УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №5.2

по предмету «Основы алгоритмизации и программирования»

Вариант 18

Выполнил:

Машевский Д.В

Гр. 351003

Проверил:

Данилова Г. В.

Минск 2024

**Задание:**

Вывести номера вершин, для которых высота левого поддерева не равна высоте правого поддерева. Дерево визуализировать.

**Код программы Delphi:**

**unit Unit4Laba52;**

interface

uses

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

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

Vcl.ComCtrls, Vcl.StdCtrls, Vcl.ExtCtrls, Vcl.Grids, Vcl.Menus,

Vcl.Samples.Spin, Vcl.Buttons, Clipbrd;

Const

MAX\_LENGTH\_VALUE = 1000;

MAX\_VALUE = 1000;

CORRECT\_SYMBOLS: Set of Char = ['-', '0' .. '9', #8];

CORRECT\_FILE\_SYMBOLS: Set of Char = ['-', '0' .. '9', #10, #13, ' '];

type

PBinaryTreeNode = ^TBinaryTreeNode;

TBinaryTreeNode = record

Data: Integer;

Left, Right: PBinaryTreeNode;

end;

TArr = array of Integer;

TMainForm = class(TForm)

ButtonPushNodeTree: TButton;

TreePaint: TPaintBox;

MainMenuMain: TMainMenu;

NFile: TMenuItem;

NInstruction: TMenuItem;

NDeveloper: TMenuItem;

NOpenFile: TMenuItem;

NSaveFile: TMenuItem;

NExitProgram: TMenuItem;

PopupMenuMain: TPopupMenu;

LabelGiveInfoSpiEdit: TLabel;

SpinEditPushElement: TSpinEdit;

ButtonDelFullTree: TBitBtn;

MemoGiveFinalResult: TMemo;

LabelPaint: TLabel;

Button1: TButton;

ButtonClearMemo: TBitBtn;

procedure DrawBinaryTree(X, Y, CurLength: Integer;

Node: PBinaryTreeNode);

procedure TreePaintPaint(Sender: TObject);

procedure ButtonPushNodeTreeClick(Sender: TObject);

procedure NDeveloperClick(Sender: TObject);

procedure NInstructionClick(Sender: TObject);

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

procedure NExitProgramClick(Sender: TObject);

procedure ButtonDelFullTreeClick(Sender: TObject);

procedure FindNodesWithDifferentHeights(Node: PBinaryTreeNode;

var NodesWithDifferentHeights: string);

procedure SpinEditPushElementKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

procedure WriteStrIntoFile(Path: string; Massiv: TArr);

procedure SpinEditPushElementClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure SpinEditPushElementKeyPress(Sender: TObject; var Key: Char);

function GetTreeHeight(Node: PBinaryTreeNode): Integer;

procedure Button1Click(Sender: TObject);

procedure NOpenFileClick(Sender: TObject);

procedure NSaveFileClick(Sender: TObject);

procedure ButtonClearMemoClick(Sender: TObject);

private

treeRoot: PBinaryTreeNode;

buttonClickCount: Integer;

public

{ Public declarations }

end;

var

MainForm: TMainForm;

MainRoot: PBinaryTreeNode;

TreeSize: Integer;

implementation

{$R \*.dfm}

uses UntFor52;

Function IsInt(Value: Char): Integer;

External 'C:\Users\Даниил\Documents\Embarcadero\Studio\Projects\Win32\Debug\Library52.dll';

procedure TMainForm.DrawBinaryTree(X, Y, CurLength: Integer;

Node: PBinaryTreeNode);

var

D, DataWidth: Integer;

begin

D := 20;

DataWidth := TreePaint.Canvas.TextWidth(IntToStr(Node.Data));

if Node.Left <> nil then

begin

TreePaint.Canvas.MoveTo(X, Y + 13);

TreePaint.Canvas.LineTo(X - CurLength, Y + 50);

DrawBinaryTree(X - CurLength, Y + 50, CurLength div 2, Node.Left);

end;

if Node.Right <> nil then

begin

TreePaint.Canvas.MoveTo(X, Y + 13);

TreePaint.Canvas.LineTo(X + CurLength, Y + 50);

DrawBinaryTree(X + CurLength, Y + 50, CurLength div 2, Node.Right);

end;

TreePaint.Canvas.Ellipse(X - 13, Y - 13, X + 13, Y + 13);

