Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
Кафедра вычислительной техники

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ № 22

Двоичное дерево поиска
Вариант № 19

Преподаватель _____ Пушкарев К. В.
подпись, дата

Студент КИ18-09б, 031831293 _____ Овсянников В.А.
подпись, дата

Красноярск 2019

1 Назначение функции (Упражнение № 2 вариант 1)

Для представленного фрагмента программы, оформленного как функция и предназначенного для решения конкретной задачи обработки двоичного дерева поиска, выполнить следующее:

1. определить назначение функции;
2. прокомментировать смысл формальных параметров и возвращаемого значения, а также ход решения задачи;
3. привести графическую схему алгоритма для представленной функции;
4. дополнить предложенный фрагмент функциями построения дерева, обхода дерева и главной функции;
5. подготовить наборы тестовых данных, сопровождая их рисунками, для отладки программы, отладить и продемонстрировать преподавателю полученную программу;
6. подготовить отчет согласно выше перечисленным пунктам и защитить работу.

Вариант	Фрагмент программы

1.	`bool tree_equality (node *p1, node *p2) {`
	`  if (p1 == NULL && p2 == NULL) return 1;`
	`  if (p1 != NULL && p2 != NULL && p1->info == p2->info)`
	`  return tree_equality(p1->rl, p2->rl) &&`
	`  tree_equality(p1->ll, p2->ll);`
	`  else return 0;`
	`}`

Данная функция предназначена для сравнения двух деревьев.

2 Комментарии к функции

Формальные параметры принимают значения узлов первого и второго дерева, функция может возвращать 0 или false, 1 или true и саму себя со следующими значениями. Ход решения: Функция принимает на вход корневые узлы двух деревьев, если они равны и деревья имеют другие узлы функция вызывает саму себя, сравнивая последующие узлы деревьев, если все они равны, то функция возвращает 1, а если значение хотя бы одного узла не совпадает - функция возвращает 0.

3 Графическая схема алгоритма функции

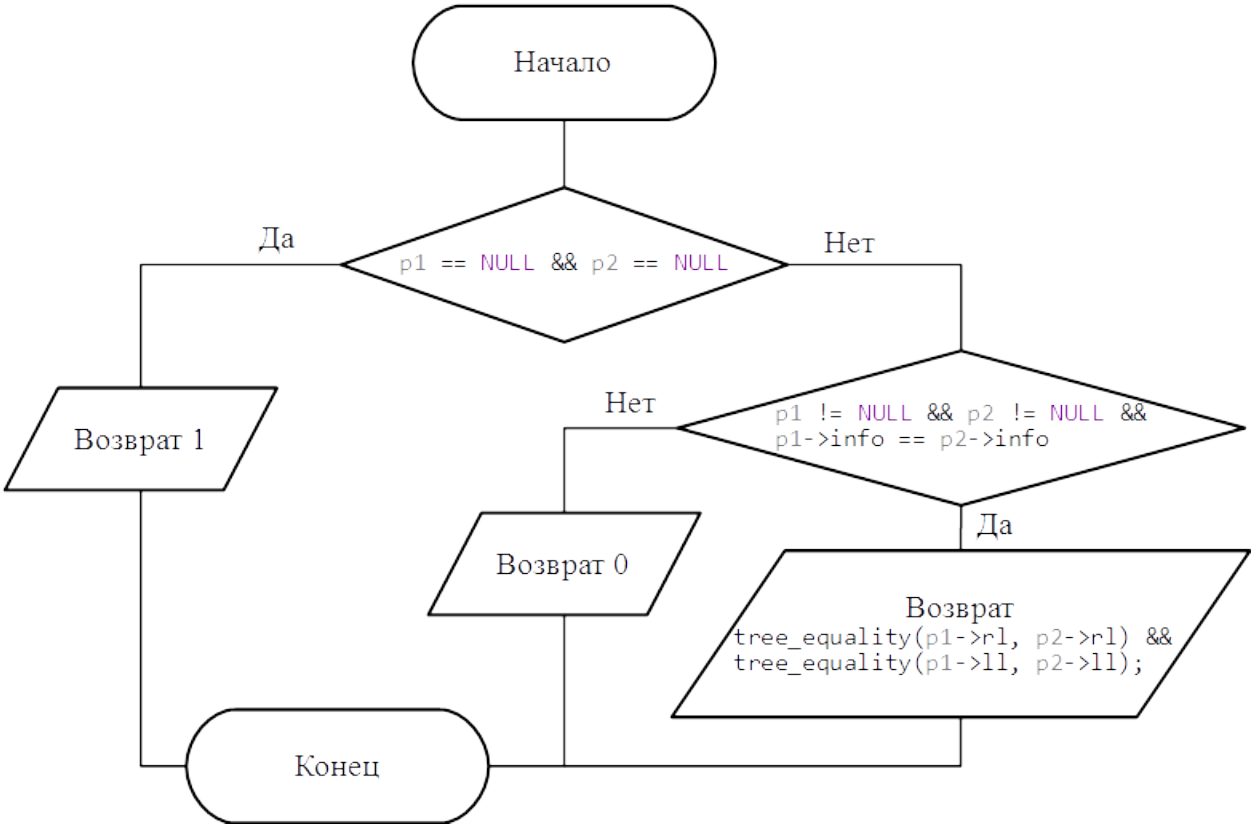


Рисунок 1 – функция bool tree_equality(node *p1, node *p2)

