Министерство высшего образования и науки

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 21ВВ2:

Хабибулин А.М.

Щеглов Д.А.

Приняли:

Митрохин М.А.

Юрова О.В.

**Название**

Бинарное дерево поиска.

**Лабораторное задание.**

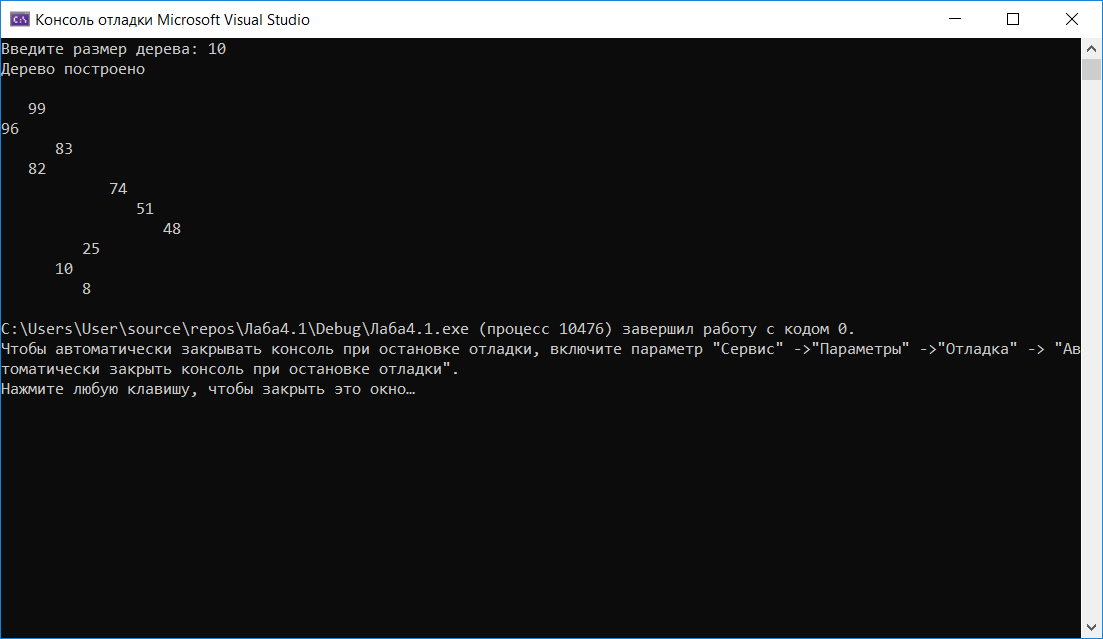
**Задание 1.** Реализовать алгоритм поиска вводимого с клавиатуры значения в уже созданном дереве.

**Задание 2.** Реализовать функцию подсчёта числа вхождений заданного элемента в дерево.

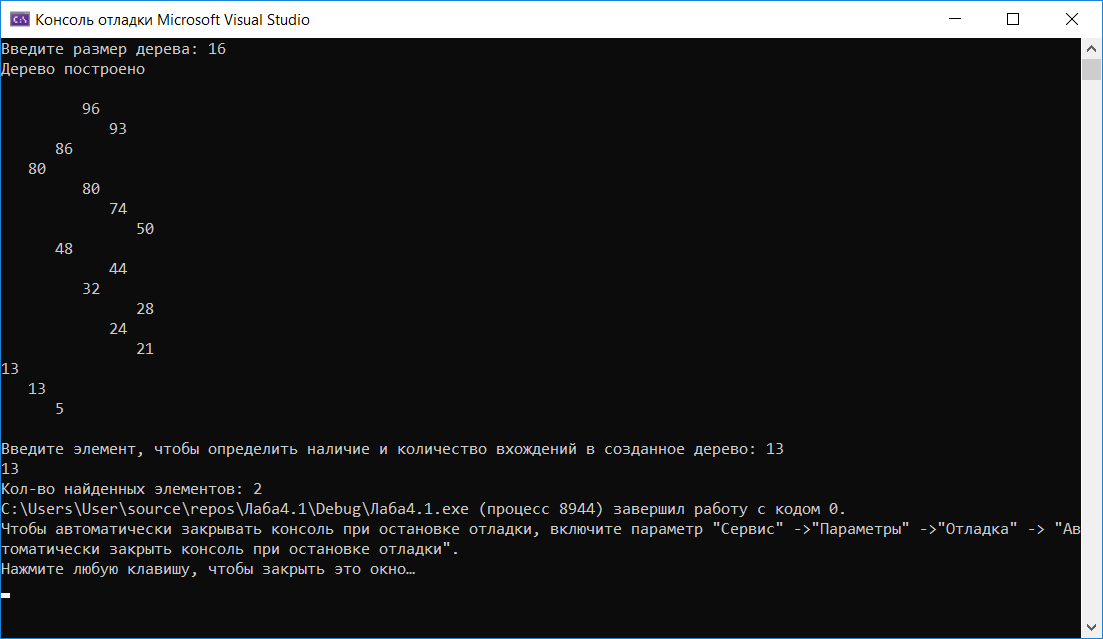
**Задание 3.** \*Изменить функцию добавления элементов для исключения добавления одинаковых символов.

