МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Практическое занятие № 20.

НЕЛИНЕЙНЫЕ СТРУКТУРЫ ДАННЫХ

**Выполнил** студент:

Герасимов Константин Сергеевич,

МОиАИС, 1 курс.

Тюмень-2023

**Работа в аудитории**

**1. Решение задачи 1**

***1.1 Постановка задачи***

A picture containing text, screenshot, font, number

Description automatically generated

***1.2 Текст программы***

**main.cpp:**

#include <iostream>

#include <cmath>

#include "tree.h"

#include "menu.h"

int main() {

Node\* root = nullptr;

int choice, value;

bool exitProgram = false;

int count;

long long product;

double sum = 0.0;

while (!exitProgram) {

printMenu();

std::cin >> choice;

switch (choice) {

case 1:

std::cout << "Enter a number to add to the tree: ";

std::cin >> value;

root = insertNode(root, value);

std::cout << "Number added successfully!" << std::endl;

break;

case 2:

std::cout << "Enter a number to remove from the tree: ";

std::cin >> value;

root = deleteNode(root, value);

std::cout << "Number removed successfully!" << std::endl;

break;

case 3:

freeMemory(root);

root = nullptr;

std::cout << "All numbers are removed from the tree." << std::endl;

break;

case 4:

count = 0;

product = 1;

sum = findPositiveGeometricMean(root, count, product);

if (count > 0) {

double geometricMean = std::pow(product, 1.0 / count);

std::cout << "Geometric mean of positive numbers: " << geometricMean << std::endl;

}

else {

std::cout << "There are no positive numbers in the tree." << std::endl;

}

break;

case 5:

std::cout << "Numbers in tree: ";

printTree(root);

std::cout << std::endl;

break;

case 6:

exitProgram = true;

break;

default:

std::cout << "Incorrect operation selection. Try again." << std::endl;

break;

}

std::cout << std::endl;

}

// Освобождение памяти

freeMemory(root);

return 0;

}

**menu.cpp:**

#include "menu.h"

#include <iostream>

void printMenu() {

std::cout << "===== Menu =====" << std::endl;

std::cout << "1. Add number to the tree" << std::endl;

std::cout << "2. Remove number from the tree" << std::endl;

std::cout << "3. Remove all number from the tree" << std::endl;

std::cout << "4. Find the geometric mean of positive numbers" << std::endl;

std::cout << "5. View all numbers in the tree" << std::endl;

std::cout << "6. Exit from the program" << std::endl;

std::cout << "================" << std::endl;

std::cout << "Select an operation: ";

}

**menu.h:**

#ifndef MENU\_H

#define MENU\_H

void printMenu();

#endif

**tree.cpp:**

#include <iostream>

#include <cmath>

#include "tree.h"

Node\* insertNode(Node\* root, int value) {

if (root == nullptr) {

return new Node(value);

}

if (value < root->data) {

root->left = insertNode(root->left, value);

}

else if (value > root->data) {

root->right = insertNode(root->right, value);

}

return root;

}

Node\* deleteNode(Node\* root, int value) {

if (root == nullptr) {

return root;

}

if (value < root->data) {

root->left = deleteNode(root->left, value);

}

else if (value > root->data) {

root->right = deleteNode(root->right, value);

}

else {

if (root->left == nullptr) {

Node\* temp = root->right;

delete root;

return temp;

}

else if (root->right == nullptr) {

Node\* temp = root->left;

delete root;

return temp;

}

Node\* successor = root->right;

while (successor->left != nullptr) {

successor = successor->left;

}

root->data = successor->data;

root->right = deleteNode(root->right, successor->data);

}

return root;

}

double findPositiveGeometricMean(Node\* root, int& count, long long& product) {

if (root == nullptr) {

return 0.0;

}

if (root->data > 0) {

count++;

product \*= root->data;

}

double leftMean = findPositiveGeometricMean(root->left, count, product);

double rightMean = findPositiveGeometricMean(root->right, count, product);

return (leftMean + rightMean + root->data);

}

void printTree(Node\* root) {

if (root == nullptr) {

return;

}

printTree(root->left);

std::cout << root->data << " ";

printTree(root->right);

}

void freeMemory(Node\* root) {

if (root == nullptr) {

return;

}

freeMemory(root->left);

freeMemory(root->right);

delete root;

}

**tree.h:**

#ifndef TREE\_H

#define TREE\_H

struct Node {

int data;

Node\* left;

Node\* right;

Node(int value) : data(value), left(nullptr), right(nullptr) {}

};

Node\* insertNode(Node\* root, int value);

Node\* deleteNode(Node\* root, int value);

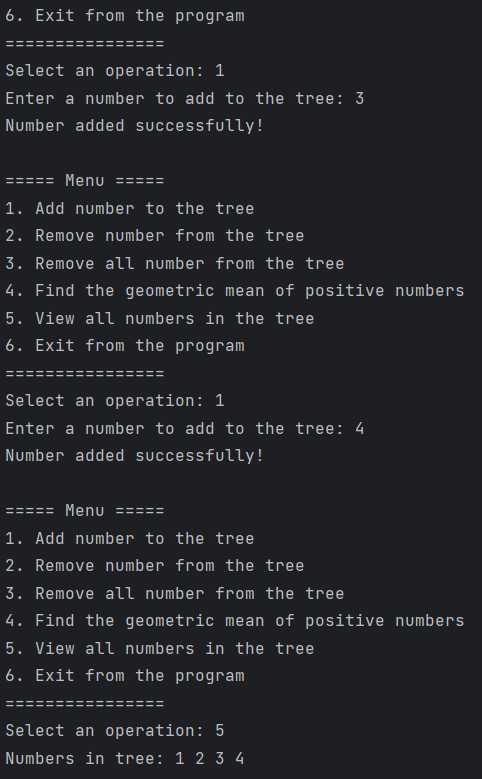
double findPositiveGeometricMean(Node\* root, int& count, long long& product);

void printTree(Node\* root);

void freeMemory(Node\* root);

#endif

***1.3 Результат тестирования программы***

A screenshot of a computer code

Description automatically generated with low confidence

A screenshot of a computer program

Description automatically generated with medium confidence

A screenshot of a computer program

Description automatically generated