Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра вычислительная техника

**ОТЧËТ**  
по лабораторной работе №5  
по дисциплине: «Определение характеристик графов»

Выполнили студенты группы 22ВВВ2:  
Беляев Д.

Приняли:  
Акифьев И. В.

Митрохин М. А.

Пенза 2023

**Название**

Определение характеристик графов

**Цель работы**

Научиться создавать графы, строить матрицы смежности и инцидентности. Определять характеристики графа.

**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.

**Листинг**

* **main.c**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define MATRIX\_SIZE 4

int\*\* GenerateMatrix(int size\_x, int size\_y);

void ClearMatrix(int\*\* matrix, int size);

void PrintMatrix(int\*\* matrix, int size\_x, int size\_y);

int\*\* GenerateAdjacencyMatrix(int size);

int CalculateAdjSize(int\*\* matrix, int size);

void CheckAdjPeaks(int\*\* matrix, int size);

int\*\* GenerateIncidenceMatrix(int\*\* matrix, int size);

int RebrCount(int\*\* matrix, int size);

int CalculateIncSize(int\*\* matrix, int size\_x, int size\_y);

void CheckIncPeaks(int\*\* matrix, int size\_x, int size\_y);

int main(){

srand(time(NULL));

printf("----- ADJACENCY MATRIX -----\n");

int\*\* matrixAdj = GenerateAdjacencyMatrix(MATRIX\_SIZE);

printf("---- Print matrix ----\n");

PrintMatrix(matrixAdj, MATRIX\_SIZE, MATRIX\_SIZE);

printf("---- Calculate matrix size ----\n");

printf("Size: %d\n", CalculateAdjSize(matrixAdj, MATRIX\_SIZE));

printf("---- Peaks info ----\n");

CheckAdjPeaks(matrixAdj, MATRIX\_SIZE);

printf("\n----- INCEDENCE MATRIX -----\n");

int r = RebrCount(matrixAdj, MATRIX\_SIZE);

int\*\* matrixInc = GenerateIncidenceMatrix(matrixAdj, MATRIX\_SIZE);

printf("---- Print matrix ----\n");

PrintMatrix(matrixInc, MATRIX\_SIZE, r);

printf("---- Calculate matrix size ----\n");

printf("Size: %d\n", CalculateIncSize(matrixInc, MATRIX\_SIZE, r));

printf("---- Peaks info ----\n");

CheckIncPeaks(matrixInc, MATRIX\_SIZE, r);

ClearMatrix(matrixAdj, MATRIX\_SIZE);

ClearMatrix(matrixInc, MATRIX\_SIZE);

return 0;

}

int\*\* GenerateMatrix(int size\_x, int size\_y)

{

int\*\* matrix = (int\*\*)malloc(size\_x \* sizeof(int\*));

if (matrix == NULL)

return NULL;

for (int i = 0; i < size\_x; i++)

{

matrix[i] = (int\*)malloc(size\_y \* sizeof(int));

if (matrix[i] == NULL)

return NULL;

}

for (int i = 0; i < size\_x; i++)

{

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

{

matrix[i][j] = 0;

}

}

return matrix;

}

void ClearMatrix(int\*\* matrix, int size)

{

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

{

free(matrix[i]);

}

free(matrix);

}

void PrintMatrix(int\*\* matrix, int size\_x, int size\_y)

{

for (int i = 0; i < size\_x; i++)

{

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

{

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

}

printf("\n");

}

}

#pragma region Adjacency Matrix

int\*\* GenerateAdjacencyMatrix(int size)

{

int\*\* matrix = GenerateMatrix(size, size);

if (matrix == NULL)

return NULL;

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

{

for (int j = i; j < size; j++)

{

if (i == j)

continue;

int rnd = rand() % 2;

int izolate = rand() % 2;

matrix[i][j] = rnd;

if (rnd == 1)

matrix[j][i] = 1;

}

}

return matrix;

}

int CalculateAdjSize(int\*\* matrix, int size)

{

int count = 0;

int\*\* rebra = GenerateMatrix(size, size);

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

{

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

{

rebra[i][j] = matrix[i][j];

}

}

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

{

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

{

if (rebra[i][j] == 1)

{

count++;

rebra[j][i] = 0;

}

}

}

return count;

}

void CheckAdjPeaks(int\*\* matrix, int size)

