Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №4

# по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Обход графа в глубину»

**Выполнили:**

студенты группы 20ВВ3

Шадрин Данила

**Приняли:**

Юрова О.В.

Митрохин М.А.

Пенза 2021

# Название

Обход графа в глубину

**Цель работы**–изучение функций, предназначенных для обхода графа в глубину.

# 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\*

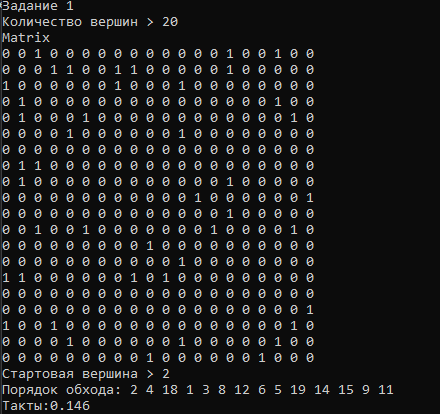
1. Для матричной формы представления графов выполните

преобразование рекурсивной реализации обхода графа к не рекурсивной.

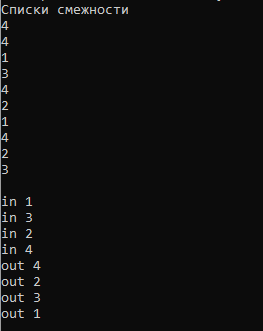
**Теоретические сведения**

Одним из способов обхода графов является поиск в глубину. Идея такого обхода состоит в том, чтобы начав обход из какой-либо вершины всегда переходить по первой встречающейся в процессе обхода связи в следующую вершину, пока существует такая возможность.

# Результат работы программы



**Рисунок 1 – Результат работы программы**



**Рисунок 2 – Результат работы программы**

**Листинг**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <conio.h>

#include <time.h>

#include <stack>

#include <vector>

using namespace std;

void dfs\_spis(int, const vector<vector<int>>&);

vector<bool> vis(0);

void DFS\_f(int\*\* graph, bool\* visited, int st, int n)

{

int r;

cout << st + 1 << " ";

visited[st] = true;

for (r = 0; r <= n; r++)

if ((graph[st][r] != 0) && (!visited[r]))

DFS\_f(graph, visited, r, n);

}

void Create\_matrix() {

int start, n, t = clock();

cout << "Задание 1" << endl;

cout << "Количество вершин > "; cin >> n;

bool\* visited = new bool[n];

int\*\* matrix;

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

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

//matrix[i] = new int[n];

visited[i] = false;

}

int s = 1;

srand(time(NULL));

for (int i = 0; i < n; i++) // создание матрицы

{

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

matrix[i][i] = 0;

for (int j = s; j < n; j++)

{

if (rand() % 100 > 10)

{

matrix[i][j] = 0;

}

else {

matrix[i][j] = 1;

}

}

s++;

}

s = 1;

for (int i = 0; i < n; i++) // неориентированный граф

{

for (int j = s; j < n; j++)

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

s++;

}

cout << "Matrix"; // вывод матрицы на экран

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

{

cout << endl;

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

{

cout << matrix[i][j] << " ";

}

}

cout << endl;

cout << "Стартовая вершина > "; cin >> start;

cout << "Порядок обхода: ";

DFS\_f(matrix, visited, start - 1, n);

delete[]visited;

//DFS\_stack(matrix, visited, start - 1, n);

cout << "\nТакты:" << ((float)t / CLOCKS\_PER\_SEC) << "\n";

system("pause");

}

void sp\_smej() {

int n, m;

cout << "Списки смежности" << endl;

cin >> n >> m; // строки и столбцы

//иницилизируем список смежности и массив посещенных вершин

vector<vector<int>> adj\_list(n, vector<int>());

vis.resize(n, false);

//ввод ребер

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

int x, y;

cin >> x >> y;

adj\_list[x - 1].push\_back(y - 1);

adj\_list[y - 1].push\_back(x - 1);

}

cout << endl;

dfs\_spis(0, adj\_list); //поиск в глубину

}

void dfs\_spis(int u, const vector<vector<int>>& matrix\_adjacency) {

vis[u] = true;

cout << "in " << u + 1 << endl;

for (auto i : matrix\_adjacency[u])

if (!vis[i]) dfs\_spis(i, matrix\_adjacency);

cout << "out " << u + 1 << endl;

}

void main()

{

system("chcp 1251"); system("cls");

Create\_matrix();

//vector\_matrix();

sp\_smej();

}