TreePaint.Canvas.TextOut(X - 5, Y - 5, IntToStr(Node.Data));

end;

procedure TMainForm.TreePaintPaint(Sender: TObject);

begin

// Очищаем холст перед каждой отрисовкой

TreePaint.Canvas.FillRect(TreePaint.ClientRect);

// Получаем корневой узел дерева

if Assigned(treeRoot) then

begin

// Устанавливаем начальную позицию для отображения дерева

DrawBinaryTree(TreePaint.Width div 2, 50, TreePaint.Width div 4,

treeRoot);

end;

end;

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

var

UserChoice: Integer;

Begin

CanClose := False;

UserChoice := MessageBox(Handle, 'Вы уверены, что хотите выйти?',

'Подтверждение', MB\_ICONQUESTION or MB\_YESNO);

If UserChoice = IDYES then

CanClose := True;

end;

procedure TMainForm.FormCreate(Sender: TObject);

begin

MemoGiveFinalResult.Text := '';

MemoGiveFinalResult.Enabled := False;

MainMenuMain.Items[0].Items[1].Enabled := False;

end;

procedure TMainForm.NInstructionClick(Sender: TObject);

Var

MessageBoxCaption: String;

MessageBoxText: String;

IsMessageShow: Boolean;

Begin

IsMessageShow := False;

if not IsMessageShow then

Begin

IsMessageShow := True;

MessageBoxCaption := 'Инструкция';

MessageBoxText :=

'1)Значения, вводимые в дерево не должны превышать 1000.'#10'2)При работе с

файлом, файл должен быть формата .txt'#10'3)В файле должны быть записаны

только числа'#10'4)Пример значений в файле:'#10' 53 432 434 321 654

231 3';

MessageBox(Handle, PChar(MessageBoxText),

PChar(MessageBoxCaption), MB\_OK);

End;

IsMessageShow := False;

end;

procedure TMainForm.SpinEditPushElementClick(Sender: TObject);

begin

Clipboard.Clear; // Очищает содержимое буфера обмена

end;

procedure TMainForm.SpinEditPushElementKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

if (Key = VK\_INSERT) and (SsShift in Shift) then

Key := 0;

if (Key = Ord('V')) and (SsCtrl in Shift) then

Key := 0;

end;

procedure TMainForm.SpinEditPushElementKeyPress(Sender: TObject; var Key: Char);

var

Value: Integer; // Declare the Value variable

begin

if TryStrToInt(SpinEditPushElement.Text + Key, Value) then

if Value > MAX\_LENGTH\_VALUE then

Key := #0;

end;

procedure TMainForm.NDeveloperClick(Sender: TObject);

Var

MessageBoxCaption: String;

MessageBoxText: String;

IsMessageShow: Boolean;

Begin

IsMessageShow := False;

if not IsMessageShow then

Begin

IsMessageShow := True;

MessageBoxCaption := 'О разработчике';

MessageBoxText :=

'Машевский Даниил Витальевич, группа 351003, Лабораторная №

5.2.'#10''#10'Задача:'#10'Вывести номера вершин, для которых высота левого

поддерева не равна высоте правого поддерева. Дерево визуализировать.';

MessageBox(Handle, PChar(MessageBoxText),

PChar(MessageBoxCaption), MB\_OK);

End;

IsMessageShow := False;

end;

procedure TMainForm.NExitProgramClick(Sender: TObject);

var

UserChoice: Integer;

Begin

MainForm.close;

end;

function TMainForm.GetTreeHeight(Node: PBinaryTreeNode): Integer;

var

LeftHeight, RightHeight: Integer;

begin

if Node = nil then

Result := 0

else

begin

// Рекурсивно вычисляем высоты левого и правого поддеревьев

LeftHeight := GetTreeHeight(Node.Left);

RightHeight := GetTreeHeight(Node.Right);

// Возвращаем максимальную высоту поддерева плюс 1 (для текущего узла)

if LeftHeight > RightHeight then

Result := LeftHeight + 1

else

Result := RightHeight + 1;

end;

end;

procedure TMainForm.FindNodesWithDifferentHeights(Node: PBinaryTreeNode;

var NodesWithDifferentHeights: string);

var

LeftHeight, RightHeight: Integer;

begin

if Node = nil then

Exit;

