Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра “Вычислительная техника”

**Отчет**

по лабораторной работе № 5

по курсу “Логика и основы алгоритмизации в инженерных задачах”

на тему “Определение характеристик графов”

Выполнили студенты группы 22ВВВ3:

Байков А. В.

Гераськина Д. А.

Приняли:

Юрова О. В.

Акифьев И. В.

Пенза 2023

**Лабораторное задание**

1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного графа G. Выведите матрицу на экран.

2. Определите размер графа G, используя матрицу смежности графа.

3. Найдите изолированные, концевые и доминирующие вершины.

4. Постройте для графа G матрицу инцидентности.

5. Определите размер графа G, используя матрицу инцидентности графа.

6. Найдите изолированные, концевые и доминирующие вершины.

**Листинг**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

#include <malloc.h>

int main()

{

setlocale(LC\_ALL, "RUS");

int num\_vertices, count\_size\_graph = 0, count\_edges = 0;

// Ввод количества вершин графа

printf("Введите количество вершин графа: ");

scanf("%d", &num\_vertices);

// Расширение памяти для матрицы инцендентности

int\*\* graph = (int\*\*)malloc(num\_vertices \* sizeof(int\*));

for (int i = 0; i < num\_vertices; i++)

{

graph[i] = (int\*)malloc(num\_vertices \* sizeof(int));

}

int degree\_dominant = num\_vertices - 1; // Степень доминантной вершины = количество вершин - 1

srand(time(NULL));

// Генерируем случайную матрицу смежности

for (int i = 0; i < num\_vertices; i++)

{

for (int j = i; j < num\_vertices; j++)

{

if (i == j)

{

graph[i][j] = 0; // Заполняем обратную диагональ 0

}

else

{

graph[i][j] = graph[j][i] = rand() % 2; // Случайным образом присваиваем 0 или 1

}

}

}

// Распечатываем матрицу смежности

printf("\n Матрица смежности:\n");

for (int i = 0; i < num\_vertices; i++)

{

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

{

printf("%d ", graph[i][j]);

}

printf("\n");

}

// Определение размера графа по ребрам

for (int i = 0; i < num\_vertices; i++)

{

for (int j = i; j < num\_vertices; j++) // Отбрасываем уже просмотренные элементы

{

if (graph[i][j] == 1) // Если есть ребро

{

count\_size\_graph++; // Увеличиваем размер на 1

}

}

}

printf("\nРазмер графa = %d\n", count\_size\_graph); // Выводим размер графика

// Находим изолированные вершины по ребрам

for (int i = 0; i < num\_vertices; i++)

{

count\_edges = 0; // Счетчик краев

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

{

if (graph[i][j] == 1) // Если есть ребро

{

count\_edges++; //Увеличиваем счетчик

}

}

if (count\_edges == 0) // Если ребер нет, то вершина изолирована

{

printf("\n%d-я вершина изолированая\n", i + 1); // Отображение номера изолированной вершины

}

if (count\_edges == 1) // Если ребро только одно, то вершина терминальная

{

printf("\n%d-я вершина концевая\n", i + 1); // Выводим номер концевой вершины

}

if(count\_edges == degree\_dominant)

{

printf("\n%d-я вершина доминирующая\n", i + 1); // Отображение номера доминирующей вершины

}

}

// -------------------------------------------------------------------------- 2 задание

// Расширение памяти для матрицы инцидентности

int\*\* Incident\_matrix = (int\*\*)malloc(num\_vertices \* sizeof(int\*));

for (int i = 0; i < num\_vertices; i++)

{

Incident\_matrix[i] = (int\*)malloc(count\_size\_graph \* sizeof(int));

}

// Инициализация матрицы инцидентности

for (int i = 0; i < num\_vertices; i++) {

for (int j = 0; j < count\_size\_graph; j++) {

Incident\_matrix[i][j] = 0;

}

}

// Заполняем матрицу инцидентности

int edge\_index = 0;

for (int i = 0; i < num\_vertices; i++) {

for (int j = i + 1; j < num\_vertices; j++) {

if (graph[i][j] == 1) {

Incident\_matrix[i][edge\_index] = 1;

Incident\_matrix[j][edge\_index] = 1;

edge\_index++;

}

}

}

// Вывод матрицы инцидентности

printf("\nМатрица инцидентности: \n");

for (int i = 0; i < num\_vertices; i++)

{

for (int j = 0; j < count\_size\_graph; j++)

{

printf("%d ", \*(\*(Incident\_matrix + i) + j));

}

printf("\n");

}

printf("\nРазмер графа = %d\n", count\_size\_graph);

for (int i = 0; i < num\_vertices; i++)

{

count\_edges = 0; // Счетчик краев

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

{

if (graph[i][j] == 1) // Если есть ребро

{

count\_edges++; //Увеличиваем счетчик

}

}

if (count\_edges == 0) // Если ребер нет, то вершина изолирована

{

printf("\n%d-я вершина изолированная\n", i + 1); // Отображение номера изолированной вершины

}

if (count\_edges == 1) // Если ребро только одно, то вершина терминальная

{

printf("\n%d-я вершина концевая\n", i + 1); // Выводим номер концевой вершины

}

if (count\_edges == degree\_dominant)

{

printf("\n%d-я вершина доминирующая\n", i + 1); // Отображение номера доминирующей вершины

}

}

// Освобождаем память матрицы смежности

for (int i = 0; i < num\_vertices; i++) {

free(graph[i]);

}

free(graph);

// Освобождаем память матрицы инцидентов

for (int i = 0; i < num\_vertices; i++) {

free(Incident\_matrix[i]);

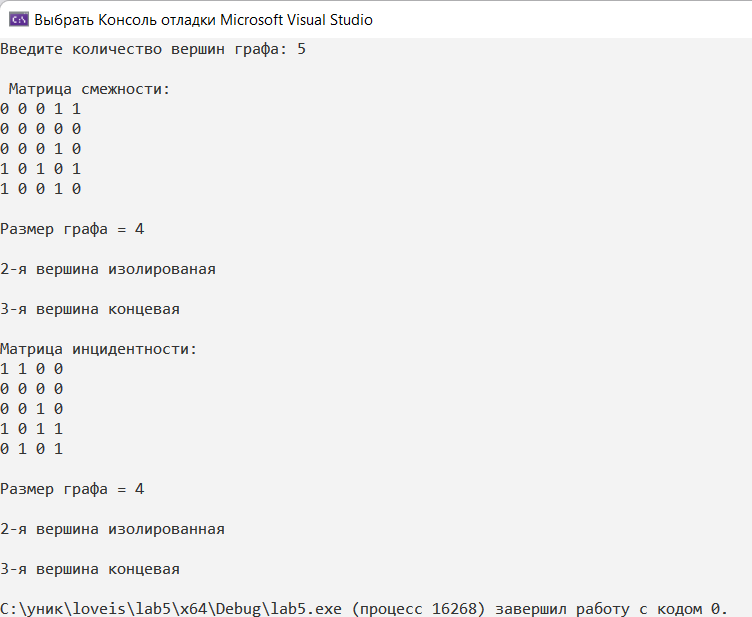
}

free(Incident\_matrix);

return 0;

}

**Результат работы программы**



**Рисунок 1 - Результат работы программы**

**Вывод:** в ходе выполнения лабораторной работы сгенерировали матрицу смежности для неориентированного графа G, вывели матрицу на экран, определили размер графа G, используя матрицу смежности графа,

нашли изолированные, концевые и доминирующие вершины. Построили для графа G матрицу инцидентности, определили размер графа G, используя матрицу инцидентности графа, нашли изолированные, концевые и доминирующие вершины.