МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

 Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

Отчет по лабораторной работе №3

по дисциплине

Алгоритмы и структуры данных

«Рекурсия. Графы. Деревья.»

Вариант 21

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Капранов С.Н.

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сухоруков В.А.

19-ИВТ-3

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2020

# Текст задачи

Дано N-дерево. Найти поддерево, не включающее ни одну из заданных вершин.

# Код библиотеки MyTree.h

#pragma once

#ifndef \_MyTree\_H\_

#define \_MyTree\_H\_

#include<iostream>

#include<iomanip>

#define N 5 //Определение максимального количества сыновей

using namespace std;

// Класс узла дерева

template <typename T>

struct Node {

T key; //Значение узла

int number; //Номер узла

Node\* child[N]; //Указатель на массив сыновей

Node\* parent; //Указатель на родителя

int height; //Высота узла

int count\_child; //Текущее количество детей

Node() {

parent = nullptr;

count\_child = 0;

key = T();

height = 0;

number = 0;

}

~Node() {}

};

int num = 0; //Счётчик узлов

/\*Функция для создания узла

Параметры функции:

1)Значение узла

Результат - указатель на узел

\*/

template <typename T>

Node<T>\* newNode(T key) {

num++;

Node<T>\* temp = new Node<T>;

temp->key = key;

temp->number = num;

for (int i = 0; i < N; i++) {

temp->child[i] = NULL;

}

return temp;

}

/\* Функция для добавления узла в дерево по индексу родителя

Параметры функции:

1)Указатель на корень дерева

2)Индекс родительского узла

3)Значение узла

\*/

template <typename T>

void push(Node<T>\* PNode, int index, T key) {

if (PNode->number == index) {

if (PNode->count\_child == N) {

cout << "\u001B[31mУ этого узла максимальное

количество сыновей" << endl;

}

else {

if (PNode->parent == nullptr) {

PNode->height = 1;

}

PNode->child[PNode->count\_child] = newNode(key);

PNode->child[PNode->count\_child]->parent = PNode;

PNode->child[PNode->count\_child]->height = PNode->height + 1;

PNode->count\_child++;

}

}

for (int i = 0; i < PNode->count\_child; i++) {

push(PNode->child[i], index, key);

}

}

/\*Функция заполнения дерева

Параметры функции:

1)Указатель на корень дерева

2)Количество узлов

\*/

template <typename T>

void filling(Node<T>\* PNode, int count) {

int p\_i;

T k;

for (int i = 1; i < count + 1; i++) {

cout << "\u001B[32m\n\nДобавление узла " << i + 1

<< "\u001B[36m\n\tВведите номер узла родителя ";

cin >> p\_i;

if (p\_i > num) {

cout << "\u001B[31mУзел с таким номером ещё не записан\n";

i--;

}

else {

cout << "\tВведите значение узла ";

cin >> k;

push(PNode, p\_i, k);

}

}

}

/\*Функция удаления дерева

Параметры функции:

1)Указатель на корень дерева

\*/

template <typename T>

void deletion(Node<T>\* PNode) {

for (int i = 0; i < PNode->count\_child; i++) {

deletion(PNode->child[i]);

}

delete PNode;

}

/\*Перегрузка оператора << для вывода содержимого узла

Параметры функции:

1)Ссылка на поток вывода

2)Указатель на узел

Результат- ссылка на поток вывода

\*/

template <typename T>

ostream& operator<<(ostream& os, Node<T>\* PNode) {

if (PNode->parent != nullptr) {

os << "\u001B[32m\n\nНомер вершины: " << PNode->number

<<"\u001B[33m\n\tЗначение вершины: " << PNode->key

<< "\n\tВысота вершины: " << PNode->height

<< "\n\tНомер родителя: " << PNode->parent->number;

if (PNode->count\_child != 0) {

os << "\n\tНомера вершин сыновей: ";

for (int i = 0; i < PNode->count\_child; i++) {

os << PNode->child[i]->number << " ";

}

}

else {

os << "\n\tУ этого узла нет сыновей";

}

}

else {

os << "\u001B[32m\n\nНомер вершины: " << PNode->number

<< "\u001B[33m\n\tЗначение вершины: " << PNode->key

<< "\n\tВысота вершины: " << PNode->height;

if (PNode->count\_child != 0) {

os << "\n\tНомера вершин сыновей: ";

for (int i = 0; i < PNode->count\_child; i++) {

os << PNode->child[i]->number << " ";

}

}

else {

os << "\n\tУ этого узла нет сыновей";

}

}

return os;

}

/\*Функция для вывода дерева

Параметры функции:

1)Указатель на корень дерева

\*/

template <typename T>

void Print(Node<T>\* PNode) {

cout << PNode;

for (int i = 0; i < PNode->count\_child; i++) {

Print(PNode->child[i]);

}

}

#endif

# Код программы

/\*Условие задачи:

Дано N-дерево. Найти поддерево не включающее ни одну из заданных вершин.

