Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №7

по курсу «Логика и основы алгоритмизации в инженерных задачах»

## на тему «Поиск расстояний во взвешенном графе»

Выполнили:

студенты группы 20ВВ2

Зиновьев Я.М.

Аверочкин П.С.

Приняли:

к. т. н. доцент Юрова О.В.

д. т. н., профессор Митрохин М.А.

Пенза 2021

**Цель работы**

Реализовать алгоритм поиска расстояний во взвешенном графе.

**Лабораторное задание**

**Задание 1**

1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного взвешенного графа G. Выведите матрицу на экран.

2. Для сгенерированного графа осуществите процедуру поиска расстояний, реализованную в соответствии с приведенным выше описанием. При реализации алгоритма в качестве очереди используйте класс queue из стандартной библиотеки С++.

3.\* Сгенерируйте (используя генератор случайных чисел) матрицу смежности для ориентированного взвешенного графа G. Выведите матрицу на экран и осуществите процедуру поиска расстояний, реализованную в соответствии с приведенным выше описанием.

**Описание программы:**

Программа осуществляет поиск расстояний для каждой вершины графа. Реализована данная функция с помощью обхода в ширину.  
Создается массив, который представляет нам матрицу смежности ориентированного и неориентированного графа.  
Подсчет расстояний производится в отдельный массив, члены которого выводятся на экран.

### **Листинг**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <iostream>

#include <locale.h>

#include <malloc.h>

#include <queue>

using namespace std;

void BFSD(int\*\* G, int n, int num, int\* dist)

{

    queue <int> q;

    q.push(num);

    dist[num] = 0;

    while (!q.empty())

    {

        num = q.front();

        q.pop();

        cout << num + 1 << " ";

        for (int i = 0; i < n; i++)

        {

            if ((G[num][i] > 0) && (dist[i] > dist[num] + G[num][i]))

            {

                q.push(i);

                dist[i] = dist[num] + G[num][i];

            }

        }

    }

}

void main()

{

    setlocale(LC\_ALL, "RUS");

    srand(time(NULL));

    int n, num, v; //N-кол-во вершин, num-вершина для ввода, v-вершина для ввода

    int\*\* G1, \*\* G2;

    cout << "\nУкажите размер матрицы:";

    cin >> n;

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

    G1 = (int\*\*)malloc(n \* sizeof(int\*));

    for (int i = 0; i < n; i++)

    {

        G1[i] = (int\*)malloc(n \* sizeof(int));

    }

    for (int i = 0; i < n; i++)

    {

        G1[i][i] = 0;

        for (int j = i + 1; j < n; j++)

        {

            G1[i][j] = rand() % 11;

            G1[j][i] = G1[i][j];

        }

    }

    cout << "\n\nВывод матрицы смежности:\n";

    for (int i = 0; i < n; i++)

    { // вывод матриц смежности

        cout << endl;

        for (int j = 0; j < n; j++) {

            cout << G1[i][j] << " ";

        }

    }

    for (int i = 0; i < n; i++)

    {

        dist[i] = 1000;

    }

    cout << endl << endl;

    for (int i = 0; i < n; i++)

    {

        BFSD(G1, n, i, dist);

        cout << "\nРасстояние от вершины " << i+1 << "до остальных:\n";

        for (int j = 0; j < n; j++)

        {

            cout << dist[j];

        }

        for (int k = 0; k < n; k++)

        {

            dist[k] = 1000;

        }

    }

    G2 = (int\*\*)malloc(n \* sizeof(int\*));

    for (int i = 0; i < n; i++)

    {

        G2[i] = (int\*)malloc(n \* sizeof(int));

    }

    for (int i = 0; i < n; i++)

    {

        for (int j = 0; j < n; j++)

        {

            G2[i][j] = rand() % 11;

        }

        G2[i][i] = 0;

    }

    cout << "\n\nВывод матрицы смежности ориентированного графа:\n ";

    for (int i = 0; i < n; i++) { // вывод матриц смежности

        cout << endl;

        for (int j = 0; j < n; j++) {

            cout << G2[i][j] << " ";

        }

    }

    for (int i = 0; i < n; i++)

