 Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЁТ**

по лабораторной работе №4

по курсу «Логика и основы алгоритмизации в ИЗ»

на тему «Бинарное дерево поиска»

Выполнили:

студенты группы 22ВВВ2

Перфилов А.В.

Приняли:

доцент, профессор Митрохин М.А.

к.э.н., доцент Акифьев И.В.

Пенза 2023

**Название**

Бинарное дерево поика

**Цель работы**

Научиться реализовывать бинарное дерево

**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 <locale>

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 if (data < root->data) root->right = r;

else {

free(r);

return root;

}

return r;

}

if (data > r->data)

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

else if (data < r->data)

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

return root;

}

/\*

struct node\* search(struct node\* root, int key) {

if (root == NULL || root->data == key) {

return root;

}

if (key < root->data) {

return search(root->left, key);

}

return search(root->right, key);

}

\*/

int countOccurrences(struct Node\* root, int key) {

if (root == NULL) {

return 0;

}

if (root->data == key) {

return 1 + countOccurrences(root->left, key) + countOccurrences(root->right, key);

}

else if (key < root->data) {

return countOccurrences(root->right, key);

}

else {

return countOccurrences(root->left, key);

}

}

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

{

int col = 0;

if (r == NULL) {

return 1;

}

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

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

printf(" ");

}

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

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

}

int main()

{

setlocale(LC\_ALL, "");

int D,item, 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);

}

printf("%d\n",print\_tree(root, 0)-1);

printf("Введите элемент для поиска\n");

scanf("%d", &item);

int ans = countOccurrences(root, item);

printf("%d", ans);

return 0;

}

**Результат работы программы**

Результат работы программы показан на рисунках 1.

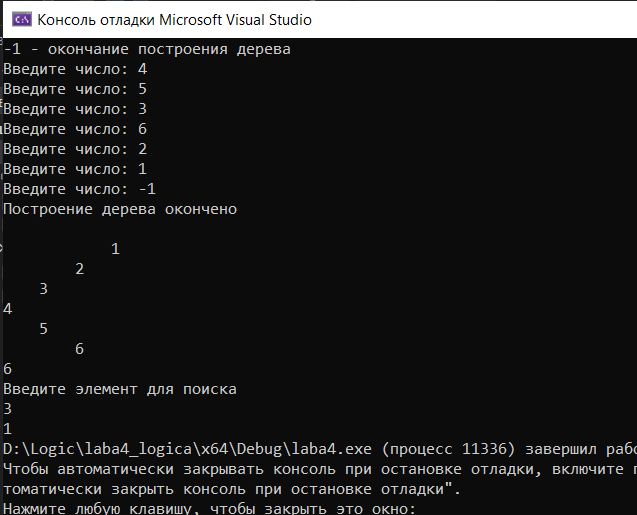


Рис. 1

**Вывод:**

В ходе выполнения лабораторной работы мы научились оценивать время выполнения программы и анализировать алгоритмы сортировки.

Ссылка на репозиторий: https://github.com/chif45/laba4\_logica