Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №5

по теме «Определение характеристик графов»

Выполнили:

Студенты группы 21ВВ2

Назаров Е.А.

Макаров И.С.

Принял:

Митрохин М.А.

Юрова О.В.

Пенза, 2022

**Цель работы:**

Изучить основы теоретико-множественного и графического представлений графов, простейших свойств графов, получить практический навык задания и визуализации графа на плоскости.

**Лабораторные работы:**

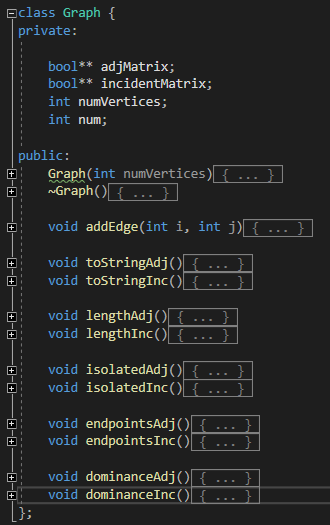
**Задание 1**

1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного взвешенного графа *G*. Выведите матрицу на экран.
2. Определите размер графа *G*, используя матрицу смежности графа.
3. Найдите изолированные, концевые и доминирующие вершины.

**Задание 2\***

1. Постройте для графа G матрицу инцидентности.
2. Определите размер графа *G*, используя матрицу смежности графа.
3. Найдите изолированные, концевые и доминирующие вершины.

**Пояснительный текст к программе:**

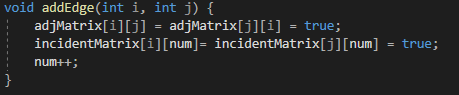


Для выполнения заданий был реализован класс Graph.

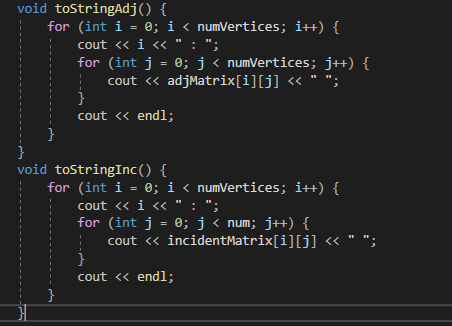
adjMatrix-двухмерная матрица, для работы с матрицей смежности. incidentMatrix-двухмерная матрица, для работы с матрицей инцидентности. Переменная numVerties хранит в себе кол-во вершин, переменная num хранит в себе кол-во рёбер.



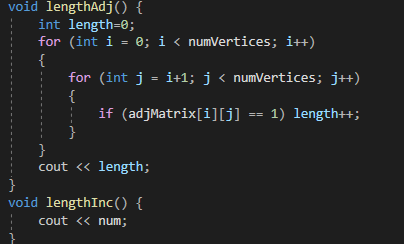
В конструкторе и деструкторе выделяется и освобождается динамическая память для матриц.



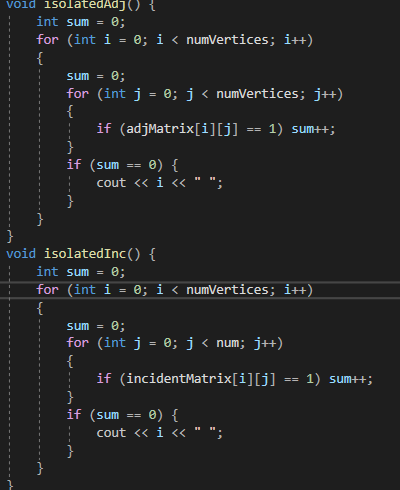
Функция добавления связей между вершинами для обеих матриц.



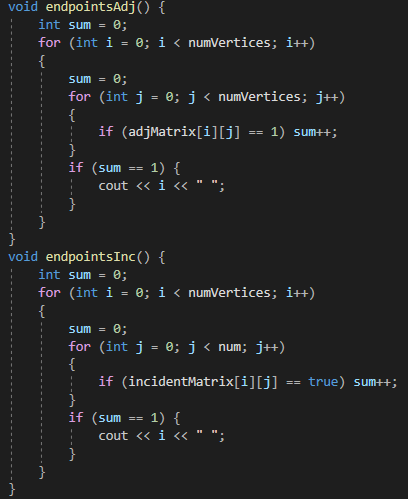
Вывод на консоль матрицы смежности(функция toStringAdj) и матрицы инцидентности(функция toStringInc).



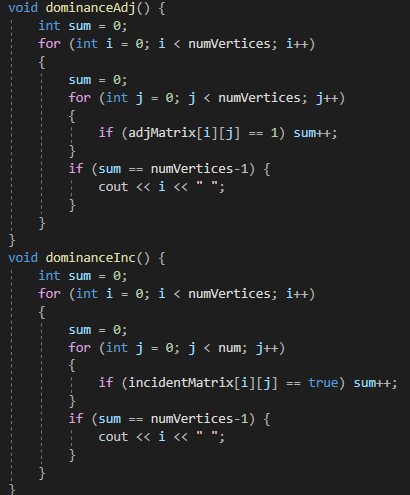
Функции подсчета размера графа используя матрицу смежности и матрицу инцидентности.



Функции нахождения изолирующих вершин в графе используя матрицу смежности и матрицу инцидентности.

