УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №6.2

по предмету «Основы алгоритмизации и программирования»

Вариант 13

Выполнил:

Захвей И.В.

Гр. 351005

Проверил:

Данилова Г. В.

Минск 2024

**Задание:**

Составить программу, которая строит магические квадраты четного-нечетного порядка (6, 10, …). Визуализировать!

**Код программы Delphi:**

**MainUnit.pas**

Type

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

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

TGridCracker = Class(TStringGrid);

TSquare = Array Of Array Of Integer;

Const

INFTEXT = 'Построить Магический квадрат четно-нечетного порядка';

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

VOID = #0;

BACKSPACE = #8;

MAX\_SIZE = 18;

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;

TotalCounter: Integer;

FindedSquare: TSquare;

Implementation

{$R \*.dfm}

Procedure SwapElem(Arr: TSquare; I1, J1, I2, J2: Integer);

Var

Buff: Integer;

Begin

Buff := Arr[I1, J1];

Arr[I1, J1] := Arr[I2, J2];

Arr[I2, J2] := Buff;

End;

Function MagicBox(Size: Integer): TSquare;

Var

HalfSize, I, J, Number: Integer;

Square: TSquare;

Begin

SetLength(Square, Size, Size);

HalfSize := Size Div 2;

// Построение четырех квадратов порядка n / 2

I := 0;

J := (HalfSize - 1) Div 2;

Number := 1;

While Number <= HalfSize \* HalfSize Do

Begin

If I < 0 Then

Inc(I, HalfSize);

If I >= HalfSize Then

Dec(I, HalfSize);

If J >= HalfSize Then

Dec(J, HalfSize);

If J < 0 Then

Inc(J, HalfSize);

Square[J, I] := Number;

Square[J + HalfSize, I + HalfSize] := Number + HalfSize \* HalfSize;

Square[J, I + HalfSize] := Number + 2 \* HalfSize \* HalfSize;

Square[J + HalfSize, I] := Number + 3 \* HalfSize \* HalfSize;

If Number Mod HalfSize = 0 Then

Begin

Inc(I);

Dec(J); // for iteration

Inc(I);

End;

Inc(Number);

Inc(J);

Dec(I);

End;

// Меняем местами ломанные

SwapElem(Square, 0, 0, HalfSize, 0);

SwapElem(Square, HalfSize - 1, 0, Size - 1, 0);

For I := 1 To HalfSize - 2 Do

Begin

SwapElem(Square, I, 1, I + HalfSize, 1)

End;

// Меняем местами столбцы

For J := HalfSize - ((HalfSize - 3) Div 2) To HalfSize + (HalfSize - 3)

Div 2 - 1 Do

Begin

For I := 0 To HalfSize - 1 Do

Begin

SwapElem(Square, J, I, J, I + HalfSize)

End;

End;

MagicBox := Square;

End;

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

Var

I: Integer;

Begin

Grid.Visible := True;

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

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

Grid.ColCount := Size;

Grid.RowCount := Size;

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.SizeEditChange(Sender: TObject);

Begin

CheckButton.Enabled := False;

SaveButtonMenu.Enabled := False;

If SizeEdit.Text = '' Then

Begin

MatrixLabel.Visible := False;

ClearGrid(SquareGrid);

CheckButton.Enabled := False;

End

Else If (StrToInt(SizeEdit.Text) Mod 2 = 0) And

(StrToInt(SizeEdit.Text) Mod 4 <> 0) And

(StrToInt(SizeEdit.Text) > 2) Then

Begin

MatrixLabel.Visible := True;

ClearGrid(SquareGrid);

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

CheckButton.Enabled := True;

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 (SquareGrid.Visible) Then

SquareGrid.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.TimerTimer(Sender: TObject);

Var

I, J: Integer;

Begin

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

For J := Low(FindedSquare) To High(FindedSquare) Do

if FindedSquare[I, J] = TotalCounter then

SquareGrid.Cells[J, I] := IntTostr(FindedSquare[I, J]);

Inc(TotalCounter);

if TotalCounter > SquareGrid.ColCount \* SquareGrid.ColCount then

Timer.Interval := 0;

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): 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 (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: Integer;

Err: ERRORS\_CODE;

Begin

Err := SUCCESS;

