1;

Error := CORRECT;

While (Error = CORRECT) And (I <= Length(BufStr)) Do

Begin

If BufStr[I] <> ' ' Then

Error := EXTRA\_DATA;

Inc(I);

End;

CheckSpaceInFile := Error;

End;

Function TMainForm.ReadFileData(Var F: TextFile; Sender: TObject): ERRORS\_CODE;

Var

Error: ERRORS\_CODE;

Num, Order: Integer;

Matrix: TMatrix;

I, J: Integer;

BufStr: String;

Begin

I := 0;

J := 0;

Reset(F);

Error := ReadFileNum(F, Num, MIN\_N + 1, MAX\_N);

If Error = CORRECT Then

Begin

Readln(F, BufStr);

Error := CheckSpaceInFile(BufStr);

End;

If Error = CORRECT Then

OrderEdit.Text := IntToStr(Num);

Order := Num;

SetLength(Matrix, Order, Order);

While (Error = CORRECT) And (I < Order) Do

Begin

J := 0;

While (Error = CORRECT) And Not EOF(F) And (J < Order) Do

Begin

Error := ReadFileNum(F, Matrix[I, J], MIN\_NUM, MAX\_NUM);

Inc(J);

End;

Readln(F, BufStr);

Error := CheckSpaceInFile(BufStr);

Inc(I);

End;

If (Error = CORRECT) And (J <> Order) Then

Error := INCORRECT\_NUMS\_AMOUNT;

If Error = CORRECT Then

Begin

FillGrid(Num, MatrixGrid);

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

Begin

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

MatrixGrid.Cells[J, I] := IntToStr(Matrix[I - 1, J - 1]);

End;

ResultButton.Enabled := True;

End;

CloseFile(F);

ReadFileData := Error;

End;

Procedure CreateAdjacencyLists(Matrix: TMatrix);

Var

I, J, K: Integer;

Temp: PVertex;

Begin

SetLength(AdjacencyArr, Length(Matrix));

For I := 0 To Length(AdjacencyArr) - 1 Do

Begin

New(AdjacencyArr[I]);

AdjacencyArr[I].Next := Nil;

Temp := AdjacencyArr[I];

For J := 0 To Length(Matrix) - 1 Do

Begin

For K := 1 To Matrix[I, J] Do

Begin

New(Temp.Next);

Temp := Temp.Next;

Temp.Value := Chr(Ord('A') + J);

Temp.Next := Nil;

End;

End;

End;

End;

Procedure MakeResultGrid(RowNum: Integer; Grid: TStringGrid);

Begin

Grid.ColCount := 2;

Grid.RowCount := RowNum;

Grid.ColWidths[0] := Grid.DefaultColWidth;

Grid.ColWidths[1] := Grid.DefaultColWidth + 20 \* RowNum;

If RowNum < 4 Then

Begin

Grid.Height := (Grid.DefaultRowHeight + 4 - Grid.GridLineWidth) \*

Grid.RowCount + 2;

Grid.Width := Grid.ColWidths[0] + Grid.ColWidths[1] + 10;

End

Else

Begin

Grid.Height := (Grid.DefaultRowHeight + 4 - Grid.GridLineWidth) \* 3 + 2;

Grid.Width := Grid.ColWidths[0] + Grid.ColWidths[1] + 42;

End;

Grid.Visible := True;

End;

Procedure FillResultGrid(RowNum: Integer; Grid: TStringGrid);

Var

I: Integer;

ResultStr: String;

Temp: PVertex;

Begin

For I := 0 To RowNum - 1 Do

Begin

ResultStr := '';

Grid.Cells[0, I] := Chr(Ord('A') + I) + ':';

Temp := AdjacencyArr[I];

While Temp.Next <> Nil Do

Begin

If ResultStr <> '' Then

ResultStr := ResultStr + ', ';

ResultStr := ResultStr + Temp.Next.Value;

Temp := Temp.Next;

End;

Grid.Cells[1, I] := ResultStr;

End;

End;

Procedure TMainForm.ResultButtonClick(Sender: TObject);

Var

Matrix: TMatrix;

I, J: Integer;

Begin

SetLength(Matrix, MatrixGrid.RowCount - 1, MatrixGrid.ColCount - 1);

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

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

Matrix[I - 1][J - 1] := StrToInt(MatrixGrid.Cells[J, I]);

CreateAdjacencyLists(Matrix);

MakeResultGrid(MatrixGrid.RowCount - 1, ResultGrid);