**Задание 4.** \*Оценить сложность процедуры поиска по значению в бинарном дереве.

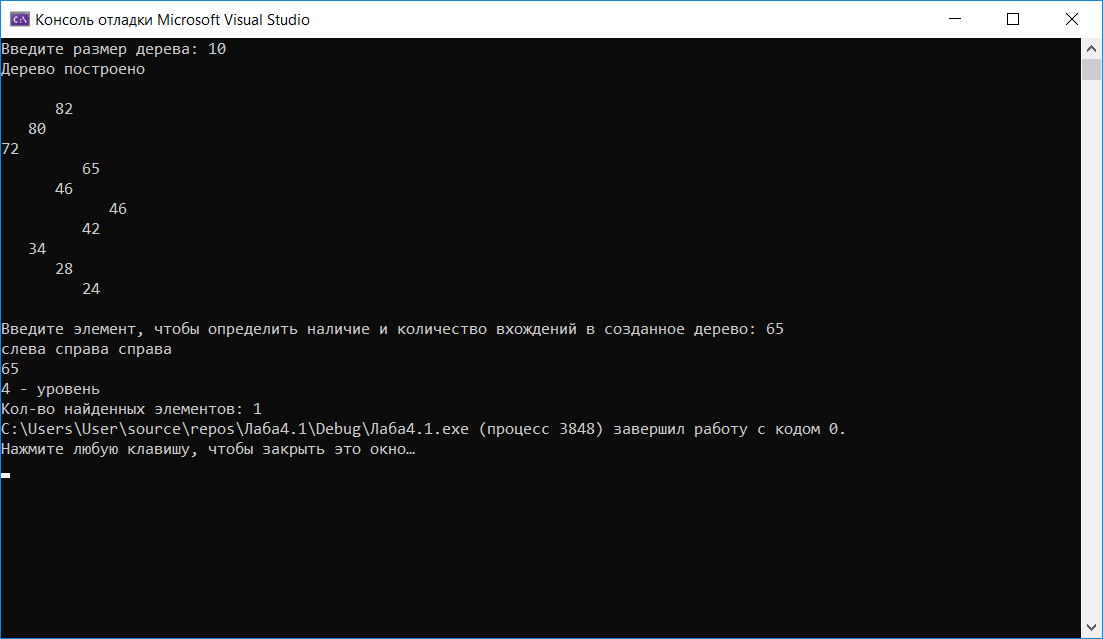
**Задание 1. Результат работы программы.**



**Задание 2. Результат работы программы.**

****

**Задание 3. Результат работы программы.**

****

**Листинг программы**

#include <iostream>

#include <stdio.h>

#include <locale.h>

#include <malloc.h>

#include <stdlib.h>

#include <time.h>

#include <stdlib.h>

using namespace std;

struct Node {

int data; //данные в узле

struct Node\* left;//левая нода

struct Node\* right;//правая нода

};

struct Node\* CreateTree(struct Node\* root, struct Node\* r, int data) // функция создания дерева

{

if (r == NULL)

{

r = (struct Node\*)malloc(sizeof(struct Node)); //динамическое выделение памяти под структуру

if (r == NULL)

{

cout << "Ошибка выделения памяти";

exit(0);

}

r->left = NULL;

r->right = NULL;

r->data = data;

if (root == NULL)

return r;

if (data > root->data) root->right = r;

else root->left = r;

return r;

}

if (data > r->data)

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

else

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

return root;

}

void print\_tree(struct Node\* r, int l) //функция вывода дерева

{

if (r == NULL)

return;

print\_tree(r->right, l + 1);

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

cout << (" ");

cout << r->data << endl;

print\_tree(r->left, l + 1);

}

void search (struct Node\* r, int data, int count) //поиск по заданному элементу

{

if (r == NULL)

return;

if (r->data < data) {

cout << "справа ";

search(r->right, data, count + 1);

}

else if (r->data > data) {

cout << "слева ";

search(r->left, data, count + 1);

}

if (r->data == data)

cout << endl << r->data << endl << count << " - уровень" << endl;

}

int quantity (Node\* root, int N) // подсчет количества элементв в дереве

{

if (root == 0) return 0;

return (root->data == N) + quantity(root->left, N) + quantity(root->right, N);

}

int main()

{

setlocale(LC\_ALL, "");

srand(time(NULL));

int N, num;

struct Node\* root = NULL;

cout << "Введите размер дерева: ";

cin >> N;

while (N) {

root = CreateTree(root, root, rand() % 100);

N--;

}

cout << "Дерево построено" << endl << endl;

print\_tree(root, 0);

cout << endl << "Введите элемент, чтобы определить наличие и количество вхождений в созданное дерево: ";

cin >> N;

search (root, N, 1);

num = quantity (root, N);

if (num)

cout << "Кол-во найденных элементов: " << num;

else

cout << "Ничего не найдено";

return 0;

getchar();

}

**Вывод**

Бинарные деревья – это деревья, у каждого узла которого возможно наличие

только двух сыновей. Двоичные деревья являются упорядоченными. Самый первый элемент называется родительским. У родительского элемента есть два наследника - правый и левый. Слева находятся листы с меньшим значением по отношению к узлу, из которого они выходят, а справа большие.