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 21вв1:**

Кирьянов В.Е.

Аляев А.О.

**Приняли**

Юрова О.В.

Акифьев И.В.

Пенза 2022

**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.

**Листинг программы:**

**Задание 1:**

// Lab\_4\_task1.cpp: определяет точку входа для консольного приложения.

//

#include "StdAfx.h"

#include <stdlib.h>

#include "stdafx.h"

#include "malloc.h"

#include "locale.h"

#include "iostream"

using namespace std;

struct Node {

int data;

struct Node \*left;

struct Node \*right;

};

struct Node \*root;

int amount = 0;

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);

}

int findint(Node \*tree, int find)

{

if (tree != NULL){

if (find == tree->data){

amount++;

}

findint(tree->left, find);

findint(tree->right,find);

return 0;

}

}

int replay(Node \*tree, int D)

{

if (tree != NULL){

if (D == tree->data)

tree->data = rand()%100;

replay(tree->left, D);

replay(tree->right,D);

}

return 0;

}

int main()

{

struct Node\* root;

setlocale(LC\_ALL, "");

int D, start = 1;

int k = 1;

root = NULL;

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

printf("Введите число: ");

while (start)

{

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

replay(root,D);

if (D == -1)

{

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

start = 0;

}

else{

root = CreateTree(root, root, D);

}

}

print\_tree(root,0);

while (k)

{

cout << "\n1 - поиск элемента по дереву\n2 - колличество элементов в дереве\n" << "0 - выход" << endl;

cin >> k;

if (k == 1)

{

int find;

cout << "Введите элемент поиска: \n";

cin >> find;

findint(root, find);

if (amount != 0){

cout << "Элемент найден! \n";

cout << "Число: " << amount << "\n";

}

else

cout << "Элемент не найден! \n";

}

amount = 0;

if (k == 0)

return 0;

}

system("pause");

return 0;

}

**Ход работы:**

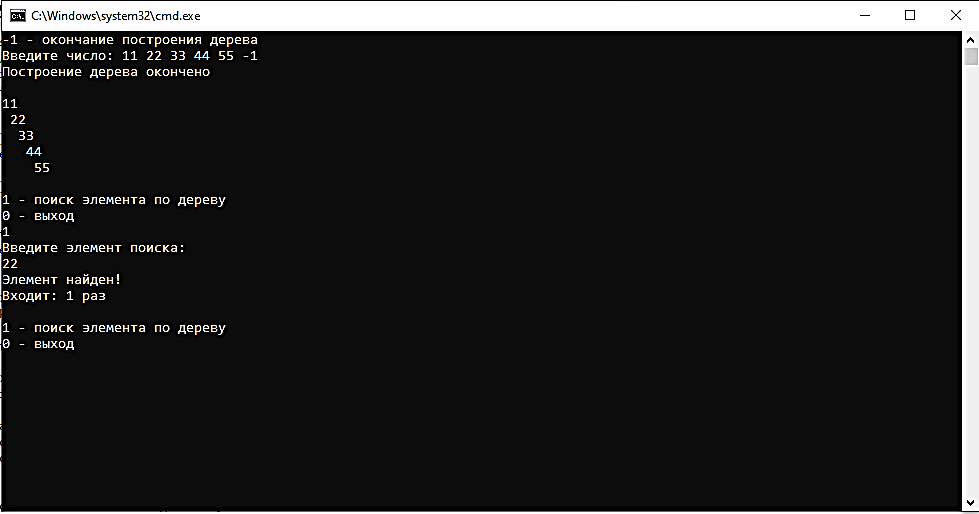
**Задание 1:**

Рисунок 1 – результат работы программы.

**Задание 2:**

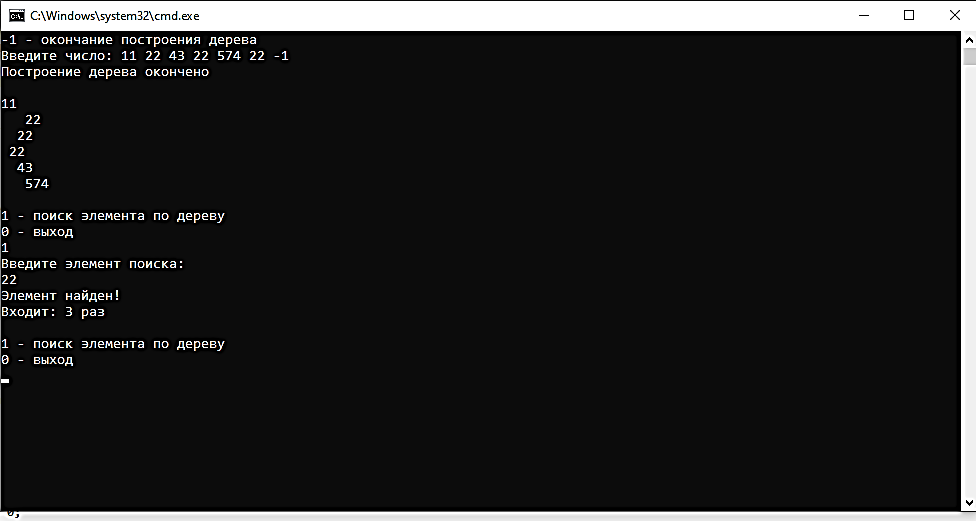
****

Рисунок 2 – результат работы программы.

**Задание 3:**

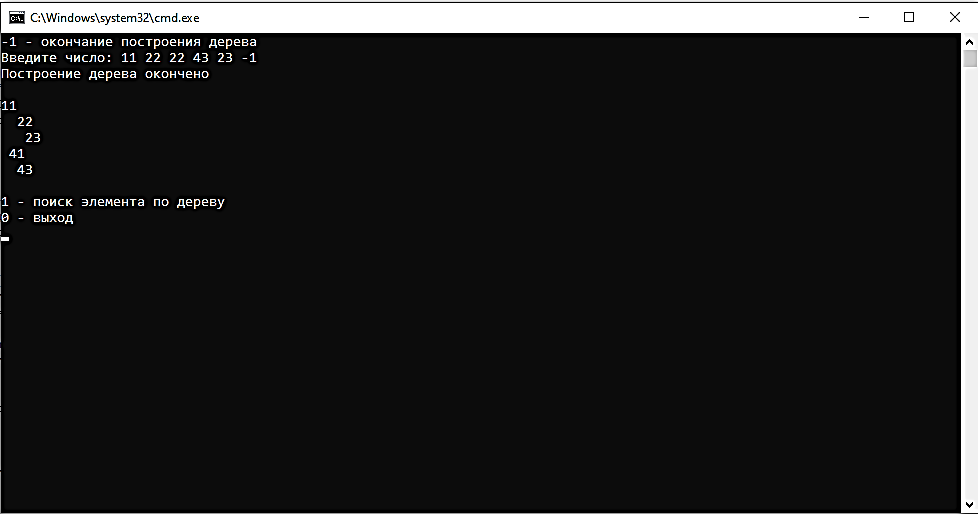
****

Рисунок 3 – результат работы программы.

**Задание 4:**

Так как при обходе дерева каждый элемент проходится 1 раз и скорость обработки каждого узла постоянна, то сложно алгоритма поиска – O(n).

**Вывод:** в ходе выполнения лабораторной работы были разработаны программы, выполняющие работу с бинарными деревьями.

Получили опыт в создании проектов в среде Microsoft Visual Studio, научились писать и отлаживать программы с применением бинарных деревьев на языке Си.