// Рекурсивно вычисляем высоты левого и правого поддеревьев узла

LeftHeight := GetTreeHeight(Node.Left);

RightHeight := GetTreeHeight(Node.Right);

// Проверяем, если высоты левого и правого поддеревьев не равны

if LeftHeight <> RightHeight then

begin

// Добавляем номер узла в список вершин с разными высотами поддеревьев

NodesWithDifferentHeights := NodesWithDifferentHeights +

IntToStr(Node.Data) + ', ';

end;

// Рекурсивно выполняем для левого и правого поддеревьев

FindNodesWithDifferentHeights(Node.Left, NodesWithDifferentHeights);

FindNodesWithDifferentHeights(Node.Right, NodesWithDifferentHeights);

end;

procedure TMainForm.Button1Click(Sender: TObject);

var

NodesWithDifferentHeights: string;

begin

NodesWithDifferentHeights := '';

// Поиск вершин с разными высотами поддеревьев

FindNodesWithDifferentHeights(treeRoot, NodesWithDifferentHeights);

// Вывод результатов в компонент TMemo

MemoGiveFinalResult.Lines.Text := 'Вершины с разными высотами поддеревьев: '

+ NodesWithDifferentHeights;

MainMenuMain.Items[0].Items[1].Enabled := True;

end;

procedure TMainForm.ButtonClearMemoClick(Sender: TObject);

begin

MemoGiveFinalResult.Clear;

MainMenuMain.Items[0].Items[1].Enabled := False;

end;

procedure TMainForm.ButtonDelFullTreeClick(Sender: TObject);

begin

ClearTree(treeRoot);

// Удаление всех значений из дерева (treeRoot - корень дерева)

TreePaint.Canvas.FillRect(TreePaint.ClientRect); // Очистка PaintBox

buttonClickCount := 0; // Сброс счетчика нажатий кнопки

ButtonPushNodeTree.Enabled := True;

// Включение кнопки для добавления узлов в дерево

end;

function CheckMaxValueInMatrices(const FilePath: string): Boolean;

var

F: TextFile;

Value: Integer;

Line: string;

FoundMaxValue: Boolean;

begin

Result := True;

FoundMaxValue := False;

AssignFile(F, FilePath);

Reset(F);

try

while not Eof(F) do

begin

ReadLn(F, Line); // Читаем строку из файла

if Line <> '' then

begin

var

Elements := Line.Split([' ']);

for var Element in Elements do

begin

if TryStrToInt(Element, Value) then

begin

if Value > MAX\_VALUE then

begin

Result := False;

FoundMaxValue := True;

end;

end;

end;

end;

if FoundMaxValue then

Break;

end;

finally

CloseFile(F);

end;

if not Result then

begin

ShowMessage

('В файле некорректные данные:'#10'Было найдено значение, превышающее

максимальное.');

CheckMaxValueInMatrices := False;

end

else

CheckMaxValueInMatrices := True;

end;

Function CheckFileMatrixValues(FileInputPath: String): Boolean;

Var

InputFile: TextFile;

Value: Char;

IsChecked, Fl: Boolean;

Counter: Integer;

IsHigher: Boolean;

Begin

IsChecked := False;

IsHigher := False;

Counter := 0;

AssignFile(InputFile, FileInputPath);

Reset(InputFile);

While (not(Eof(InputFile)) and (IsChecked = False)) do

Begin

Read(InputFile, Value);

If IsInt(Value) > 0 then

Begin

IsChecked := True;

Inc(Counter);

End;

End;

If (Counter > 0) then

Begin

ShowMessage

('В файле некорректные данные:'#10'Был найден символ,не являющийся

числом');

CheckFileMatrixValues := False;

End

Else

CheckFileMatrixValues := True;

CloseFile(InputFile);

End;

Function CheckFile(Path: String): Boolean;

Var

Fl: Boolean;

InputFile: TextFile;

Begin

Fl := True;

Try

AssignFile(InputFile, Path);

Reset(InputFile);

Except

ShowMessage('Произошла ошибка:файл занят другим процессом');

Fl := False;

End;

CloseFile(InputFile);

CheckFile := Fl;

End;

procedure TMainForm.NOpenFileClick(Sender: TObject);

var

FileLines: TStringList;

i, j, Value: Integer;

OpenDialog: TOpenDialog;

FileContent: string;