    {

        dist[i] = 1000;

    }

    cout << endl << endl;

    for (int i = 0; i < n; i++) {

        BFSD(G2, n, i, dist);

        cout << "\nРасстояние от вершины " << i + 1 << " до остальных : \n";

        for (int j = 0; j < n; j++)

        {

            cout << dist[j];

        }

        for (int k = 0; k < n; k++)

        {

            dist[k] = 1000;

        }

    }

}

**Результаты работы программы**

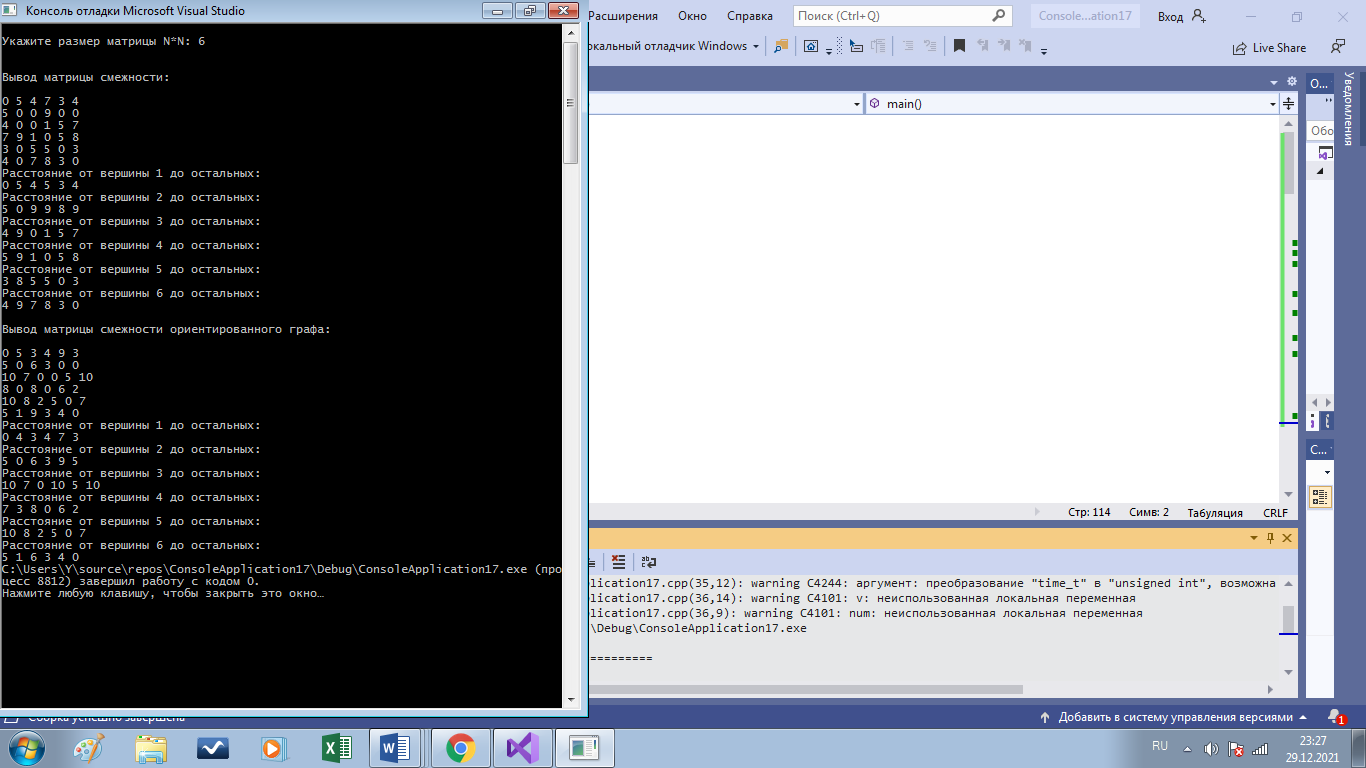


Рисунок 1

## Вывод

В ходе выполнения данной лабораторной работы мы научились работать с алгоритмом поиска расстояний во взвешенном графе. Научились реализовывать его с помощью стандартной библиотеки <queue>.