Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра вычислительная техника

**ОТЧËТ**  
по лабораторной работе №4  
по дисциплине: «Логика и основы алгоритмизации в инженерных задачах»  
на тему «Бинарное дерево поиска»

Выполнили студенты группы 22ВВВ2:  
Демин М. С.

Амиров И. Р.

Приняли:  
Акифьев И. В.

Митрохин М. А.

Пенза 2023

**Название**

Бинарное дерево поиска

**Цель работы**

Разработать алгоритм бинарного дерева поиска. Научиться работать с данным типом данных

**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.

**Листинг**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <locale.h>

#include<windows.h>

typedef struct Node {

struct Node\* left;

struct Node\* right;

int data;

} Node;

struct Node\* root;

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

if (r == NULL) {

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

if (r == NULL) {

printf("Ошибка выделения памяти");

exit(0);

}

r->left = NULL;

r->right = NULL;

r->data = data;

if (rooot == NULL)

return r;

if (data > rooot->data)

rooot->left = r;

else

rooot->right = r;

return r;

}

if (data > r->data)

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

else

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

return rooot;

}

void print\_tree(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);

}

void search\_tree(Node\* r, int l, int query) {

if (r == NULL)

return;

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

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

printf(" ");

}

if (r->data == query)

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

else

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

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

}

int \_count\_tree(Node\* r, int l, int query, int count) {

if (r == NULL)

return count;

count = \_count\_tree(r->right, l + 1, query, count);

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

printf(" ");

}

if (r->data == query) {

count++;

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

}

else {

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

}

count = \_count\_tree(r->left, l + 1, query, count);

return count;

}

int count\_tree(Node\* r, int l, int query) {

return \_count\_tree(r, l, query, 0);

}

int main() {

setlocale(LC\_ALL, "Russian");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setbuf(stdout, NULL);

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);

}

print\_tree(root, 0);

printf("Поиск числа:\n");

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

search\_tree(root, 0, D);

printf("Поиск количества вхождений числа:\n");

count\_tree(root, 0, D);

root = NULL;

start = 1;

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

while (start) {

printf("Введите число: ");

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

if (count\_tree(root, 0, D)) {

printf("Число уже есть\n\n");

continue;

}if (D == -1) {

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

start = 0;

}

else {

root = CreateTree(root, root, D);

}

}

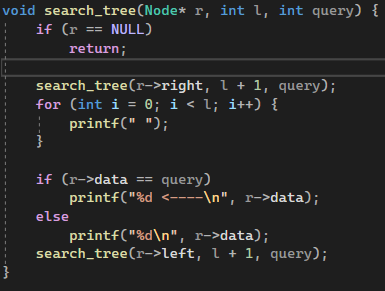
print\_tree(root, 0);

return 0;

}

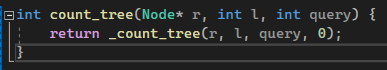
**Задание 1.**

Реализовали алгоритм поиска вводимого с клавиатуры значения в уже созданном дереве

****

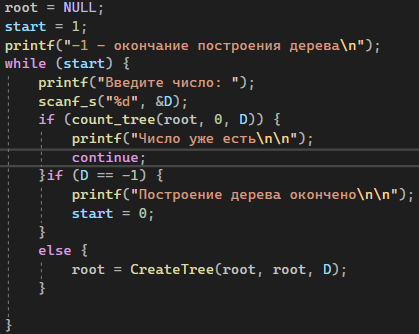
**Задание 2.**

Реализовать функцию подсчёта числа вхождений заданного элемента в дерево



**Задание 3.**

Изменить функцию добавления элементов для исключения добавления одинаковых символов



