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;

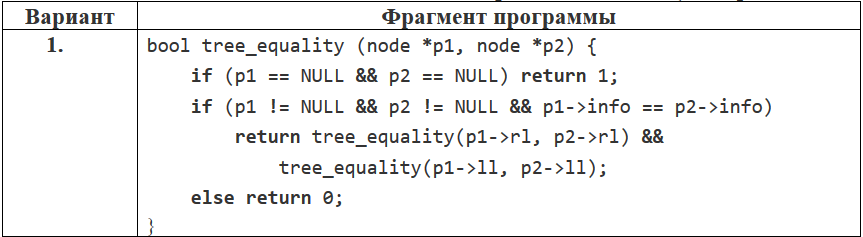
2. прокомментировать смысл формальных параметров и возвращаемого значения, а также ход решения задачи;

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.



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**

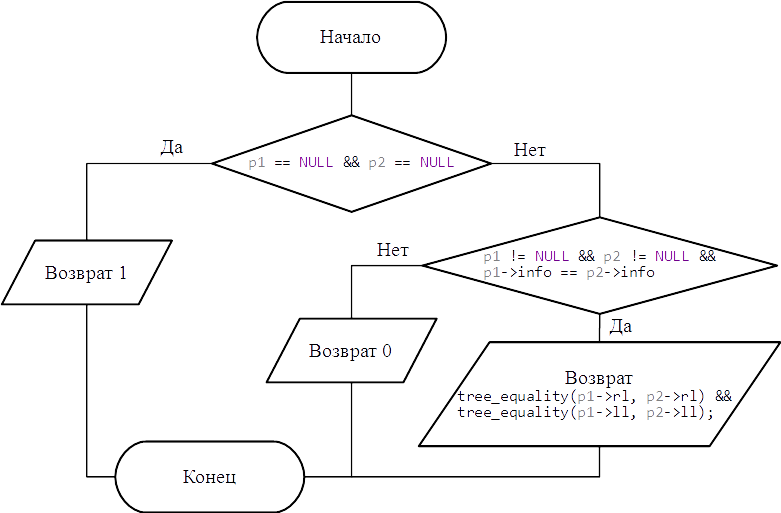
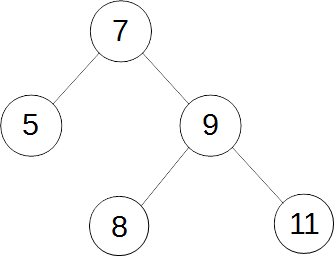
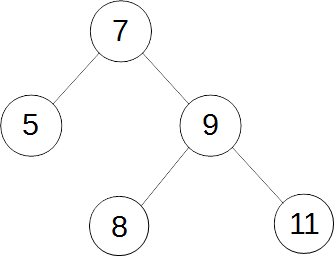


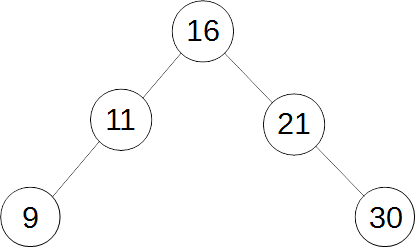
Рисунок 1 – функция bool tree\_equality(node \*p1, node \*p2)

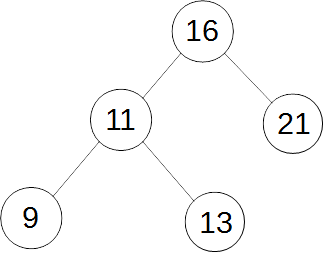
**4 Код программы**

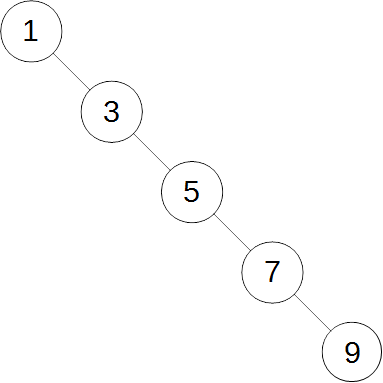
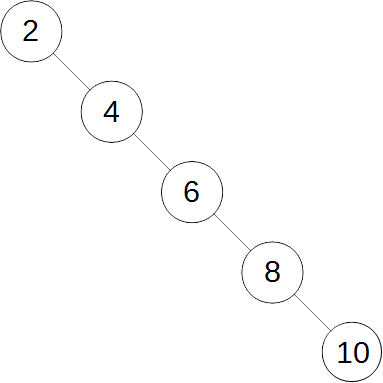
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 28 29 30 31 32 33 34 35 36 37 38 39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83 | #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  node \*tree(node \*p, int w) {  if (p == NULL) {  p = new node;  p->info = w;  p->ll = nullptr;  p->rl = nullptr;  }  else {  if (w < p->info) {  p->ll = tree(p->ll, w);  }  else {  p->rl = tree(p->rl, w);  }  }  return p;  }  //Функция обхода дерева  void treeprint(node \*p) {  if (p != NULL) {  treeprint(p->ll);  cout << " " << p->info;  treeprint(p->rl);  }  }  int main(){  setlocale(LC\_ALL, "");  int t1, t2;  node \*root1 = nullptr, \*root2 = nullptr;  cout << "Введите узел первого дерева: ";  cin >> t1;  cout << "Введите узел второго дерева: ";  cin >> t2;  while (cin) {  root1 = tree(root1, t1);  root2 = tree(root2, t2);  cout << "Введите узел первого дерева: ";  cin >> t1;  cout << "Введите узел второго дерева: ";  cin >> t2;  }  cout << "\n\nПервое дерево: ";  treeprint(root1);  cout << "\nВторое дерево: ";  treeprint(root2);  if (tree\_equality(root1, root2)) {  cout << "\n\nДеревья равны" << endl;  }  else {  cout << "\n\nДеревья не равны" << endl;  }  return 0;  } |

**5 Наборы тестовых данных**

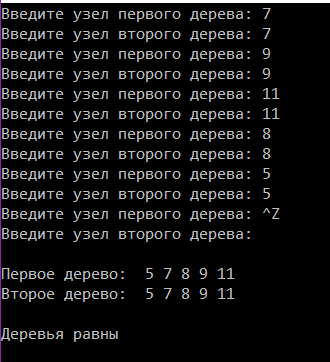
**** root 1 root 2

**** root 1 root 2



**** root 1 root 2

**6 Результаты работы программы**

Результаты запуска программы с различными входными значениями.