FillResultGrid(MatrixGrid.RowCount - 1, ResultGrid);

ResultLabel.Visible := True;

IsSaved := False;

SaveFileButton.Enabled := True;

End;

Procedure TMainForm.ResultEditChange(Sender: TObject);

Begin

If ResultGrid.Visible = False Then

Begin

IsSaved := True;

SaveFileButton.Enabled := False;

End

Else

Begin

IsSaved := False;

SaveFileButton.Enabled := True;

End;

End;

Procedure TMainForm.ResultGridDrawCell(Sender: TObject; ACol, ARow: Integer; Rect: TRect; State: TGridDrawState);

Begin

If ACol = 1 Then

Begin

Text := ResultGrid.Cells[ACol, ARow];

ResultGrid.Canvas.Brush.Color := ClWindow;

ResultGrid.Canvas.FillRect(Rect);

DrawText(ResultGrid.Canvas.Handle, PChar(Text), Length(Text), Rect,

DT\_CENTER Or DT\_VCENTER Or DT\_SINGLELINE);

End;

End;

Procedure TMainForm.OpenOnClick(Sender: TObject);

Var

Error: ERRORS\_CODE;

F: TextFile;

FileName: String;

Begin

If OpenFile.Execute Then

Begin

FileName := OpenFile.FileName;

AssignFile(F, FileName);

Error := IsReadable(F);

If Error = CORRECT Then

Error := ReadFileData(F, Sender);

If Error <> CORRECT Then

Application.MessageBox(PWideChar(ERRORS[Error]), 'Ошибка', MB\_OK Or MB\_ICONINFORMATION);

End;

End;

Function IsWriteable(Var F: TextFile): ERRORS\_CODE;

Var

Error: ERRORS\_CODE;

Begin

Error := CORRECT;

Try

Try

Append(F);

Finally

CloseFile(F);

End;

Except

Error := Is\_NOT\_WRITEABLE;

End;

IsWriteable := Error;

End;

Procedure TMainForm.SaveOnClick(Sender: TObject);

Var

Error: ERRORS\_CODE;

F: TextFile;

FileName: String;

I: Integer;

Begin

If SaveFile.Execute Then

Begin

FileName := SaveFile.FileName;

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

AssignFile(F, FileName);

If FileExists(FileName) Then

Begin

Error := IsWriteable(F);

If Error = CORRECT Then

Begin

Append(F);

Writeln(F, #13#10, 'Списки инцидентности: ');

For I := 0 To Length(AdjacencyArr) - 1 Do

Writeln(F, Chr(Ord('A') + I) + ': ' +

ResultGrid.Cells[1, I]);

CloseFile(F);

IsSaved := True;

End

Else

Begin

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

MB\_OK Or MB\_ICONINFORMATION);

IsSaved := False;

End;

End

Else

Begin

Rewrite(F);

Writeln(F, 'Списки инцидентности: ');

For I := 0 To Length(AdjacencyArr) - 1 Do

Writeln(F, Chr(Ord('A') + I) + ': ' + ResultGrid.Cells[1, I]);

CloseFile(F);

IsSaved := True;

End;

End;

End;

End.

Unit InstructionUnit;

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)

CloseButton: TButton;

Button1: TButton;

InstructionLabel1: TLabel;

InstructionLabel4: TLabel;

InstructionLabel2: TLabel;

Procedure CloseButtonClick(Sender: TObject);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

InstructionForm: TInstructionForm;

Implementation

{$R \*.dfm}

Procedure TInstructionForm.CloseButtonClick(Sender: TObject);

Begin

Close;

End;

End.

Unit DeveloperUnit;

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)

CloseButton: TButton;

DeveloperLabel1: TLabel;

DeveloperLabel2: TLabel;

Procedure CloseButtonClick(Sender: TObject);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

DeveloperForm: TDeveloperForm;

Implementation

{$R \*.dfm}

Procedure TDeveloperForm.CloseButtonClick(Sender: TObject);

Begin

Close;

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\_O = 1;

public static final int MAX\_O = 9;

public static final int MIN\_MAT = 0;

public static final int MAX\_MAT = 1;

public static class Vertex {

String value;

Vertex next;

public Vertex(String value) {

this.value = value;

this.next = null;

