Министерство 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 23ВВВ2

Герасимов В.Р.

Мадамкин В. М.

Приняли:

Юрова О. В.

Митрохин М. А.

Пенза 2024

**Лабораторное задание.**

* Реализовать алгоритм поиска вводимого с клавиатуры значения в уже созданном дереве.
* Реализовать функцию подсчёта числа вхождений заданного элемента в дерево.
* Изменить функцию добавления элементов для исключения добавления одинаковых символов.
* Оценить сложность процедуры поиска по значению в бинарном дереве.

**Ход работы**

Задание 1:

Реализовали алгоритм поиска вводимого с клавиатуры значения в уже созданном дереве, код программы находится в Приложении А/lab4.1.

Результат работы программы изображен на рисунке №1:

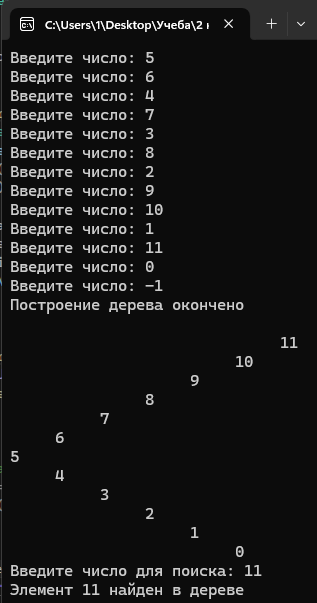


Рисунок №1

Задание 2:

Реализовали функцию подсчёта числа вхождений заданного элемента в дерево, код программы находится в Приложении А/lab4.1.

Результат работы программы изображен на рисунке №2:

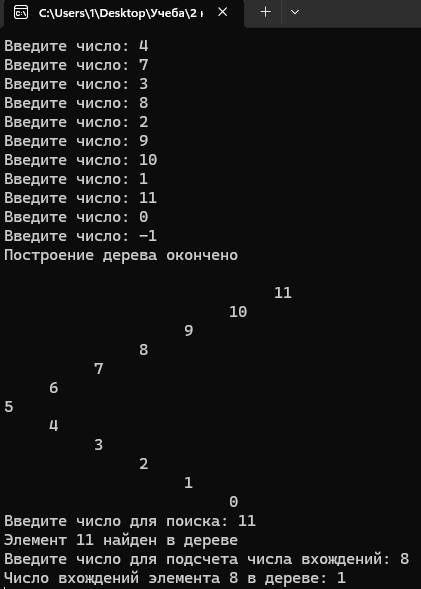


Рисунок №2

Задание 3:

Изменить функцию добавления элементов для исключения добавления одинаковых символов, код программы находится в Приложении А/lab4.1.

Результат работы программы изображен на рисунке №3:

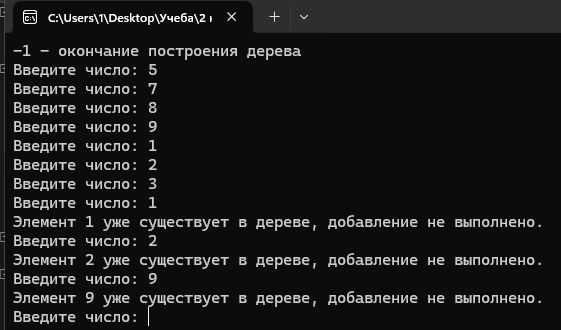


Рисунок №3

Задание 4:

Оценка сложности поиска в бинарном дереве

Лучший случай (O(1)): Если элемент находится в корне дерева.

Средний и худший случай (O(h)), где h — высота дерева. В среднем бинарное дерево поиска сбалансировано, и его высота составляет O(log n), где n — количество узлов. В худшем случае, когда дерево полностью вырождено, сложность поиска будет O(n)

**Вывод:** Реализовали алгоритм поиска вводимого с клавиатуры значения в уже созданном дереве, функцию подсчёта числа вхождений заданного элемента в дерево, изменили функцию добавления элементов для исключения добавления одинаковых символов, оценили сложность процедуры поиска по значению в бинарном дереве.

**Приложение A**

**Lab4.1**

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

struct Node\* root;

// Функция для создания нового узла дерева

struct Node\* CreateNode(int data) {

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

if (newNode == NULL) {

printf("Ошибка выделения памяти\n");

exit(0);

}

newNode->data = data;

newNode->left = NULL;

newNode->right = NULL;

return newNode;

}

// Функция для добавления элемента в дерево

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

if (r == NULL) {

r = CreateNode(data);

return r;

}

// Проверка на дублирование значения

if (data == r->data) {

printf("Элемент %d уже существует в дереве, добавление не выполнено.\n", data);

return r;

}

// Добавление элемента в левое или правое поддерево

if (data < r->data) {

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

}

else {

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

}

return r;

}

// Функция для поиска элемента в дереве

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

if (r == NULL) {

return 0;

}

if (data == r->data) {

return 1;

}

else if (data < r->data) {

return Search(r->left, data);

}

else {

return Search(r->right, data);

}

}

// Функция для подсчета числа вхождений элемента в дереве

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

if (r == NULL) {

return 0;

}

int count = 0;

if (data == r->data) {

count = 1;

}

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

}

// Функция для вывода дерева на экран

void PrintTree(struct Node\* r, int space) {

if (r == NULL) {

return;

}

space += 5;

PrintTree(r->right, space);

for (int i = 5; i < space; i++) {

printf(" ");

}

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

PrintTree(r->left, space);

}

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, root, D);

}

}

PrintTree(root, 0);

// Поиск элемента в дереве

printf("Введите число для поиска: ");

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

if (Search(root, D)) {

printf("Элемент %d найден в дереве\n", D);

}

else {

printf("Элемент %d не найден в дереве\n", D);

}

// Подсчет числа вхождений элемента в дереве

printf("Введите число для подсчета числа вхождений: ");

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

int count = CountOccurrences(root, D);

printf("Число вхождений элемента %d в дереве: %d\n", D, count);

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

return 0;

}