Министерство науки и высшего образования РФ

Пензенский государственный университет

Кафедра «Вычислительная кафедра»

**ОТЧЕТ**

по лабораторной работе №5

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Определение характеристик графов»

Выполнила:

студентка группы 22ВВВ2

Расторгуева К.В.

Приняли:

Акифьев И.В.

Митрохин М.А.

Пенза 2023

**Название**

Определение характеристик графов.

**Цель работы**

Изучение неориентированных графов и их характеристик.

**Лабораторное задание**

### Задание 1

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.

### Задание 2\*

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

**Пояснительный текст к программе**

Задание 1

Для начала работы программы пользователю необходимо ввести размер матрицы.

setlocale(0, "rus");

printf("Введите размер матрицы -> ");

scanf("%d", &n); //польщоватлеь вводит размер матрицы

printf("\n");

После чего происходит выделение памяти для матрицы смежности и её формирование с помощью генератора случайных чисел *srand()*.

int\*\* G = (int\*\*)malloc(n \* sizeof(int\*)); //выделение памяти для двумерного массива - матрицы смежности неориентированного графа G

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

G[i] = (int\*)malloc(n \* sizeof(int));

}

В неориентированном графе представлены двухсторонние (симметричные) связи, поэтому матрица будет симметрична относительно главной диагонали. Для организации такой матрицы используется вложенный цикл, генерирующий случайные значения (0 или 1) для области матрицы, находящейся над главной диагональю (в главной диагонали номер столбца = номеру строки => область над диагональю будет содержать элементы, у которых номер строки > номера столбца). Главная диагональ заполняется «0». Область матрицы, находящаяся под главной диагональю симметрично отражает сгенерированные до этого значения.

srand(time(0)); //граф неориентированный -> матрица симметричная

for (int i = 0; i < n; i++) { //формирование матирицы смежности

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

if (j >= i) { //генерируем только бласть матрицы над главной диагональю

if (i == j) G[i][j] = 0; //главная диагональ заполняется нулями (петель в вершинах нет)

else {

G[i][j] = rand() % 2; //генерируется случайное значение [0;1]

G[j][i] = G[i][j]; //значение дублируется симметрично относительно главной диагонали

} //т.е. заполняется область под главной диагональю

}

}

}

Затем происходит вывод матрицы, во время которого подсчитывается количество единиц для определения размера графа. Матрица симметрична, следовательно, каждая связь между вершинами дублируется. Поэтому общее количество «1» делится на 2 – получаем размер графа.

for (int i = 0; i < n; i++) { //вывод матрицы и её размера

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

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

if (G[i][j] == 1) size\_G++; //подсчет "1"

}

printf("\n");

}

size\_G = size\_G / 2; //размер графа = (количество "1")/2

printf("\nРазмер графа = %d\n", size\_G);

Далее определяются изолированные (количество «1» = 0), концевые (количество «1» = 1) и доминирующие вершины (количество «1» = размер матрицы - 1). Организуется вложенный цикл, который подсчитывает количество «1» в каждой строке матрицы.

for (int i = 0; i < n; i++) { //поиск изолированных, концевых и доминирующих вершин

col\_one = 0; //начало подсчета количества "1" для каждой строки матрицы

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

if (G[i][j] == 1) col\_one++; //считаем кол-во "1"

}

if (col\_one == 0) printf("Вершина %d изолировнная\n", i+1); //изолированная вершина - связей нет

if (col\_one == 1) printf("Вершина %d концевая\n", i+1); //концевая - только одна связь

if (col\_one == n-1) printf("Вершина %d доминирующая\n", i+1); //доминирующая - смежна со всеми другими вершинами

}

printf("\n\n");

В программе нумерация вершин начинается с 0, для пользователя выводится номер вершины +1.

Задание 2\*

Матрица инцидентности формируется на основе матрицы смежности: если элемент матрицы смежности равен 1, то в соответствующий столбец для каждой вершины записываются «1». Размер графа = количеству столбцов. Поиск изолированных, концевых и доминирующих вершин осуществляется аналогично матрице смежности

Назначение отдельных операторов программы указывается в виде комментариев. Программа завершает свою работу после нажатия на любую клавишу на клавиатуре.

**Результат выполнения программы**

Результат работы программы для заданий 1 и 2 представлены на рис. 1-3.

