УО «Белорусский государственный университет информатики и

радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №4.2

по предмету

Основы Алгоритмизации и Программирования

Вариант 4

Выполнил

Воривода М.А.

Проверила

Данилова Г.В.

Группа:

951007

Минск 2020

**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 Unit1;

interface

uses

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

System.Classes, Vcl.Graphics,

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

Vcl.Grids;

type

MyMatrix = Array of Array of Integer;

TForm1 = class(TForm)

Button1: TButton;

MEdit: TComboBox;

Label2: TLabel;

SaveTextFileDialog1: TSaveTextFileDialog;

OpenTextFileDialog1: TOpenTextFileDialog;

MainMenu1: TMainMenu;

File1: TMenuItem;

Open1: TMenuItem;

Save1: TMenuItem;

N1: TMenuItem;

Exit1: TMenuItem;

About1: TMenuItem;

Program1: TMenuItem;

Author1: TMenuItem;

BackGround: TStringGrid;

MatrixGrid: TStringGrid;

ErrorLabel: TLabel;

Result: TLabel;

FocusStopper: TEdit;

Hotkeys1: TMenuItem;

N2: TMenuItem;

Label1: TLabel;

Label3: TLabel;

Label4: TLabel;

Label5: TLabel;

I1Edit: TEdit;

I2Edit: TEdit;

J2Edit: TEdit;

J1Edit: TEdit;

FindButton: TButton;

Label6: TLabel;

NEdit: TComboBox;

Label7: TLabel;

PointsGrid: TStringGrid;

ClearButton: TButton;

procedure FormCreate(Sender: TObject);

procedure Open1Click(Sender: TObject);

procedure Save1Click(Sender: TObject);

procedure Exit1Click(Sender: TObject);

procedure Program1Click(Sender: TObject);

procedure Author1Click(Sender: TObject);

procedure FormCloseQuery(Sender: TObject; var

CanClose: Boolean);

procedure MEditExit(Sender: TObject);

procedure MatrixGridKeyPress(Sender: TObject; var key:

Char);

procedure MEditKeyPress(Sender: TObject; var key:

Char);

procedure SetLenClick(Sender: TObject);

procedure MEditChange(Sender: TObject);

procedure MatrixGridSelectCell(Sender: TObject; ARow,

ACol: Integer;

var CanSelect: Boolean);

procedure MatrixGridExit(Sender: TObject);

procedure Hotkeys1Click(Sender: TObject);

procedure IKeyPress(Sender: TObject; var key: Char);

procedure JKeyPress(Sender: TObject; var key: Char);

procedure CoordsChange(Sender: TObject);

procedure FindButtonClick(Sender: TObject);

procedure FindTheWay(ElementInWay, I1, J1, Sum: Integer);

procedure ClearButtonClick(Sender: TObject);

procedure PrintWay();

private

{ Private declarations }

public

{ Public declarations }

end;

const

MAX\_ABS\_VALUE = 100;

var

Form1: TForm1;

LastKey: Char;

MLength, NLength, NextCell, LastRow, LastCol: Integer;

Matrix, CurrentWay, BestWay: MyMatrix;

MaxSum: Integer;

I2, J2: Integer;

implementation

{$R \*.dfm}

procedure TForm1.FindTheWay(ElementInWay, I1: Integer; J1, Sum: Integer);

var

I: Integer;

begin

if (I1 = I2) and (J1 = J2) then

begin

if (MaxSum = 1000000) or (Sum > MaxSum) then

begin

MaxSum := Sum;

BestWay := CurrentWay;

SetLength(BestWay, 2, ElementInWay + 2);

end;

end

else

begin

Inc(ElementInWay);

if I1 < I2 then

begin

CurrentWay[0][ElementInWay] := I1 + 1;

CurrentWay[1][ElementInWay] := J1;

FindTheWay(ElementInWay, I1 + 1, J1, Sum +

Matrix[I1 + 1, J1]);

end;

if J1 < J2 then

begin

CurrentWay[0][ElementInWay] := I1;

CurrentWay[1][ElementInWay] := J1 + 1;