{

int isolate = 0;

int end = 0;

int max = 0;

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

{

int c = 0;

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

{

if (matrix[i][j] == 1)

c++;

}

if (c == 0)

isolate++;

else if (c == 1)

end++;

if (c == size - 1)

{

max = i;

}

}

printf("Isolate peaks: %d\n", isolate);

printf("End peaks: %d\n", end);

printf("Dominant peaks: %d\n", max);

}

#pragma endregion

#pragma region Incidence Matrix

int RebrCount(int\*\* matrix, int size)

{

int rCount = 0;

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

{

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

{

if (matrix[i][j] == 1)

rCount++;

}

}

return rCount / 2;

}

int\*\* GenerateIncidenceMatrix(int\*\* matrix, int size)

{

int rCount = RebrCount(matrix, size);

int\*\* incMatrix = GenerateMatrix(size, rCount);

int curR = 0;

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

{

for (int j = i; j < size; j++)

{

if (matrix[i][j] == 1)

{

incMatrix[i][curR] = 1;

incMatrix[j][curR] = 1;

curR++;

}

}

}

return incMatrix;

}

int CalculateIncSize(int\*\* matrix, int size\_x, int size\_y)

{

int count = 0;

for (int i = 0; i < size\_x; i++)

{

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

{

if (matrix[i][j] == 1)

count++;

}

}

return count / 2;

}

void CheckIncPeaks(int\*\* matrix, int size\_x, int size\_y)

{

int isolate = 0;

int end = 0;

int max = 0;

for (int i = 0; i < size\_x; i++)

{

int c = 0;

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

{

if (matrix[i][j] == 1)

c++;

}

if (c == 0)

isolate++;

else if (c == 1)

end++;

if (c == size\_x - 1)

{

max = i;

}

}

printf("Isolate peaks: %d\n", isolate);

printf("End peaks: %d\n", end);

printf("Dominant peaks: %d\n", max);

