Министерство образования Российской Федерации   
Пензенский государственный университет   
Кафедра «Вычислительная техника»

ОТЧЕТ

По лабораторной №7   
На тему: «Поиск расстояний во взвешенном графе»  
По дисциплине: «Логика и основы алгоритмизации в инженерных задачах»

Выполнили студенты группы 20ВВ2:

Борунов Д.А

Сурков М.С

Приняли:

Митрохин М.А. Юрова О.В.

**Пенза 2020г**

**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).

**Методы решения заданий**

**Задание 1.1 и 1.2.**

Создать матрицу смежности графа (двумерный динамический массив). Описать функцию:

void BSFD(int st, int\*\* b, int num\_of\_elem, int \*vis)

int st **–**текущая и стартовая вершина

int\*\* b – указатель на сгенерированную матрицу смежности

int \*vis – указатель на массив посещенных вершин

int num\_of\_elem – количество элементов в сгенерированной матрице 11

1. Перед вызовом функции **BSFD** заполнить массив не посещенных вершин 1000(1000 индикатор того, что вершина не посещена, так как 0 и -1 может быть расстоянием).
2. В теле функции реализовать алгоритм обхода графа в ширину на основе очереди Q с небольшими изменениями. В теле цикла, проверяющего наличие смежной вершины, не посещенной по отношению к текущей, изменить условие на if(b[t][i]>0 && vis[i]==1000). При выполнении условия обновить расстояние для смежной посещенной вершины следующим образом: vis[i] = vis[t] + b[t][i]; Затем поместить ее в очередь, пометив текущей.
3. Вывести массив посещенных вершин в консоль

**Задание 1.3**

Процедура генерации ориентированного графа отличается от процедуры генерации **Не**ориентированного лишь тем, что матрица смежности не обязательно должна быть зеркальной

Код для генерации матрицы смежности ориентированного графа:

for (i = 0; i < max\_size; i++)

{

for (j = 0; j < max\_size; j++)

{

num = rand() % 7;

if (num <= 3)

{

a[i][j] = 0;

}

else

{

a[i][j] = (rand() % 6) + 1;

}

}

}

Код для генерации матрицы смежности НЕориентированного графа:

for (i = 0; i < max\_size; i++)

{

for (j = i + 1; j < max\_size; j++)

{

num = rand() % 7;

if (num <= 3)

{

a[i][j] = 0;

a[j][i] = a[i][j];

}

else

{

a[i][j] = (rand() % 6) + 1;

a[j][i] = a[i][j];

}

}

}

Алгоритм поиска расстояний во взвешенном графе идентичен для матриц смежности ориентированного и Неориентированного графа.

**Задание 3.1**

Модернизация программы для запуска с параметрами:

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

setlocale(LC\_ALL, "Rus");

srand(time(NULL));

if (argc == 2)

{

if (strcmp(argv[1], "Neor") == 0)

no\_orient\_graph();

else if (strcmp(argv[1], "Or") == 0)

orient\_graph();

}

else if (argc == 3)

{

if (strcmp(argv[1], "Neor") == 0 and strcmp(argv[2], "Or") == 0)

{

no\_orient\_graph();

orient\_graph();

}

else if (strcmp(argv[1], "Or") == 0 and strcmp(argv[2], "Neor") == 0)

{

orient\_graph();

no\_orient\_graph();

