Министерство образования Российской федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №4

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Бинарное дерево поиска»

Выполнили

студенты группы 21ВВ3:

Савицкий Макар

Приняли:

Митрохин М. А.

Юрова О. В.

Пенза 2022

**Цель работы**

Реализация бинарного дерева посредством массивов структур и работа с поиском элементов в нем.

**Лабораторное задание**

**Задание 1:**

Реализовать 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.

**Теоретический материал**

*Известно, что слева от узла располагается элемент, который меньше чем текущий узел. Из чего следует, что если у узла нет левого наследника, то он является минимумом в дереве. Таким образом, можно найти минимальный элемент дерева.*

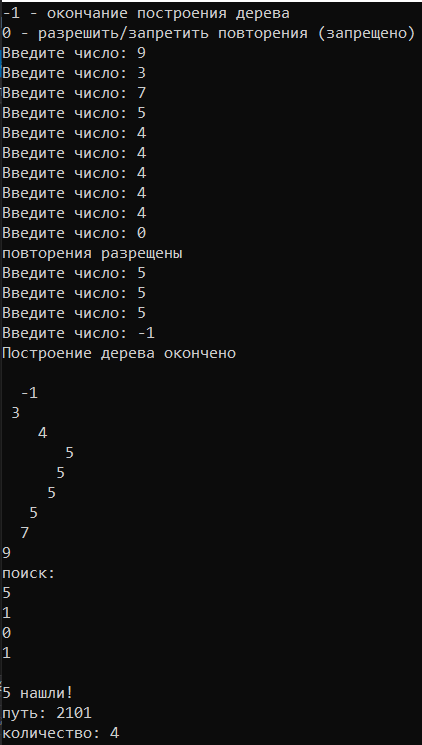
*Опять же, если дерево хорошо сбалансировано, то поиск минимума и максимума будет иметь сложность порядка log(n), а в случае плохой балансировки стремится к n.*

*Поиск нужного узла по значению похож на алгоритм бинарного поиска в отсортированном массиве. Если значения больше узла, то продолжаем поиск в правом поддереве, если меньше, то продолжаем в левом. Если узлов уже нет, то элемент не содержится в дереве.*

**Описание метода решения задачи**

1. *Выводим памятку по использованию программы*
2. *Вводим значения ветвей дерева*
3. *Вводим значение, которое нужно найти*
4. *Выводим результат поиска (путь к нужной ветке и количество совпадений)*

**Результаты работ программы**

****

**Листинг 1**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h> //указание прототипов

#include <stdlib.h> //библиотечных функций

#include <windows.h>

#include <malloc.h>

#include <string.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, int d){

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 (d == 1 && data == r->data)

return root;

if (data > r->data)

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

else

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

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);

}

int schet(struct Node\* r, int l) {

if (r == NULL) {

return 0;

}

if (r->data == l) {

return 1 + schet(r->right, l);

}

if (r->data > l) {

return schet(r->right, l);

}

if (r->data < l) {

return schet(r->left, l);

}

}

int poisk(struct Node\* r, int l){

if (r == NULL){

return 3;

}

if (r->data == l) {

printf("\n%d нашли!\n", r->data);

return 2;

}

if (r->data > l){

printf("1\n");

return 10 \* poisk(r->right, l) + 1;

}

if (r->data < l) {

printf("0\n");

return 10 \* poisk(r->left, l);

}

}

int main(){

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int D, start = 1, ddd = 1;

root = NULL;

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

printf("0 - разрешить/запретить повторения (запрещено)\n");

while (start){

printf("Введите число: ");

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

if (D == -1){

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

start = 0;

}

if (D == 0) {

if (ddd == 1) {

ddd = 0;

printf("повторения разрещены\n");

}

else {

ddd = 1;

printf("повторения запрещены\n");

}

}

else

root = CreateTree(root, root, D, ddd);

}

print\_tree(root, 0);

printf("поиск:\n");

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

printf("путь: %d\n", poisk(root, D));

printf("количество: %d\n", schet(root, D));

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

return 0;

}

**Вывод**

В ходе выполнения данной лабораторной работы мы научились реализовывать бинарные дерева посредством массивов структур и искать элементы в нем.