Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №7

# по дисциплине: «Логика и основы алгоритмизации в инженерных задачах»

на тему: «Поиск расстояний во взвешенном графе»

Выполнили:

студенты группы 20ВВ3

Горожанин Я. А.

Тихонов А. А.

Скирдова В. М.

Приняли:

Юрова О.В.

Митрохин М.А.

Пенза 2021

**Название**

Поиск расстояний во взвешенном графе

**Цель работы**

Разработка алгоритма 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.

**Задание 2\***

1. Модернизируйте программу так, чтобы получить возможность запуска программы с параметрами командной строки (см. описание ниже). В качестве параметра должны указываться тип графа (взвешенный или нет) и наличие ориентации его ребер (есть ориентация или нет).

**Листинг**

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include <locale.h>

#include <iostream>

#include <stack>

#include <queue>

using namespace std;

struct node {

int numb;

node\* next;

};

struct graf {

node\*\* nodes;

int size;

};

void Google\_search(int\*\* a, int num, int\* dist, int size) {

queue <int> q;

dist[num] = 0;

q.push(num);

while (!q.empty()) {

num = q.front();

q.pop();

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

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

q.push(i);

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

}

}

}

}

int main(int argc, char\* argv[]) {

srand(time(0));

setlocale(LC\_ALL, "Russian");

int rand\_limit;

if (strcmp(argv[1], "no"))

rand\_limit = 100;

else

rand\_limit = 2;

int size;

cout << " Введите размерность матрицы: ";

cin >> size;

cout << endl;

int\*\* arr = new int\* [size];

for (int count = 0; count < size; ++count)

arr[count] = new int[size];

if (strcmp(argv[2], "no")) {

for (int row = 0; row < size; row++) {

arr[row][row] = 0;

for (int col = row + 1; col < size; col++) {

arr[row][col] = rand() % rand\_limit;

arr[col][row] = rand() % rand\_limit;

}

}

}

else {

for (int row = 0; row < size; row++) {

arr[row][row] = 0;

for (int col = row + 1; col < size; col++) {

arr[row][col] = rand() % rand\_limit;

arr[col][row] = arr[row][col];

}

}

}

cout << " ";

for (int row = 0; row < size; row++)

{

for (int col = 0; col < size; col++) {

cout << arr[row][col] << " ";

}

cout << endl << " ";

}

cout << endl;

auto distG = new int[size];

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

distG[i] = INT\_MAX;

cout << endl << " Поиск кратчайшего пути в графе, представленным через матрицу смежности, реализованный на основе алгоритма поиска в ширину с использованием очереди" << endl << endl;

cout << " Введите номер вершины, с которой хотите начать обход: ";

int start;

cin >> start;

Google\_search(arr, start, distG, size);

cout << endl;

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

if (distG[i] == -1)

cout << " Кратчайший путь до вершины " << i << " = " << 0 << endl;

else

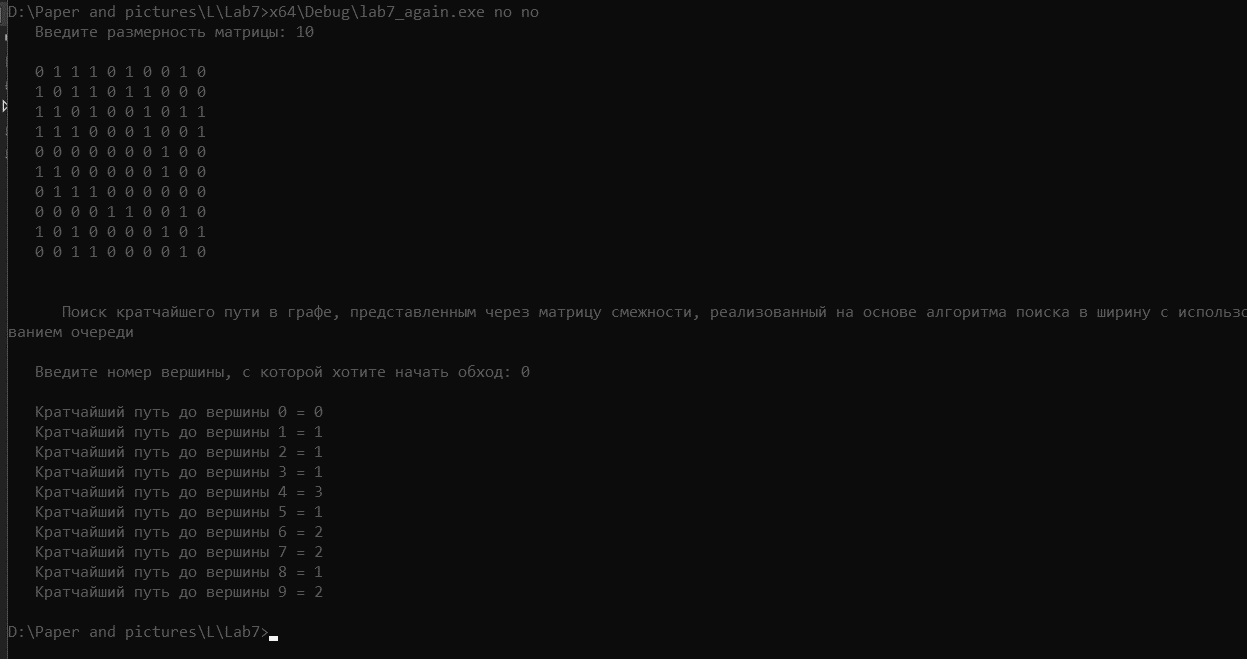
cout << " Кратчайший путь до вершины " << i << " = " << distG[i] << endl;

