Министерство 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»

Выполнил:

Студент группы 23ВВВ2

Стрельцов А.П.

Федоров Б.М.

Приняли:

Юрова О. В.

Митрохин М. А.

Пенза 2024

**Цель работы:** поработать с графами, выполнить задания и усвоить материал.

**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.

**ЛИСТИНГ**

**laba 5-1.cpp**

#include <stdio.h>

#include <random>

#include <Windows.h>

#include <conio.h>

#include <malloc.h>

#include <time.h>

int main(void) {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

//Задание 1.1

srand(time(NULL));

int n = 10, size = 0, size2 = 0, \* deg = NULL;

printf("Введите количество вершин: ");

scanf\_s("%d", &n);

int\*\* G = (int\*\*)malloc(n \* sizeof(int\*));

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

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

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

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

G[i][j] = rand() % 2;

G[j][i] = G[i][j];

if (i == j)

G[i][j] = 0;

}

}

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

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

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

}

printf("\n");

}

//Задание 1.2

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

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

size2 += G[i][j];

if (G[i][j] != 0)

size++;

}

}

size /= 2;

printf("Размер графа G: %d\n", size);

//Задание 1.3

deg = (int\*)malloc(n \* sizeof(int));

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

deg[i] = 0;

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

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

if (G[i][j] != 0)

deg[i]++;

}

}

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

printf("Степень вершины %d = %d\n", i, deg[i]);

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

if (deg[i] == 0) printf("Вершина %d - изолированная\n", i);

if (deg[i] == 1) printf("Вершина %d - концевая\n", i);

if (deg[i] == n - 1) printf("Вершина %d - доминирующая\n", i);

}

return 0;

}

Результат работы программы

На рисунке 1 показана реализация задания №1.



Рисунок 1 - Результат работы программы №1.

**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.

**ЛИСТИНГ**

**laba 5-2.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

void generateAdjacencyMatrix(int n, int\*\* G) {

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

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

if (i != j) {

G[i][j] = rand() % 2; // Генерация случайного ребра

G[j][i] = G[i][j]; // Симметричное значение для неориентированного графа

}

else {

G[i][j] = 0; // Нет петель

}

}

}

}

void printAdjacencyMatrix(int n, int\*\* G) {

setlocale(LC\_ALL, "RUS");

printf("Матрица смежности графа G:\n");

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

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

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

}

printf("\n");

}

}

int\*\* createIncidenceMatrix(int n, int\*\* G, int\* edgeCount) {

\*edgeCount = 0;

// Сначала подсчитываем количество рёбер

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

for (int j = i + 1; j < n; j++) {

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

(\*edgeCount)++;

}

}

}

// Создаем матрицу инцидентности

int\*\* incidenceMatrix = (int\*\*)malloc(n \* sizeof(int\*));

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

incidenceMatrix[i] = (int\*)calloc(\*edgeCount, sizeof(int));

}

// Заполняем матрицу инцидентности

int edgeIndex = 0;

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

for (int j = i + 1; j < n; j++) {

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

incidenceMatrix[i][edgeIndex] = 1; // Инцидентность для первой вершины

incidenceMatrix[j][edgeIndex] = 1; // Инцидентность для второй вершины

edgeIndex++;

}

}

}

return incidenceMatrix;

}

void printIncidenceMatrix(int n, int\*\* incidenceMatrix, int edgeCount) {

setlocale(LC\_ALL, "RUS");

printf("Матрица инцидентности графа G:\n");

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

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

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

}

printf("\n");

}

}

void findVertexTypes(int n, int\*\* G) {

setlocale(LC\_ALL, "RUS");

int\* degrees = (int\*)malloc(n \* sizeof(int));

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

degrees[i] = 0; // Инициализация степеней вершин

}

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

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

degrees[i] += G[i][j]; // Подсчет степени вершины

}

}

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

if (degrees[i] == 0) {

printf("Вершина %d - изолированная\n", i);

