МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**Липецкий Государственный Технический Университет**

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

Лабораторная работа

по технологиям программирования №5

# “ Программирование алгоритмов реализации и обработки графов”

Студент \_\_\_\_\_\_\_\_\_\_\_\_ Станиславчук С.М.

(подпись, дата)

Группа АС-21-1

Руководитель

Доцент, кандидат наук \_\_\_\_\_\_\_\_\_\_\_\_ Хромов А. М.

(подпись, дата)

Липецк 2022 г.

**Содержание:**

1. Титульный лист.

2. Задание кафедры, соответствующее варианту, номер варианта.

3. Цель работы.

4. Блок-схема алгоритма разработанной программы

5. Текст программы

6. Контрольный пример

7. Выводы по работе.

Цель работы

Получение навыков реализации алгоритмов обхода графов и построения минимальных остовных деревьев.

Задание

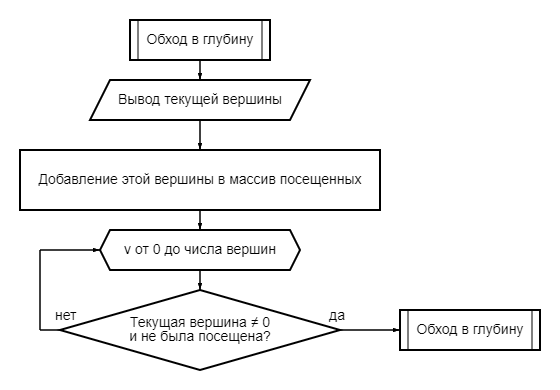
Осуществить программную реализацию алгоритма обработки графа, выбрав из табл. 3 приложения алгоритм и начальные параметры, в соответствии с заданным номером варианта. В программе необходимо предусмотреть удобный пользовательский интерфейс, позволяющий создавать граф (пользователем или генерировать автоматически), выполнять требуемый алгоритм и визуализировать этапы выполнения алгоритма и результаты.

Алгоритм: обход в глубину

Граф: неориентированный

Степень графа: 4

3. Блок-схема алгоритма



4. Код программы

#include <iostream>

#include <audiere.h>

#include <conio.h>

using namespace std;

using namespace audiere;

const int n = 5;

int i, j;

bool\* visited = new bool[n];

int graph[n][n]; //матрица смежности графа

void customGenerate() {

cout << "Вводи матрицу смежности графа: \n";

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

{

visited[i] = false;

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

cin >> graph[i][j];

}

}

}

void randomGenerate(){

cout << "Рандомная матрица смежности графа: \n";

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

{

visited[i] = false;

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

graph[i][j] = 0 + rand() % 2;

cout << graph[i][j] << " ";

}

cout << endl;

}

}

void DFS(int st) //поиск в глубину

{

cout << st + 1 << " ";

visited[st] = true;

for (int v = 0; v <= n; v++)

if ((graph[st][v] != 0) && (!visited[v]))

DFS(v);

}

void main()

{

AudioDevicePtr device = OpenDevice();

OutputStreamPtr sound = OpenSound(device, "D:\\main1.mp3", true);

OutputStreamPtr sound1 = OpenSound(device, "D:\\func1.mp3", true);

OutputStreamPtr sound2 = OpenSound(device, "D:\\func2.mp3", true);

OutputStreamPtr sound3 = OpenSound(device, "D:\\goodending.mp3", true);

sound->play();

sound->setVolume(3.0f); sound1->setVolume(3.0f); sound2->setVolume(3.0f);

srand(time(NULL)); //долгожданный рандом

setlocale(LC\_ALL, "Rus");

int start; bool flag = true; size\_t N;

while (flag == true) {

cout << "Хотите вписать свою матрицу смежности или предпочтёте довериться рандому?";

cout << "\n1. Хочу сам!";

cout << "\n2. Время рандома!\n";

cin >> N;

if (N == 1) {

sound->stop();

sound1->play();

customGenerate();

cout << "\nМатрица введена";

flag = false;

}

if (N == 2) {

sound->stop();

sound2->play();

randomGenerate();

flag = false;

