Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра "Вычислительная техника"

Отчет

по лабораторной работе №4

по курсу “Л и ОА в ИЗ”

на тему “Бинарное дерево поиска”

Выполнил студенты группы 22ВВС1:

Агапов И.Е.

Братчиков А.А.

Приняли:

Юрова О.В.

Акифьев И.В.

Пенза 2023

### 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 <time.h>

#include <stdlib.h>

#include <string>

#include <iostream>

struct Node

{

int data;// полезная информация

struct Node\* right;

struct Node\* left;

};

int manytimes(int data, struct Node\* root);

struct Node\* find(int data, struct Node\* root);

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

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

int main()

{

setlocale(LC\_ALL, "");

int D, start = 1;

struct Node\* root;

root = NULL;

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

while (start)

{

printf("Введите число: ");

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

/\*if (find(D, root) != NULL) {

printf("Такой уже есть\n");

}

else\*/

if (D == -1)

{

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

start = 0;

}

else

root = CreateTree(root, root, D);

}

print\_tree(root, 0);

printf("Введите искомое число\n");

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

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

int y = manytimes(D, root);

printf("Число %d встречается %d раз(a) \n",r->data, y);

return 0;

}

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)

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(int data, struct Node\* root) {

if(root==NULL){

return NULL;

}

if (root->data == data) {

return root;

}

if (data > root->data) {

find(data, root->left);

}

else {

find(data, root->right);

}

}

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

int many = 0;

struct Node\* r;

r = find(data, root);

while (1) {

if (r != NULL) {

many++;

r = r->right;

}

else {

break;

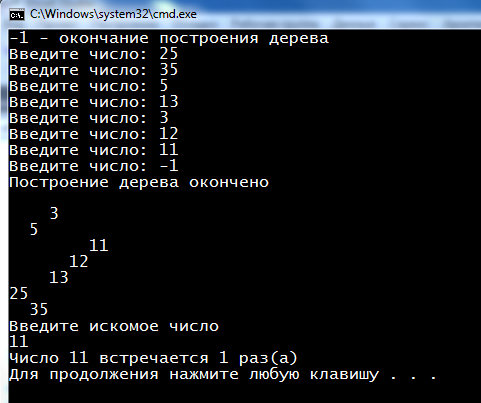
}

}

return many;

}

**Результаты работ программ**



**Вывод**

### В ходе выполнения лабораторной работы были обобщены знания о структуре данных типа дерево. Были реализованы функции поиска, нахождения количества вхождений в дерево, было добавлено предупреждение о повторном вводе.

### Была оценена сложность поиска в бинарном дереве. O(n) = log2n.