If OpenTextFileDialog.Execute() Then

Begin

AssignFile(InfFile, OpenTextFileDialog.FileName);

Reset(InfFile);

Err := ReadOneFromFile(Size, InfFile);

If Not EoF(InfFile) Then

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

If Err = SUCCESS Then

Begin

SizeEdit.Text := IntToStr(Size);

FillGrid(Size, SquareGrid);

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);

With SquareGrid Do

Begin

For I := 0 To ColCount - 1 Do

Begin

For J := 0 To ColCount - 1 Do

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

Writeln(OutFile);

End;

End;

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.CheckButtonClick(Sender: TObject);

Var

Size: Integer;

Begin

Size := StrToInt(SizeEdit.Text);

SetLength(FindedSquare, Size, Size);

FindedSquare := MagicBox(Size);

Timer.Interval := 500;

TotalCounter := 1;

SaveButtonMenu.Enabled := True;

End;

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

Shift: TShiftState);

Begin

If (Key = VK\_UP) Then

SquareGrid.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;

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,

IN\_OUT\_FILE\_EXCEPTION,

}

static final int MIN\_SIZE = 3,

MAX\_SIZE = 18,

MIN\_CHOICE = 1,

MAX\_CHOICE = 2;

static final String START\_INFO = """

Program calculate magic square with size 6, 10, 14, ...

""";

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

"Data is not correct, it should be like in instruction\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.equals(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, MIN\_CHOICE, MAX\_CHOICE, false);

if (err != ErrCode.SUCCESS) {

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

}

} while (err != ErrCode.SUCCESS);

return choiceArr[0];

}

static int inputSizeConsole(Scanner input) {

int[] sizeArr = {0};

ErrCode err;

do {

System.out.println("Enter size of square like (6, 10, 14, ...)");

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 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 readSizeFile(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[] arrSize = new int[1];

ErrCode err;

String fileName;

do {

fileName = getFileName(input);

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

err = readSizeFile(arrSize, file);

} catch (IOException e) {

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

}

if (err != ErrCode.SUCCESS) {

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

}

} while (err != ErrCode.SUCCESS);

return arrSize[0];

}

static int inputInf(Scanner input) {

int size;

int choice;

choice = userChoice(input);

if (choice == 1) {

size = inputSizeConsole(input);

} else {

size = inputFromFile(input);

}

return size;

}

static void outputToConsole(int[][] square) {

for (int[] row : square) {

for (int element : row) {

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

}

System.out.println();

}

}

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

ErrCode err;

do {

err = ErrCode.SUCCESS;

String fileName = getFileName(input);

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

for (int[] row : square) {

for (int element : row) {

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

}

file.println();

}

} catch (IOException e) {

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

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

}

} while (err != ErrCode.SUCCESS);

}

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

int choice = userChoice(input);

if (choice == 1) {

outputToConsole(square);

} else {

outputToFile(square, input);

}

}

static void printInf() {

System.out.println(START\_INFO);

}

static int[][] calcSquare(int size) {

int[][] square = new int[size][size];

int n2 = size / 2;

// Построение четырех квадратов порядка size / 2

for (int i = 0, j = (n2 - 1) / 2, number = 1; number <= n2 \* n2; number++, j++, i--) {

if (i < 0)

i += n2;

if (i >= n2)

i -= n2;

if (j >= n2)

j -= n2;

if (j < 0)

j += n2;

square[i][j] = number;

square[i + n2][j + n2] = number + n2 \* n2;

square[i][j + n2] = number + 2 \* n2 \* n2;

square[i + n2][j] = number + 3 \* n2 \* n2;

if (number % n2 == 0) {

i++;

j--; // for iteration

i++;

}

}

// Меняем местами ломанные

int temp = square[0][0];

square[0][0] = square[n2][0];

square[n2][0] = temp;

temp = square[n2 - 1][0];

square[n2 - 1][0] = square[size - 1][0];

square[size - 1][0] = temp;

for (int i = 1, j = 1; i < n2 - 1; i++) {

temp = square[i][j];

square[i][j] = square[i + n2][j];

square[i + n2][j] = temp;

