**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по курсовой работе**

**по дисциплине «Программирование»**

Тема: "AVL-Дерево"

|  |  |  |
| --- | --- | --- |
| Студент гр. 0324 |  | Стамбровский Т.C. |
| Преподаватель |  | Глущенко А.Г |

Санкт-Петербург

2021

**Цель работы**: изучение свойств и организации сбалансированных деревьев; получение практических навыков в работе с АВЛ-деревьями; определение преимуществ и недостатков подобных структур данных; проведение сравнительной характеристики скорости вставки, удаления и поиска элемента в АВЛ-деревьях.

**Постановка задачи**:

Программа должна поддерживать весь описанный функционал:

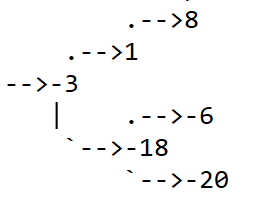
1.   Формирование АВЛ-дерева из N элементов:

a) пользователь вводит количество элементов N АВЛ-дерева, которое автоматически заполняется случайными числами (–99 до 99);

б) пользователь вводит в консоль элементы массива, N определяется автоматически по количеству введенных элементов;

Определение скорости формирования АВЛ-дерева.

2. Вывод в консоль AVL\_tree АВЛ-дерева.



3.   Определение скорости вставки, удаления и получения элемента  дерева.

4.   Определение скорости проверки на сбалансированность.

5.   Генерация заданий к практической работе по АВЛ-деревьям. Необходимо сгенерировать задания в файл output\_task в количестве вариантов, которые введет пользователь. В файл output\_key необходимо вывести короткие ответы к заданиям. В файл output\_ans необходимо вывести развернутые ответы к заданиям. Должны быть представлены задания на создание АВЛ-дерева из массива значений, удаление и вставка элементов в АВЛ-дерево.

Выполнение работы:

#include <iostream>

#include <string>

#include <time.h>

#include <fstream>

#include <chrono>

#include <windows.h>

HANDLE N **=** GetStdHandle**(**STD\_OUTPUT\_HANDLE**);**

**using** **namespace** std**;**

int maximum**(**int num1**,** int num2**)** **{**

**if** **(**num1 **>** num2**)** **{**

**return** num1**;**

**}**

**else** **{**

**return** num2**;**

**}**

**}**

struct tree **{**

int key**;**

int height **=** 1**;**

tree**\*** left **=** **NULL;**

tree**\*** right **=** **NULL;**

**};**

int heightTree**(**tree**\*** unit**)** **{**

**if** **(!**unit**)** **{**

**return** 0**;**

**}**

**else** **{**

**return** unit**->**height**;**

**}**

**}**

void streamTree**(**ofstream **\***file**,** tree**\*** unit**,** int depth **=** 0**,** int branch **=** 0**)** **{**

**if** **(!**unit**)** **{**

**return;**

**}**

streamTree**(**file**,** unit**->**right**,** depth **+** 1**,** **-**1**);**

**for** **(**int i **=** 0**;** i **!=** depth**;** i**++)** **{**

**\***file **<<** " "**;**

**}**

**if** **(**branch **==** **-**1**)** **{**

**\***file **<<** unit**->**key **<<** endl**;**

**}**

**else** **if** **(**branch **==** 1**)** **{**

**\***file **<<** unit**->**key **<<** endl**;**

**}**

**else** **{**

**\***file **<<** unit**->**key **<<** endl**;**

**}**

streamTree**(**file**,** unit**->**left**,** depth **+** 1**,** 1**);**

**}**

void printTree**(**tree**\*** unit**,** int depth **=** 0**,** int branch **=** 0**)** **{**