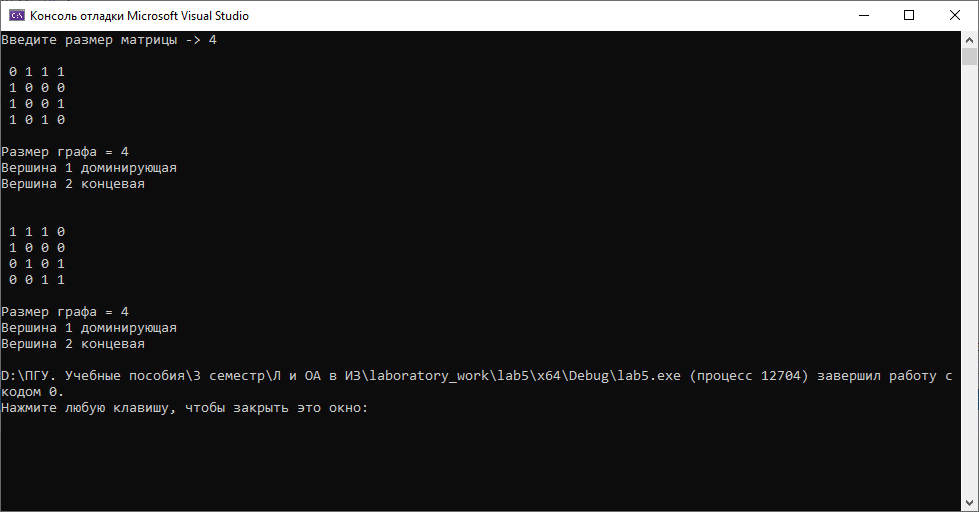


Рисунок 1 – результат выполнения программы

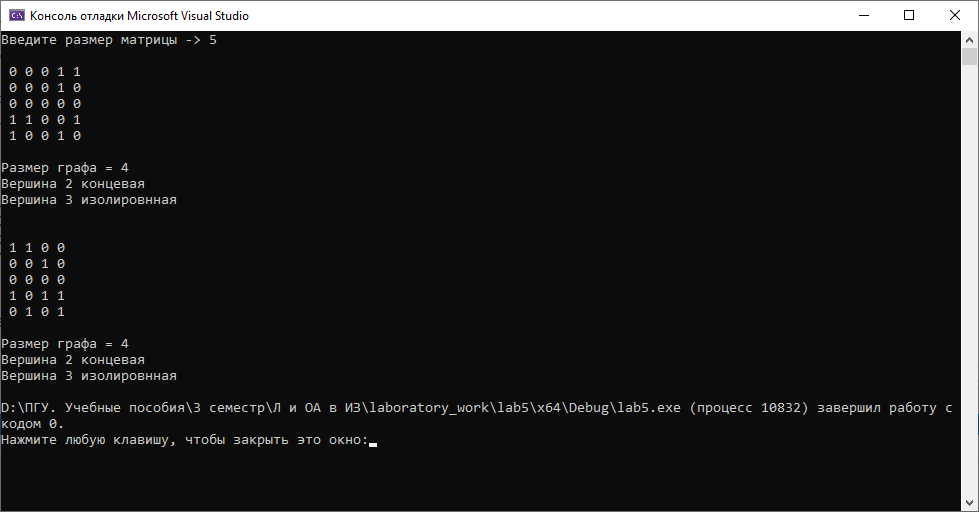


Рисунок 2 – результат выполнения программы

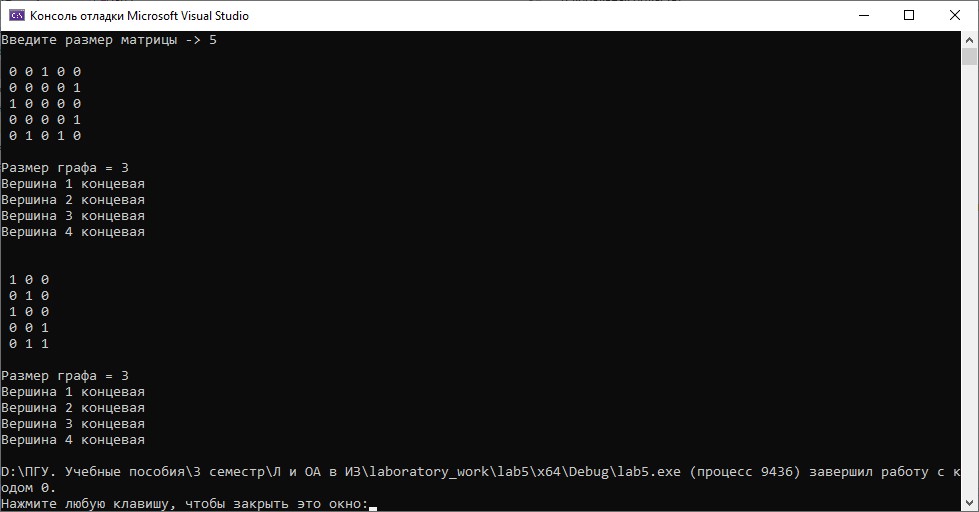


Рисунок 3 – результат выполнения программы

**Выводы**

В ходе выполнения лабораторной работы была разработана на языке Си программа, осуществляющая работу с неориентированным графом:

– сгенерирована матрица смежности;

– построена матрица инцидентности;

– организован подсчет размера графа, используя матрицу смежности;

– организован подсчет размера графа, используя матрицу инцидентности;

– найдены изолированные, концевые и доминирующие вершины.

Результаты работы программы совпали с результатами расчета вручную.

**Приложение**

**Программа**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <locale.h>

#include <stdlib.h>

#include <time.h>

int main(void){

int n; //размер матрицы

int size\_G = 0; //переменная для подсчета размера графа G

int col\_one; //переменная для подсчета количества единиц

setlocale(0, "rus");

printf("Введите размер матрицы -> ");

scanf("%d", &n); //польщоватлеь вводит размер матрицы

printf("\n");

/\*Матрица смежности\*/

int\*\* G = (int\*\*)malloc(n \* sizeof(int\*)); //выделение памяти для двумерного массива - матрицы смежности неориентированного графа G

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

G[i] = (int\*)malloc(n \* sizeof(int));

}

srand(time(0)); //граф неориентированный -> матрица симметричная

for (int i = 0; i < n; i++) { //формирование матирицы смежности

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

if (j >= i) { //генерируем только бласть матрицы над главной диагональю

if (i == j) G[i][j] = 0; //главная диагональ заполняется нулями (петель в вершинах нет)

else {

G[i][j] = rand() % 2; //генерируется случайное значение [0;1]

G[j][i] = G[i][j]; //значение дублируется симметрично относительно главной диагонали

} //т.е. заполняется область под главной диагональю

}

}

}