}

}

private static Vertex[] adjacencyArray;

public enum ErrorsCode {

CORRECT,

INCORRECT\_CHOICE,

INCORRECT\_NUMBER,

FILE\_NOT\_TXT,

FILE\_NOT\_EXIST,

FILE\_NOT\_READABLE,

FILE\_IS\_EMPTY,

FILE\_NOT\_WRITABLE,

INCORRECT\_RANGE,

NOT\_A\_NUM,

EXTRA\_DATA

}

public static final String[] ERRORS = {

"Правильно",

"Некорректный выбор!",

"Ошибка!Введите 1 или 2",

"Расширение файла не.txt!",

"Проверьте корректность ввода пути к файлу!",

"Файл закрыт для чтения!",

"Файл пуст!",

"Файл закрыт для записи!",

"Число не попадает в диапазон!",

"Проверьте корректность ввода данных!",

"Лишние данные в файле!"};

public static void printTask() {

System.out.println("Данная программа преобразует матрицу смежности в списки инцидентности.");

}

public static int chooseOption(Scanner inputScanner) {

ErrorsCode error;

int option;

String optionStr;

option = 0;

optionStr = "";

do {

error = ErrorsCode.CORRECT;

try {

optionStr = inputScanner.nextLine();

option = Integer.parseInt(optionStr);

} catch (NumberFormatException e) {

error = ErrorsCode.INCORRECT\_CHOICE;

}

if ((error == ErrorsCode.CORRECT) && (option != 1) && (option != 2))

error = ErrorsCode.INCORRECT\_NUMBER;

if (error != ErrorsCode.CORRECT && (optionStr != ""))

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

} while (error != ErrorsCode.CORRECT);

return option;

}

public static String readPathFile(Scanner inputScanner) {

ErrorsCode error;

String pathToFile;

do {

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

pathToFile = inputScanner.nextLine();

if (pathToFile.endsWith(".txt"))

error = ErrorsCode.CORRECT;

else

error = ErrorsCode.FILE\_NOT\_TXT;

if (error != ErrorsCode.CORRECT)

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

} while (error != ErrorsCode.CORRECT);

return pathToFile;

}

public static File getReadableFile(Scanner inputScanner) {

File file;

ErrorsCode error;

String pathToFile;

do {

error = ErrorsCode.CORRECT;

pathToFile = readPathFile(inputScanner);

file = new File(pathToFile);

if (!file.exists())

error = ErrorsCode.FILE\_NOT\_EXIST;

if (error == ErrorsCode.CORRECT && !file.canRead())

error = ErrorsCode.FILE\_NOT\_READABLE;

if (error == ErrorsCode.CORRECT && (file.length() == 0))

error = ErrorsCode.FILE\_IS\_EMPTY;

if (error != ErrorsCode.CORRECT)

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

} while (error != ErrorsCode.CORRECT);

return file;

}

public static File getWritableFile(Scanner inputScanner) {

File file;

ErrorsCode error;

String pathToFile;

do {

error = ErrorsCode.CORRECT;

pathToFile = readPathFile(inputScanner);

file = new File(pathToFile);

if (!file.exists())

error = ErrorsCode.FILE\_NOT\_EXIST;

if (error == ErrorsCode.CORRECT && !file.canWrite())

error = ErrorsCode.FILE\_NOT\_WRITABLE;

if (error != ErrorsCode.CORRECT)

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

} while (error != ErrorsCode.CORRECT);

return file;

}

public static ErrorsCode checkSpaceInFile(String bufStr) {

int i;

ErrorsCode error;

i = 0;

error = ErrorsCode.CORRECT;

while ((error == ErrorsCode.CORRECT) && (i < bufStr.length())) {

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

error = ErrorsCode.EXTRA\_DATA;

i++;

}

return error;

}

public static ErrorsCode readOneNum(Scanner inputScanner, int[] numberArr, final int MIN,

final int MAX, boolean isFile) {

int number = 0;

ErrorsCode error;

error = ErrorsCode.CORRECT;

try {

if (isFile)

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

else

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

} catch (NumberFormatException e) {

error = ErrorsCode.NOT\_A\_NUM;

}

if (error == ErrorsCode.CORRECT && ((number < MIN) || (number > MAX)))

error = ErrorsCode.INCORRECT\_RANGE;

numberArr[0] = error == ErrorsCode.CORRECT ? number : 0;

return error;

}

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

int i;

int j;

int[][] matrix;

String bufStr;

ErrorsCode error;

error = ErrorsCode.CORRECT;

i = 0;

matrix = new int[order][order];

while (i < order && error == ErrorsCode.CORRECT) {

j = 0;

while (j < order && error == ErrorsCode.CORRECT) {

if (option == 1) {

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

if (error == ErrorsCode.CORRECT)