**if** **(!**unit**)** **{**

**return;**

**}**

printTree**(**unit**->**right**,** depth **+** 1**,** **-**1**);**

**for** **(**int i **=** 0**;** i **!=** depth**;** i**++)** **{**

cout **<<** " "**;**

**}**

**if** **(**branch **==** **-**1**)** **{**

cout **<<** unit**->**key **<<** endl**;**

**}**

**else** **if** **(**branch **==** 1**)** **{**

cout **<<** unit**->**key **<<** endl**;**

**}**

**else** **{**

cout **<<** unit**->**key **<<** endl**;**

**}**

printTree**(**unit**->**left**,** depth **+** 1**,** 1**);**

**}**

void leftTurn**(**tree**\*** unit**)** **{**

tree**\*** temp\_tree**;**

int temp\_num**;**

**if** **(**heightTree**(**unit**->**right**->**right**)** **>=** heightTree**(**unit**->**right**->**left**))** **{**

temp\_tree **=** unit**->**right**->**right**;**

temp\_num **=** unit**->**key**;**

unit**->**key **=** unit**->**right**->**key**;**

unit**->**right**->**key **=** temp\_num**;**

unit**->**right**->**right **=** unit**->**right**->**left**;**

unit**->**right**->**left **=** unit**->**left**;**

unit**->**left **=** unit**->**right**;**

unit**->**right **=** temp\_tree**;**

unit**->**left**->**height **=** max**(**heightTree**(**unit**->**left**->**left**),** heightTree**(**unit**->**left**->**right**))** **+** 1**;**

unit**->**height **=** max**(**heightTree**(**unit**->**left**),** heightTree**(**unit**->**right**))** **+** 1**;**

**}**

**else** **{**

temp\_num **=** unit**->**key**;**

unit**->**key **=** unit**->**right**->**left**->**key**;**

unit**->**right**->**left**->**key **=** temp\_num**;**

temp\_tree **=** unit**->**right**->**left**;**

unit**->**right**->**left **=** unit**->**right**->**left**->**right**;**

temp\_tree**->**right **=** temp\_tree**->**left**;**

temp\_tree**->**left **=** unit**->**left**;**

unit**->**left **=** temp\_tree**;**

unit**->**left**->**height **=** max**(**heightTree**(**unit**->**left**->**left**),** heightTree**(**unit**->**left**->**right**))** **+** 1**;**

unit**->**right**->**height **=** max**(**heightTree**(**unit**->**right**->**left**),** heightTree**(**unit**->**right**->**right**))** **+** 1**;**

unit**->**height **=** max**(**heightTree**(**unit**->**left**),** heightTree**(**unit**->**right**))** **+** 1**;**

**}**

**}**

void rightTurn**(**tree**\*** unit**)** **{**

tree**\*** temp\_tree**;**

int temp\_num**;**

**if** **(**heightTree**(**unit**->**left**->**left**)** **>=** heightTree**(**unit**->**left**->**right**))** **{**

temp\_num **=** unit**->**key**;**

unit**->**key **=** unit**->**left**->**key**;**

unit**->**left**->**key **=** temp\_num**;**

temp\_tree **=** unit**->**left**->**left**;**

unit**->**left**->**left **=** unit**->**left**->**right**;**

unit**->**left**->**right **=** unit**->**right**;**

unit**->**right **=** unit**->**left**;**

unit**->**left **=** temp\_tree**;**

unit**->**right**->**height **=** max**(**heightTree**(**unit**->**right**->**left**),** heightTree**(**unit**->**right**->**right**))** **+** 1**;**

unit**->**height **=** max**(**heightTree**(**unit**->**left**),** heightTree**(**unit**->**right**))** **+** 1**;**

**}**

**else** **{**

temp\_num **=** unit**->**key**;**

unit**->**key **=** unit**->**left**->**right**->**key**;**

unit**->**left**->**right**->**key **=** temp\_num**;**

temp\_tree **=** unit**->**left**->**right**;**

unit**->**left**->**right **=** unit**->**left**->**right**->**left**;**

temp\_tree**->**left **=** temp\_tree**->**right**;**

temp\_tree**->**right **=** unit**->**right**;**

unit**->**right **=** temp\_tree**;**

unit**->**left**->**height **=** max**(**heightTree**(**unit**->**left**->**left**),** heightTree**(**unit**->**left**->**right**))** **+** 1**;**

unit**->**right**->**height **=** max**(**heightTree**(**unit**->**right**->**left**),** heightTree**(**unit**->**right**->**right**))** **+** 1**;**

unit**->**height **=** max**(**heightTree**(**unit**->**left**),** heightTree**(**unit**->**right**))** **+** 1**;**

**}**

**}**

void balance**(**tree**\*** unit**)** **{**

unit**->**height **=** max**(**heightTree**(**unit**->**left**),** heightTree**(**unit**->**right**))** **+** 1**;**