}

if (degrees[i] == 1) {

printf("Вершина %d - концевая\n", i);

}

}

int max\_degree = 0;

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

if (degrees[i] > max\_degree) {

max\_degree = degrees[i]; // Находим максимальную степень

}

}

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

if (degrees[i] == max\_degree) {

printf("Вершина %d - доминирующая\n", i);

}

}

free(degrees);

}

int main(void) {

setlocale(LC\_ALL, "RUS");

srand(time(NULL)); // Инициализация генератора случайных чисел

int n;

printf("Введите количество вершин: ");

scanf("%d", &n);

// Создание матрицы смежности

int\*\* G = (int\*\*)malloc(n \* sizeof(int\*));

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

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

}

// Генерация матрицы смежности

generateAdjacencyMatrix(n, G);

printAdjacencyMatrix(n, G);

// Создание матрицы инцидентности

int edgeCount;

int\*\* incidenceMatrix = createIncidenceMatrix(n, G, &edgeCount);

printIncidenceMatrix(n, incidenceMatrix, edgeCount);

// Нахождение изолированных, концевых и доминирующих вершин

findVertexTypes(n, G);

// Освобождение памяти

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

free(G[i]);

free(incidenceMatrix[i]);

}

free(G);

free(incidenceMatrix);

return 0;

}

Результат работы программы

На рисунке 2 показана реализация задания №2.

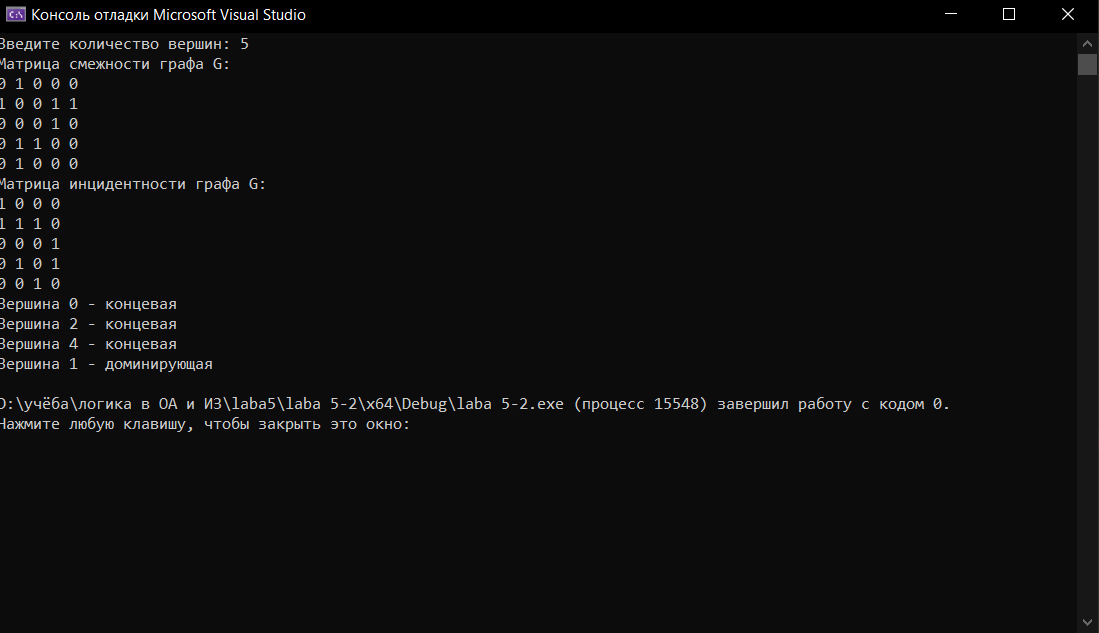


Рисунок 2 - Результат работы программы №2.

**Вывод**

Познакомились с графами, изучили новые функции и применили их в лабораторной работе.