УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №4.3

по предмету «Основы алгоритмизации и программирования»

Вариант 4

Выполнил:

Бражалович А. И.

Гр. 351004

Проверил:

Данилова Г. В.

Минск 2024

**Задание:**

Решить головоломку «Ханойская башня» с n количеством дисков.

**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

TMatrix = Array Of Array Of Integer;

// TStepArr = Array [1..2] Of Array Of Integer;

TVector = Array [1 .. 2] Of Array [1 .. 2] Of Real;

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

FILE\_EMPTY, EXTRA\_DATA);

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;

EnterNEdit: TEdit;

EnterNLabel: TLabel;

SaveTextFile: TSaveTextFileDialog;

OutGrid: TStringGrid;

OutLabel: TLabel;

Procedure DeveloperMenuClick(Sender: TObject);

Procedure InstructionMenuClick(Sender: TObject);

Procedure FormCreate(Sender: TObject);

Procedure EnterNEditContextPopup(Sender: TObject; MousePos: TPoint;

Var Handled: Boolean);

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

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

Procedure ResultButtonClick(Sender: TObject);

Procedure EnterNEditChange(Sender: TObject);

Procedure QuitMenuClick(Sender: TObject);

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

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

Shift: TShiftState);

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

Procedure EnterNEditExit(Sender: TObject);

Procedure EnterNEditClick(Sender: TObject);

Procedure SaveMenuClick(Sender: TObject);

Procedure OpenMenuClick(Sender: TObject);

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 MainFunction();

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!', 'Лишние данные!');

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

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

BACKSPACE = #8;

NONE = #0;

MIN\_N = 1;

MAX\_N = 15;

MAX\_SIGNS = 4;

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

Var

MainTaskForm: TMainTaskForm;

Implementation

Uses

Math;

{$R \*.dfm}

Var

Saved: Boolean = True;

PerformCloseQuery: Boolean = True;

CtrlPressed: Boolean = False;

Procedure TMainTaskForm.DeveloperMenuClick(Sender: TObject);

Var

DeveloperForm: TDeveloperForm;

Begin

DeveloperForm := TDeveloperForm.Create(Self);

DeveloperForm.ShowModal;

DeveloperForm.Free;

End;

Procedure TMainTaskForm.FormCreate(Sender: TObject);

Begin

ResultButton.Enabled := False;

TaskLabel.Caption := 'Данная программа находит решение головоломки' + #13#10

+ '"Ханойская башня" с n количеством дисков' + #13#10 + '';

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.EnterNEditContextPopup(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';

If (Length(Edit.Text) > 0) And (Edit.Text[1] = ',') Then

Edit.Text := '';

End;

Procedure TMainTaskForm.EnterNEditExit(Sender: TObject);

Begin

GiveZeroOrNone(EnterNEdit);

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

Var

ERRORS: ERRORS\_LIST;

Value: Integer;

Begin

ERRORS := CORRECT;

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 + 1, MAX\_N);

End;

If Not EOF(F) Then

ERRORS := EXTRA\_DATA;

CloseFile(F);

CheckFileData := ERRORS;

End;

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

Var

NumStr: String;

Begin

Reset(F);

Readln(F, NumStr);

EnterNEdit.Text := NumStr;

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

ResultButton.Enabled := True;

FileReading := ERRORS;

End;

Procedure TMainTaskForm.OpenMenuClick(Sender: TObject);

Var

Error: ERRORS\_LIST;

F: TextFile;

Num, 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;

Procedure FillGridFromMatrix(ResMatrix: TMatrix; OutGrid: TStringGrid);

Var

I, J, MaxI, MaxJ: Integer;

Counter: Integer;

Begin

MaxI := High(ResMatrix[0]);

MaxJ := High(ResMatrix);

Counter := 0;

For I := 0 To MaxJ Do

Begin

Inc(Counter);

OutGrid.Cells[I + 1, 0] := Concat(IntToStr(Counter), ' Шаг');

End;

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

OutGrid.Cells[0, 1] := 'Откуда';

OutGrid.Cells[0, 2] := 'Куда';

For I := 0 To MaxI Do

Begin

For J := 0 To MaxJ Do

Begin

OutGrid.Cells[J + 1, I + 1] := IntToStr(ResMatrix[J, I]);

End;

End;

End;

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

Var

I, J: Integer;

NumArr: TMatrix;

Begin

If Columns > 4 Then

Begin

Grid.Width := (Grid.DefaultColWidth + 3) \* 4;

Grid.Height := Grid.DefaultRowHeight \* 4;

End

Else

Begin

Grid.Width := (Grid.DefaultColWidth) \* (Columns + 1);