**if** **(**heightTree**(**unit**->**left**)** **-** heightTree**(**unit**->**right**)** **<** **-**1**)** **{**

leftTurn**(**unit**);**

balance**(**unit**);**

**}**

**else** **if** **(**heightTree**(**unit**->**left**)** **-** heightTree**(**unit**->**right**)** **>** 1**){**

rightTurn**(**unit**);**

balance**(**unit**);**

**}**

**}**

tree**\*** addToTree**(**int new\_key**,** tree**\*** unit**)** **{**

**if** **(**unit**)** **{**

**if** **(**new\_key **<=** unit**->**key**)** **{**

unit**->**left **=** addToTree**(**new\_key**,** unit**->**left**);**

**}**

**else** **if(**new\_key **>** unit**->**key**){**

unit**->**right **=** addToTree**(**new\_key**,** unit**->**right**);**

**}**

balance**(**unit**);**

**return** unit**;**

**}**

**else** **{**

unit **=** **new** tree**;**

unit**->**key **=** new\_key**;**

**return** unit**;**

**}**

**}**

tree**\*** removeFromTree**(**int num**,** tree**\*** unit**)** **{**

tree**\*** temp\_tree**,** **\***temp\_tree2**;**

**if** **(!**unit**)** **{**

**return** unit**;**

**}**

**else** **if** **(**num **<** unit**->**key**)** **{**

unit**->**left **=** removeFromTree**(**num**,** unit**->**left**);**

balance**(**unit**);**

**return** unit**;**

**}**

**else** **if** **(**num **>** unit**->**key**)** **{**

unit**->**right **=** removeFromTree**(**num**,** unit**->**right**);**

balance**(**unit**);**

**return** unit**;**

**}**

**else** **{**

**if** **(**unit**->**height **==** 1**)** **{**

**delete** unit**;**

**return** **NULL;**

**}**

**else** **if(**unit**->**left **&&** unit**->**right**){**

**if** **(!**unit**->**right**->**left**)** **{**

temp\_tree **=** unit**->**right**;**

unit**->**key **=** unit**->**right**->**key**;**

unit**->**right **=** unit**->**right**->**right**;**

**delete** temp\_tree**;**

balance**(**unit**);**

**}**

**else** **{**

temp\_tree **=** unit**->**right**;**

int count **=** 0**;**

**while** **(**temp\_tree**->**left**->**left**)** **{**

temp\_tree **=** temp\_tree**->**left**;**

count**++;**

**}**

unit**->**key **=** temp\_tree**->**left**->**key**;**

temp\_tree2 **=** temp\_tree**->**left**;**

temp\_tree**->**left **=** temp\_tree**->**left**->**right**;**

**delete** temp\_tree2**;**

**while** **(**count **>** **-**1**)** **{**

temp\_tree **=** unit**->**right**;**

**for** **(**int i **=** 0**;** i **!=** count**;** i**++)** **{**

temp\_tree **=** temp\_tree**->**left**;**

**}**

balance**(**temp\_tree**);**

count**--;**

**}**

balance**(**unit**);**

**}**

**return** unit**;**

**}**

**else** **{**

**if** **(**unit**->**left**)** **{**

temp\_tree **=** unit**->**left**;**

**delete** unit**;**

**return** temp\_tree**;**

**}**

**else** **{**

temp\_tree **=** unit**->**right**;**

**delete** unit**;**

**return** temp\_tree**;**

**}**

**}**

**}**

**}**

tree**\*** removeTree**(**tree**\*** root**)** **{**

**if** **(!**root**)** **{**

**return** **NULL;**

**}**

**else** **{**

root**->**left **=** removeTree**(**root**->**left**);**

root**->**right **=** removeTree**(**root**->**right**);**

**delete** root**;**

**return** **NULL;**

**}**

**}**

tree**\*** findUnit**(**int num**,** tree**\*** unit**)** **{**

**if** **(!**unit**)** **{**

**return** **NULL;**

**}**

**else** **{**

**if** **(**unit**->**key **==** num**)** **{**

**return** unit**;**

**}**

**else** **{**

**if** **(**num **>** unit**->**key**)** **{**

**return** findUnit**(**num**,** unit**->**right**);**

**}**

**else** **{**

**return** findUnit**(**num**,** unit**->**left**);**

**}**

**}**

**}**

**}**

tree**\*** createRandTree**(**int size**)** **{**

tree**\*** root **=** **new** tree**;**

srand**(**time**(NULL));**

root**->**key **=** rand**()** **%** 100 **-** rand**()** **%** 100**;**

**for** **(**int i **=** 1**;** i **!=** size**;** i**++)** **{**

addToTree**(**rand**()** **%** 100 **-** rand**()** **%** 100**,** root**);**

**}**

**return** root**;**

**}**

tree**\*** inputTree**()** **{**

tree**\*** root **=** **NULL;**

int num**;**

SetConsoleTextAttribute**(**N**,** FOREGROUND\_GREEN **|** FOREGROUND\_BLUE **|** FOREGROUND\_INTENSITY**);**