}

// Свап столбцов

int numOfColumnsToSwap = n2 - ((n2 - 3) / 2);

for (int j = numOfColumnsToSwap; j < n2 + (n2 - 3) / 2; j++) {

for (int i = 0; i < n2; i++) {

temp = square[i][j];

square[i][j] = square[i + n2][j];

square[i + n2][j] = temp;

}

}

return square;

}

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int size;

int[][] square;

printInf();

size = inputInf(input);

square = calcSquare(size);

outputInf(square, input);

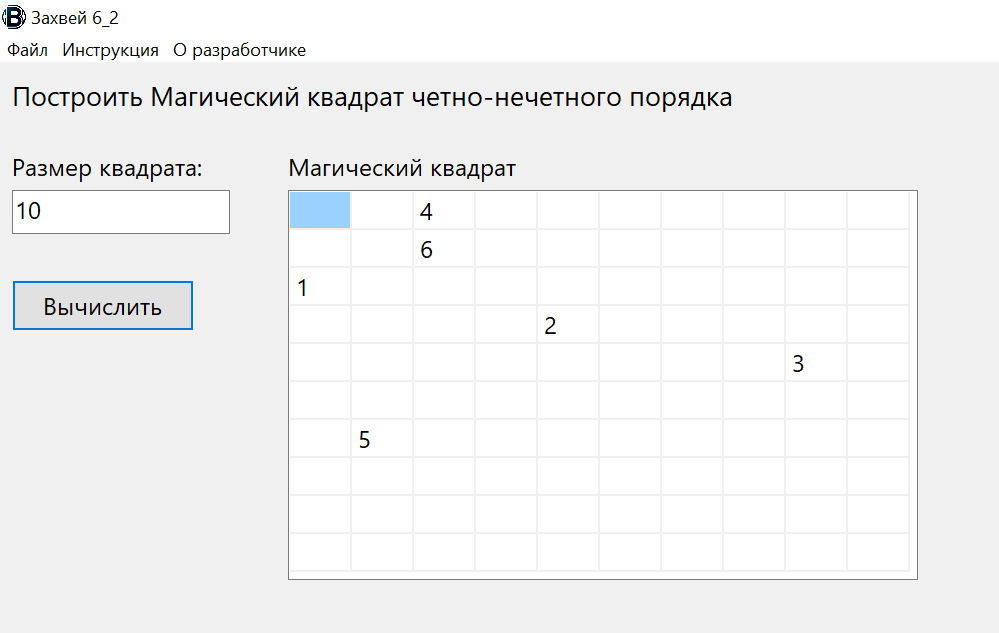
input.close();

}

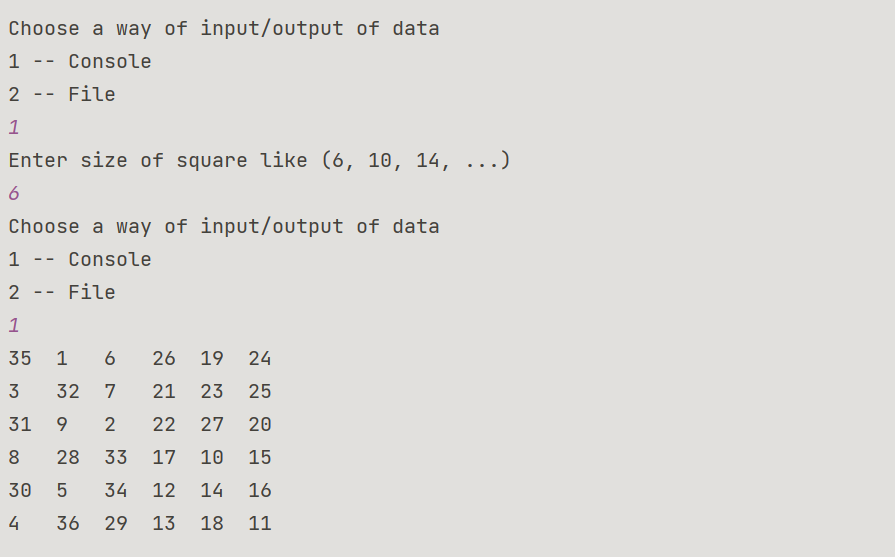
}

**Скриншоты:**

**Delphi****:**

****

**Java:**



**Блок-схема:**

