**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 23ВВВ1 :

Макаров А.С.

Приняли:

к.т.н. доцент Митрохин М.А.

к.т.н., доцент Юрова О.В.

Пенза 2024

Вспомнить работу с динамическими списками в Си и закрепить на практике.

**Задание.**

**Задания**:

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.

**Ход работы:**

**1-2 задание**

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

struct Node\* root = NULL;

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);

}

struct Node\* find(struct Node\* root, int data) {

if (root->data == data) return root;

if (root->data > data && root->right != NULL) find(root->right, data);

else if (root->left != NULL) find(root->left, data);

else return NULL;

}

int count(struct Node\* root, int data, int cnt) {

if (root->data == data) cnt++;

if (root->data >= data && root->right != NULL) cnt = count(root->right, data, cnt);

else if (root->left != NULL) cnt = count(root->left, data, cnt);

else return cnt;

}

int main()

{

setlocale(LC\_ALL, "Rus");

int D, start = 1;

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);

}

print\_tree(root, 0);

int num;

int cnt;

printf("Введите элемент для поиска: ");

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

struct Node\* r = find(root, num);

if (r) printf("Элемент найден: %d\n", r->data);

else printf("элемент не найден\n");

cnt = count(root, num, 0);

printf("Элиментов %d найдено: %d\n",num, cnt);

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

return 0;

}

**3 задани**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <locale.h>**

**struct Node {**

**int data;**

**struct Node\* left;**

**struct Node\* right;**

**};**

**struct Node\* root = NULL;**

**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)**

**return root;**

**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);**

**}**

**struct Node\* find(struct Node\* root, int data) {**

**if (root->data == data) return root;**

**if (root->data > data && root->right != NULL) find(root->right, data);**

**else if (root->left != NULL) find(root->left, data);**

**else return NULL;**

**}**

**int count(struct Node\* root, int data, int cnt) {**

**if (root->data == data) cnt++;**

**if (root->data >= data && root->right != NULL) cnt = count(root->right, data, cnt);**

**else if (root->left != NULL) cnt = count(root->left, data, cnt);**

**else return cnt;**

**}**

**int main()**

**{**

**setlocale(LC\_ALL, "Rus");**

**int D, start = 1;**

**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);**

**}**

**print\_tree(root, 0);**

**int num;**

**int cnt;**

**printf("Введите элемент для поиска: ");**

**scanf\_s("%d", &num);**

**struct Node\* r = find(root, num);**

**if (r) printf("Элемент найден: %d\n", r->data);**

**else printf("элемент не найден\n");**

**cnt = count(root, num, 0);**

**printf("Элиментов %d найдено: %d\n", num, cnt);**

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

**return 0;**

**}**