FindTheWay(ElementInWay, I1, J1 + 1, Sum +

Matrix[I1, J1 + 1]);

end;

end;

end;

procedure TForm1.FindButtonClick(Sender: TObject);

var

I, J: Integer;

I1, J1, Temp: Integer;

begin

MaxSum := 1000000;

SetLength(Matrix, MLength, NLength);

for I := 0 to MLength - 1 do

for J := 0 to NLength - 1 do

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

SetLength(BestWay, 2, MLength \* NLength);

SetLength(CurrentWay, 2, MLength \* NLength);

I1 := StrToInt(I1Edit.Text);

I2 := StrToInt(I2Edit.Text);

J1 := StrToInt(J1Edit.Text);

J2 := StrToInt(J2Edit.Text);

if (I1 \* NLength + J1 + 1) > (I2 \* NLength + J2 + 1) then

begin

Temp := I1;

I1 := I2;

I2 := Temp;

Temp := J1;

J1 := J2;

J2 := Temp;

end;

FindTheWay(0, I1, J1, Matrix[I1, J1]);

BestWay[0][0] := I1;

BestWay[1][0] := J1;

PrintWay();

end;

procedure TForm1.PrintWay;

var

I: Integer;

begin

PointsGrid.ColCount := Length(BestWay[0]);

PointsGrid.Cells[0, 1] := 'X';

PointsGrid.Cells[0, 2] := 'Y';

for I := 1 to High(BestWay[0]) do

begin

PointsGrid.Cells[I, 0] := 'Point #' + IntToStr(I);

PointsGrid.Cells[I, 1] := IntToStr(BestWay[0][I - 1]);

PointsGrid.Cells[I, 2] := IntToStr(BestWay[1][I - 1]);

end;

Label7.Visible := true;

PointsGrid.Visible := true;

end;

procedure TForm1.FormCreate(Sender: TObject);

var

I, J: Integer;

begin

MatrixGrid.Col := 0;

MatrixGrid.Row := 0;

MLength := StrToInt(MEdit.Text);

NLength := StrToInt(NEdit.Text);

with MatrixGrid do

begin

Height := 3 + MLength \* (DefaultRowHeight + 1);

Width := 3 + NLength \* (1 + DefaultColWidth);

for I := 0 to ColCount do

for J := 0 to ColCount do

begin

Cells[I, J] := '0';

BackGround.Cells[I, J] := '0';

end;

end;

end;

procedure TForm1.ClearButtonClick(Sender: TObject);

var

I, J: Integer;

begin

MatrixGrid.Col := 0;

MatrixGrid.Row := 0;

Label7.Visible := false;

PointsGrid.Visible := false;

MEdit.Text := '9';

NEdit.Text := '9';

MLength := StrToInt(MEdit.Text);

NLength := StrToInt(NEdit.Text);

with MatrixGrid do

begin

Height := 3 + MLength \* (DefaultRowHeight + 1);

Width := 3 + NLength \* (1 + DefaultColWidth);

for I := 0 to ColCount do

for J := 0 to ColCount do

begin

Cells[I, J] := '0';

BackGround.Cells[I, J] := '0';

end;

end;

end;

procedure TForm1.CoordsChange(Sender: TObject);

begin

if (I1Edit.Text = '') or (I2Edit.Text = '') or

(J1Edit.Text = '') or (J2Edit.Text = '') then

FindButton.Enabled := false

else if (StrToInt(I2Edit.Text) < StrToInt(I1Edit.Text)) or

(StrToInt(J2Edit.Text) < StrToInt(J1Edit.Text)) then

FindButton.Enabled := false

else

FindButton.Enabled := true;

end;

procedure TForm1.IKeyPress(Sender: TObject; var key: Char);

begin

case key of

#8:

;

'0':

if Length((Sender as TEdit).Text) = 1 then

key := #0;

'1' .. '9':

if StrToInt((Sender as TEdit).Text + key) >

MLength - 1 then

key := #0;

else

key := #0;

end;

end;

procedure TForm1.JKeyPress(Sender: TObject; var key: Char);

begin

case key of

#8:

;