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

End;

Grid.Enabled := True;

Grid.ColCount := Columns + 2;

Grid.RowCount := 3;

Grid.FixedCols := 1;

Grid.FixedRows := 1;

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.EnterNEditChange(Sender: TObject);

Begin

If (EnterNEdit.Text = '') OR (EnterNEdit.Text = '1') Then

Begin

Saved := True;

SaveMenu.Enabled := False;

OutGrid.Visible := False;

ClearGrid(OutGrid);

ResultButton.Enabled := False;

End

Else

Begin

ResultButton.Enabled := True;

ClearGrid(OutGrid);

End;

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

Rewrite(F);

For I := 0 To OutGrid.ColCount Do

Begin

For J := 0 To OutGrid.ColCount Do

Begin

Write(F, OutGrid.Cells[I, J]);

If J < OutGrid.ColCount - 1 Then

Write(F, ' ');

End;

Writeln(F);

End;

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);

For I := 0 To OutGrid.ColCount Do

Begin

For J := 0 To OutGrid.ColCount Do

Begin

Write(F, OutGrid.Cells[I, J], ' ');

End;

Writeln(F);

End;

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;

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.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;

Procedure BlockClick(Edit: TEdit);

Begin

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

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

End;

Procedure MoveDisk(FromStick, ToStick: Integer; Var StepArr: TMatrix;

Var Counter: Integer);

Begin

StepArr[Counter][0] := FromStick;

StepArr[Counter][1] := ToStick;

Inc(Counter);

End;

Procedure HanoiTower(N: Integer; FromStick, ToStick, BufStick: Integer;

Var StepArr: TMatrix; Var Counter: Integer);

Begin

If N > 0 Then

Begin

HanoiTower(N - 1, FromStick, BufStick, ToStick, StepArr, Counter);

MoveDisk(FromStick, ToStick, StepArr, Counter);

HanoiTower(N - 1, BufStick, ToStick, FromStick, StepArr, Counter);

End;

End;

Procedure TMainTaskForm.MainFunction();

Var

StepArr: TMatrix;

N: Byte;

Len: Double;

Counter: Integer;

Begin

Counter := 0;

N := StrToInt(EnterNEdit.Text);

SetLength(StepArr, Round(Power(2, N)) - 1, 2);

HanoiTower(N, 1, 3, 2, StepArr, Counter);

FillGrid(High(StepArr), OutGrid);

FillGridFromMatrix(StepArr, OutGrid);

End;

Procedure TMainTaskForm.EnterNEditClick(Sender: TObject);

Begin

BlockClick(EnterNEdit);

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);

Begin

MainFunction();

OutGrid.Visible := True;

OutGrid.Enabled := True;

If Not IsGridFull(OutGrid) Then

Begin

Saved := True;

SaveMenu.Enabled := False;

End

Else

Begin

Saved := False;

SaveMenu.Enabled := True;

End;

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;

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;

CloseButton: TButton;

Procedure CloseButtonClick(Sender: TObject);

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.CloseButtonClick(Sender: TObject);

Begin

Close;

End;

Procedure TInstructionForm.FormCreate(Sender: TObject);

Begin

CenterFormOnScreen(Self);

InstructionLabel.Caption := '1. Введите количество дисков у башни [1; 15].'

+ #13#10 +

'2. В файле должно содержаться количество дисков башни на первой строке';

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\_DISK = 1;

public static final int MAX\_DISK = 20;

public static int stepCounter = 0;

public static int[][] steps;

public enum ERRORS\_LIST {

CORRECT,

RANGE\_ERR,

NUM\_ERR,

NOT\_TXT,

NOT\_EXIST,

NOT\_READABLE,

NOT\_WRITEABLE,

CHOICE\_ERR,

FILE\_EMPTY

}

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!",

"Проверьте корректность выбора!",

"Файл пуст!",

"Лишние данные!"

};

public static void printTask() {

System.out.println("Данная программа решает головоломку 'Ханойская башня':\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 = Integer.parseInt(inputScanner.nextLine());

else

number = inputScanner.nextInt();

} 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 int inputOption(Scanner inpuScanner) {

int option;

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 readFileNum (Scanner inputScanner) {

int disks = 0;

int[] numArr = new int[1];

File file;

ERRORS\_LIST error;

do {

file = fileReading(inputScanner);

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

error = readOneNum(scanFile, numArr, MIN\_DISK, MAX\_DISK, true);

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

printError(error);

} else {

disks = numArr[0];

}

} catch (Exception e) {

error = ERRORS\_LIST.NOT\_READABLE;

printError(error);

}

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

return disks;

