Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №4

по курсу «Логика и основы алгоритмизации

в инженерных задачах»

на тему «Бинарное дерево поиска»

Выполнили:

студенты группы 22ВВВ1

Дунюшкин В.А.

Ивлюшин М.Д.

Проверили:

к.т.н, доцент Юрова О.В.

к.э.н, доцент Акифьев И.В.

Пенза 2023

**Цель работы:**

Цель данной лабораторной работы заключается в изучении и практическом применении бинарных деревьев в программировании

**Лабораторное задание:**

1. Реализовать алгоритм поиска вводимого с клавиатуры значения в уже созданном дереве.

2. Реализовать функцию подсчёта числа вхождений заданного элемента в дерево.

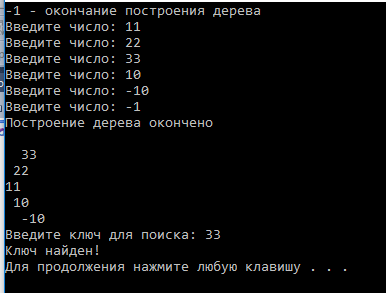
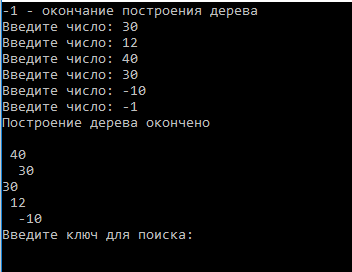
3. \* Изменить функцию добавления элементов для исключения добавления одинаковых символов.

4. \* Оценить сложность процедуры поиска по значению в бинарном дереве.

**Ход работы:**

Объявляем структуру, которая представляет узел дерева. Создаем функции для построения, создания, вывода дерева на экран и поиска искомого значения. В главной функции вызываем заданные функции для заполнения, вывода дерева, поиска искомого значения и вывода количества вхождений определенного значения.

**Результат работы программы:**



**Вывод**

В ходе лабораторной работы нами было изучено бинарное дерево поиска, которое является структурой данных, используемой для хранения и поиска элементов. Реализовали такие функции как: построение, создание, вывод дерева на экран и поиск искомого значения. Оценили сложность кода, она равна O(log(n)).

**Листинг:**

#include "stdafx.h"

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* left;

Node\* right;

};

Node\* root = nullptr;

Node\* CreateTree(Node\* root, int data) {

if (root == nullptr) {

Node\* newNode = new Node;

newNode->data = data;

newNode->left = newNode->right = nullptr;

return newNode;

}

if (data < root->data) {

root->left = CreateTree(root->left, data);

}

else {

root->right = CreateTree(root->right, data);

}

return root;

}

void PrintTree(Node\* r, int l) {

if (r == nullptr) {

return;

}

PrintTree(r->right, l + 1);

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

cout << "\t";

}

cout << r->data << endl;

PrintTree(r->left, l + 1);

}

Node\* Search(Node\* root, int key) {

if (root == nullptr || root->data == key)

return root;

if (key < root->data)

return Search(root->left, key);

else

return Search(root->right, key);

}

int CountOccurrences(Node\* root, int data) {

if (root == nullptr)

return 0;

if (root->data == data)

return 1 + CountOccurrences(root->left, data) + CountOccurrences(root->right, data);

return CountOccurrences(root->left, data) + CountOccurrences(root->right, data);

}

int main() {

setlocale(LC\_ALL, "Russian");

int vubor;

cout <<"Выбирите задание"<<endl;

cout <<"1"<<endl;

cout <<"2"<<endl;

std::cin >> vubor;

system("cls");

switch (vubor)

{

case 1:

{

int D, start = 1;

root = nullptr;

cout << "-1 - окончание построения дерева" << endl;

while (start) {

cout << "Введите число: ";

cin >> D;

if (D == -1) {

cout << "Построение дерева окончено" << endl << endl;

start = 0;

}

else

root = CreateTree(root, D);

}

PrintTree(root, 0);

int key;

cout << "Введите ключ для поиска: ";

cin >> key;

Node\* found = Search(root, key);

if (found == nullptr) {

cout << "Ключ к сожалению не найден" << endl;

}

else {

cout << "Ключ найден!" << endl;

}

int searchKey;

cout << "Введите ключ для поиска количества вхождений: ";

cin >> searchKey;

int count = CountOccurrences(root, searchKey);

cout << "Количество вхождений элемента " << searchKey << " в дереве: " << count << endl;

break;

}

case 2:

{

int D, start = 1;

root = nullptr;

cout << "-1 - окончание построения дерева" << endl;

while (start) {

cout << "Введите число: ";

cin >> D;

Node\* found = Search(root, D);

if (D == -1) {

cout << "Построение дерева окончено" << endl << endl;

start = 0;

}

else

{

if (found == nullptr)

{

root = CreateTree(root, D);

}

}

}

PrintTree(root, 0);

int key;

cout << "Введите ключ для поиска: ";

cin >> key;

Node\* found = Search(root, key);

if (found == nullptr) {

cout << "Ключ к сожалению не найден" << endl;

}

else {

cout << "Ключ найден!" << endl;

}

break;

}

}

system("pause");

}