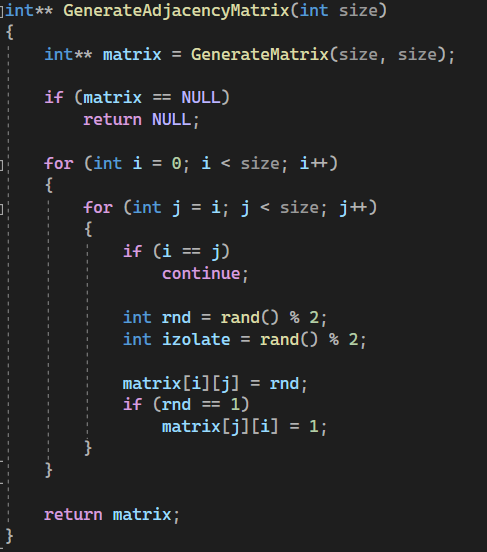
}

#pragma endregion

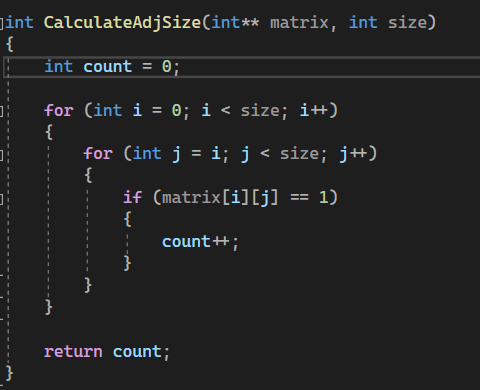
**Задания**

Задание 1

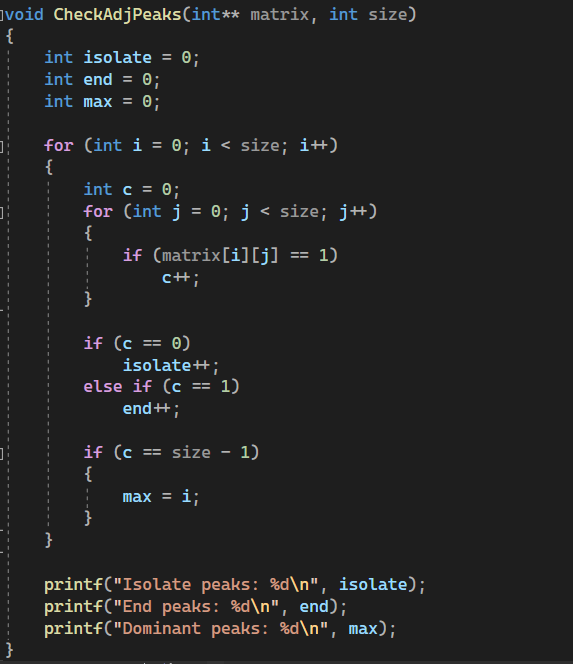
1. Матрица смежности



2. Размер графа

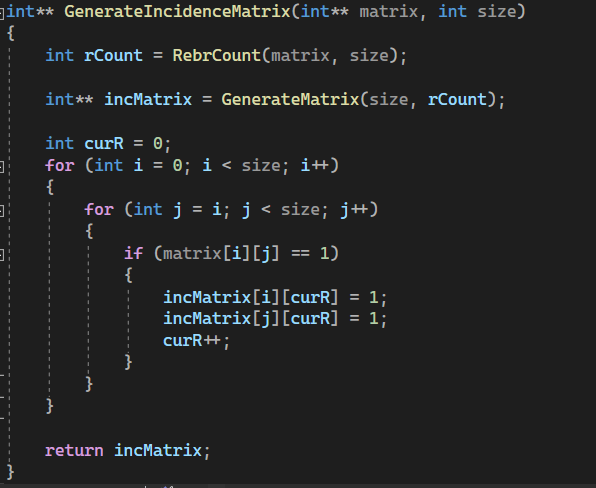


3. Характеристики графа

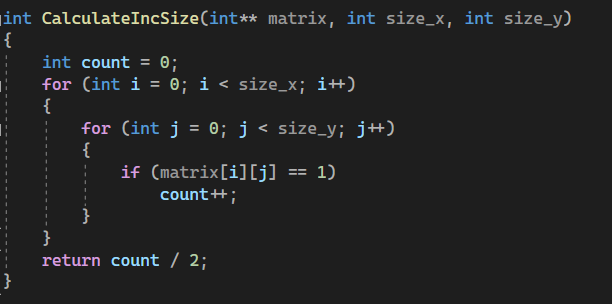


Задание 2

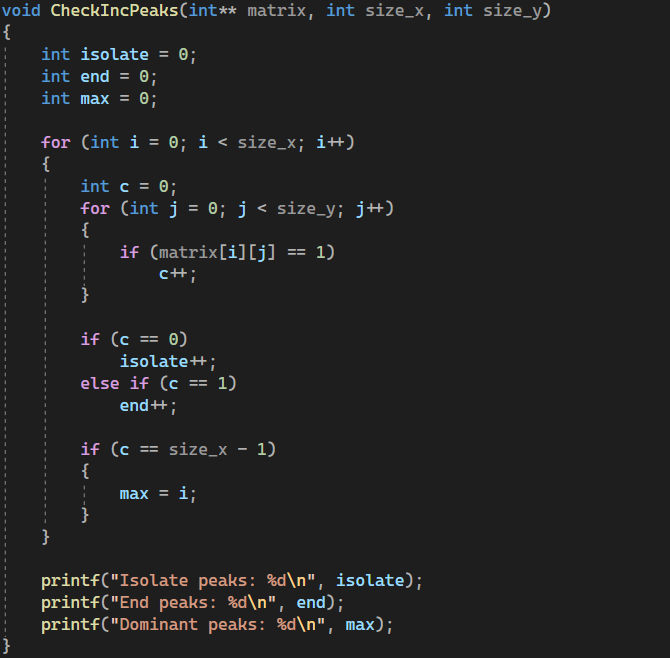
1. Матрица инцидентности



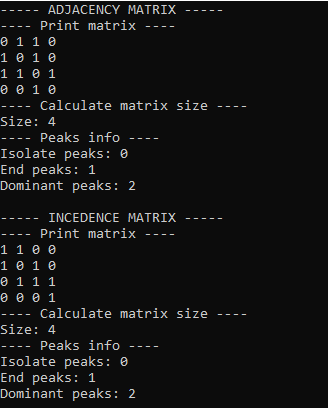
2. Размер графа



3. Характеристики графа



**Результат работы программы**

****

**Вывод**

Научился создавать графы, строить матрицы смежности и инцидентности. Определять характеристики графа.