**Министерство науки и высшего образования РФ**

**Пензенский государственный университет**

**Кафедра "Вычислительная техника"**

**Отчёт**

по лабораторной работе № 4

по курсу «Логика и основы алгоритмизации инженерных задач»

на тему: “Бинарное дерево поиска ”

Выполнили студ. группы 23ВВВ1:

Галкин А.П.\_\_\_\_

Швырёв А.И.\_\_\_\_

Панин А.С.\_\_\_\_

Приняли работу доценты:

к.т.н. Митрохин М.А.

к.т.н. Юрова О.В.

Пенза 2024

**Цель работы:** изучить бинарное дерево и научиться работать с ним.

**Задание:**

1. Реализовать алгоритм поиска вводимого с клавиатуры значения в уже созданном дереве.
2. Реализовать функцию подсчёта числа вхождений заданного элемента в дерево.

**Ход работы**

**Код:**

#define \_CRT\_SECURE\_NO\_WARNINGS\_

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

struct Node\* root;

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

// Если дерево пустое, создаем новый узел

if (root == NULL) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

if (newNode == NULL) {

printf("Ошибка выделения памяти");

exit(0);

}

newNode->data = data;

newNode->left = NULL;

newNode->right = NULL;

return newNode;

}

// Если дерево не пустое, рекурсивно добавляем узел

if (data < root->data) {

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

}

else {

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

}

return root;

}

void print\_tree(struct Node\* r, int l) {

if (r == NULL) {

return;

}

print\_tree(r->right, l + 1);

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

printf(" ");

}

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

print\_tree(r->left, l + 1);

}

struct Node\* SearchTree(struct Node\* r, int data) {

if (r == NULL || r->data == data) {

return r;

}

if (data < r->data) {

return SearchTree(r->left, data);

}

else {

return SearchTree(r->right, data);

}

}

int CountOccurrences(struct Node\* r, int data) {

if (r == NULL) {

return 0; // Если узел пустой, возвращаем 0

}

int count = (r->data == data) ? 1 : 0; // Если узел равен искомому значению, увеличиваем счетчик

// Рекурсивно считаем вхождения в левом и правом поддеревьях

count += CountOccurrences(r->left, data);

count += CountOccurrences(r->right, data);

return count; // Возвращаем общее количество вхождений

}

int main() {

setlocale(LC\_ALL, "");

int D, start = 1;

root = NULL;

printf("-1 - окончание построения дерева\n");

while (start) {

printf("Введите число: ");

scanf\_s("%d", &D);

if (D == -1) {

printf("Построение дерева окончено\n\n");

start = 0;

}

else {

root = CreateTree(root, D);

}

}

print\_tree(root, 0);

printf("\nВведите число для поиска: ");

scanf\_s("%d", &D);

struct Node\* result = SearchTree(root, D);

if (result != NULL) {

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

}

else {

printf("Значение %d не найдено в дереве.\n", D);

}

// Подсчёт вхождений заданного числа

int occurrences = CountOccurrences(root, D);

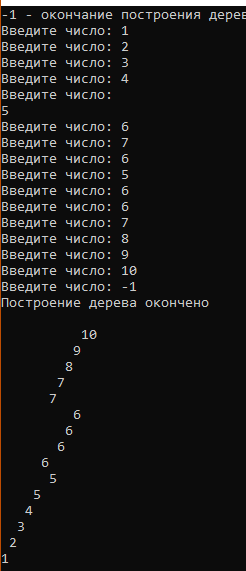
printf("Количество вхождений значения %d в дереве: %d\n", D, occurrences);

return 0;

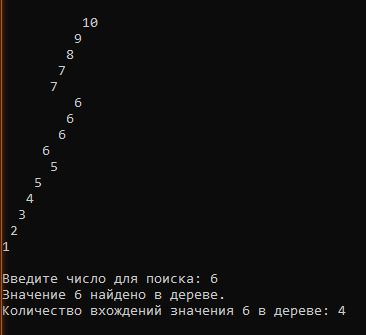
}

**Схема программы:**

**Трассеровка:**

**1) Ввели значения и построили дерево**

**2)Нашли нужное число в дереве**

****

**Вывод:** изучилибинарное дерево и научились работать с ним; реализовали алгоритм поиска вводимого с клавиатуры значения; реализовать функцию подсчёта числа вхождений заданного элемента.