CurrentValue: string;

begin

OpenDialog := TOpenDialog.Create(nil);

OpenDialog.Title := 'Выберите файл';

OpenDialog.Filter := 'Текстовые файлы (\*.txt)|\*.txt';

if OpenDialog.Execute then

begin

if CheckMaxValueInMatrices(OpenDialog.FileName) = True then

begin

if CheckFileMatrixValues(OpenDialog.FileName) = True Then

Begin

if CheckFile(OpenDialog.FileName) = True then

Begin

// Создайте корневой узел дерева

treeRoot := nil;

// Откройте файл для чтения

FileLines := TStringList.Create;

try

FileLines.LoadFromFile(OpenDialog.FileName);

// Прочитайте значения из файла и добавьте их в дерево

for i := 0 to FileLines.Count - 1 do

begin

FileContent := FileLines[i];

CurrentValue := '';

for j := 1 to Length(FileContent) do

begin

if FileContent[j] in ['0' .. '9'] then

CurrentValue := CurrentValue +

FileContent[j]

else if CurrentValue <> '' then

begin

if TryStrToInt(CurrentValue, Value) then

begin

if treeRoot = nil then

begin

New(treeRoot);

treeRoot^.Data := Value;

treeRoot^.Left := nil;

treeRoot^.Right := nil;

end

else

begin

AddToTree(treeRoot, Value);

ButtonPushNodeTreeClick(Sender);

end;

end;

CurrentValue := '';

end;

end;

if CurrentValue <> '' then

begin

if TryStrToInt(CurrentValue, Value) then

begin

if treeRoot = nil then

begin

New(treeRoot);

treeRoot^.Data := Value;

treeRoot^.Left := nil;

treeRoot^.Right := nil;

end

else

begin

AddToTree(treeRoot, Value);

ButtonPushNodeTreeClick(Sender);

end;

end;

end;

end;

// Перерисовываем окно для отображения обновленного дерева

TreePaint.Repaint;

finally

FileLines.Free;

end;

End;

End;

end;

end;

end;

procedure TMainForm.WriteStrIntoFile(Path: string; Massiv: TArr);

var

F: TextFile;

i: Integer;

begin

AssignFile(F, Path);

Rewrite(F);

try

// Write the header message

Writeln(F, 'Результат вычислений:');

Writeln(F);

Writeln(F, MemoGiveFinalResult.Text, '');

finally

CloseFile(F);

end;

end;

procedure TMainForm.NSaveFileClick(Sender: TObject);

var

SaveDialog: TSaveDialog;

MText, S1, S2: String;

FileName, Path: String;

MessageBoxCaption: String;

MessageBoxText: String;

IsMessageShow: Boolean;

InputFile: TextFile;

Massiv: TArr;

i, j, ArrLen: Integer;

begin

SaveDialog := TSaveDialog.Create(nil);

SaveDialog.Title := 'Сохранить файл';

SaveDialog.Filter := 'Текстовые файлы (\*.txt)|\*.txt';

SaveDialog.DefaultExt := 'txt';

if SaveDialog.Execute then

begin

FileName := SaveDialog.FileName;

if not FileName.EndsWith('.txt') then

FileName := FileName + '.txt';

Path := FileName;

WriteStrIntoFile(Path, Massiv);

SaveDialog.Free;

IsMessageShow := False;

if not IsMessageShow then

Begin

IsMessageShow := True;

MessageBoxCaption := 'Работа с файлом:';

MessageBoxText := 'Запись в файл прошла успешно!';

MessageBox(Handle, PChar(MessageBoxText),

PChar(MessageBoxCaption), MB\_OK);

End;

IsMessageShow := False;

end;

end;

procedure TMainForm.ButtonPushNodeTreeClick(Sender: TObject);

var

Value: Integer;

begin

// Получите значение элемента, который нужно добавить

Value := SpinEditPushElement.Value;

// Если корень еще не был установлен, создайте новый узел с заданным значением

if treeRoot = nil then

begin

New(treeRoot);

treeRoot^.Data := Value;

treeRoot^.Left := nil;

treeRoot^.Right := nil;

end

else

begin

// Вызов функции для добавления элемента в дерево

AddToTree(treeRoot, Value);

end;

// Перерисовываем окно для отображения обновленного дерева

TreePaint.Repaint;

// Проверяем количество нажатий

if buttonClickCount >= 20 then

begin

// Блокируем кнопку

ButtonPushNodeTree.Enabled := False;

ShowMessage('Вы достигли ограничения на добавление элементов в дерево');

end;

MemoGiveFinalResult.Lines.Text := '';

end;

end.