cout **<<** "Leave spaces between elements, press any letter to stop: "**;**

SetConsoleTextAttribute**(**N**,** FOREGROUND\_RED **|** FOREGROUND\_BLUE **|** FOREGROUND\_GREEN **|** FOREGROUND\_INTENSITY**);**

cin **>>** num**;**

**while** **(!**cin**.**fail**())** **{**

root **=** addToTree**(**num**,** root**);**

cin **>>** num**;**

**}**

cin**.**ignore**(**1024**);**

cin**.**clear**();**

**return** root**;**

**}**

bool areBalance**(**tree**\*** root**)** **{**

**if** **(!**root**)** **{**

**return** **true;**

**}**

**else** **{**

**if** **(**areBalance**(**root**->**left**)** **&&** areBalance**(**root**->**right**))** **{**

**if** **(-**2 **>** heightTree**(**root**->**left**)** **-** heightTree**(**root**->**right**)** **<** 2**)** **{**

**return** **true;**

**}**

**else** **{**

**return** **false;**

**}**

**}**

**else** **{**

**return** **false;**

**}**

**}**

**}**

void createTest**()** **{**

srand**(**time**(NULL));**

int quantity**,** size**,** num**;**

tree**\*** root**;**

cout **<<** endl**;**

cout **<<** "Quantity of the test options: "**;**

cin **>>** quantity**;**

ofstream task**;**

ofstream key**;**

ofstream ans**;**

task**.**open**(**"task.txt"**);**

key**.**open**(**"key.txt"**);**

ans**.**open**(**"ans.txt"**);**

**if** **(!**task **||** **!**key **||** **!**ans**)** **{**

cout **<<** "Error!" **<<** endl**;**

**return;**

**}**

**for** **(**int i **=** 0**;** i **!=** quantity**;** i**++)** **{**

root **=** **NULL;**

task **<<** "Option: " **<<** i **+** 1 **<<** endl**;**

task **<<** "Create an AVL tree from: "**;**

ans **<<** "Option: " **<<** i **+** 1 **<<** endl**;**

ans **<<** "AVL tree creation" **<<** endl**;**

ans **<<** "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" **<<** endl**;**

key **<<** "Option " **<<** i **+** 1 **<<** endl**;**

size **=** rand**()** **%** 8 **+** 7**;**

**for** **(**int j **=** 0**;** j **!=** size**;** j**++)** **{**

num **=** rand**()** **%** 10**;**

task **<<** num **<<** " "**;**

root **=** addToTree**(**num**,** root**);**

streamTree**(&**ans**,** root**);**

ans **<<** "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" **<<** endl**;**

**}**

key **<<** "AVL tree: " **<<** endl**;**

streamTree**(&**key**,** root**);**

num **=** rand**()** **%** 10**;**

task **<<** "Add element " **<<** num **<<** endl**;**

root **=** addToTree**(**num**,** root**);**

ans **<<** "Adding element " **<<** num **<<** endl**;**

streamTree**(&**ans**,** root**);**

ans **<<** "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" **<<** endl**;**

key **<<** "Tree after adding element " **<<** num **<<** endl**;**

streamTree**(&**key**,** root**);**

num **=** rand**()** **%** 10**;**

task **<<** "Delete element " **<<** num **<<** endl**;**

cout **<<** endl**;**

**if** **(**findUnit**(**num**,** root**))** **{**

ans **<<** "Element was deleted" **<<** endl**;**

root **=** removeFromTree**(**num**,** root**);**

streamTree**(&**ans**,** root**);**

**}**

**else** **{**

ans **<<** "Element is not present in the tree" **<<** endl**;**

**}**

key **<<** "Tree after deleting element " **<<** num **<<** endl**;**

streamTree**(&**key**,** root**);**

removeTree**(**root**);**

**}**

task**.**close**();**

ans**.**close**();**

key**.**close**();**

**}**

int main**()** **{**

setlocale**(NULL,** ""**);**

tree**\*** root **=** **NULL;**

int choise **=** 1**;**

int num**;**

string trash**;**

int answer **=** 0**;**

**while** **(**answer **!=** 6**)** **{**

SetConsoleTextAttribute**(**N**,** FOREGROUND\_GREEN **|** FOREGROUND\_INTENSITY**);**

cout **<<** "Menu" **<<** endl**;**

SetConsoleTextAttribute**(**N**,** FOREGROUND\_RED **|** FOREGROUND\_BLUE **|** FOREGROUND\_GREEN **|** FOREGROUND\_INTENSITY**);**