'0':

if Length((Sender as TEdit).Text) = 1 then

key := #0;

'1' .. '9':

if StrToInt((Sender as TEdit).Text + key) >

NLength - 1 then

key := #0;

else

key := #0;

end;

end;

procedure TForm1.Hotkeys1Click(Sender: TObject);

begin

MessageBox(0, 'Alt+O - load from file' + #13#10 + 'Alt+S - save to file' +

#13#10 + 'Alt+P - about program' + #13#10 + 'Alt+A - about author' +

#13#10 + 'Alt+H - hot keys' + #13#10 + 'Esc - exit', 'Hot keys',

MB\_OK + MB\_ICONASTERISK);

end;

procedure TForm1.Open1Click(Sender: TObject);

var

InF: TextFile;

I, J, Reader: Integer;

Valid: Boolean;

begin

MatrixGrid.Col := 0;

MatrixGrid.Row := 0;

Label7.Visible := false;

PointsGrid.Visible := false;

MEdit.Text := '9';

NEdit.Text := '9';

MLength := StrToInt(MEdit.Text);

NLength := StrToInt(NEdit.Text);

with MatrixGrid do

begin

Height := 3 + MLength \* (DefaultRowHeight + 1);

Width := 3 + NLength \* (1 + DefaultColWidth);

for I := 0 to ColCount do

for J := 0 to ColCount do

begin

Cells[I, J] := '0';

BackGround.Cells[I, J] := '0';

end;

end;

Valid := true;

I := 0;

J := 0;

if OpenTextFileDialog1.Execute then

try

AssignFile(InF, OpenTextFileDialog1.FileName);

Reset(InF);

try

while (I < MLength) and (Valid) do

begin

J := 0;

while (J < NLength) and (Valid) do

begin

Read(InF, Reader);

if abs(Reader) > MAX\_ABS\_VALUE then

begin

Valid := false;

ErrorLabel.Caption :=

'Error! Element is out of

bounds. Aborted on Cell['

+ IntToStr(J) + ', ' +

IntToStr(I) + '].';

end

else

MatrixGrid.Cells[J, I] :=

IntToStr(Reader);

Inc(J);

end;

Inc(I);

end;

CloseFile(InF);

except

ErrorLabel.Caption := 'Error! Corrupted data.

Aborted on Cell['

+ IntToStr(J) + ', ' + IntToStr(I) + '].';

end;

except

ErrorLabel.Caption := 'Error! Unknown file

error.';

end;

end;

procedure TForm1.Save1Click(Sender: TObject);

var

OutF: TextFile;

i : Integer;

begin

if SaveTextFileDialog1.Execute then

try

AssignFile(OutF, SaveTextFileDialog1.FileName);

Rewrite(OutF);

Write(OutF, 'Step :');

for i := 0 to High(BestWay[0]) do

Write(OutF, ' ' + IntToStr(i + 1));

WriteLn;

Write(OutF, 'i :');

for i := 0 to High(BestWay[0]) do

Write(OutF, ' ' + IntToStr(myMatrix[0][i]));

WriteLn;

Write(OutF, 'j :');

for i := 0 to High(BestWay[0]) do

Write(OutF, ' ' + IntToStr(myMatrix[1][i]));

CloseFile(OutF);

except

ErrorLabel.Caption := 'Error! Unknown file

error.';

end;

end;

procedure TForm1.Exit1Click(Sender: TObject);

begin

Form1.Close();

end;

procedure TForm1.Program1Click(Sender: TObject);

begin

MessageBox(0, 'This program search way from [i1, j1]-

element to [i2, j2]-element' +

' with the largest sum of elements on this way.', 'About

program', MB\_OK + MB\_ICONASTERISK);

end;

procedure TForm1.Author1Click(Sender: TObject);

begin

ErrorLabel.Caption := IntToStr(MatrixGrid.Height);

MessageBox(0, 'Created by Matvey Vorivoda. Student of group 951007',

'About author', MB\_OK + MB\_ICONASTERISK);

end;

