Пензенский государственный университет

Кафедра «Вычислительная техника»

**Отчёт**

По лабораторной работе №7

по дисциплине: «Логика и основы оптимизации в инженерных задачах»

## на тему «Поиск расстояний во взвешенном графе»

Выполнил студент группы 19ВВ4:

Шахов К.М

Приняли:

Митрохин М. А.

Юрова О.В.

Пенза 2020

**Цель работы:** Освоить алгоритм поиска расстояний во взвешенном графе.

**Общие сведения:**

Во взвешенном графе в отличие от не взвешенного каждое ребро имеет вес, отличный от нуля. Поэтому в матрице смежности взвешенного графа содержится информация не только о наличии ребра, но и о его весе.

Поиск расстояний между вершинами в таком графе также возможно построить используя процедуры обхода графа. Отличие от поиска расстояний в не взвешенном графе будет состоять в том, что при обновлении расстояния до вершины при ее посещении оно будет увеличиваться не на 1, а на величину веса ребра.

**Практическая часть**

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++.

**Листинг :**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <stdio.h>

#include <conio.h>

#include <malloc.h>

#include <stdlib.h>

#include <time.h>

#include <queue>

using namespace std;

int\*\* G;

int\* dist;

int ver;

void BFSD(int s, int rows) {

int v;

queue <int> q;

q.push(s);

dist[s] = 0;

while (!q.empty()) {

v = q.front();

q.pop();

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

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

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

q.push(i);

}

}

}

}

int main()

{

int rows;

int n = 0;

int k = 0;

system("cls");

printf("rows:");

scanf("%d", &rows);

srand(time(NULL));

G = (int\*\*)malloc(rows \* sizeof(int\*));

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

{

G[i] = (int\*)malloc(rows \* sizeof(int));

}

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

{

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

{

G[i][j] = rand() % 5;

if (i == j) {

G[i][j] = 0;

}

G[j][i] = G[i][j];

}

}

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

{

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

{

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

}

printf("\n");

}

printf("\n");

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

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

dist[i] = 1000;

}

printf("\n");

printf("Start Vershina:");

scanf("%d", &ver);

printf("\n");

BFSD(ver, rows);

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

printf("%d ", dist[i]);

}

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

free(G[i]);

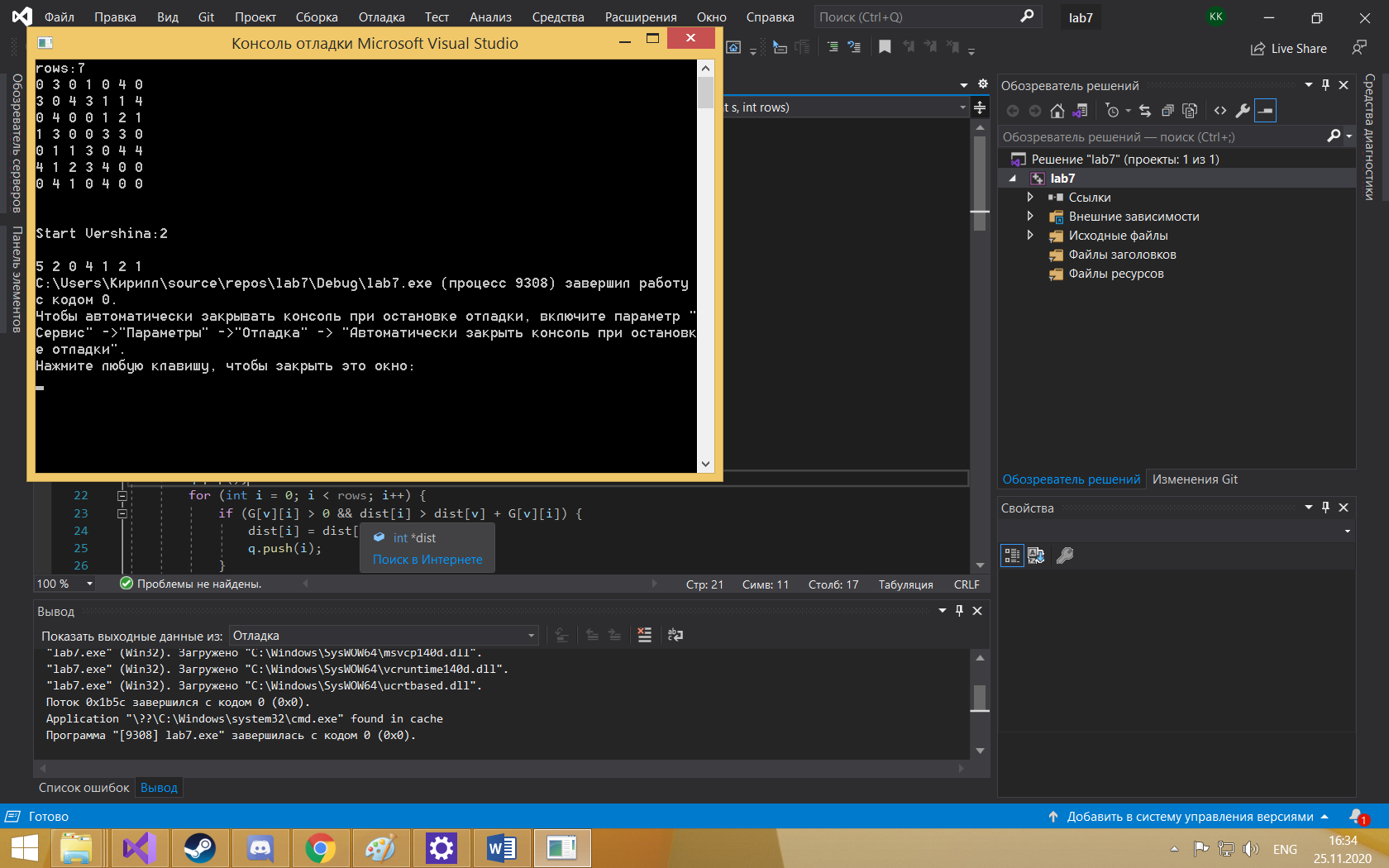
free(G);

getchar();

return 0;

}

## Результат работы.



**Вывод:** Освоили алгоритм поиска расстояний во взвешенном графе.