**Внешний файл:**

**unit UntFor52;**

interface

uses Unit4Laba52;

type

PNode = ^TNode;

TNode = record

Data: Integer;

Next: PNode;

end;

Pt = ^TList;

TArr = Unit4Laba52.TArr;

TList = Record

Data: Integer;

Next: Pt;

Prev: Pt;

End;

procedure AddToTree(Var Root: PBinaryTreeNode; Num: Integer);

procedure ClearTree(var Node: PBinaryTreeNode);

implementation

procedure AddToTree(Var Root: PBinaryTreeNode; Num: Integer);

begin

if Root = nil then

begin

// Создание нового узла с заданным значением

New(Root);

Root^.Data := Num;

Root^.Left := nil;

Root^.Right := nil;

end

else if Num < Root^.Data then

begin

// Рекурсивный вызов для левого поддерева

AddToTree(Root^.Left, Num);

end

else if Num > Root^.Data then

begin

// Рекурсивный вызов для правого поддерева

AddToTree(Root^.Right, Num);

end;

end;

procedure ClearTree(var Node: PBinaryTreeNode);

begin

if Assigned(Node) then

begin

ClearTree(Node^.Left);

ClearTree(Node^.Right;

Dispose(Node);

Node := nil;

end;

end;

end.

**DLL:**

library Library52;

uses

System.SysUtils,

System.Classes;

{$R \*.res}

Const

CORRECT\_FILE\_SYMBOLS: Set of Char = ['-', '0' .. '9', #10, #13, ' '];

type

PBinaryTreeNode = ^TBinaryTreeNode;

TBinaryTreeNode = record

Data: Integer;

Left, Right: PBinaryTreeNode;

end;

Function IsInt(Value: Char): Integer;

Var

Counter: Integer;

Begin

Counter := 0;

If not(Value in CORRECT\_FILE\_SYMBOLS) Then

Inc(Counter);

IsInt := Counter;

End;

Exports

IsInt;

Begin

End.

**Код программы на Java:**

import java.io.File;

import java.util.LinkedList;

import java.util.Queue;

import java.util.Scanner;

import java.util.concurrent.atomic.AtomicBoolean;

import java.io.FileNotFoundException;

public class Main {

static class TreeNode {

int data;

TreeNode left;

TreeNode right;

public TreeNode(int data) {

this.data = data;

this.left = null;

this.right = null;

}

public TreeNode getLeft() {

return left;

}

public TreeNode getRight() {

return right;

}

public Object getData() {

return data;

}

}

public static void main(String[] args) {

System.out.println("Это программа для построения бинарного дерева и подсчета

номеров вершин, для которых высота левого поддерева не

равна высоте правого поддерева");

TreeNode root = null;

String line;

Scanner scanner = new Scanner(System.in);

label:

while (true) {

System.out.println("Вам доступны следующие команды: добавить,

просмотреть, показать");

System.out.println("Для выхода напишите выход");

line = scanner.nextLine();

switch (line) {

case "добавить":

System.out.println("Как вы хотите добавить элементы?

(файл/консоль)");

String inputMethod = scanner.nextLine();

if (inputMethod.equalsIgnoreCase("файл")) {

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

String filePath = scanner.nextLine();

try {

File file = new File(filePath);

Scanner fileScanner = new Scanner(file);

while (fileScanner.hasNextInt()) {

int value = fileScanner.nextInt();

root = insert(root, value);

}

fileScanner.close();

} catch (FileNotFoundException e) {

System.out.println("Файл не найден.");

}

} else if (inputMethod.equalsIgnoreCase("консоль")) {

System.out.println("Введите значения (для завершения введите

'стоп'):");

while (true) {

String input = scanner.nextLine();

if (input.equals("стоп")) {

break;

}

int value = Integer.parseInt(input);

root = insert(root, value);

}

} else {

System.out.println("Некорректный метод ввода.");

}

break;

case "просмотреть":

System.out.println("Дерево:");

printBinaryTree(root, 64);

break;

case "показать":

System.out.println("Номера вершин, для которых высота левого

поддерева не равна высоте правого

поддерева:");