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

} else {

do {

System.out.print("Введите в " + (i + 1) + " строке " + (j + 1) + " столбце

элемент матрицы [" + MIN\_MAT + "; " + MAX\_MAT + "]: ");

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

if (error != ErrorsCode.CORRECT)

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

else

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

} while (error != ErrorsCode.CORRECT);

}

j++;

}

if (option == 1 && error == ErrorsCode.CORRECT) {

bufStr = inputScanner.nextLine();

error = checkSpaceInFile(bufStr);

}

i++;

}

if (error != ErrorsCode.CORRECT) {

matrix = null;

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

}

return matrix;

}

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

int[] numberArr = new int[1];

int order;

int[][] matrix = {};

ErrorsCode error;

do {

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

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

if (error != ErrorsCode.CORRECT)

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

else {

order = numberArr[0];

matrix = readMatrix(inputScanner, numberArr, order, 2);

}

} while (error != ErrorsCode.CORRECT);

return matrix;

}

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

String bufStr;

int[] numberArr = new int[1];

int order;

int[][] matrix = {};

ErrorsCode error;

File file;

do {

file = getReadableFile(inputScanner);

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

error = readOneNum(scanFile, numberArr, MIN\_O, MAX\_O, true);

if (error == ErrorsCode.CORRECT) {

bufStr = scanFile.nextLine();

error = checkSpaceInFile(bufStr);

}

if (error != ErrorsCode.CORRECT)

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

else {

order = numberArr[0];

matrix = readMatrix(scanFile, numberArr, order, 1);

}

} catch (Exception e) {

error = ErrorsCode.FILE\_NOT\_READABLE;

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

}

if (matrix == null)

error = ErrorsCode.INCORRECT\_NUMBER;

} while (error != ErrorsCode.CORRECT);

return matrix;

}

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

int[][] matrix;

int option;

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

System.out.println("Через файл - 1");

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

option = chooseOption(inputScanner);

if (option == 1) {

matrix = readFileMatrix(inputScanner);

} else {

matrix = readConsoleMatrix(inputScanner);

}

return matrix;

}

public static void createAdjacencyLists(int[][] matrix) {

Vertex temp;

int n = matrix.length;

adjacencyArray = new Vertex[n];

for (int i = 0; i < n; i++) {

adjacencyArray[i] = new Vertex("");

adjacencyArray[i].next = null;

temp = adjacencyArray[i];

for (int j = 0; j < n; j++) {

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

temp.next = new Vertex("");

temp = temp.next;

temp.value = String.valueOf((char)(65 + j));

temp.next = null;

}

}

}

}

public static void printResult(Scanner inputScanner, int n) {

ErrorsCode error;

File file;

StringBuilder resultString;

Vertex temp;

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

System.out.println("Через файл - 1");

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

int option = chooseOption(inputScanner);

if (option == 1)

{

file = getWritableFile(inputScanner);

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

writer.write("Списки инцидентности:\n");

for (int i = 0; i < n; i++) {

resultString = new StringBuilder();

writer.write(((char)(65 + i)) + ": ");

temp = adjacencyArray[i];

while (temp.next != null) {

if (resultString.toString() != "") {

resultString.append(", ");

}

resultString.append(temp.next.value);

temp = temp.next;

}

writer.write(resultString + "\n");

}

} catch (Exception e) {

error = ErrorsCode.FILE\_NOT\_WRITABLE;

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

}

}

else {

System.out.println("Списки инцидентности:");

for (int i = 0; i < n; i++) {

resultString = new StringBuilder();

System.out.print(((char)(65 + i)) + ": ");

temp = adjacencyArray[i];

while (temp.next != null) {

if (resultString.toString() != "") {

resultString.append(", ");

}

resultString.append(temp.next.value);

temp = temp.next;

}

System.out.println(resultString);

}

}

}

public static void main(String[] args) {

Scanner inputScanner = new Scanner(System.in);

int[][] matrix;

printTask();

matrix = inputMatrix(inputScanner);

createAdjacencyLists(matrix);

printResult(inputScanner, matrix.length);