cout **<<** "1. Create an AVL tree" **<<** endl**;**

cout **<<** "2. Print the AVL tree" **<<** endl**;**

cout **<<** "3. Delete/Print an element from the AVL tree" **<<** endl**;**

cout **<<** "4. Check if the AVL tree is balanced" **<<** endl**;**

cout **<<** "5. Generate a test" **<<** endl**;**

SetConsoleTextAttribute**(**N**,** FOREGROUND\_GREEN **|** FOREGROUND\_INTENSITY**);**

cout **<<** "6. Exit" **<<** endl**;**

SetConsoleTextAttribute**(**N**,** FOREGROUND\_RED **|** FOREGROUND\_BLUE **|** FOREGROUND\_GREEN **|** FOREGROUND\_INTENSITY**);**

cin **>>** answer**;**

system**(**"CLS"**);**

**switch** **(**answer**)**

**{**

**case** **(**1**):** **{**

int key **=** 0**;**

**while** **(**key **!=** 3**)** **{**

**if** **(**root**)** **{**

root **=** removeTree**(**root**);**

**}**

cout **<<** endl**;**

SetConsoleTextAttribute**(**N**,** FOREGROUND\_GREEN **|** FOREGROUND\_INTENSITY**);**

cout **<<** "Select input method: " **<<** endl**;**

SetConsoleTextAttribute**(**N**,** FOREGROUND\_RED **|** FOREGROUND\_BLUE **|** FOREGROUND\_GREEN **|** FOREGROUND\_INTENSITY**);**

cout **<<** "1. Create a tree from random numbers (-99 - 99)" **<<** endl**;**

cout **<<** "2. Create a tree from numbers entered by user" **<<** endl**;**

SetConsoleTextAttribute**(**N**,** FOREGROUND\_GREEN **|** FOREGROUND\_INTENSITY**);**

cout **<<** "3. Exit: " **<<** endl**;**

SetConsoleTextAttribute**(**N**,** FOREGROUND\_RED **|** FOREGROUND\_BLUE **|** FOREGROUND\_GREEN **|** FOREGROUND\_INTENSITY**);**

cin **>>** key**;**

system**(**"CLS"**);**

**switch** **(**key**)** **{**

**case** **(**1**):** **{**

cout **<<** endl**;**

cout **<<** "Amount of elements in a tree: "**;**

cin **>>** num**;**

auto start **=** chrono**::**high\_resolution\_clock**::**now**();**

root **=** createRandTree**(**num**);**

auto end **=** chrono**::**high\_resolution\_clock**::**now**();**

chrono**::**duration**<**float**>** duration **=** end **-** start**;**

SetConsoleTextAttribute**(**N**,** FOREGROUND\_BLUE **|** FOREGROUND\_INTENSITY**);**

cout **<<** "Time: "**;**

SetConsoleTextAttribute**(**N**,** FOREGROUND\_RED **|** FOREGROUND\_BLUE **|** FOREGROUND\_GREEN **|** FOREGROUND\_INTENSITY**);**

cout**.**setf**(**ios**::**fixed**,** ios**::**floatfield**);**

cout**.**precision**(**7**);**

cout **<<** duration**.**count**()** **<<** endl**;**

cout **<<** endl**;**

key **=** 3**;**

**break;**

**}**

**case** **(**2**):** **{**

auto start **=** chrono**::**high\_resolution\_clock**::**now**();**

root **=** inputTree**();**

auto end **=** chrono**::**high\_resolution\_clock**::**now**();**

chrono**::**duration**<**float**>** duration **=** end **-** start**;**

getline**(**cin**,** trash**);**

SetConsoleTextAttribute**(**N**,** FOREGROUND\_BLUE **|** FOREGROUND\_INTENSITY**);**

cout **<<** "Time: "**;**

SetConsoleTextAttribute**(**N**,** FOREGROUND\_RED **|** FOREGROUND\_BLUE **|** FOREGROUND\_GREEN **|** FOREGROUND\_INTENSITY**);**

cout**.**setf**(**ios**::**fixed**,** ios**::**floatfield**);**

cout**.**precision**(**7**);**

cout **<<** duration**.**count**()** **<<** endl**;**

cout **<<** endl**;**

key **=** 3**;**

**break;**

**}**

**}**

**}**

**break;**

**}**

**case** **(**2**):** **{**

printTree**(**root**);**

**break;**

**}**

**case** **(**3**):** **{**

int key **=** 0**;**

**while** **(**key **!=** 3**)** **{**

cout **<<** endl**;**

SetConsoleTextAttribute**(**N**,** FOREGROUND\_GREEN **|** FOREGROUND\_INTENSITY**);**