StringBuilder nodesWithDifferentHeights = new StringBuilder();

showUnequalHeightNodes(root, nodesWithDifferentHeights);

System.out.println(nodesWithDifferentHeights.toString());

break;

case "выход":

break label;

default:

System.out.println("такой команды нет");

break;

}

}

}

public static void showUnequalHeightNodes(TreeNode node, StringBuilder nodesWithDifferentHeights) {

if (node == null) {

return;

}

// Recursively calculate the heights of the left and right subtrees of the node

int leftHeight = getTreeHeight(node.getLeft());

int rightHeight = getTreeHeight(node.getRight());

// Check if the heights of the left and right subtrees are not equal

if (leftHeight != rightHeight) {

// Add the node's data to the list of nodes with different subtree heights

nodesWithDifferentHeights.append(node.getData()).append(", ");

}

// Recursively call showUnequalHeightNodes for the left and right subtrees

showUnequalHeightNodes(node.getLeft(), nodesWithDifferentHeights);

showUnequalHeightNodes(node.getRight(), nodesWithDifferentHeights);

}

public static int getTreeHeight(TreeNode node) {

if (node == null) {

return 0;

}

int leftHeight = getTreeHeight(node.getLeft());

int rightHeight = getTreeHeight(node.getRight());

return Math.max(leftHeight, rightHeight) + 1;

}

private static int readValue(int maxValue, AtomicBoolean isStop, Scanner scanner) {

System.out.println("Введите значение:");

String value = scanner.nextLine();

if (value.equals("стоп")) {

isStop.set(true);

return 0;

}

return Integer.parseInt(value);

}

private static TreeNode insert(TreeNode root, int value) {

if (root == null) {

return new TreeNode(value);

}

if (value < root.data) {

root.left = insert(root.left, value);

} else {

root.right = insert(root.right, value);

}

return root;

}

private static void printBinaryTree(TreeNode root, double spaces) {

if (root == null) {

return;

}

Queue<TreeNode> q = new LinkedList<>();

q.add(root);

int levelSize = 1;

while (levelSize <= Math.pow(2, maxDepth(root) - 1)) {

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

TreeNode node = q.poll();

if (node != null) {

if (i != 0) {

for (int j = 0; j < 2 \* spaces; ++j) {

System.out.print(" ");

}

} else {

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

System.out.print(" ");

}

}

if (Math.abs(node.data) < 10) {

System.out.print(" " + "(" + node.data + ")");

} else if (Math.abs(node.data) < 100) {

System.out.print(" " + "(" + node.data + ")");

}

q.add(node.left);

q.add(node.right);

} else {

q.add(null);

q.add(null);

if (i != 0) {

for (int j = 0; j < 2 \* spaces - 1; ++j) {

System.out.print(" ");

}

} else {

for (int j = 0; j < spaces - 1; ++j) {

System.out.print(" ");

}

}

System.out.print("\_\_");

}

}

System.out.println();

spaces /= 2;

levelSize \*= 2;

}

}

private static int maxDepth(TreeNode root) {

if (root == null) {

return 0;

}

int leftDepth = maxDepth(root.left);

