Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Алтайский государственный технический университет

им. И.И. Ползунова»

Факультет (институт) Информационных технологий

Кафедра Прикладная математика

Отчет защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_

А.И.Потупчик

(подпись преподавателя) (инициалы, фамилия)

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Отчет

по Расчетному заданию

**«Бинарные деревья»**

по дисциплине Типы и структуры данных

(наименование дисциплины)

РЗ 09.03.04.21.000 ОТ

(обозначение документа)

Студент группы ПИ-02 Р.А.Чередов

(инициалы, фамилия)

Преподаватель доцент, доцент А.И.Потупчик

(должность, ученое звание) (инициалы, фамилия)

Барнаул 2022

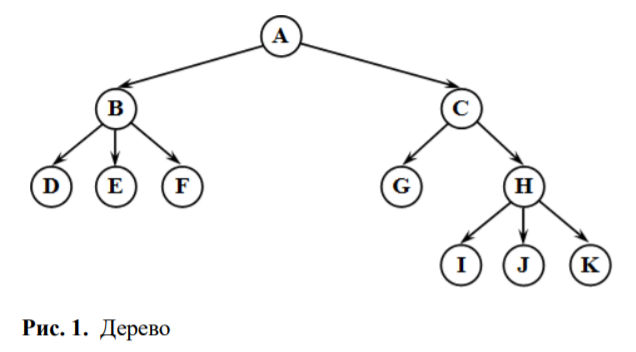
Вариант 4

Задания

1. Разработать и отладить программу на языке C++, реализующую работу с деревом в соответствии с вариантом. Выполнить оценку временной и емкостной сложности программы.
2. Исходные данные поместить в файл input.dat
3. Результаты вывести на экран. Исходные данные и результаты вывести также в выходной файл output.dat
4. Предусмотреть вывод исходного и выходного деревьев в табличной форме или в виде рисунка. Вывод в табличной форме означает вывод в прямом порядке, т.е. Корень, Левое поддерево, Правое поддерево.

Задание в соответствии с вариантом

1. Проверить, содержится ли второе **дерево** как **поддерево** в первом дереве, и если содержится, определить **уровень**, на котором находится в первом дереве **узел**, содержащий **ключ**, равный ключу корня второго дерева.

**Дерево** – это структура данных, представляющая собой совокупность элементов, образующих иерархическую структуру (рис. 1). 

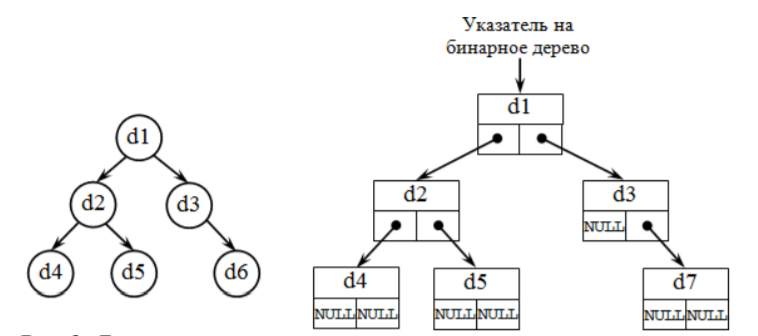
**Поддерево** – часть древовидной структуры данных, которая может быть представлена в виде отдельного дерева.

**Вершиной** **(узлом)** дереваназываетсякаждый элемент дерева.

Вершины дерева соединены направленными дугами, которые называют **ветвями** дерева.

Начальный узел дерева называют **корнем** дерева, ему соответствует нулевой **уровень**. Листьями дерева называют вершины, в которые входит одна ветвь и не выходит ни одной ветви.

**Бинарное (двоичное)** дерево – это динамическая структура данных, представляющая собой дерево, в котором каждая вершина имеет не более двух потомков (рис. 2). Таким образом, бинарное дерево состоит из элементов, каждый из которых содержит информационное поле и не более двух ссылок на различные бинарные поддеревья. На каждый элемент дерева имеется ровно одна ссылка.



**Рис. 2.** Бинарное дерево и его организация

Каждая вершина бинарного дерева является структурой, состоящей из четырех видов полей. Содержимым этих полей будут соответственно:

• информационное поле (**ключ** вершины);

• служебное поле (их может быть несколько или ни одного);

• указатель на левое поддерево;

• указатель на правое поддерево.

Текст программы:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <fstream>

#include <sstream>

#include <conio.h>

#include <string>

using namespace std;

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

//Создание нового узла

struct Node\* newNode(int item) {

struct Node\* temp = (struct Node\*)malloc(sizeof(struct Node));

temp->data = item;

temp->left = temp->right = NULL;

return temp;

}

//Добавление нового узла

struct Node\* insert(Node\* t, int x)

{

// Возвращаем новый узел, если дерево пустое

if (t == NULL) return newNode(x);

// Проходим в нужное место и вставляет узел

if (x < t->data)

t->left = insert(t->left, x);

else

t->right = insert(t->right, x);

return t;

}

// поиск в бинарном дереве

Node\* find(Node\* t, int x, int& r) {

r++;

if (t == NULL)

return NULL;

else if (x < t->data)

return find(t->left, x, r);

else if (x > t->data)

return find(t->right, x, r);

else

return t;