cout **<<** "Select the operation: " **<<** endl**;**

SetConsoleTextAttribute**(**N**,** FOREGROUND\_RED **|** FOREGROUND\_BLUE **|** FOREGROUND\_GREEN **|** FOREGROUND\_INTENSITY**);**

cout **<<** "1. Search an element" **<<** endl**;**

cout **<<** "2. Delete an element" **<<** endl**;**

cin **>>** key**;**

**switch** **(**key**)** **{**

**case** **(**1**):** **{**

cout **<<** endl**;**

cout **<<** "Element: "**;**

cin **>>** num**;**

auto start **=** chrono**::**high\_resolution\_clock**::**now**();**

**if** **(**findUnit**(**num**,** root**))** **{**

cout **<<** endl**;**

cout **<<** "Element was found!" **<<** endl**;**

**}**

**else** **{**

cout **<<** endl**;**

cout **<<** "Element is not present in the tree" **<<** endl**;**

**}**

auto end **=** chrono**::**high\_resolution\_clock**::**now**();**

chrono**::**duration**<**float**>** duration **=** end **-** start**;**

SetConsoleTextAttribute**(**N**,** FOREGROUND\_BLUE **|** FOREGROUND\_INTENSITY**);**

cout **<<** "Time: "**;**

SetConsoleTextAttribute**(**N**,** FOREGROUND\_RED **|** FOREGROUND\_BLUE **|** FOREGROUND\_GREEN **|** FOREGROUND\_INTENSITY**);**

cout**.**setf**(**ios**::**fixed**,** ios**::**floatfield**);**

cout**.**precision**(**7**);**

cout **<<** duration**.**count**()** **<<** endl**;**

cout **<<** endl**;**

key **=** 3**;**

**break;**

**}**

**case** **(**2**):** **{**

cout **<<** '\n' **<<** "Element: "**;**

cin **>>** num**;**

auto start **=** chrono**::**high\_resolution\_clock**::**now**();**

root **=** removeFromTree**(**num**,** root**);**

auto end **=** chrono**::**high\_resolution\_clock**::**now**();**

chrono**::**duration**<**float**>** duration **=** end **-** start**;**

cout **<<** "Element was removed!" **<<** endl**;**

SetConsoleTextAttribute**(**N**,** FOREGROUND\_BLUE **|** FOREGROUND\_INTENSITY**);**

cout **<<** "Time: "**;**

SetConsoleTextAttribute**(**N**,** FOREGROUND\_RED **|** FOREGROUND\_BLUE **|** FOREGROUND\_GREEN **|** FOREGROUND\_INTENSITY**);**

cout**.**setf**(**ios**::**fixed**,** ios**::**floatfield**);**

cout**.**precision**(**7**);**

cout **<<** duration**.**count**()** **<<** endl**;**

cout **<<** endl**;**

key **=** 3**;**

**break;**

**}**

**}**

**}**

**break;**

**}**

**case** **(**4**):** **{**

auto start **=** chrono**::**high\_resolution\_clock**::**now**();**

**if** **(**areBalance**(**root**))** **{**

cout **<<** endl**;**

cout **<<** "The tree is balanced" **<<** endl**;**

**}**

**else** **{**

cout **<<** endl**;**

cout **<<** "The tree is not balanced" **<<** endl**;**

**}**

auto end **=** chrono**::**high\_resolution\_clock**::**now**();**

chrono**::**duration**<**float**>** duration **=** end **-** start**;**

SetConsoleTextAttribute**(**N**,** FOREGROUND\_BLUE **|** FOREGROUND\_INTENSITY**);**

cout **<<** "Time: "**;**

SetConsoleTextAttribute**(**N**,** FOREGROUND\_RED **|** FOREGROUND\_BLUE **|** FOREGROUND\_GREEN **|** FOREGROUND\_INTENSITY**);**

cout**.**setf**(**ios**::**fixed**,** ios**::**floatfield**);**

cout**.**precision**(**7**);**

cout **<<** duration**.**count**()** **<<** endl**;**

cout **<<** endl**;**

**break;**

**}**

**case** **(**5**):** **{**

createTest**();**

**break;**

**}**

**break;**

**}**

**}**

**return** 0**;**

**}**