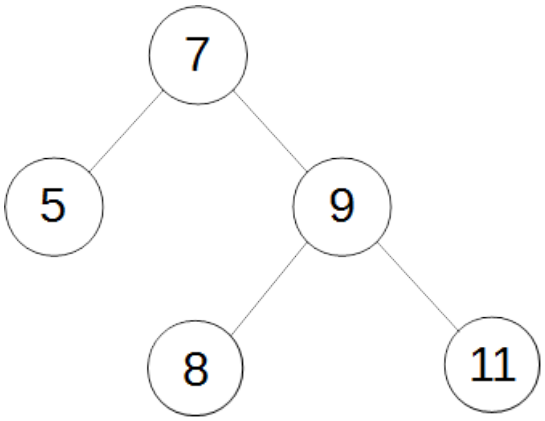
4 Код программы

```
#include "pch.h"
#include <iostream>
#include <clocale>

using namespace std;

struct node {
	int info;
	node *ll, *rl;
};
//Функция сравнения двух деревьев
bool tree_equality(node *p1, node *p2) {

	if (p1 == NULL && p2 == NULL) {
		return 1;
	}
	if (p1 != NULL && p2 != NULL && p1->info == p2->info) {

		return tree_equality(p1->rl, p2->rl) && tree_equality(p1->ll, p2->ll);
	}
	else {
		return 0;
	}
}
//Функция построения дерева
```

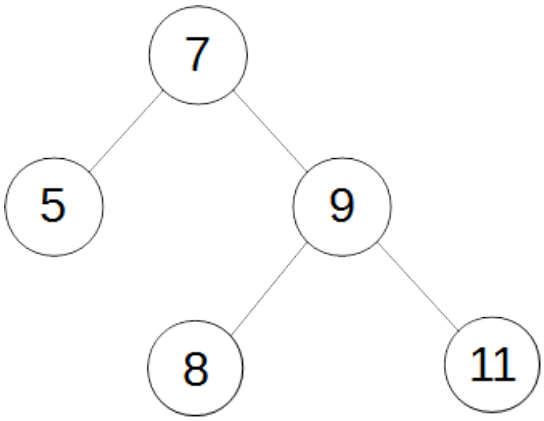
```
26 node *tree(node *p, int w) {
27         if (p == NULL) {
28                 p = new node;
29                 p->info = w;
30                 p->ll = nullptr;
31                 p->rl = nullptr;
32         }
33         else {
34                 if (w < p->info) {
35                         p->ll = tree(p->ll, w);
36                 }
37                 else {
38                         p->rl = tree(p->rl, w);
39                 }
40         }
41         return p;
42 }
43 //Функция обхода дерева
44 void treeprint(node *p) {
45         if (p != NULL) {
46                 treeprint(p->ll);
47                 cout << " " << p->info;
48                 treeprint(p->rl);
49         }
50 }
51
52 int main(){
53         setlocale(LC_ALL, "");
54
55         int t1, t2;
56         node *root1 = nullptr, *root2 = nullptr;
57
58         cout << "Введите узел первого дерева: ";
59         cin >> t1;
60         cout << "Введите узел второго дерева: ";
61         cin >> t2;
62         while (cin) {
63                 root1 = tree(root1, t1);
64                 root2 = tree(root2, t2);
65                 cout << "Введите узел первого дерева: ";
66                 cin >> t1;
67                 cout << "Введите узел второго дерева: ";
68                 cin >> t2;
69         }
70         cout << "\n\nПервое дерево: ";
71         treeprint(root1);
72         cout << "\nВторое дерево: ";
73         treeprint(root2);
74
75         if (tree_equality(root1, root2)) {
76                 cout << "\n\nДеревья равны" << endl;
77         }
78         else {
79                 cout << "\n\nДеревья не равны" << endl;
80         }
81
82         return 0;
83 }
```

