Федеральное государственное бюджетное образовательное учреждение высшего образования «Пензенский Государственный университет»

Кафедра «Вычислительная техника»

ОТЧЕТ

**По лабораторной работе №4**

по курсу «Логика и основы алгоритмизации в ИЗ»

на тему «Бинарное дерево поиска»

Выполнили: ст. гр. 21ВВ1

Сурков А.А

Хорошильцев Е.А

Кривенков И.В

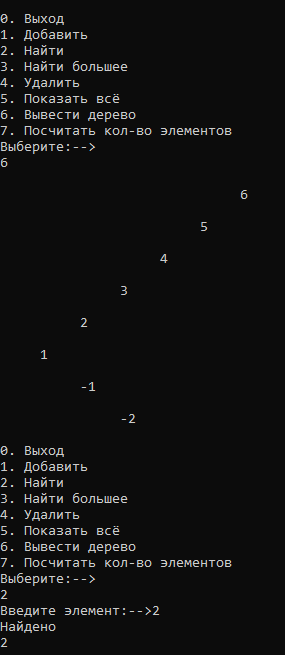
Приняли: Юрова О.В.

Акифьев И.В.

2022

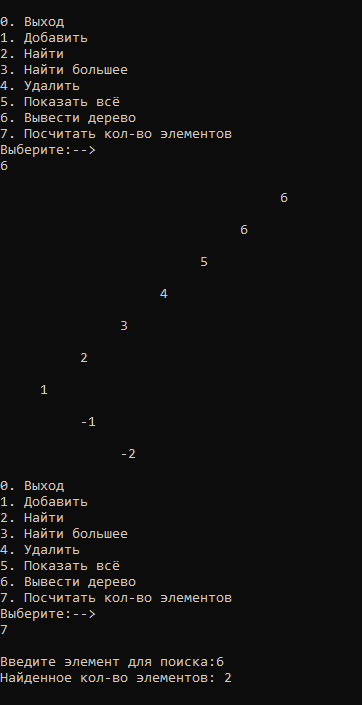
**Задание 1:**

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

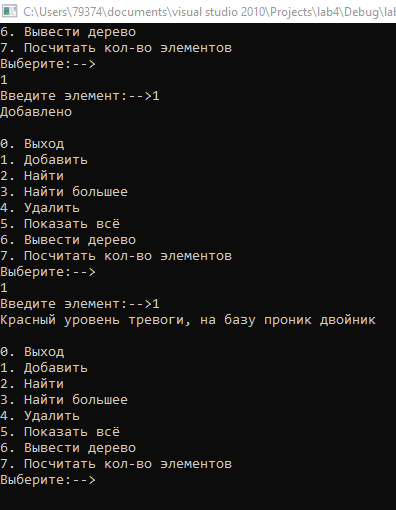


**Задание 2:**

Реализовать функцию подсчёта числа вхождений заданного элемента в дерево.



**Задание 3:**



**Задание 4:**

Сложность: в среднем; O(n) в худшем случае.

**Вывод:**

В ходе данной лабораторной работы мы научились создавать бинарное дерево поиска, а также отточили свои навыки программирования благодаря функция поиска значений в дереве и подсчёта количества определенных значений в дереве.

**Листинг:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <malloc.h>

#include <locale.h>

typedef struct Node {

int key;

struct Node \*left;

struct Node \*right;

struct Node \*parent;

} Node;

int getInt (int \*);

Node \*find (Node \*, int);

Node \*minNode (Node \*);

Node \*maxNode (Node \*);

Node \*nextNode (Node \*);

Node \*insert (Node \*, int);

Node \*del (Node \*, Node \*);

void deltree(Node \*);

int dialog (const char \*\*, int);

Node \*T\_Add (Node \*);

Node \*T\_Find (Node \*);

Node \*T\_Delete (Node \*);

Node \*T\_ShowTable (Node \*);

Node \*T\_ShowTree (Node \*);

void ShowTree (Node \*, int);

Node \*T\_MaxNode (Node \*);

Node \*T\_Counter (Node \*);

int main()

{

setlocale(LC\_ALL, "Rus");

Node \*root = NULL;

int rc, N = 8;

const char \*msgs[] = {"0. Выход","1. Добавить","2. Найти","3. Найти большее","4. Удалить","5. Показать всё","6. Вывести дерево", "7. Посчитать кол-во элементов"};

Node \*(\*fptr[])(Node\*) = {NULL, T\_Add, T\_Find, T\_MaxNode, T\_Delete, T\_ShowTable, T\_ShowTree, T\_Counter};

while(rc = dialog(msgs, N))

root = fptr[rc](root);

printf("Биби бай-бай!\n");

deltree(root);

return 0;

}

Node \*find (Node \*root, int key) {

Node \*ptr = root;

while (ptr != NULL){

if (ptr->key == key)

return ptr;

else if (ptr->key > key)

ptr = ptr->left;

else

ptr = ptr->right;

}

return 0;

}

Node \*minNode (Node \*ptr) {

if (ptr == NULL)

return 0;

else

while (ptr->left != NULL)

ptr = ptr->left;

return ptr;