\*/

#include<iostream>

#include"MyTree.h"

#include<vector>

using namespace std;

/\*Функция для нахождения поддерева без заданных вершин

Параметры функции:

1)Указатель на корень дерева

2)Вектор номеров заданных узлов

\*/

template <typename T>

void find\_subtree(Node<T>\* PNode, vector<int>numb) {

for (int i = 0; i < PNode->count\_child; i++) {

find\_subtree(PNode->child[i], numb);

}

for (int j = 0; j < numb.size(); j++) {

if (PNode->number == numb[j]) {

Node<T>\* cur = PNode->parent;

int i = 0;

while (cur->child[i] != PNode) {

i++;

}

for (i; i < cur->count\_child - 1; i++) {

cur->child[i] = cur->child[i + 1];

}

cur->count\_child--;

}

}

}

int main() {

setlocale(LC\_ALL, "ru"); //Включение русского языка

Node<int>\* root; //Указатель на корень дерева

int count; //Количество узлов

int root\_key; //Значение корня

int n; //Количество вершин, которых не

//должно быть в поддереве

vector<int> numbers; //Номера вершин, которых не должно

//быть в поддереве

cout << "\u001B[36mВведите количество узлов в дереве ";

cin >> count;

cout << "Узлы номеруются от 1 до " << count

<< "\n\nВведите значние корня дерева ";

cin >> root\_key;

root = newNode(root\_key);

filling(root, count - 1); //Заполнение дерева

cout << endl << "\u001B[36mСформированное дерево: " << endl;

Print(root);

while (n == 0) {

cout << "\n\u001B[36mВведите количество вершин, которых не должно

быть в поддереве ";

cin >> n;

if (n > count) {

cout << "\u001B[31m В дереве меньше вершин";

n = 0;

}

}

for (int i = 1; i <= n; i++) { //Формирование вектора номеров

cout << endl << "\u001B[32mВведите номер " << i << " вершины ";

int a;

cin >> a;

if (a > count) {

cout << "\u001B[31mВершины с таким номером в дереве нет.

Повторите ввод.\n";

i--;

}

numbers.push\_back(a);

}

cout << "\u001B[36m\n\nСформированное поддерево: \n" ;

find\_subtree(root, numbers);

Print(root);

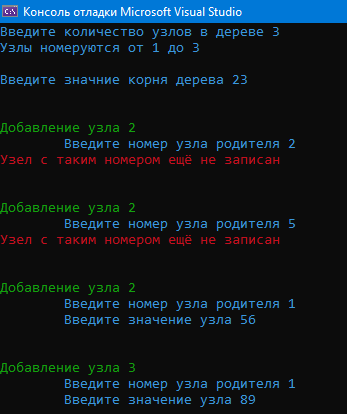
deletion(root); //Освобождение памяти

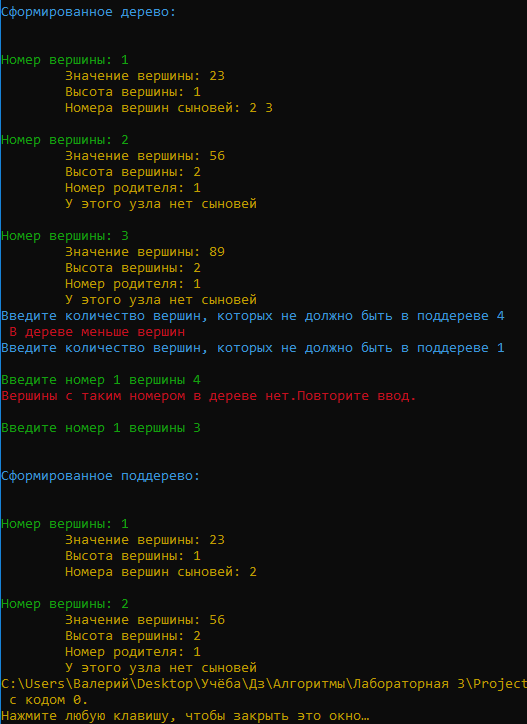
return 0;