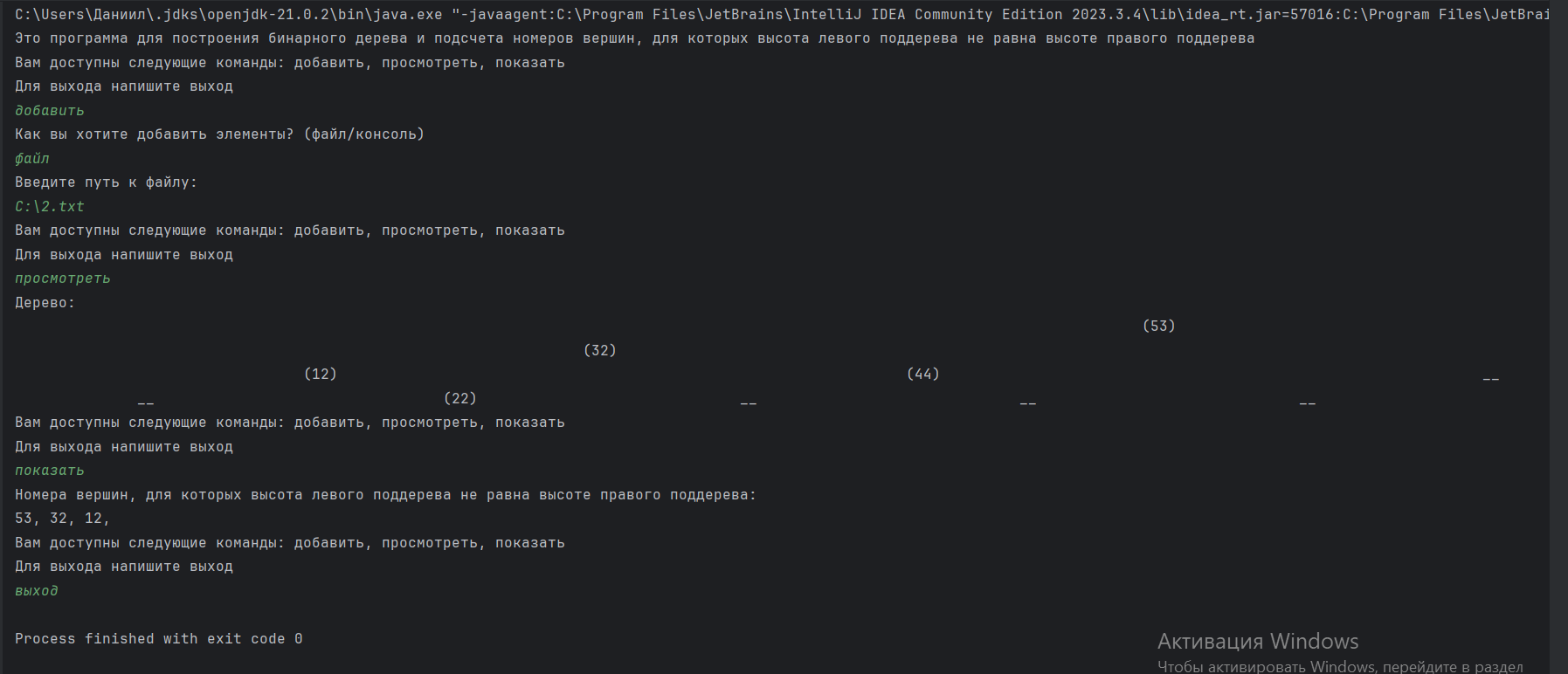
int rightDepth = maxDepth(root.right);

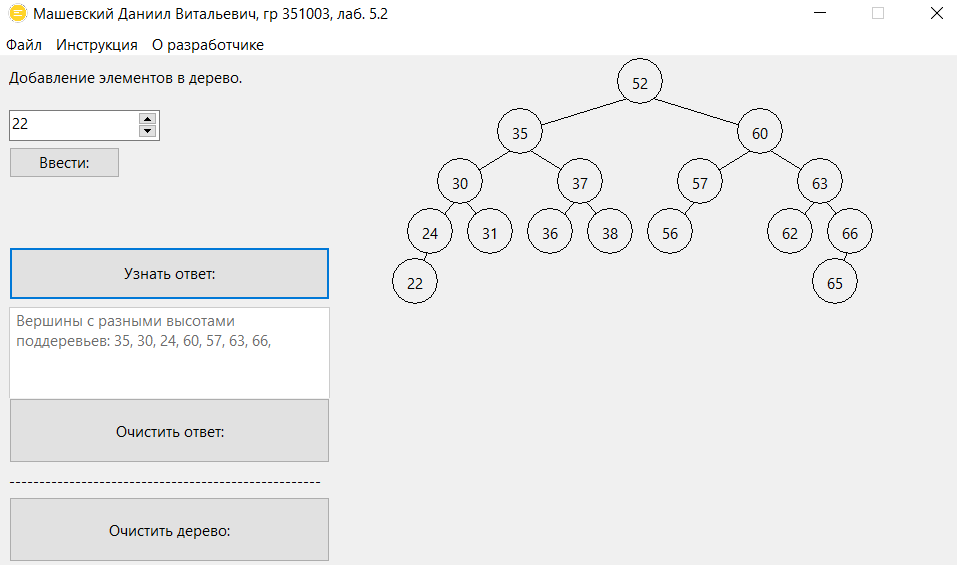
return 1 + Math.max(leftDepth, rightDepth);

}

}

**Скриншоты:**



****

**Блок-схема:**