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

begin

if MessageBox(0, 'Do you want to exit?', 'Exit',

MB\_ICONQUESTION + MB\_YESNO) = 6 then

CanClose := true

else

CanClose := false;

end;

procedure TForm1.MEditExit(Sender: TObject);

begin

MatrixGrid.Col := 0;

MatrixGrid.Row := 0;

if MEdit.Text = '' then

MEdit.ItemIndex := 8;

if NEdit.Text = '' then

NEdit.ItemIndex := 8;

MLength := StrToInt(MEdit.Text);

NLength := StrToInt(NEdit.Text);

with MatrixGrid do

begin

Height := 3 + MLength \* (DefaultRowHeight + 1);

Width := 3 + NLength \* (1 + DefaultColWidth);

end;

end;

procedure TForm1.SetLenClick(Sender: TObject);

begin

MatrixGrid.Col := 0;

MatrixGrid.Row := 0;

MLength := StrToInt(MEdit.Text);

NLength := StrToInt(NEdit.Text);

I1Edit.Text := '';

I2Edit.Text := '';

J1Edit.Text := '';

J2Edit.Text := '';

with MatrixGrid do

begin

Height := 3 + MLength \* (DefaultRowHeight + 1);

Width := 3 + NLength \* (1 + DefaultColWidth);

end;

end;

procedure TForm1.MEditChange(Sender: TObject);

begin

(Sender as TComboBox).SetFocus();

end;

procedure TForm1.MEditKeyPress(Sender: TObject; var key: Char);

begin

case key of

#8:

;

#13:

FocusStopper.SetFocus();

'1' .. '9':

if Length((Sender as TComboBox).Text) = 1 then

key := #0;

else

key := #0;

end;

end;

procedure TForm1.MatrixGridKeyPress(Sender: TObject; var key: Char);

begin

with MatrixGrid do

begin

LastRow := Col;

LastCol := Row;

case key of

#8:

;

#13:

begin

if (Row = MLength - 1) and (Col = NLength

- 1) then

begin

Row := 0;

Col := 0;

end

else

begin

NextCell := Row \* MLength + Col + 1;

Row := NextCell div MLength;

Col := NextCell - Row \* MLength;

end;

end;

'0':

if Cells[Col, Row] = '0' then

key := #0

else

begin

if Length(Cells[Col, Row]) > 0 then

if abs(StrToInt(Cells[Col, Row] +

key)) > MAX\_ABS\_VALUE

then

begin

ErrorLabel.Caption :=

'Error! You are trying to enter

' + Cells[Col, Row] + key + '

(max absolute value is '

+ IntToStr(MAX\_ABS\_VALUE) + ')';

key := #0

end;

end;

'1' .. '9':

if abs(StrToInt(Cells[Col, Row] + key)) >

MAX\_ABS\_VALUE then

begin

ErrorLabel.Caption := 'Error! You are

trying to enter ' + Cells[Col, Row] +

key + ' (max absolute value is ' +

IntToStr(MAX\_ABS\_VALUE) + ')';

key := #0

end;

'-':

if Length(Cells[Col, Row]) > 0 then

key := #0;

else

key := #0;

end;

end;

BackGround.Cells[MatrixGrid.Row, MatrixGrid.Col] :=

MatrixGrid.Cells[MatrixGrid.Row, MatrixGrid.Col] + key;

end;

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

var CanSelect: Boolean);

begin

if MatrixGrid.Cells[LastRow, LastCol] = '' then

MatrixGrid.Cells[LastRow, LastCol] := '0';

CanSelect := true;

end;

procedure TForm1.MatrixGridExit(Sender: TObject);

begin

with MatrixGrid do

if Cells[LastRow, LastCol] = '' then

Cells[LastRow, LastCol] := '0';

end;

end.