}

//Поиск корня из второго дерева в узлах первого

Node\* Nach(Node\* root1, Node\* root2) {

if (root1 == NULL)

return NULL;

else if (root1->data == root2->data)

return root1;

else if (root1->data > root2->data)

Nach(root1->left, root2);

else

Nach(root1->right, root2);

}

//Проверка, содержится ли второе дерево в первом

int Sravn(Node\* root1, Node\* root2) {

if (root2 == NULL)

return -1;

if (root1 == NULL && root2 != NULL)

return 0;

if (root1->data == root2->data) {

if (Sravn(root1->left, root2->left) && Sravn(root1->right, root2->right))

return 1;

}

else

return 0;

}

// проверка пустоты бинарного дерева

int isempty(Node\* root1, Node\* root2)

{

if (root1 == NULL && root2 == NULL)

{

return 0;

}

else

return 1;

}

//печать бинарного дерева в виде дерева повернутого

void Pr90(Node\* t, int l) {

if (t == NULL)

return;

Pr90(t->right, l + 1);

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

cout << "\t";

cout << t->data << endl;

Pr90(t->left, l + 1);

}

//печать бинарного дерева повернутого в файл

void Pr90File(Node\* t, ofstream& fout, int l) {

if (t == NULL)

return;

Pr90File(t->right, fout, l + 1);

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

fout << "\t";

fout << t->data << endl;

Pr90File(t->left, fout, l + 1);

}

int main()

{

setlocale(LC\_ALL, "Russian");

struct Node\* root1 = NULL;//1е дерево

struct Node\* root2 = NULL;//2е дерево

string buf1, buf2; //Строки из файла

int intbuf; //Цифра строки

int r = 0; //Для определения уровня

string word; //слово строки

stringstream sstream; //строковый поток для её разделения на слов

ifstream fin; //файл для чтения

ofstream fout; //файл для записи

fin.open("input.dat");

getline(fin, buf1);

sstream << buf1;

while (sstream >> word) {

intbuf = stoi(word); //перевод слова в число

root1 = insert(root1, intbuf);

}

getline(fin, buf2);

sstream.clear();

sstream << buf2;

while (sstream >> word) {

intbuf = stoi(word); //перевод слова в число

root2 = insert(root2, intbuf);

}

fin.close();

cout << "Первое дерево:\n\n";

Pr90(root1, 0);

cout << "Второе дерево:\n\n";

Pr90(root2, 0);

sstream.clear();

sstream << buf2;

sstream >> word;

intbuf = stoi(word); //перевод слова в число

find(root1, intbuf, r);

if (Sravn(Nach(root1, root2), root2))

printf("\n\n 2е дерево содержится в 1м\n Уровень %d", r - 1);

else

printf("\n\n 2е дерево не содержится в 1м");

fout.open("output.dat");

fout << "Исходные данные:\n";

fout << buf1;

fout << endl;

fout << buf2;

fout << "\n\nПервое дерево:\n\n";

Pr90File(root1, fout, 0);

fout << "Второе дерево:\n\n";

Pr90File(root2, fout, 0);

if (Sravn(Nach(root1, root2), root2))

fout << "\n\n 2е дерево содержится в 1м\n Уровень " << r - 1;

else

fout << "\n\n 2е дерево не содержится в 1м";

fout.close();

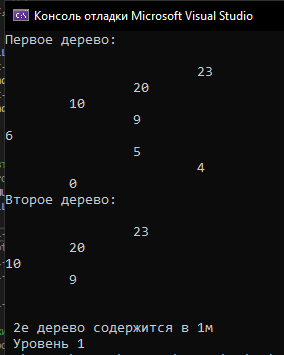
return 0;

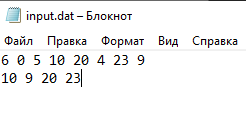
}

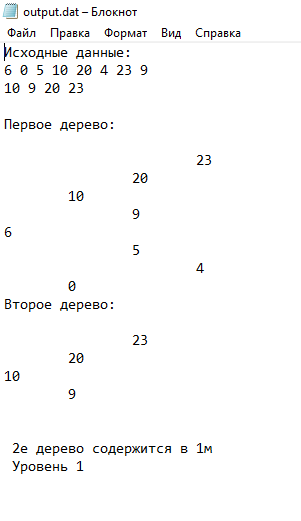
Тест программы:

1е дерево: 6 0 5 10 20 4 23 9

2е дерево: 10 9 20 23



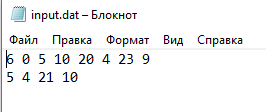


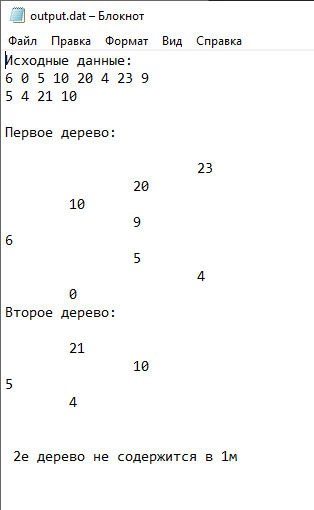


Тест программы:

1е дерево: 6 0 5 10 20 4 23 9

2е дерево: 5 4 21 10





Тест программы:

1е дерево: 6 0 5 10 20 4 23 9

2е дерево: 3 10 9 20

