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

**ОТЧËТ**  
по лабораторной работе №7  
по дисциплине: «Обход графа в глубину»

Выполнили студенты группы 22ВВВ2:  
Демин М. С.

Захаров А. С.

Амиров И. Р.

Приняли:  
Акифьев И. В.

Митрохин М. А.

Пенза 2023

**Название**

Обход графа в глубину

**Цель работы**

Разобраться в графах и матрицах. Научиться определять характеристики графов.

**Лабораторное задание**

**Задание 1**

1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного графа *G*. Выведите матрицу на экран.
2. Для сгенерированного графа осуществите процедуру обхода в глубину, реализованную в соответствии с приведенным выше описанием.

**3.**\* Реализуйте процедуру обхода в глубину для графа, представленного списками смежности.

**Задание 2\***

1. Для матричной формы представления графов выполните преобразование рекурсивной реализации обхода графа к не рекурсивной.

**Листинг**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int isolTest(int\*\* matrix, int count, int clearVertex, int\*\* NUM)

{

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

{

int isol = 0;

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

{

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

isol++;

}

if (isol == 0)

{

printf("%d\n", i + 1);

NUM[1][i] = false;

}

}

return 0;

}

int DFS(int\*\* vertex, int first, int\*\* matrix, int count, int clearVertex)

{

vertex[1][first] = false;

printf("%d ", vertex[0][first]);

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

{

if (matrix[first][i] == 1 and vertex[1][i] == true)

{

DFS(vertex, i, matrix, count, clearVertex);

}

}

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

{

if (vertex[1][i] == 0)

clearVertex--;

}

return clearVertex;

}

int POG(int\*\* matrix, int count)

{

int\*\* NUM = (int\*\*)malloc(2 \* sizeof(int\*));

for (int i = 0; i < 2; i++)

{

NUM[i] = (int\*)malloc(count \* sizeof(int));

}

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

{

NUM[0][i] = i + 1;

NUM[1][i] = true;

}

int clearVertex = count;

int i=0;

printf("Enter vertex number from which would you prefer to start: ");

scanf("%d", &i);

i--;

isolTest(matrix, count, clearVertex, NUM);

while (clearVertex > 0)

{

while (NUM[1][i] == false)

{

i++;

}

clearVertex = DFS(NUM, i, matrix, count, clearVertex);

}

return 0;

}

// Структура для представления узла графа

struct Node {

int value;

struct Node\* next;

};

// Функция для создания нового узла графа

struct Node\* createNode(int value) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->value = value;

newNode->next = NULL;

return newNode;

}

// Процедура обхода в глубину

void DFS3(int vertex, struct Node\*\* adjacencyList, int\* visited) {

// Помечаем текущую вершину как посещенную

visited[vertex] = 1;

// Выводим значение текущей вершины

printf("%d ", vertex);

// Получаем список смежных вершин для текущей вершины

struct Node\* adjacentVertex = adjacencyList[vertex];

// Рекурсивно вызываем DFS для каждой смежной вершины, если она еще не была посещена

while (adjacentVertex != NULL) {

int connectedVertex = adjacentVertex->value;

if (!visited[connectedVertex]) {

DFS3(connectedVertex, adjacencyList, visited);

}

adjacentVertex = adjacentVertex->next;

}

}

int task1\_3() {

int numVertices = 6; // Количество вершин в графе

// Создаем список смежности для графа

struct Node\*\* adjacencyList = (struct Node\*\*)malloc((numVertices +3) \* sizeof(struct Node\*));

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

adjacencyList[i] = NULL;

}

// Добавляем ребра в граф

// Пример графа: 0-1-2-3, 1-4, 2-4-5

adjacencyList[0] = createNode(1);

adjacencyList[1] = createNode(2);

adjacencyList[2] = createNode(3);

adjacencyList[3] = NULL;

adjacencyList[4] = createNode(4);

adjacencyList[5] = NULL;

adjacencyList[6] = createNode(4);

adjacencyList[7] = createNode(5);

adjacencyList[8] = NULL;

// Массив для отслеживания посещенных вершин

int\* visited = (int\*)malloc(numVertices \* sizeof(int));

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

visited[i] = 0;

}

// Вызываем DFS для каждой вершины графа

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

if (!visited[i]) {

DFS3(i, adjacencyList, visited);

}

}

return 0;

}

int main()

{

int count = 0;

printf("Enter count of vertices: ");

scanf("%d", &count);

//count = 5;

srand(time(NULL));

int\*\* matrix = (int\*\*)malloc(count \* sizeof(int\*));

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

{

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

}

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

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

matrix[j][i] = 0;

}

}

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

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

int random = rand() % 2; // Ãåíåðàöèÿ ñëó÷àéíîãî ÷èñëà 0 èëè 1

matrix[i][j] = random;

matrix[j][i] = random; // Çàïîëíåíèå ýëåìåíòîâ è èõ ñèììåòðè÷íûõ ïàð

}

}

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

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

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

}

printf("\n");

