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 21ВВ4:

Шутихин А.Э.

Ошкин И.В

Приняли:

Юрова О. В.

Акифьев И.В.

Пенза 2022

**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.

**Задание 1-2**

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <time.h>

struct Node {

struct Node \*left;

struct Node\* right;

int data=0;

};

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)

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);

}

void search(struct Node\* r, int data) {

if (r == NULL)

return;

if (r->data < data) {

search(r->right, data);

}

else if (r->data > data) {

search(r->left, data);

}

else if (r->data == data) {

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

}

}

int count(Node\* root, int k) {

if (root == 0)

return 0;

return (root->data == k) + count(root->left, k) + count(root->right, k);

}

int main()

{

setlocale(LC\_ALL, "");

srand(time(NULL));

int D, numb;

printf ("Введите размер дерева: ");

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

while (D)

{

root = CreateTree(root, root, rand()%20);

D--;

}

print\_tree(root, 0);

printf("Введите искомый элемент: ");

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

search(root, D);

numb = count(root, D);

if (numb)

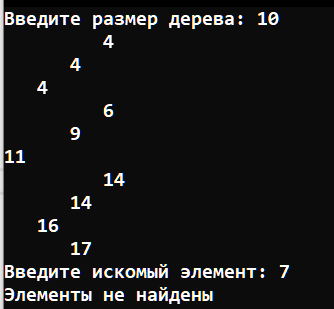
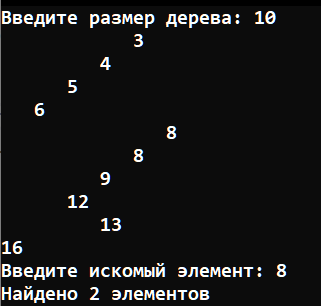
printf("Найдено %d элементов", numb);

else

printf("Элементы не найдены");

return 0;

}

****

**Задание 3**

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <time.h>

struct Node {

struct Node \*left;

struct Node\* right;

int data=0;

};

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)

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);

}

void search(struct Node\* r, int data) {

if (r == NULL)

return;

if (r->data < data) {

search(r->right, data);

}

else if (r->data > data) {

search(r->left, data);

}

else if (r->data == data) {

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

}

}

int count(Node\* root, int k) {

if (root == 0)

return 0;

return (root->data == k) + count(root->left, k) + count(root->right, k);

}

int check(Node\* root, int k) {

if (root == 0)

return 0;

return (root->data == k) + check(root->left, k) + check(root->right, k);

}

int main()

{

setlocale(LC\_ALL, "");

srand(time(NULL));

int D, numb;

printf ("Введите размер дерева (0,20): ");

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

while (D)

{

do {

numb = rand() % 20;

} while (check(root, numb));

root = CreateTree(root, root, numb);

D--;

}

print\_tree(root, 0);

printf("Введите искомый элемент: ");

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

search(root, D);

numb = count(root, D);

if (numb)

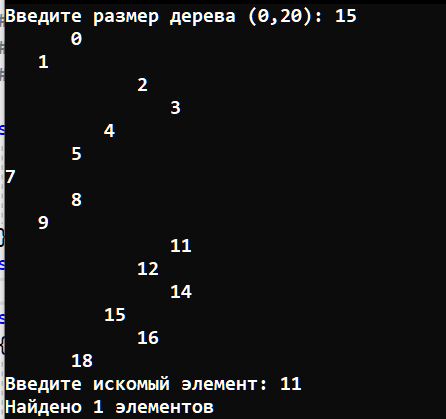
printf("Найдено %d элементов", numb);

else

printf("Элементы не найдены");

return 0;

}

****

**Задание 4**

Сложность функции - O(log n)

**Вывод:** Мы научились создавать функции построения бинарных деревьев, поиска элементов в деревьях.