for (int i = 0; i < n; i++) { //вывод матрицы и её размера

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

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

if (G[i][j] == 1) size\_G++; //подсчет "1"

}

printf("\n");

}

size\_G = size\_G / 2; //размер графа = (количество "1")/2

printf("\nРазмер графа = %d\n", size\_G);

for (int i = 0; i < n; i++) { //поиск изолированных, концевых и доминирующих вершин

col\_one = 0; //начало подсчета количества "1" для каждой строки матрицы

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

if (G[i][j] == 1) col\_one++; //считаем кол-во "1"

}

if (col\_one == 0) printf("Вершина %d изолировнная\n", i+1); //изолированная вершина - связей нет

if (col\_one == 1) printf("Вершина %d концевая\n", i+1); //концевая - только одна связь

if (col\_one == n-1) printf("Вершина %d доминирующая\n", i+1); //доминирующая - смежна со всеми другими вершинами

}

printf("\n\n");

/\*Матрица инцидентности\*/

int j\_int = 0;

int\*\* Gi = (int\*\*)malloc(size\_G \* sizeof(int\*));

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

Gi[i] = (int\*)malloc(n \* sizeof(int));

}

for (int i = 0; i < n; i++) { //формирование матрицы инцидентности

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

if (j > i) { //половина матрицы G

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

Gi[i][j\_int] = 1;

Gi[j][j\_int] = 1;

j\_int++;

}

}

}

}

for (int i = 0; i < n; i++) { //вывод матрицы инцидентности и её размера

for (int j = 0; j < size\_G; j++) {

if (Gi[i][j] != 1) Gi[i][j] = 0;

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

}

printf("\n");

}

printf("\nРазмер графа = %d\n", j\_int);

for (int i = 0; i < n; i++) { //поиск изолированных, концевых и доминирующих вершин

col\_one = 0;

for (int j = 0; j < size\_G; j++) {

if (Gi[i][j] == 1) col\_one++;

}

if (col\_one == 0) printf("Вершина %d изолировнная\n", i + 1);

if (col\_one == 1) printf("Вершина %d концевая\n", i + 1);

if (col\_one == n - 1) printf("Вершина %d доминирующая\n", i + 1);

}

return(0);

}