**ОТЧЕТ**

по лабораторной работе №4

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Бинарное дерево поиска»

Выполнили:

студенты группы 23ВВВ4

Гришаев. Д.А.

Фельдман Г.О.

Приняли:

Деев М.В.

Юрова О.В.

**Название**

Бинарное дерево поиска

**Цель работы**

Познакомиться с бинарным деревом поиска, изучить основные принципы работы с ним и выполнить соответствующие задания

**Лабораторное задание**

На основе приведенных в лабораторной работе примеров выполнить следующие задания:

**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.

**Описание метода решения задачи**

На основе описанного программного кода, опираясь на теорию, дополнили программу, написав несколько функций (поиска по числу, поиска вхождений..)

### Листинг

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

struct Node\* root;

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

{

if (r == NULL){

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

if (r == NULL)

{

printf("Ошибка выделения памяти");

exit(0);

}

r->left = NULL;

r->right = NULL;

r->data = data;

if (root == NULL) return r;

if (data > root->data) root->left = r;

else root->right = r;

return r;

}

//if (data == r->data) return(root); // \*\*безповторений\*\*

if (data > r->data)

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

else

CreateTree(r, r->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\* find(struct Node\* r, int d) {

if (r == NULL){

return(NULL);

}

if (r->data == d) return(r);

if (r->data > d) {

find(r->right, d);

}

else {

find(r->left, d);

}

}

int findhowmany(struct Node\* r, int d) {

r = find(r, d);

if (r == NULL) return(0);

int num = 1;

while (r->right->data == d) {

num++;

r = r->right;

}

return(num);

}

int main()

{

system("chcp 1251");

int D, start = 1;

root = NULL;

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

while (start)

{

printf("Введитечисло: ");

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("Выберите что вы хотите сделать: \nНайти ветку по числу - 1\nНайти количество вхождений числа - 2\n");

char mode = \_getch();

printf("Введитечисло: ");

switch (mode) {

case '1':{

scanf("%d", &D);

struct Node\* temp = find(root, D);

if (temp != NULL) {

printf("Найденнаяветка:\n");

print\_tree(temp, 0);

}

else {

printf("Числоненайдено!");

}

break;

}

default: {

scanf("%d", &D);

printf("Количество вхождений числа: %i", findhowmany(root, D));

}

}

}

**Пояснительный текст к программе**

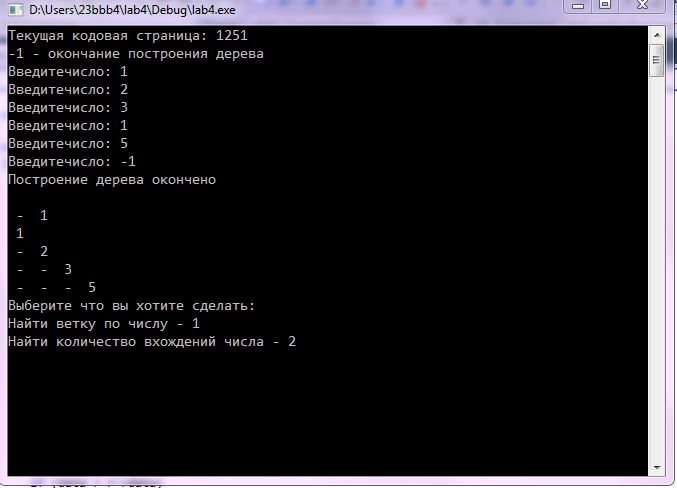
CreateTree - функция выделения памяти и создания дерева

print\_tree- функция вывода дерева

find - функция поиска по числу

findhowmany - поиск количества вхождений числа

**Результат работы программы**

****

**Рисунок 1 — Результаты работы программы**

### Выводы

Познакомиться с бинарным деревом поиска, изучить основные принципы работы с ним и выполнить соответствующие задания на языке C.