Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

Пенза 2021

**ОТЧЁТ**

по лабораторной работе №4

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Обход графа в глубину»

Выполнили:

студентки группы 20ВВ4

Кожевникова А. В.

Мамелина Ю. В.

Приняла:

д.т.н Юрова О. В.

Пенза 2021

**Цель работы:** Изучить алгоритм обхода графа в глубину, научиться применять полученные умения на практике.

**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.

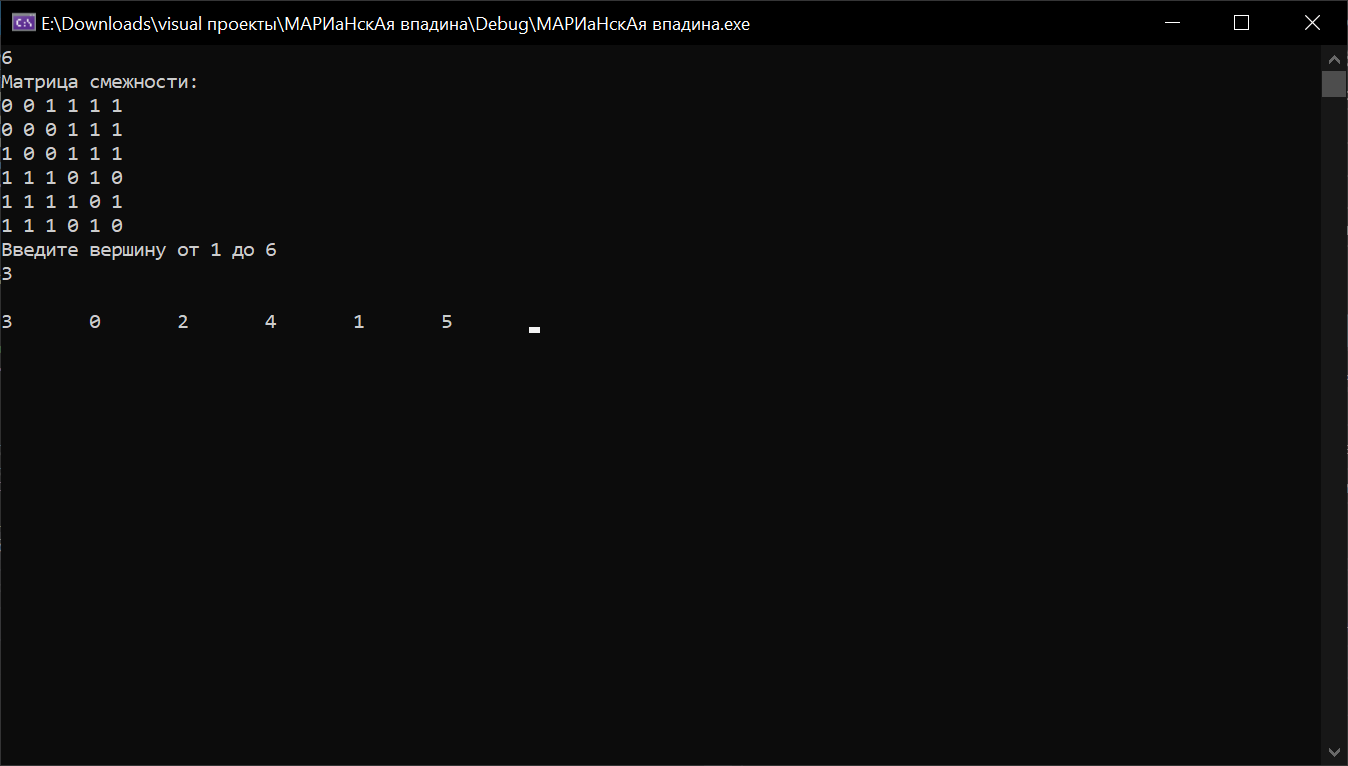


Рис 1 – результат работы программы.