}

printf("\n");

printf("\n");

POG(matrix, count);

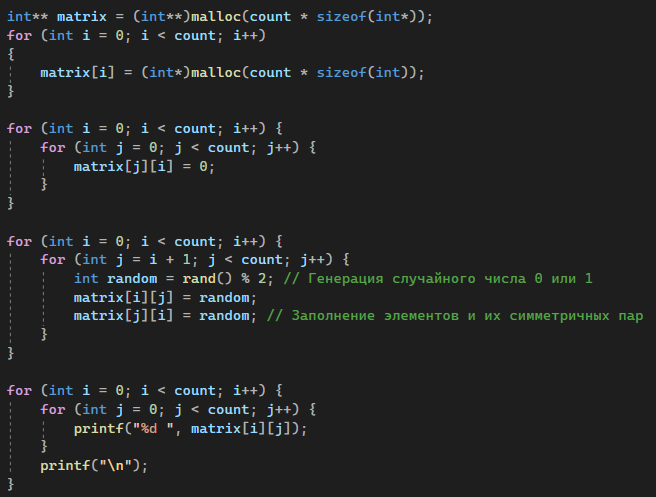
printf("\n");

printf("task 1.3\n");

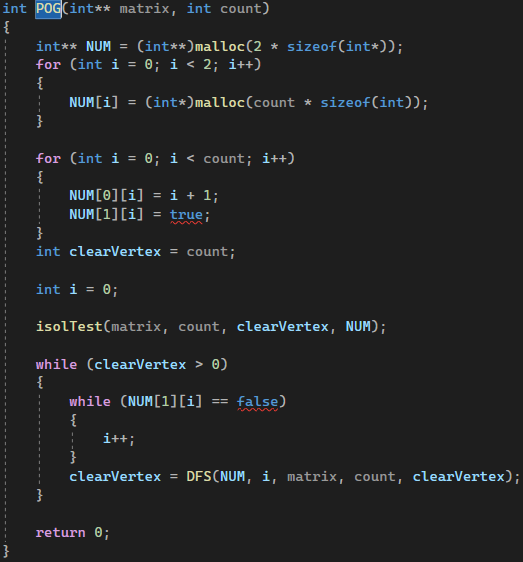
task1\_3();

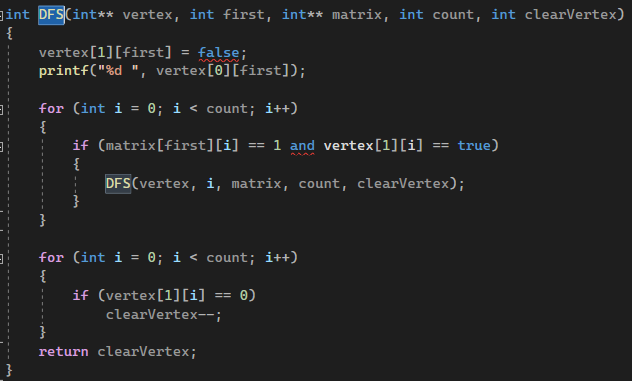
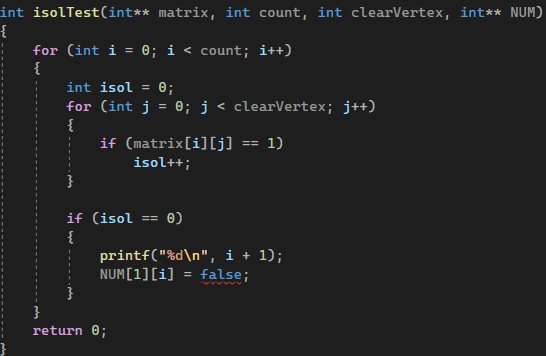
}**Задания**