}

Node \*maxNode (Node \*ptr) {

if (ptr == NULL)

return 0;

else

while (ptr->right != NULL)

ptr = ptr->right;

return ptr;

}

Node \*nextNode (Node \*x) {

Node \*ptr = NULL;

if (x == NULL)

return 0;

if (x->right != NULL){

ptr = minNode(x->right);

return ptr;

}

ptr = x->parent;

while (ptr != NULL && x == ptr->right){

x = ptr;

ptr = x->parent;

}

return ptr;

}

Node \*insert (Node \*root, int key) {

Node \*ptr, \*x, \*par;

ptr = root;

par = 0;

x = (Node \*) calloc (1, sizeof(Node));

if (ptr == NULL){

x->key = key;

ptr = x;

return ptr;

}

while (ptr != NULL){

par = ptr;

if (ptr->key > key)

ptr = ptr->left;

else

ptr = ptr->right;

}

if (par->key > key){

par->left = x;

x->key = key;

}

else {

par->right = x;

x->key = key;

}

return root;

}

Node\* del(Node\* root, int key) {

if (root == NULL)

return root;

if (key < root->key)

root->left = del(root->left, key);

else if (key > root->key)

root->right = del(root->right, key);

else {

if (root->left == NULL) {

Node\* temp = root->right;

free(root);

return temp;

}

else if (root->right == NULL) {

Node\* temp = root->left;

free(root);

return temp;

}

Node\* temp = minNode(root->right);

root->key = temp->key;

root->right = del(root->right, temp->key);

}

return root;

}

void deltree(Node \*p)

{

if (p != NULL) {

deltree(p->left);

deltree(p->right);

free(p);

}

}

int dialog (const char \*msgs[], int N){

setlocale(LC\_ALL, "Rus");

const char \*errmsg = "";

int rc,i,n;

do{

puts(errmsg);

errmsg = "Плохо, пробуй ещё!";

for (i = 0; i < N; ++i)

puts(msgs[i]);

puts("Выберите:-->");

n = getInt(&rc);

if(n==0)

rc = 0;

} while (rc < 0 || rc >= N);

return rc;

}

Node \*T\_Add (Node \*root){

setlocale(LC\_ALL, "Rus");

int k, n = 1;

char \*info;

printf("Введите элемент:-->");

n = getInt(&k);

/\*if (find(root, k) == 0 || find(root, k) != 0) {

root = insert(root, k);

puts("Добавлено");

}\*/

if (find(root, k) == 0) {

root = insert(root, k);

puts("Добавлено");

}

else puts("Красный уровень тревоги, на базу проник двойник");

return root;

}

Node \*T\_Find (Node \*root){

setlocale(LC\_ALL, "Rus");

Node \*x = NULL;

int k, n = 1;

printf("Введите элемент:-->");

n = getInt(&k);

x = find(root, k);

if (x != 0) {

puts("Найдено");

printf("%d \n", x->key);

}

else puts("Не найдено");

return root;

}

Node \*T\_Delete (Node \*root){

setlocale(LC\_ALL, "Rus");

Node \*x = NULL;

int k, n = 1;

printf("Введите элемент:-->");

n = getInt(&k);

x = find(root, k);

if (x != 0) {

root = del(root, k);

puts("Удалено");

}

else puts("Не найдено");

return root;

}

Node \*T\_MaxNode (Node \*root){

setlocale(LC\_ALL, "Rus");

Node \*x;

if (root == NULL)

return 0;

x = maxNode(root);

puts("Найдено большее");

printf("%d \n", x->key);

return root;

}

Node \*T\_ShowTable (Node \*root)

{

if (root == NULL)

return 0;

T\_ShowTable(root->right);

printf("\n");

printf(" ");

printf("%d\n", root->key);

T\_ShowTable(root->left);

return root;

}

void ShowTree (Node \*root, int flag) {

flag += 5;

if (root == NULL)

return;

ShowTree(root->right, flag);

printf("\n");

for(int i = 0; i < flag; ++i)

printf(" ");

printf("%d\n", root->key);

ShowTree(root->left, flag);

}

Node \*T\_ShowTree (Node \*root){

ShowTree(root, 0);

return root;

}

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

if (!root) return 0;

return (root->key == E ? 1 : 0) +

count(E, root->left) +

count(E, root->right);

}

int counter(int E, Node \*root)

{

setlocale(LC\_ALL, "Rus");

printf("\n");

printf("Введите элемент для поиска:");

scanf("%d",&E);

printf("Найденное кол-во элементов: %d", count(E, root));

printf("\n");

return(0);

}

Node \*T\_Counter (Node \*root)

{

counter(0, root);

return root;

}

int getInt (int \*a) {

setlocale(LC\_ALL, "Rus");

int n;

do {

n = scanf("%d", a);

if (n < 0)

return 0;

else if (n == 0){

printf("Плохо, пробуй ещё!\n");

scanf("%\*c");

}

} while (n == 0);

return 1;

}