}

}

cout << "\nСтартовая вершина >> "; cin >> start;

bool\* vis = new bool[n]; //массив посещенных вершин

cout << "Порядок обхода: ";

DFS(start - 1);

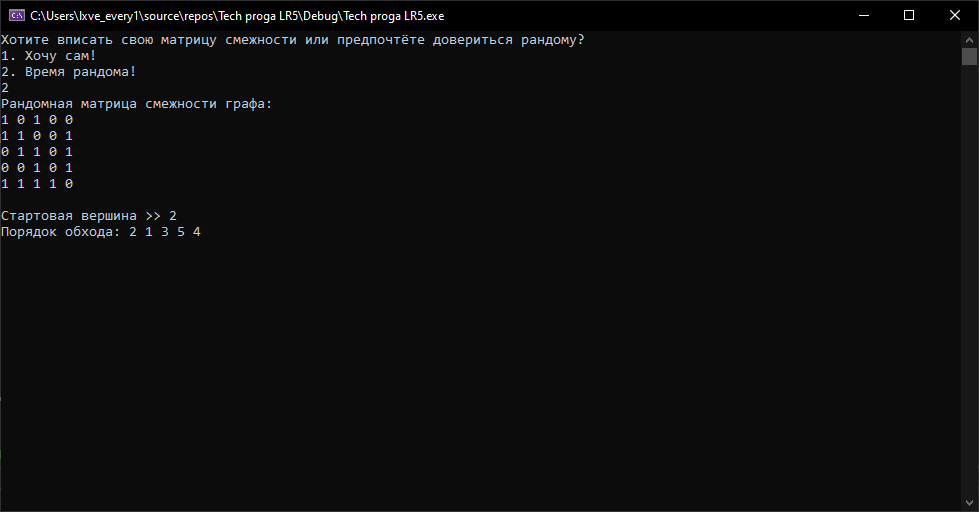
sound1->stop(); sound2->stop();

sound3->play();

\_getch();

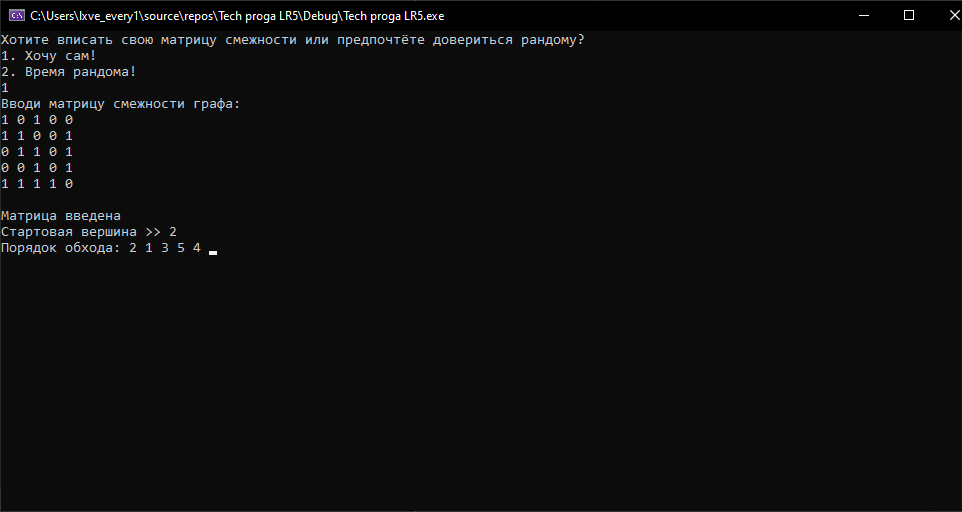
delete[] visited; // очищаем память

}

5. Тест программы.

1. Выбираем нужную функцию генерации матрицы смежности.  
2. Задаем вершину.  
3. Смотрим результат.

Сделаю то же самое, только теперь матрицу задам сам – точно такую же, чтобы проверить ответ:



6. Вывод: получил навыки реализации алгоритмов обхода графов и построения минимальных остовных деревьев.