**Код программы**

**(Java)**

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.util.Scanner;

import java.lang.Math;

import java.util.Arrays;

public class Main {

public static Scanner in = new Scanner(System.in);

public static int[][] matrix, bestWay = new int[0][0],

currentWay;

public static int maxSum = 1000000, i1, j1, i2, j2;

public static final int MAX\_ABS\_VALUE = 100;

public static final int MAX\_LENGTH = 9;

public static Boolean matrixReadError = false;

public static void inputDots() {

Boolean inValid = true;

String reader;

do {

try {

System.out.println("Coords of first dot [i1

j1]:");

reader = in.nextLine();

if (Integer.parseInt(reader.substring(0,

reader.indexOf(" "))) > matrix.length - 1

|| Integer.parseInt(reader.substring(0,

reader.indexOf(" "))) < 0) {

System.out.println("Coord is negative or

bigger than maximum");

} else {

i1 = Integer.parseInt(reader.substring(0,

reader.indexOf(" ")));

reader = reader.replace(i1 + " ", "");

}

if (Integer.parseInt(reader) >

matrix[0].length - 1 ||

Integer.parseInt(reader) < 0) {

System.out.println("Coord is negative or

bigger than maximum");

} else {

j1 = Integer.parseInt(reader);

inValid = false;

}

} catch (Exception e) {

System.out.println("INPUT ERROR");

}

} while (inValid);

inValid = true;

do {

try {

System.out.println("Coords of second dot [i2

j2]:");

reader = in.nextLine();

if (Integer.parseInt(reader.substring(0,

reader.indexOf(" "))) > matrix.length - 1 ||

Integer.parseInt(reader.substring(0,

reader.indexOf(" "))) < 0) {

System.out.println("Coord is negative or

bigger than maximum");

} else {

i2 = Integer.parseInt(reader.substring(0,

reader.indexOf(" ")));

reader = reader.replace(i2 + " ", "");

}

if (Integer.parseInt(reader) >

matrix[0].length - 1 ||

Integer.parseInt(reader) < 0) {

System.out.println("Coord is negative or

bigger than maximum");

} else {

j2 = Integer.parseInt(reader);

inValid = false;

}

} catch (Exception e) {

System.out.println("INPUT ERROR");

}

} while (inValid);

}

public static String matrixToStr(int[][] myMatrix) {

String answer = "";

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

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

answer = answer + myMatrix[i][j] + " ";

}

answer = answer + "\n";

}

return answer;

}

public static String bestWayToStr(int[][] myMatrix) {

String answer = "";

answer = answer + "Step :";

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

answer = answer + " " + (i + 1);

}

answer = answer + "\ni :";

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

answer = answer + " " + myMatrix[0][i];

}

answer = answer + "\nj :";

for (int i = 0; i < myMatrix[1].length; i++) {

answer = answer + " " + myMatrix[1][i];

}

answer = answer + "\n";

return answer;

}

public static int[] strToArr(String str, int len) {

int[] arr = new int[len];

int i = 0;

Boolean valid = true;

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

arr[j] = 0;

}

try {

while (str.indexOf(" ") > 0 && i < len && valid) {

if (Math.abs(Integer.parseInt(str.substring(0,

str.indexOf(" ")))) > MAX\_ABS\_VALUE) {

matrixReadError = true;

valid = false;

} else {

arr[i] = Integer.parseInt(str.substring(0,

str.indexOf(" ")));

str = str.replace(arr[i] + " ", "");

i++;

}

}

if (Math.abs(Integer.parseInt(str)) <

MAX\_ABS\_VALUE + 1) {

matrixReadError = true;

arr[i] = Integer.parseInt(str);

}

} catch (Exception e) {

}

return arr;

}

public static int[][] inputMatrixFromFile() throws

IOException {

Scanner fileScanner = new Scanner(System.in);

FileReader inF = new FileReader("E:\\University\\OAiP

\\LAB4\\Goal 2\\Java\\in.txt");

int[][] myMatrix = new int[0][0];

Boolean inValid = true;

int m, n;

do {

try {

System.out.println("Enter input file:");

inF = new FileReader(in.nextLine());

fileScanner = new Scanner(inF);

inValid = false;

} catch (Exception e) {

System.out.println("\nFile not found");

}

} while (inValid);

if (fileScanner.hasNext()) {

inValid = true;

try {

m = Integer.parseInt(fileScanner.nextLine());

n = Integer.parseInt(fileScanner.nextLine());

if (m > MAX\_LENGTH || n > MAX\_LENGTH) {

System.out.println("Entered size is too

large");

