Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная кафедра»

**ОТЧЕТ**

по лабораторной работе №4

по курсу «Основы алгоритмизации в инженерных задачах»

на тему «Бинарное дерево поиска»

Выполнили:

студент группы 22ВВП1

Воробьева М.М.

Приняли:

Акифьев И.В.

Митрохин М.А.

Пенза 2023

**Цель работы:** научиться работать с такой структурой данных, как бинарное дерево.

**Задания лабораторной работы:**

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.

**Листинг кода:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

// Структура для узла дерева

struct Node {

int data; // Значение узла

Node\* left; // Указатель на левого потомка

Node\* right; // Указатель на правого потомка

// Конструктор для узла, инициализирующий данные и указатели на NULL

Node(int val) : data(val), left(NULL), right(NULL) {}

};

// Функция для добавления узла в дерево

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

if (root == NULL) {

return new Node(data);

}

//if (data == root->data) {

// printf("Данное число в дереве присутствует, введите другое\n");

// return root; // Если элемент уже существует в дереве, не вставляем его повторно

//}

if (data < root->data) {

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

}

else {

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

}

return root;

}

// Функция для поиска значения в дереве

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

if (root == NULL or root->data == key) {

return root;

}

if (root->data < key) {

return Poisk(root->right, key);

}

return Poisk(root->left, key);

}

// Функция для вывода дерева на экран

void PrintT(Node\* root, int level) {

if (root == NULL) {

return;

}

PrintT(root->right, level + 1);

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

printf(" ");

}

printf("%d\n", root->data);

PrintT(root->left, level + 1);

}

// Функция для подсчета вхождений элемента в дерево

int Count(Node\* root, int k) {

if (root == NULL) {

return 0; // Если узел пустой, возвращаем 0

}

int count = 0;

if (root->data == k) {

count++; // Если узел содержит искомое значение, увеличиваем счетчик

}

count += Count(root->left, k); // Рекурсивно ищем в левом поддереве

count += Count(root->right, k); // Рекурсивно ищем в правом поддереве

return count;

}

int main() {

setlocale(LC\_ALL, "RUS");

Node\* root = NULL;

int data;

int start = 1;

int stk = 6;

int i=0;

for ( i; i < stk; i++) {

printf("Введите число: ");

scanf("%d", &data);

root = NewYZ(root, data);

}

printf("Дерево:\n");

PrintT(root, 0);

printf("Введите значение для поиска: ");

scanf("%d", &data);

Node\* result = Poisk(root, data);

if (result != NULL) {

printf("Значение найдено: %d\n", result->data);

}

else {

printf("Значение не найдено\n");

}

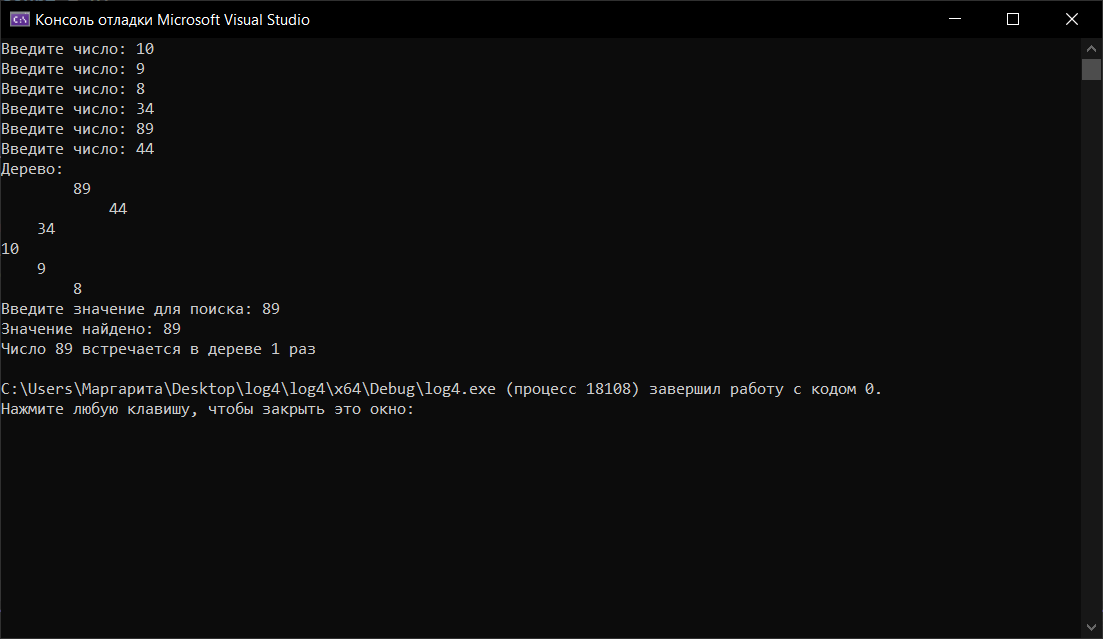
int occurrences = Count(root, data);

printf("Число %d встречается в дереве %d раз\n", data, occurrences);

return 0;

}

Результат работы:



Сложность процедуры поиска по значению в бинарном дереве = log(n), так как в самой процедуре поиска используются сравнения данных, находящихся в разных элементах структуры данных.

**Вывод:** В ходе выполнения лабораторной работы были освоены навыки создания и работы с такой структурой данных, как бинарное дерево.