Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

Пенза 2021

**ОТЧЁТ**

по лабораторной работе №5

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Обход графа в ширину»

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 ширину, изучить библиотеку <queue>, 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 ширину,

реализованную в соответствии с приведенным выше описанием. При

реализации алгоритма в качестве очереди используйте класс queue из

стандартной библиотеки С++.

3. Реализуйте процедуру обхода в ширину для графа, представленного

списками смежности.

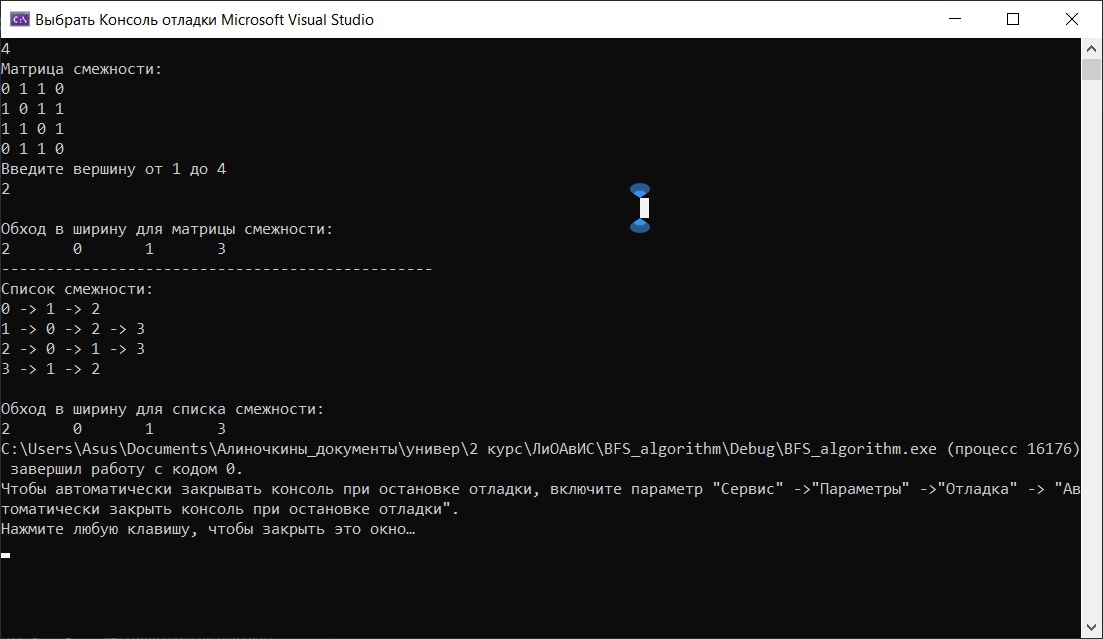


Рис 2 – результат работы задания 1.

(приложение 1)

**Задание 2:**\*

1. Для матричной формы представления графов реализуйте алгоритм

обхода в ширину с использованием очереди, построенной на основе

структуры данных «список», самостоятельно созданной в лабораторной

работе № 3.

2. Оцените время работы двух реализаций алгоритмов обхода в ширину

(использующего стандартный класс queue и использующего очередь,

реализованную самостоятельно) для графов разных порядков.

**Вывод:** Изучили работу алгоритма обхода графа в ширину, применили полученные умения на практике для написания программы в соответствии с заданием, освоили функции библиотеки <queue>.

**Приложение:**

Приложение 1

#include <iostream>

#include <vector>

#include <queue>

using namespace std;

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

void BFS\_for\_matrix(int s, int\* m[], int n);

void BFS\_for\_list(int ver, int n, vector < vector < int>> AdjList);

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;

cout << "Обход в ширину для матрицы смежности:" << endl;

BFS\_for\_matrix(v, matrix, n);

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]);

}

}

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

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;

cout << "Обход в ширину для списка смежности:" << endl;

BFS\_for\_list(v, n, AdjList);

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

delete[] matrix[i];

}

delete[] matrix;

return 0;

}

void BFS\_for\_matrix(int s, int\* m[], int n) {

vector<bool> visited(n, false);

queue<int> q;

q.push(s);

visited[s] = true;

int vis;

while (!q.empty()) {

vis = q.front();

cout << vis << "\t";

q.pop();

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

if (m[vis][i] == 1 && (!visited[i])) {

q.push(i);

visited[i] = true;

}

}

}

}

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 BFS\_for\_list(int ver, int n, vector < vector < int>> AdjList) {

queue<int> q;

q.push(ver);

vector<bool> vis(n, false);

vis[ver] = true;

while (!q.empty()) {

int node = q.front();

q.pop();

cout << node << "\t";

vector<int> ::iterator it;

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

{

if (!vis[\*it]) {

vis[\*it] = 1;

q.push(\*it);

}

}

}

}