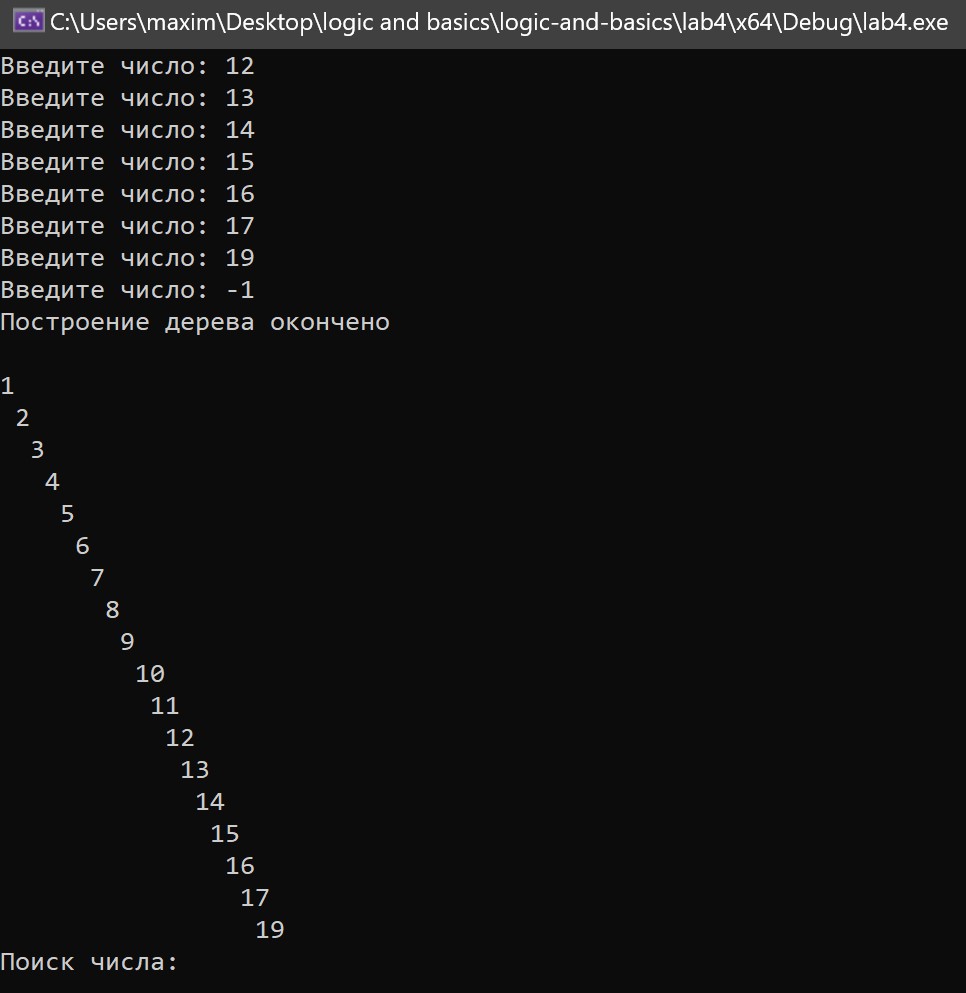
**Задание 4.**

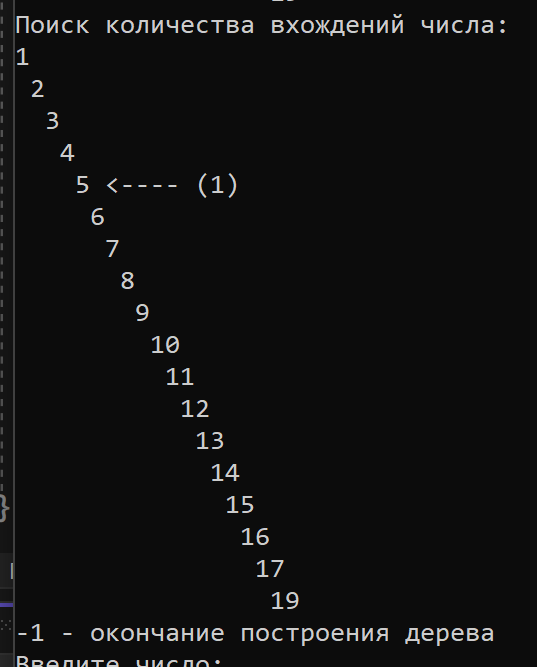
Оценить сложность процедуры поиска по значению в бинарном дереве

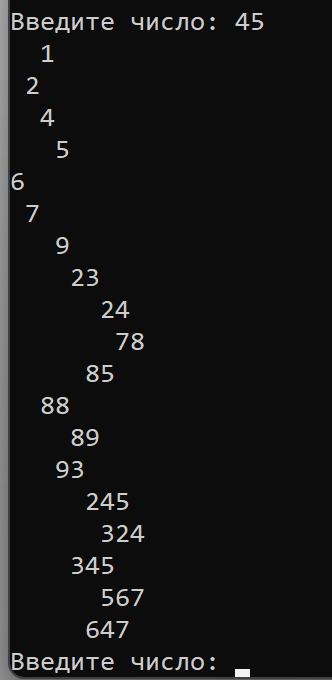
Средняя сложность – O(log(n))

Наихудшая сложность – O(n) – матрица выродилась

**Результат работы программы:**

****

****

****

**Вывод**

Разработал алгоритм бинарного дерева поиска. Научился работать с данным типом данных