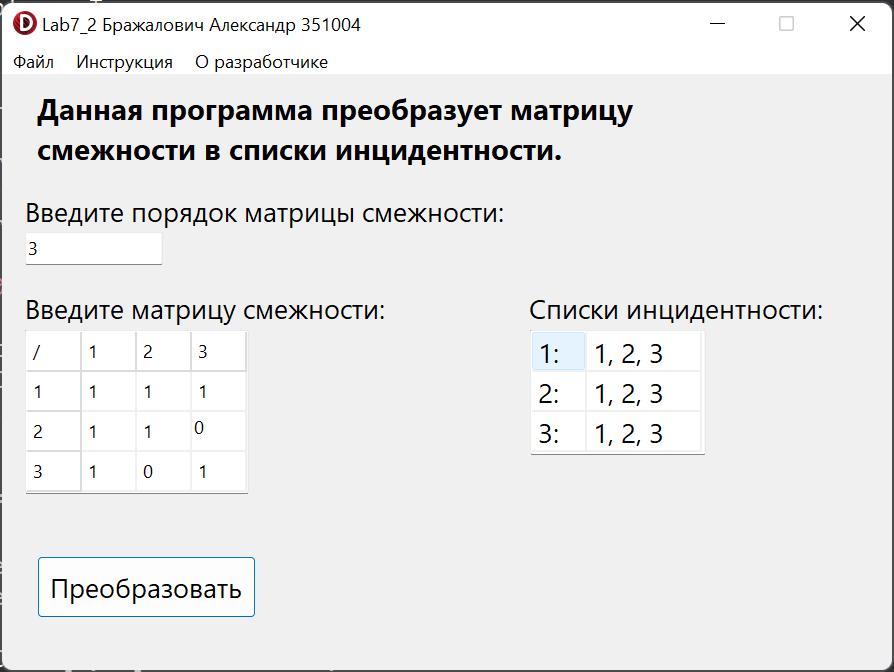
inputScanner.close();

}

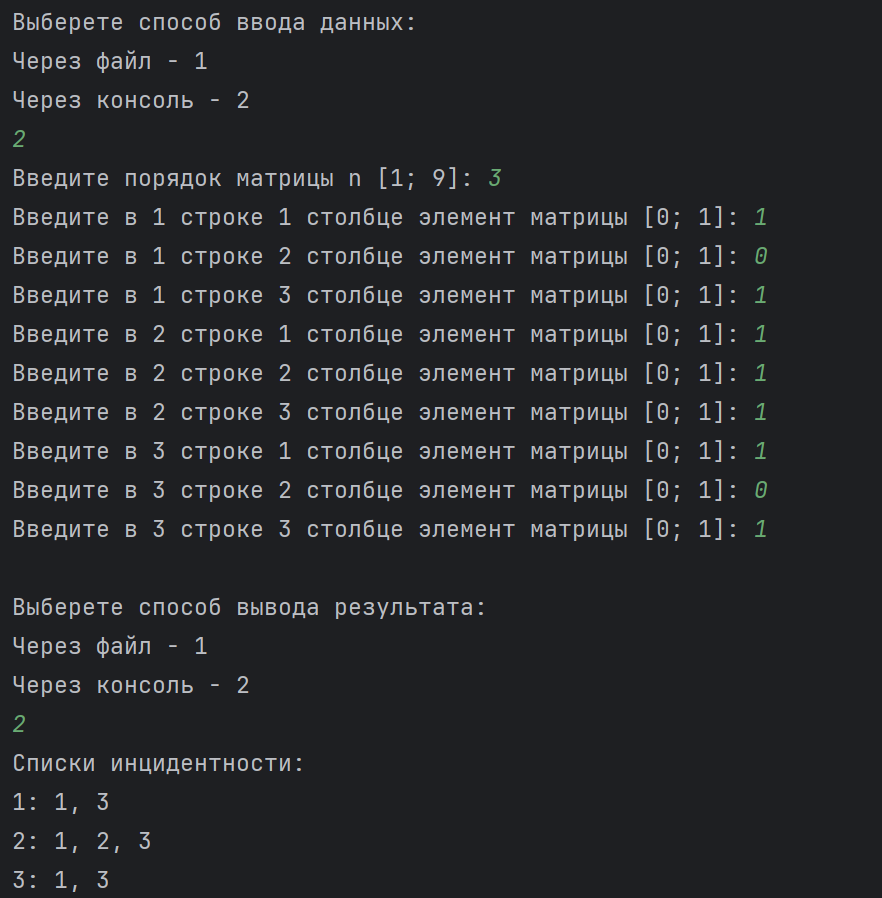
}

**Скриншоты:**

**Delphi:**

****

**Java:**

****

**Блок-схема:**

