Министерство науки и высшего образования РФ

Пензенский государственный университет

Кафедра «Вычислительная кафедра»

**ОТЧЕТ**

по лабораторной работе №7

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «**Обход графа в глубину**»

Выполнил:

студенты группы 22ВВП1

Воробьева М.М.

Приняли:

Акифьев И.В.

Митрохин М.А.

Пенза 2023

**Название**

Обход графа в глубину

**Цель работы:** освоить такую операцию, как обход графов, представленных в виде матрицы смежности и списка смежности в ширину с помощью рекурсивной и не рекурсивной функции.

**Лабораторное задание:**

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 кода:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <cstdio>

#include <vector>

#include <ctime>

#include <cstdlib>

#include <locale.h>

#include <stack>

using namespace std;

vector<bool> vis; // Вектор для хранения информации о посещенных вершинах

void DFS\_2(int start\_vertex, const vector < vector < int>>& G) {

stack<int> s;

s.push(start\_vertex);

while (!s.empty()) {

int v = s.top();

s.pop();

bool isIsolated = true;

for (int i = 1; i < G.size(); ++i) {

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

isIsolated = false;

break;

}

}

if (isIsolated) continue; // Пропустить изолированные вершины

if (!vis[v]) {

printf("%d ", v);

vis[v] = true;

for (int i = G.size() - 1; i >= 1; --i) {

if (G[v][i] == 1 && !vis[i]) {

s.push(i);

}

}

}

}

}

void DFS(int v, const vector<vector<int>>& G) {

bool isIsolated = true;

for (int i = 1; i < G.size(); ++i) {

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

isIsolated = false;

break;

}

}

if (isIsolated) return;

vis[v] = true;

printf("%d ", v);

for (int i = 1; i < G.size(); ++i) {

if (G[v][i] == 1 && !vis[i]) {

DFS(i, G);

}

}

}

void DFS\_List(int v, const vector < vector < int>>& adjacency\_list) {

if (adjacency\_list[v].empty())

return; // Пропустить изолированные вершины

vis[v] = true;

printf("%d ", v);

for (const int& neighbor : adjacency\_list[v]) {

if (!vis[neighbor]) {

DFS\_List(neighbor, adjacency\_list);

}

}

}

void printList(const vector<vector<int>>& list) {

printf("Список смежности:\n");

for (int i = 1; i < list.size(); ++i) {

printf("%d: ", i);

for (const int& j : list[i]) {

printf("%d ", j);

}

printf("\n");

}

}

int swt = 0;

vector<vector<int>> cmej\_mat(int num\_vertices, double veroj) {

vector<vector<int>> adjacency\_matrix(num\_vertices + 1, vector<int>(num\_vertices + 1, 0));

int cont = 0;

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

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

if (static\_cast<double>(rand()) / RAND\_MAX < 0.5) {

adjacency\_matrix[i][j] = 1;

adjacency\_matrix[j][i] = 1;

cont++;

}

}

}

if (cont == 0) {

swt = 1;

}

printf("Матрица смежности:\n");

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

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

printf("%d ", adjacency\_matrix[i][j]);

}

printf("\n");

}

return adjacency\_matrix;

}

vector<vector<int>> matrixToList(const vector<vector<int>>& matrix) {

int num\_vertices = matrix.size() - 1;

vector<vector<int>> adjacency\_list(num\_vertices + 1);

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

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

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

adjacency\_list[i].push\_back(j);

}

}

}

return adjacency\_list;

}

int main() {

setlocale(LC\_ALL, "RUS");

int num\_vertices;

double veroj=0.5;

printf("Введите количество вершин: ");

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

srand(static\_cast<unsigned int>(time(nullptr)));

auto matrix = cmej\_mat(num\_vertices, veroj);

auto adjacency\_list = matrixToList(matrix);

if (swt == 1) {

printf("\nОбход произвести нельзя! Первая вершина изолированна\n");

}

else {

printList(adjacency\_list);

vis = vector<bool>(num\_vertices + 1, false);

printf("Обход в глубину с применением рекурсии: ");

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

if (!vis[i]) {

DFS(i, matrix);

}

}

printf("\n");

vis = vector<bool>(num\_vertices + 1, false);

printf("Обход в глубину для матрицы: ");

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

if (!vis[i]) {

DFS\_2(i, matrix);

}

}

printf("\n");

vis = vector<bool>(num\_vertices + 1, false);

printf("Обход в глубину для списка: ");

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

if (!vis[i]) {

DFS\_List(i, adjacency\_list);

}

}

printf("\n");

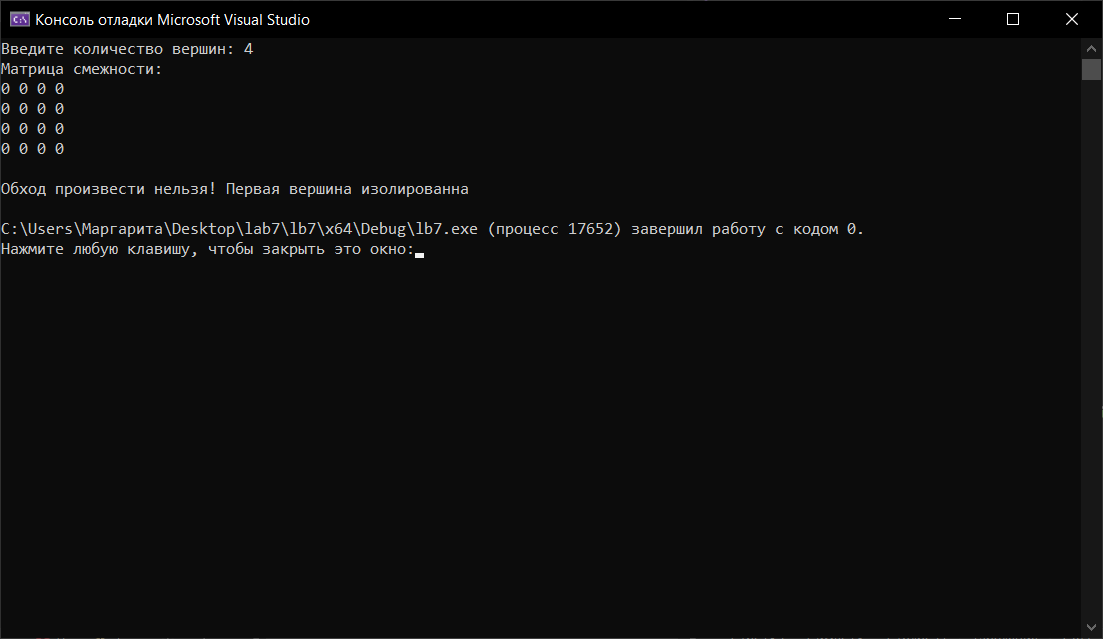
}

return 0;

}

**Результат работы программы:**





**Вывод:** в ходе выполнения лабораторной работы были освоены навыки обхода графа, представленного в виде матрицы смежности и списка смежности, с помощью рекурсивной и не рекурсивной функции.