5 Наборы тестовых данных

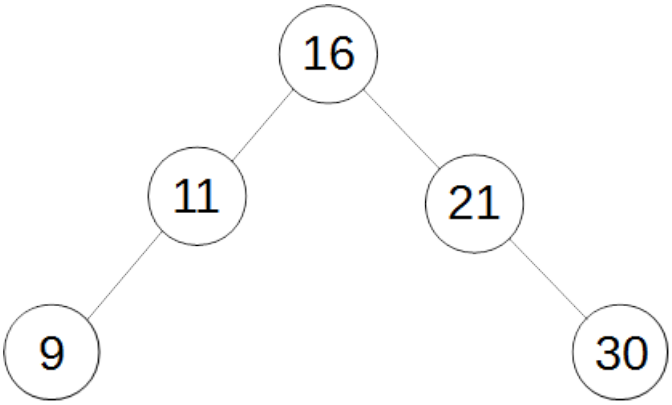
root 1



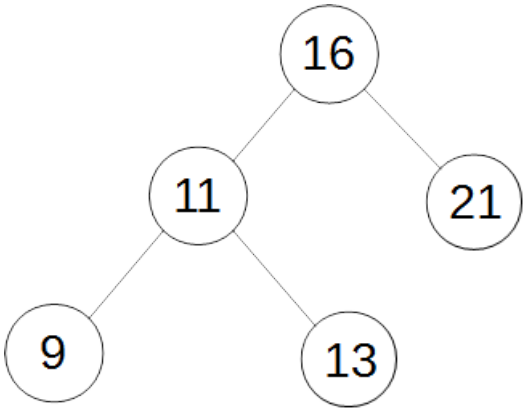
root 2



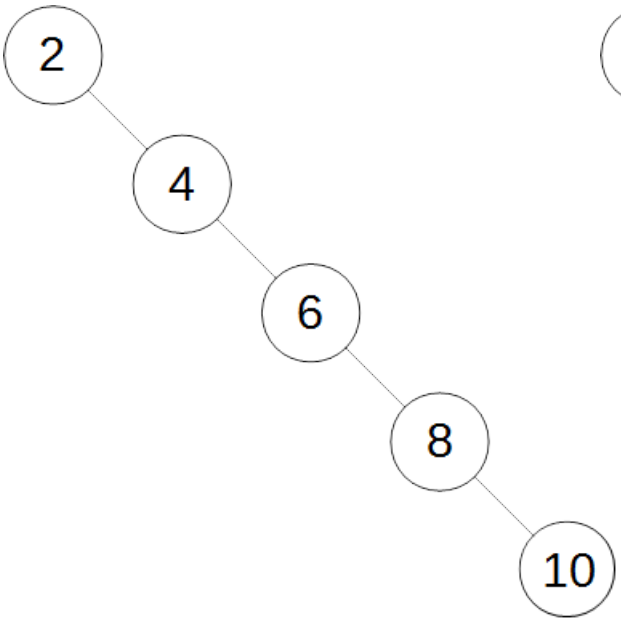
root 1



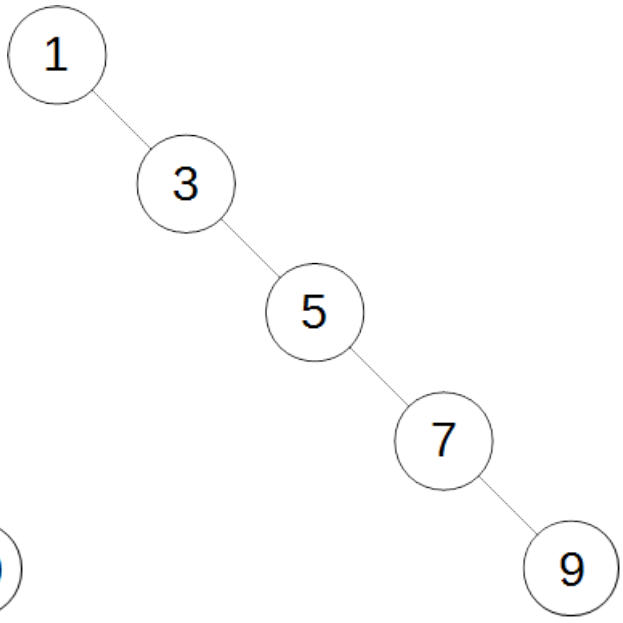
root 2



root 1



root 2



6 Результаты работы программы

Результаты запуска программы с различными входными значениями.

```
Введите узел первого дерева: 7
Введите узел второго дерева: 7
Введите узел первого дерева: 9
Введите узел второго дерева: 9
Введите узел первого дерева: 11
Введите узел второго дерева: 11
Введите узел первого дерева: 8
Введите узел второго дерева: 8
Введите узел первого дерева: 5
Введите узел второго дерева: 5
Введите узел первого дерева: ^Z
Введите узел второго дерева:

Первое дерево:  5 7 8 9 11
Второе дерево:  5 7 8 9 11

Деревья равны
```

```
Введите узел первого дерева: 16
Введите узел второго дерева: 16
Введите узел первого дерева: 21
Введите узел второго дерева: 21
Введите узел первого дерева: 30
Введите узел второго дерева: 11
Введите узел первого дерева: 11
Введите узел второго дерева: 13
Введите узел первого дерева: 9
Введите узел второго дерева: 9
Введите узел первого дерева: ^Z
Введите узел второго дерева:

Первое дерево:  9 11 16 21 30
Второе дерево:  9 11 13 16 21

Деревья не равны
```

```
Введите узел первого дерева: 2
Введите узел второго дерева: 1
Введите узел первого дерева: 4
Введите узел второго дерева: 3
Введите узел первого дерева: 6
Введите узел второго дерева: 5
Введите узел первого дерева: 8
Введите узел второго дерева: 7
Введите узел первого дерева: 10
Введите узел второго дерева: 9
Введите узел первого дерева: ^Z
Введите узел второго дерева:

Первое дерево:  2 4 6 8 10
Второе дерево:  1 3 5 7 9

Деревья не равны
```