}

public static int readConsoleNum (Scanner inputScanner) {

int disks = 0;

int[] numArr = new int[1];

ERRORS\_LIST error;

do {

System.out.print("Введите количество дисков: [" + MIN\_DISK + "; " + MAX\_DISK + "]: ");

error = readOneNum(inputScanner, numArr, MIN\_DISK, MAX\_DISK, false);

if (error != ERRORS\_LIST.CORRECT)

printError(error);

else {

disks = numArr[0];

}

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

return disks;

}

public static int inputNum(Scanner inputScanner, int option) {

int disks;

if (option == 1) {

disks = readFileNum(inputScanner);

} else {

disks = readConsoleNum(inputScanner);

}

return disks;

}

public static void moveDisk(int fromStick, int toStick, int[][] stepArr) {

stepArr[stepCounter][0] = fromStick;

stepArr[stepCounter][1] = toStick;

stepCounter++;

}

public static void hanoiTower(int n, int fromStick, int toStick, int bufStick, int[][] stepArr) {

if (n > 0) {

hanoiTower(n - 1, fromStick, bufStick, toStick, stepArr);

moveDisk(fromStick, toStick, stepArr);

hanoiTower(n - 1, bufStick, toStick, fromStick, stepArr);

}

}

public static void printResult(Scanner inputScanner, int [][] steps) {

ERRORS\_LIST error;

File file;

int option = 0;

int j = 0;

int counter = 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)) {

while (j < steps.length) {

counter++;

writer.write("\n " + counter + " шаг: " + "с " + steps[j][0] + " на " + steps[j][1]);

j++;

}

} catch (Exception e) {

error = ERRORS\_LIST.NOT\_WRITEABLE;

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

}

} else {

counter = 0;

while (j < steps.length) {

counter++;

System.out.printf("\n " + counter + " шаг: " + "с " + steps[j][0] + " на " + steps[j][1]);

j++;

}

}

}

public static void main(String[] args) {

Scanner inpScanner = new Scanner(System.in);

int disks = 0;

int option;

printTask();

option = inputOption(inpScanner);

disks = inputNum(inpScanner, option);

steps = new int[(int) (Math.pow(2, disks) - 1)][2];

hanoiTower(disks, 1, 3,2, steps);

printResult(inpScanner, steps);

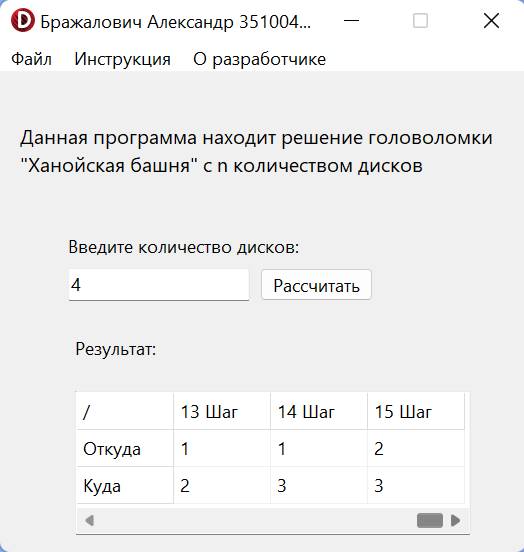
inpScanner.close();

}

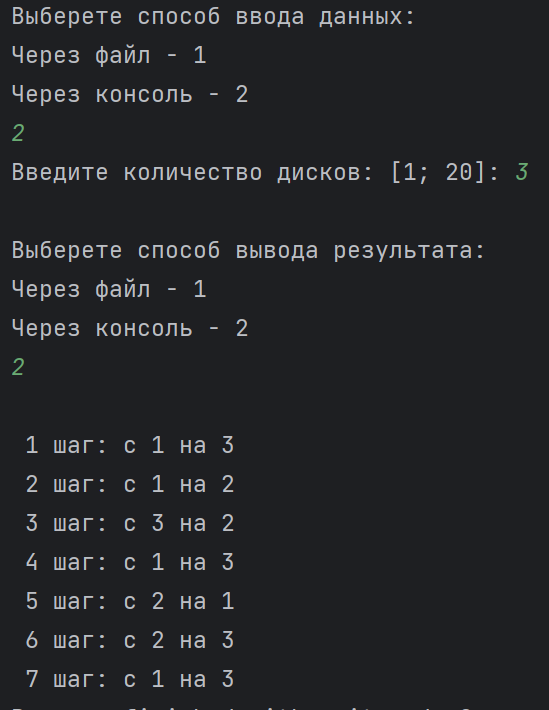
}

**Скриншоты:**

**Delphi:**

****

**Java:**

****

**Блок-схема:**

