Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

**Графы**

**Отчет по лабораторной работе №7**

**По дисциплине**

**«Структуры и алгоритмы обработки данных в ЭВМ»**

Выполнил: студент гр. 439-1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Зозуля Е.Д.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Проверил: ассистент каф. АСУ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Яблонский Я.В.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Томск 2021

# Задание на лабораторную работу

Вариант 2.

# Напишите программу, которая с помощью метода поиска в глубину, строит стягивающее дерево для произвольного связного неориентированного графа. Для представления графа в программе использовать списки смежности. Данные о графе вводятся из файла. Программа должна вывести ребра графа, вошедшие в стягивающее дерево. После завершения работы с динамическими структурами данных необходимо освободить занимаемую ими память.

# Алгоритм решения задачи

Выполняется обход по графу в глубину, ищутся не пройдённые вершины затем отмечаются. Для пройденных вершин обход не вызывается для остальных рекурсивно идёт по смежным

# Листинг программы

Lab7.cpp

#include <iostream>

#include <vector>

#include <fstream>

using namespace std;

int n; // количество вершин в орграфе

vector<int>\* adj; // список смежности

vector<bool> used; // пройденные и не пройденные вершины

// функция обхода в глубину

void dfs(int a) {

// если вершину уже проходили, то функцию не вызываем

if (used[a]) {

return;

}

used[a] = true; // маркируем вершину как пройденную

cout << a << " -> "; // выводим вершину

// обход всех вершин смежных с вершиной v

for (int i = 0; i < adj[a].size(); i++) {

int b = adj[a][i];

dfs(b); // рекурсивный обход от вершины a, смежной с вершиной b

}

}

int main()

{

setlocale(LC\_CTYPE, "RUSSIAN");

// чтение данных из файла

ifstream Filein("graph.txt"); // открываем поток для чтения из файла graph.txt

Filein >> n; // считываем количество вершин

adj = new vector<int>[n]; //инициализируем список смежности размерности n

cout << "Кол-во вершин:\n" << n << endl << endl;

cout << "Список рёбер: " << endl;

//считываем граф из файла

while (Filein) {

int a, b; // инициализируем переменные

Filein >> a >> b; // считываем из файла значения вершин

cout << a << " " << b << endl;

// добавляем ребра a и b в граф

adj[a].push\_back(b);

adj[b].push\_back(a);

}

used.assign(n, false);

cout << "\nСтягивающее дерево: ";

for (int a = 0; a < n; a++) {

dfs(a);

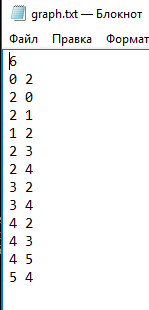
}

return 0;

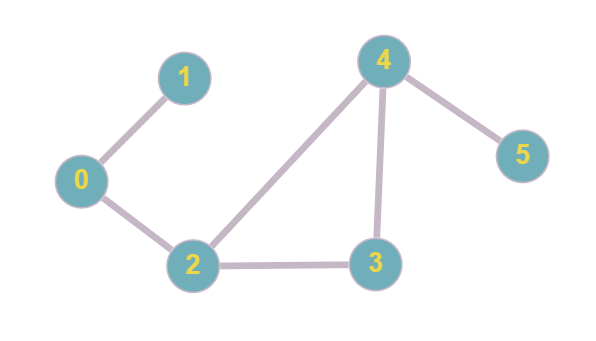
}

# Пример решения

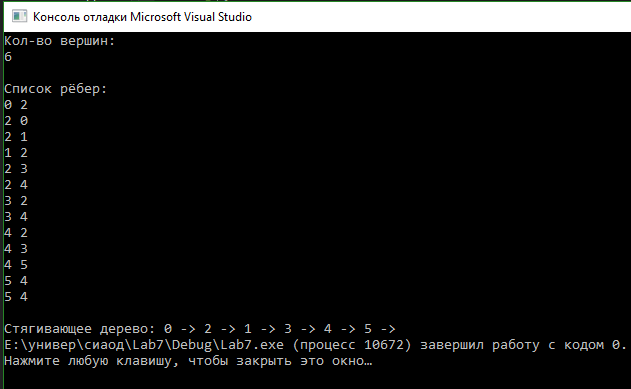
Скриншоты успешной работы программы, входные данные тестового примера, выходные данные на рисунке 4.1, 4.2 и 4.3.



*Рисунок 4.1.- Входные данные*



*Рисунок 4.3.- граф, построенный по спискам смежности*



*Рисунок 4.2.- Скриншот успешной работы программы*

# Вывод

Был изучен обход по графу в глубину и использован для постройки стягивающего дерева.