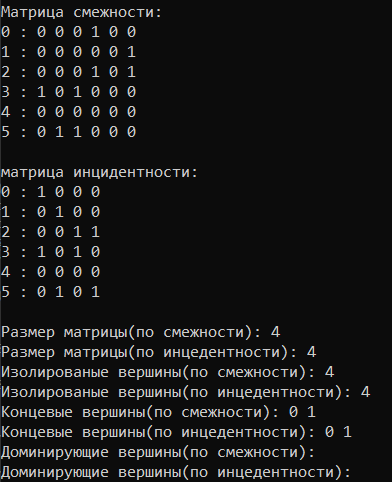


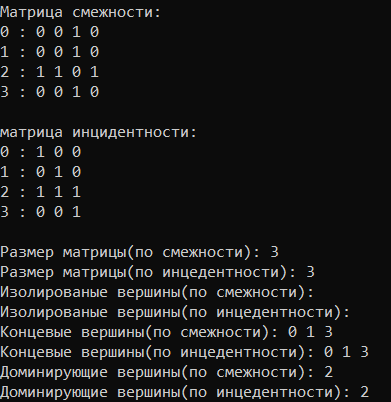
Функции нахождения концевых вершин в графе используя матрицу смежности и матрицу инцидентности.



Функции нахождения доминирующих вершин в графе используя матрицу смежности и матрицу инцидентности.

**Результат работы программы:**





**Листинг:**

**Lab6**

#include <iostream>

#include <string>

using namespace std;

class Graph {

private:

bool\*\* adjMatrix;

bool\*\* incidentMatrix;

int numVertices;

int num;

public:

Graph(int numVertices)

{

this->numVertices = numVertices;

incidentMatrix = new bool\* [numVertices];

adjMatrix = new bool\* [numVertices];

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

incidentMatrix[i] = new bool[numVertices \* numVertices];

adjMatrix[i] = new bool[numVertices];

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

adjMatrix[i][j] = false;

}

for (int j = 0; j < numVertices\* numVertices; j++) {

incidentMatrix[i][j] = false;

}

}

}

~Graph() {

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

delete[] adjMatrix[i];

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

delete[] incidentMatrix[i];

delete[] adjMatrix;

delete[] incidentMatrix;

}

void addEdge(int i, int j) {

adjMatrix[i][j] = adjMatrix[j][i] = true;

incidentMatrix[i][num]= incidentMatrix[j][num] = true;

num++;

}

void toStringAdj() {

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

cout << i << " : ";

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

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

}

cout << endl;

}

}

void toStringInc() {

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

cout << i << " : ";

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

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

}

cout << endl;

}

}

void lengthAdj() {

int length=0;

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

{

for (int j = i+1; j < numVertices; j++)

{

if (adjMatrix[i][j] == 1) length++;

}

}

cout << length;

}

void lengthInc() {

cout << num;

}

void isolatedAdj() {

int sum = 0;

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

{

sum = 0;

for (int j = 0; j < numVertices; j++)

{

if (adjMatrix[i][j] == 1) sum++;

}

if (sum == 0) {

cout << i << " ";

}

}

}

void isolatedInc() {

int sum = 0;

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

{

sum = 0;

for (int j = 0; j < num; j++)

{

if (incidentMatrix[i][j] == 1) sum++;

}

if (sum == 0) {

cout << i << " ";

}

}

}

void endpointsAdj() {

int sum = 0;

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

{

sum = 0;

for (int j = 0; j < numVertices; j++)

{

if (adjMatrix[i][j] == 1) sum++;

}

if (sum == 1) {

cout << i << " ";

}

}

}

void endpointsInc() {

int sum = 0;

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

{

sum = 0;

for (int j = 0; j < num; j++)

{

if (incidentMatrix[i][j] == true) sum++;

}

if (sum == 1) {

cout << i << " ";

}

}

}

void dominanceAdj() {

int sum = 0;

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

{

sum = 0;

for (int j = 0; j < numVertices; j++)

{

if (adjMatrix[i][j] == 1) sum++;

}

if (sum == numVertices-1) {

cout << i << " ";

}

}

}

void dominanceInc() {

int sum = 0;

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

{

sum = 0;

for (int j = 0; j < num; j++)

{

if (incidentMatrix[i][j] == true) sum++;

}

if (sum == numVertices-1) {

cout << i << " ";

}

}

}

};

int main() {

srand(time(NULL));

setlocale(0, "");

int z = 4;

Graph c(z);

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

{

for (int j = i + 1; j < z; j++)

{

if (bool(rand() % 2))

c.addEdge(i, j);

}

}

cout << endl << "Матрица смежности: \n";

c.toStringAdj();

cout << endl << "матрица инцидентности: \n";

c.toStringInc();

cout << endl << "Размер матрицы(по смежности): ";

c.lengthAdj();

cout << endl << "Размер матрицы(по инцедентности): ";

c.lengthInc();

cout << endl << "Изолированые вершины(по смежности): ";

c.isolatedAdj();

cout << endl << "Изолированые вершины(по инцедентности): ";

c.isolatedInc();

cout << endl << "Концевые вершины(по смежности): ";

c.endpointsAdj();

cout << endl << "Концевые вершины(по инцедентности): ";

c.endpointsInc();

cout << endl << "Доминирующие вершины(по смежности): ";

c.dominanceAdj();

cout << endl << "Доминирующие вершины(по инцедентности): ";

c.dominanceInc();

return 0;

}

**Вывод:** Мы изучили основы теоретико-множественного и графического представлений графов, простейших свойств графов, получили практические навыки заданий и визуализацию графа на плоскости.