return 0;

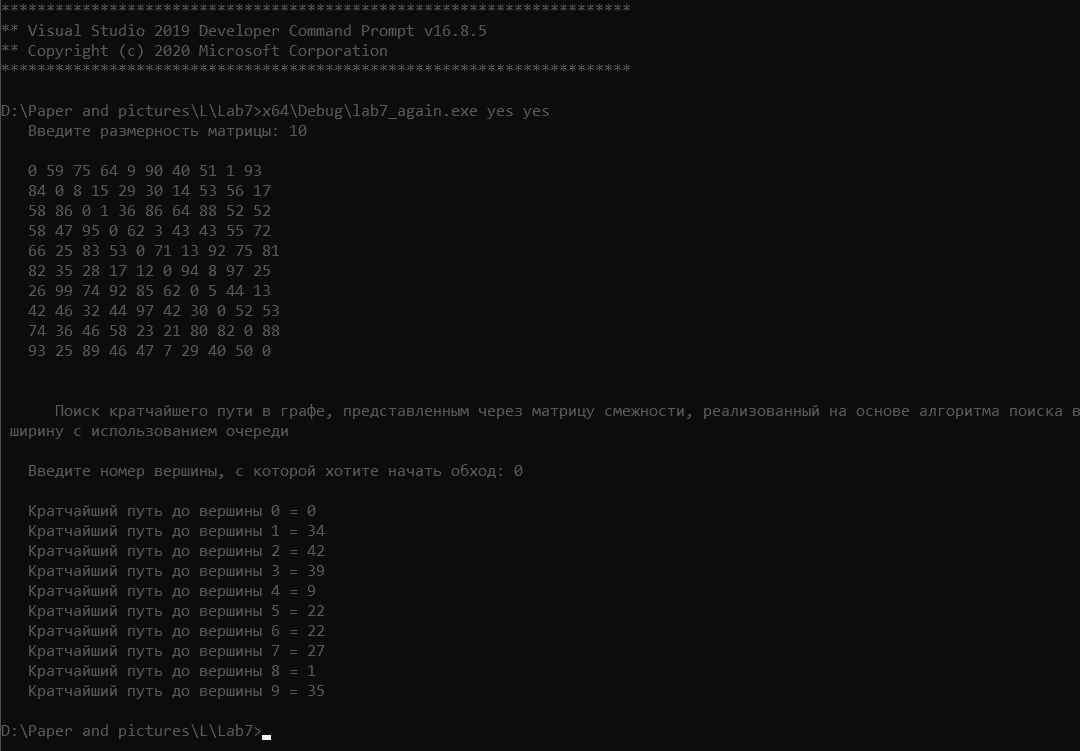
}

**Результат работы программы**

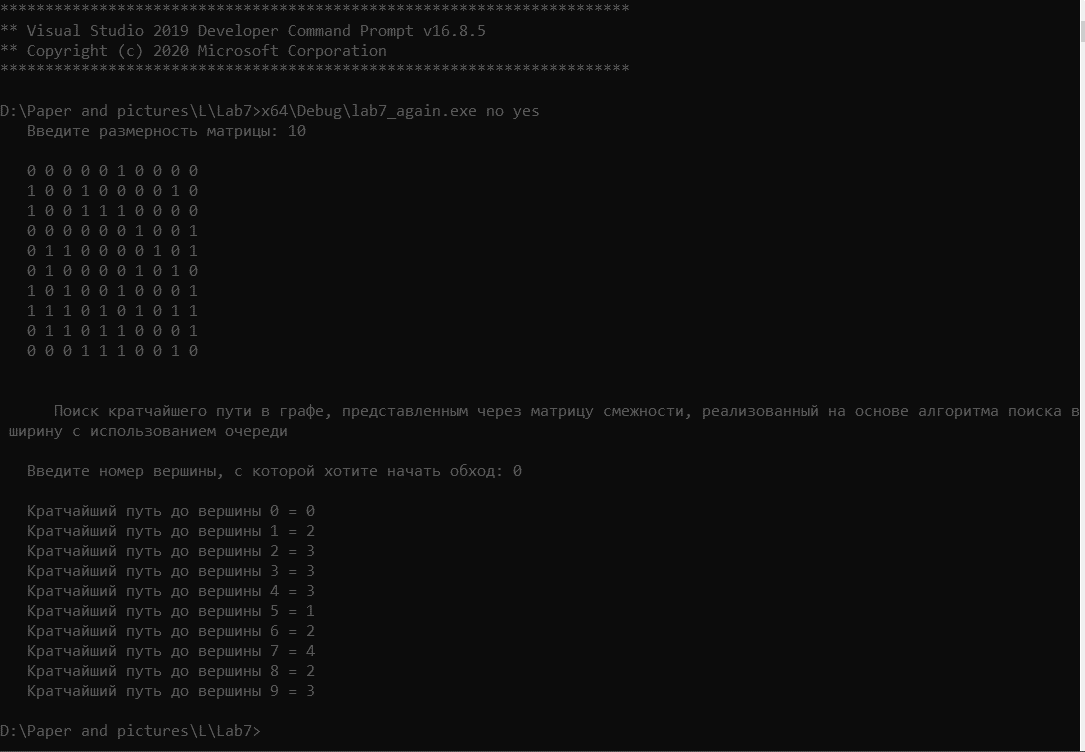
1. Невзвешенный неориентированный граф.



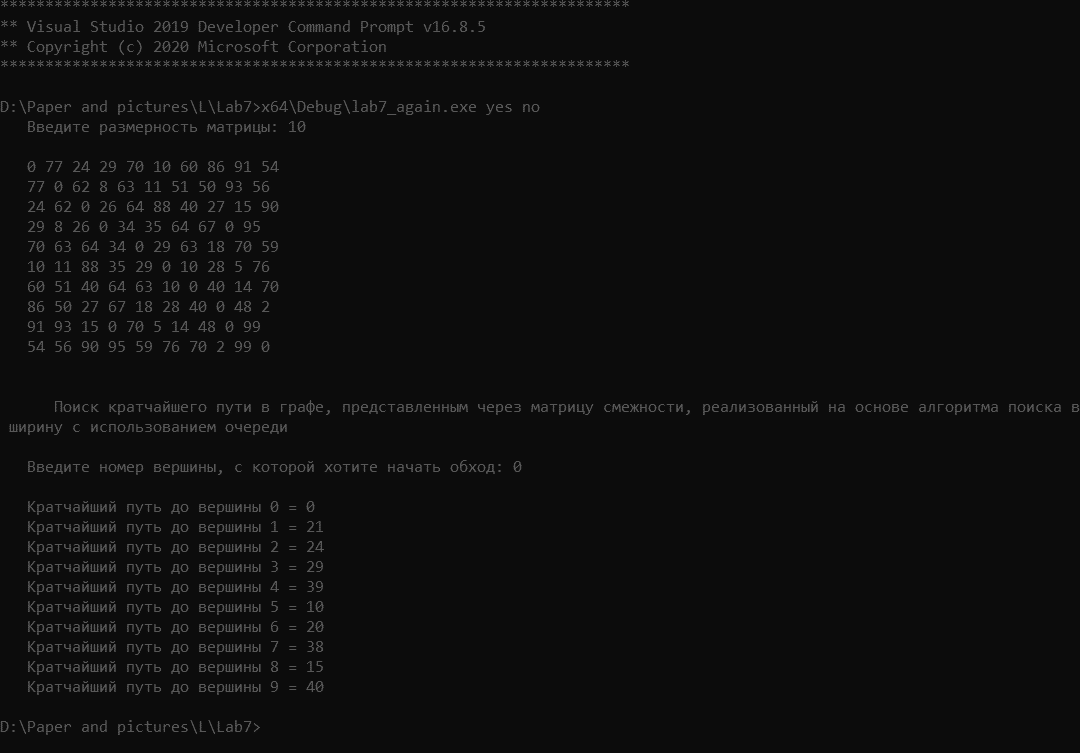
1. Взвешенный ориентированный граф.



1. Невзвешенный ориентированный граф.



1. Взвешенный неориентированный граф.



1. Сравнение времени выполнения функций.





**Вывод**

В ходе выполнения данной лабораторной работы были получены навыки разработки алгоритма поиска кратчайших расстояний в ориентированном/неориентированном взвешенном/невзвешенном графах, реализованного на основе алгоритма поиска в ширину с использованием очереди, а также была создана функция с возможностью запуска программы с параметрами командной строки. Также выяснили, что выполнение функции с использованием приоритетной очереди почти в полтора раза быстрее, чем с использованием функции очереди из стандартной библиотеки С++.