(приложение 1)

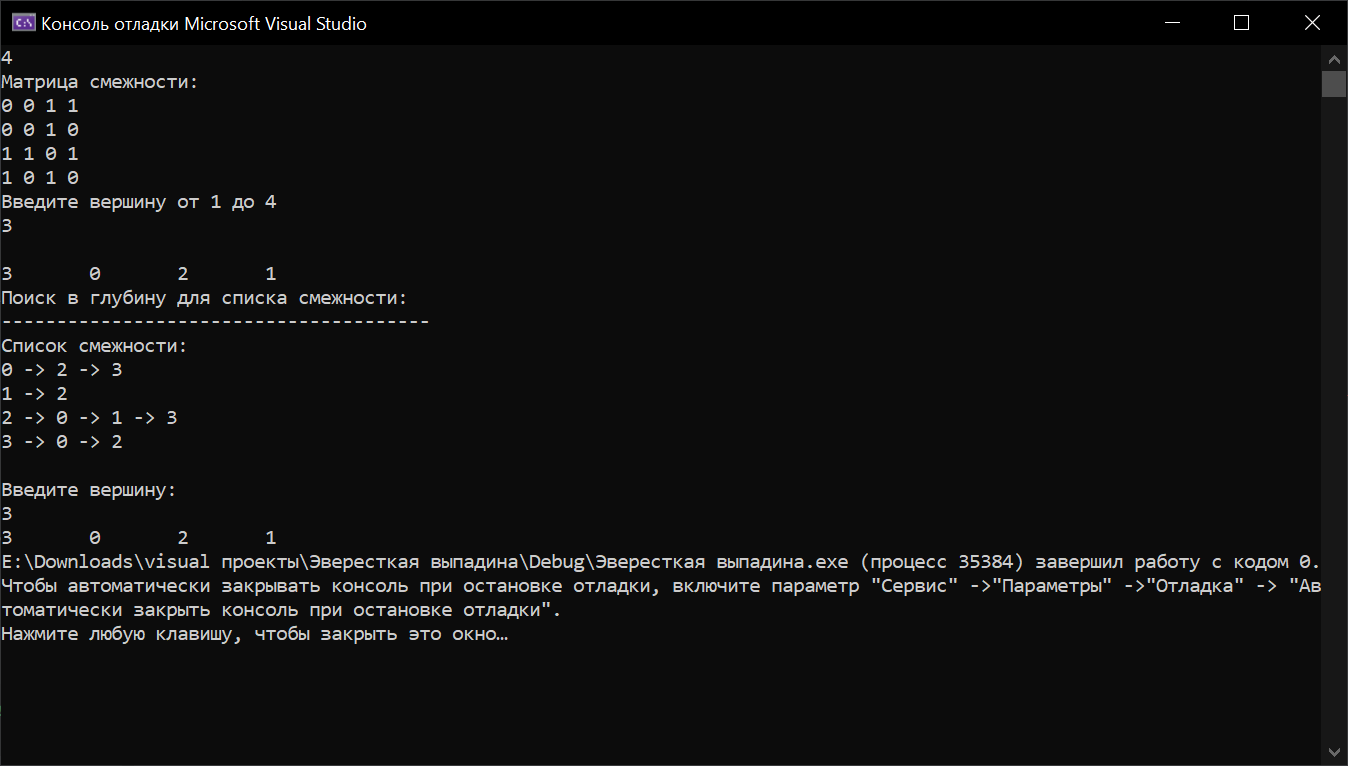


Рис 2 – результат работы задания 3.

(приложение 2)

**Задание 2:**\*

1. Для матричной формы представления графов выполните

преобразование рекурсивной реализации обхода графа к не рекурсивной.

**Вывод:** Изучили работу алгоритма обхода графа в глубину, применили полученные умения на практике для написания программы в соответствии с заданием.