* 1. Матрица смежности

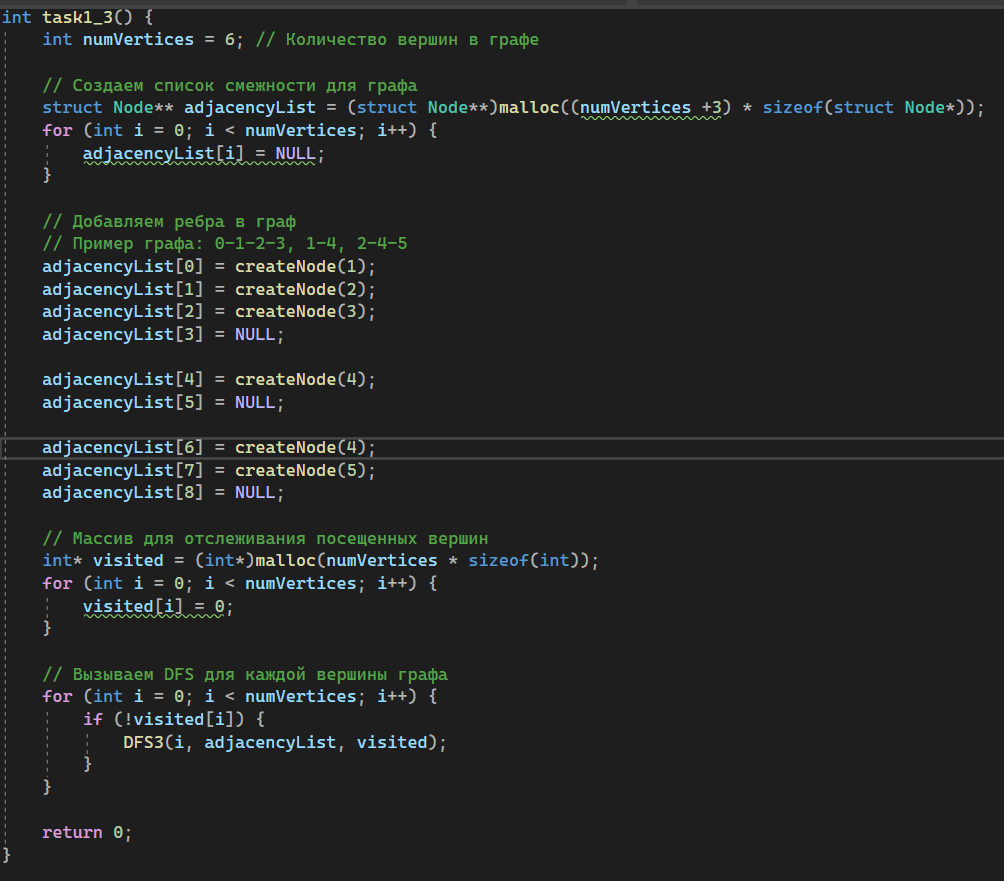


* 1. Процедура обхода в глубину для графа



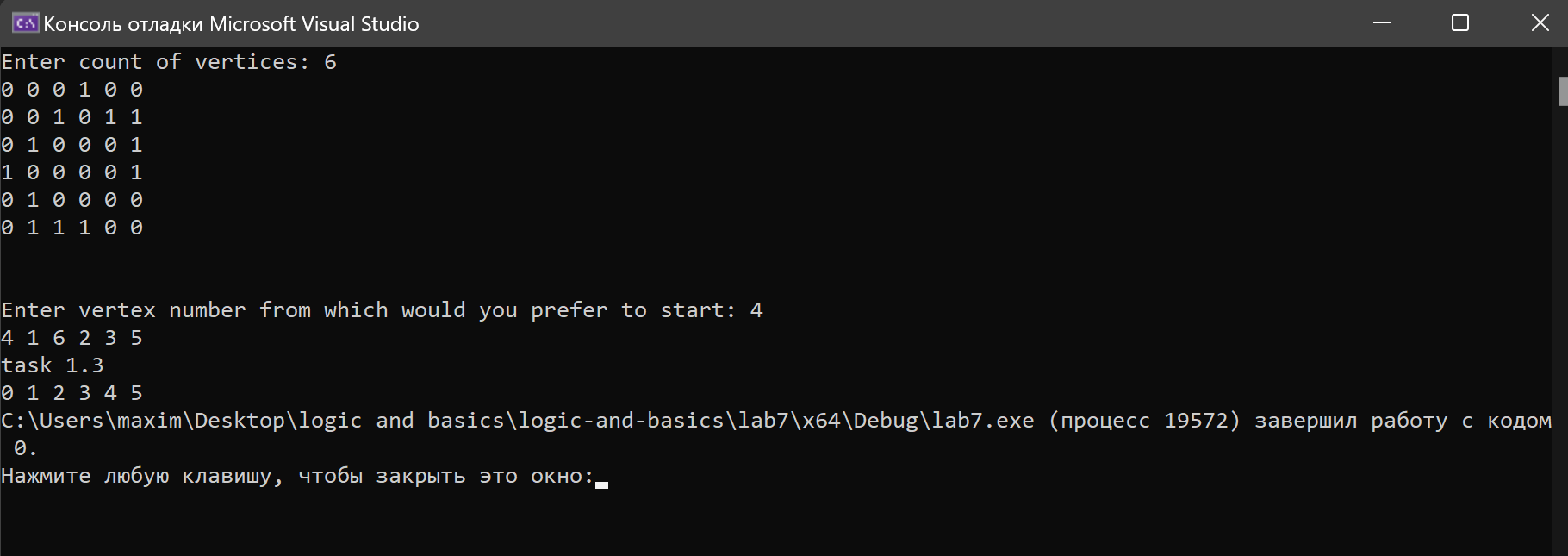
* 1. Процедура обхода в глубину для графа(списки смежности)



2.1 Преобразование рекурсивной реализации обхода графа к не рекурсивной

-----

**Результат работы программы**

****

**Вывод**

Разобрались с графами и матрицами, научились определять характеристики графов, создавать графы, строить матрицы смежности и инцидентности, создавать алгоритмы обхода графа в глубину