}

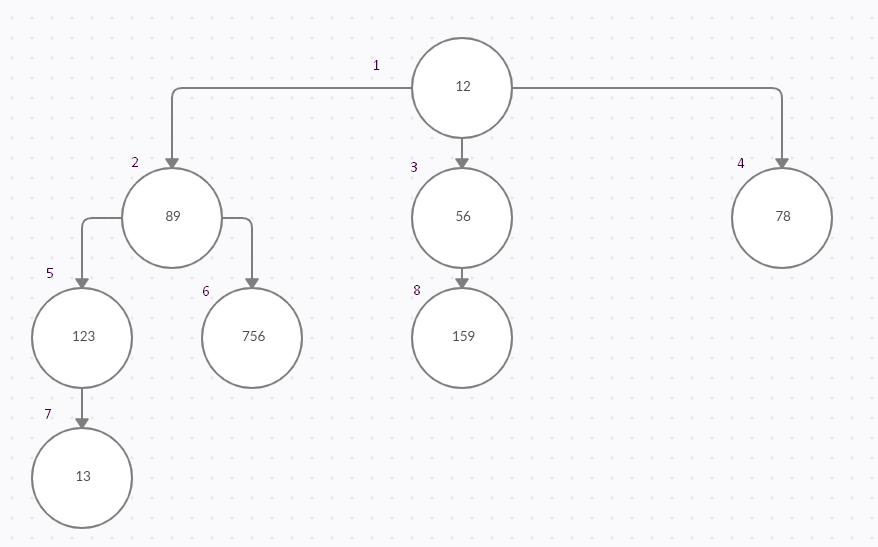
# Результат работы программы

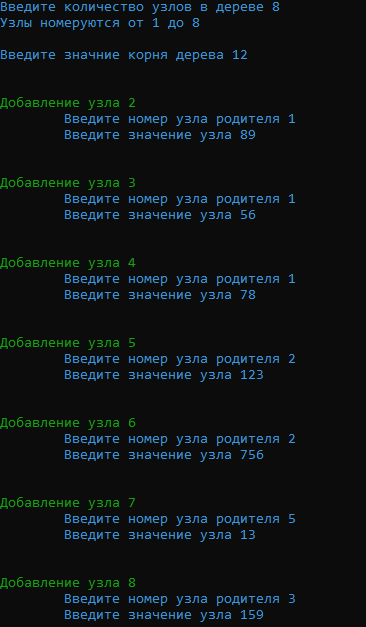
## Ввод не корректных данных

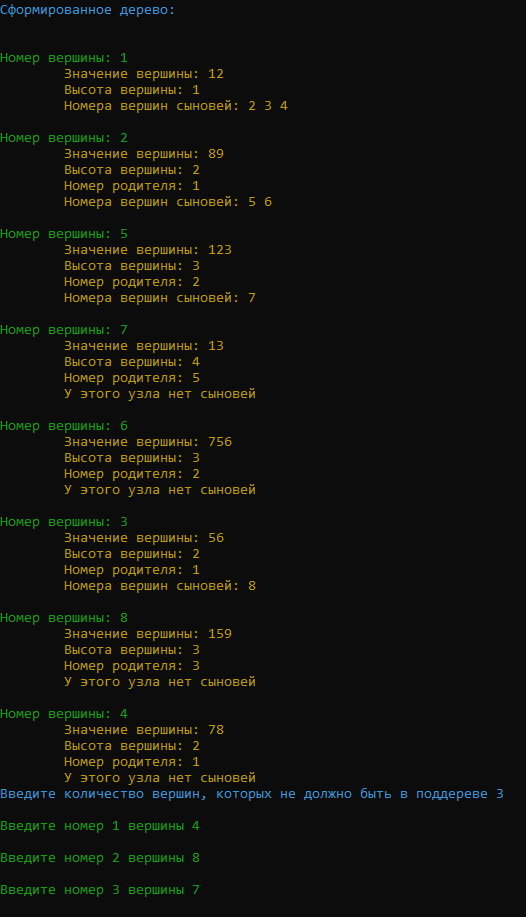


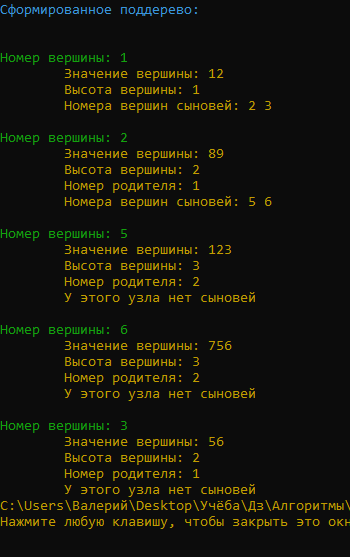


## Удаление из дерева вершин, не имеющих сыновей









## Удаление вершин, имеющих сыновей