} else {

inValid = false;

}

myMatrix = new int[m][n];

} catch (Exception e) {

System.out.println("Size of matrix is

incorrect");

}

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

Arrays.fill(myMatrix[i], 0);

}

if (!inValid) {

int i = 0;

try {

while (fileScanner.hasNextLine() && i <

myMatrix.length) {

myMatrix[i] =

strToArr(fileScanner.nextLine(),

myMatrix.length);

i++;

}

} catch (Exception e) {

System.out.println("One or more elements

of matrix is incorrect");

}

}

} else {

System.out.println("File is empty");

}

inF.close();

return myMatrix;

}

public static void outMatrixToFile(String sMatrix) throws

IOException {

FileWriter outF = new FileWriter("out.txt");

Boolean inValid = true;

do {

try {

outF = new FileWriter(in.nextLine());

inValid = false;

} catch (Exception e) {

System.out.println("\nFile not found");

}

} while (inValid);

outF.write(sMatrix);

outF.close();

}

public static void findTheWay(int elementInWay, int i1S,

int j1S, int sum) {

if (i1S == i2 && j1S == j2) {

if (maxSum == 1000000 || sum > maxSum) {

maxSum = sum;

bestWay = new int[2][elementInWay + 1];

bestWay[0] = Arrays.copyOf(currentWay[0],

elementInWay + 1);

bestWay[1] = Arrays.copyOf(currentWay[1],

elementInWay + 1);

}

;

} else {

elementInWay++;

if (i1S < i2) {

currentWay[0][elementInWay] = i1S + 1;

currentWay[1][elementInWay] = j1S;

findTheWay(elementInWay, i1S + 1, j1S, sum +

matrix[i1S + 1][j1S]);

}

if (j1S < j2) {

currentWay[0][elementInWay] = i1S;

currentWay[1][elementInWay] = j1S + 1;

findTheWay(elementInWay, i1S, j1S + 1, sum +

matrix[i1S][j1S + 1]);

}

}

}

public static void solve() {

int temp;

currentWay = new int[2][matrix.length \*

matrix[0].length];

if ((i1 \* matrix[0].length + j1) > (i2 \*

matrix[0].length + j2)) {

temp = i1;

i1 = i2;

i2 = temp;

temp = j1;

j1 = j2;

j2 = temp;

}

findTheWay(0, i1, j1, matrix[i1][j1]);

bestWay[0][0] = i1;

bestWay[1][0] = j1;

}

public static void main(String[] args)throws IOException {

matrix = inputMatrixFromFile();

if (matrix.length > 0) {

System.out.println("Entered matrix:");

System.out.println(matrixToStr(matrix));

if (matrixReadError) {

System.out.println("There is an error during

reading matrix.");

}

System.out.println("\nEnter begin/end dots:");

inputDots();

solve();

System.out.println("\nBest way:");

System.out.println(bestWayToStr(bestWay));

System.out.println("Enter output file:");

outMatrixToFile(bestWayToStr(bestWay));

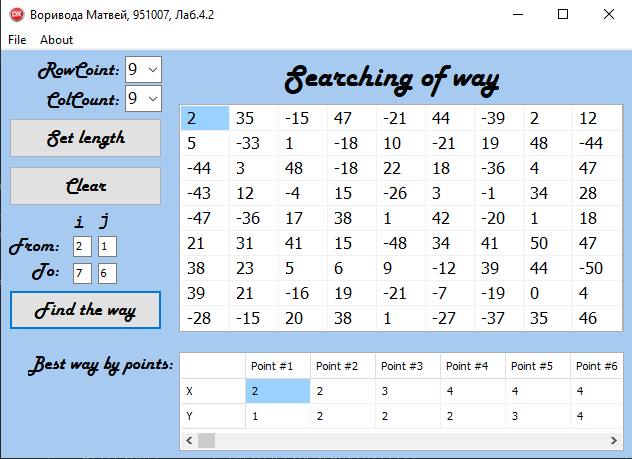
}

}

}

**Скриншоты**

А) Delphi



Б) Java