**Приложение:**

Приложение 1

#include <iostream>

#include <locale>

using namespace std;

void DFS(int v, int n, int\* M[], bool vis[]);

int main()

{

setlocale(LC\_ALL, "rus");

int n;

cin >> n;

int\*\* matrix = new int\* [n];

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

{

matrix[i] = new int[n];

}

srand(time(NULL));

cout << "Матрица смежности:" << endl;

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

{

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

{

matrix[i][j] = rand() % 100;

if (i == j) {

matrix[i][j] = 0;

}

if (matrix[i][j] >= 0 && matrix[i][j] <= 30) {

matrix[i][j] = 0;

}

if (matrix[i][j] >= 31 && matrix[i][j] <= 100) {

matrix[i][j] = 1;

}

}

}

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

{

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

{

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

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

}

cout << endl;

}

int v;

cout << "Введите вершину от 1 до " << n << endl;

cin >> v;

cout << endl;

bool\* vis = new bool[v];

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

{

vis[i] = { false };

}

DFS(v, n, matrix, vis);

//очищение памяти:

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

delete[] matrix[i];

}

delete[] matrix;

delete[] vis;

return 0;

}

void DFS(int v, int n, int\* M[], bool vis[]) {

vis[v] = true;

cout << v << "\t";

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

{

if (M[v][i] == true && vis[i] == false) {

DFS(i, n, M, vis);

}

}

}

Приложение 2

#include <iostream>

#include <locale>

#include <vector>

using namespace std;

void DFS\_for\_matrix(int v, int n, int\* M[], bool vis[]);

void DFS\_for\_list(int node, bool visited[], int vertex, vector<vector<int>> AdjList);

vector<vector<int>> convert(vector<vector<int>> a);

const int N = 1e5 + 2;

vector <int> adj[N];

int main()

{

setlocale(LC\_ALL, "rus");

int n;

cin >> n;

int\*\* matrix = new int\* [n];

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

{

matrix[i] = new int[n];

}

srand(time(NULL));

cout << "Матрица смежности:" << endl;

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

{

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

{

matrix[i][j] = rand() % 100;

if (i == j) {

matrix[i][j] = 0;

}

if (matrix[i][j] >= 0 && matrix[i][j] <= 30) {

matrix[i][j] = 0;

}

if (matrix[i][j] >= 31 && matrix[i][j] <= 100) {

matrix[i][j] = 1;

}

}

}

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

{

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

{

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

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

}

cout << endl;

}

int ver;

cout << "Введите вершину от 1 до " << n << endl;

cin >> ver;

cout << endl;

bool\* vis = new bool[ver];

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

{

vis[i] = { false };

}

DFS\_for\_matrix(ver, n, matrix, vis);

cout << endl;

cout << "Поиск в глубину для списка смежности:" << endl;

cout << "---------------------------------------" << endl;

vector<vector<int>> a;

a.resize(n);

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

{

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

{

a[i].push\_back(matrix[i][j]);

}

}

vector<vector<int>> AdjList = convert(a);

cout << "Список смежности:" << endl;

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

{

cout << i;

for (int j = 0; j < AdjList[i].size(); j++) {

cout << " -> " << AdjList[i][j];

}

cout << endl;

}

cout << endl;

bool\* visit = new bool[n];

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

{

visit[i] = false;

}

cout << "Введите вершину:" << endl;

int v;

cin >> v;

DFS\_for\_list(v, visit, n, AdjList);

//очищение памяти:

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

delete[] matrix[i];

}

delete[] matrix;

delete[] vis;

return 0;

}

void DFS\_for\_matrix(int v, int n, int\* M[], bool vis[]) {

vis[v] = true;

cout << v << "\t";

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

{

if (M[v][i] == true && vis[i] == false) {

DFS\_for\_matrix(i, n, M, vis);

}

}

}

vector<vector<int>> convert(vector<vector<int>> a)

{

vector<vector<int>> adjList(a.size());

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

{

for (int j = 0; j < a[i].size(); j++)

{

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

{

adjList[i].push\_back(j);

}

}

}

return adjList;

}

void DFS\_for\_list(int node, bool visited[], int vertex, vector<vector<int>> AdjList) {

visited[node] = true;

cout << node << "\t";

vector<int>::iterator it;

for (it = AdjList[node].begin(); it != AdjList[node].end(); it++)

{

if (visited[\*it]);

else {

DFS\_for\_list(\*it, visited, vertex, AdjList);

}

}

}