Пензенский государственный университет

Кафедра «Вычислительная техника»

ОТЧЕТ

по лабораторной работе № 4

по дисциплине: "Логика и основы алгоритмизации в инженерных задачах"

## на тему: "Бинарное дерево поиска"

Выполнили студенты гр. 22ВВП2:

Корнилов В.М.

Самофалова А.В.

Горбунов Д.А.

Приняли:

Акифьев И.В.

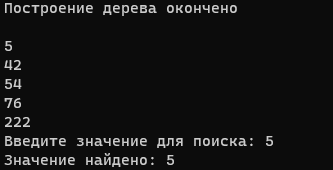
Юрова О.В.

Пенза, 2023

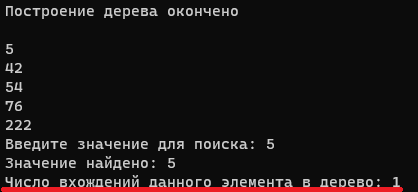
**Цель: Реализовать бинарное дерево поиска.**

**Задание.**

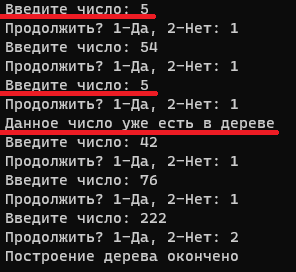
1. Реализовать алгоритм поиска вводимого с клавиатуры значения в уже созданном дереве.



1. Реализовать функцию подсчёта числа вхождений заданного элемента в дерево.



1. \* Изменить функцию добавления элементов для исключения добавления одинаковых символов.



1. \* Оценить сложность процедуры поиска по значению в бинарном дереве.

Сложность процедуры поиска по значению в бинарном дереве = log(n).

**Вывод:** научились реализовывать и использовать структуру данных – дерево, реализовывать поиск вводимого с клавиатуры значения в структуре, функцию подсчёта числа вхождений заданного элемента в дерево, исключать возможность добавления идентичных символов в структуру и оценивать сложность программы.

**Приложение А (Листинг)**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

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) {

printf("Данное число уже есть в дереве\n");

return root;

}

if (data > r->data)

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

else

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

return root;

}

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

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

return root;

if (root->data < key)

return search(root->right, key);

return search(root->left, key);

}

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);

}

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

if (root == NULL)

return 0;

int count = 0;

if (root->data == key)

count++;

count += EnterCount(root->left, key);

count += EnterCount(root->right, key);

return count;

}

int main()

{

struct Node\* root = NULL;

setlocale(LC\_ALL, "");

int D, start = 1;

int cnt = 1;

int data;

while (start)

{

printf("Введите число: ");

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

printf("Продолжить? 1-Да, 2-Нет: ");

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

if (cnt == 2)

{

root = CreateTree(root, root, D);

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

start = 0;

}

else if (cnt == 1)

root = CreateTree(root, root, D);

else

printf("Ошибка!\n");

}

print\_tree(root, 0);

printf("Введите значение для поиска: ");

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

struct Node\* res = search(root, data);

if (res != NULL)

printf("Значение найдено: %d\n", res->data);

else printf("Значение не найдено\n");

int entries = EnterCount(root, data);

printf("Число вхождений данного элемента в дерево: %d", entries);

return 0;

}