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 «Унарные и бинарные операции над графом»

Выполнил:

студент группы 19ВВ2

Гусев Д. О.

Приняли:

Митрохин М. А.

Юрова О.В.

Пенза 2020

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.

### **Листинг**

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

#include <queue>

#include <time.h>

using namespace std;

void vivod(int n, int\*\* p) {

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

{

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

{

printf("%d ", p[i][j]);

}

printf("\n");

}

}

void bfsd(int\*\* p, int n, int\* dist, int v)

{

queue <int> q{};

v--;

q.push(v);

dist[v] = 0;

while (!q.empty())

{

v = q.front();

q.pop();

cout << v + 1 << " ";

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

{

if (p[v][i] > 0 && dist[i] > dist[v] + p[v][i])

{

q.push(i);

dist[i] = dist[v] + p[v][i];

}

}

}

}

int main()

{

setlocale(LC\_ALL, "Rus");

int n, v;

cout << "Введите размер генерируемой неориентированной матрицы: ";

cin >> n;

int\*\* p, \*\* p2;

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

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

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

{

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

}

srand(time(NULL));

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

{

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

{

if (p[i][j] == 0)

{

p[i][j] = 0;

}

else

{

p[i][j] = rand() % 11;

}

}

}

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

{

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

{

p[i][j] = p[j][i];

}

}

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

{

dist[i] = 1000;

}

cout << "Выберите вершину ";

cin >> v;

cout << "исходная матрица" << endl;

vivod(n, p);

cout << "Перебранные вершины обходом в ширину: " << endl;

bfsd(p, n, dist, v);

cout << endl << "Расстояния из " << v << "-й вершины" << endl;

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

{

cout << dist[i];

cout << " ";

}

cout << endl;

cout << "Введите размер генерируемой ориентированной матрицы: ";

cin >> n;

p2 = (int\*\*)malloc(n \* sizeof(int\*));

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

{

p2[i] = (int\*)malloc(n \* sizeof(int));

}

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

{

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

{

if (p2[i][j] == 0)

{

p2[i][j] = 0;

}

else

{

p2[i][j] = rand() % 11;

}

}

}

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

{

dist[i] = 1000;

}

cout << "Выберите вершину ";

cin >> v;

cout << "исходная матрица" << endl;

vivod(n, p2);

cout << "Перебранные вершины обходом в ширину: " << endl;

bfsd(p2, n, dist, v);

cout << endl << "Расстояния из " << v << "-й вершины" << endl;

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

{

cout << dist[i];

cout << " ";

}

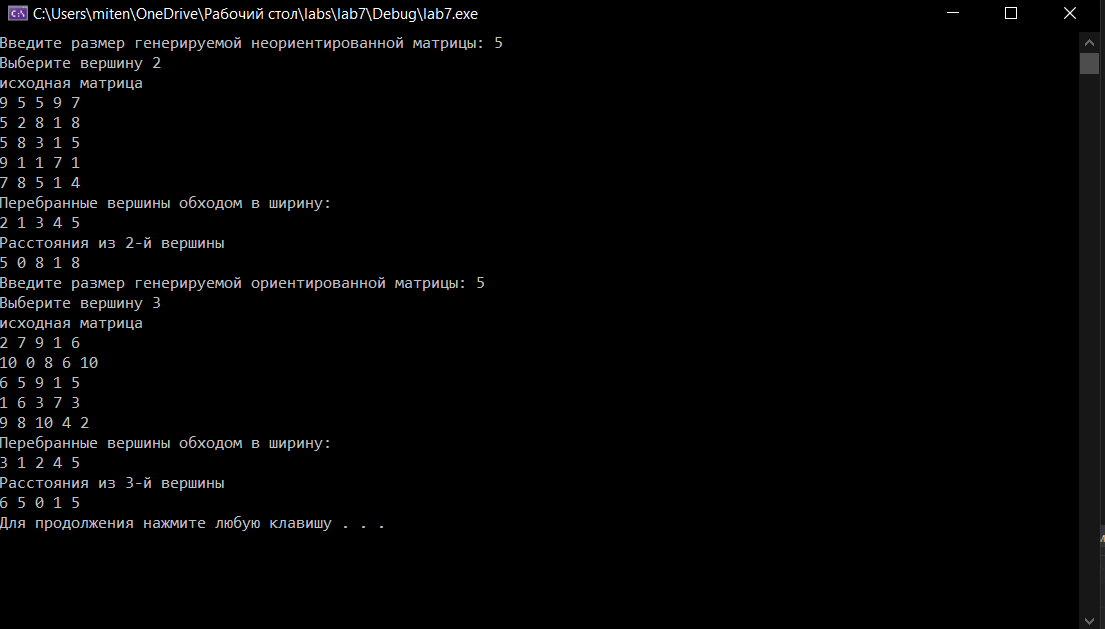
cout << endl;

system("pause");

return 0;

}

**Результаты работы программы**



## Вывод

Освоили алгоритм поиска расстояний во взвешенном графе. Научились реализовывать его с помощью стандартной библиотеки <queue>.