}

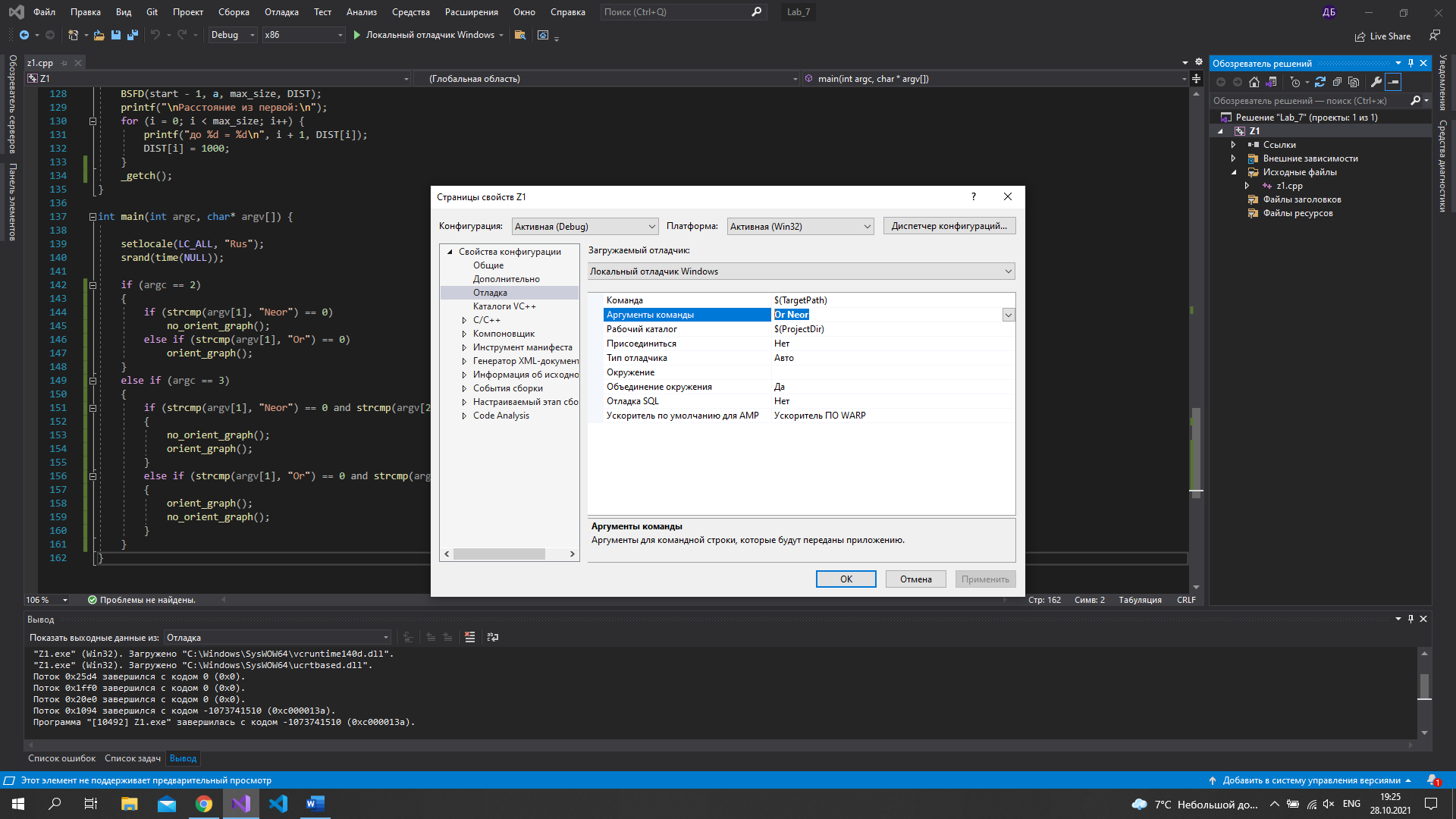
}

}

На входе функция main принимает 1 или 2, после проверки количества аргументов выполняется одна из функций orient\_graph(); и no\_orient\_graph();, либо выполняются обе в порядке заданных оргументов

* Параметр **Or –** запускает функцию orient\_graph(); которая генерирует матрицу смежности взвешенного ориентированного графа и выполняет поиск расстояний в нем.
* Параметр **Neor –** запускает функцию no\_orient\_graph();; которая генерирует матрицу смежности взвешенного НЕориентированного графа и выполняет поиск расстояний в нем.

Аргументы для запуска программы задавались в параметраx отладки Visual Studio



**Листинг программы**

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <cstdlib>

#include <windows.h>

#include <conio.h>

#include <ctime>

#include <cmath>

#include <stack>

#include <queue>

using namespace std;

int max\_size = 7, i, j, start, num;

int\* DIST = (int\*)malloc(max\_size \* sizeof(int));

void BSFD(int st, int\*\* b, int num\_of\_elem, int\* vis) {

queue<int> Q;

int t;

Q.push(st);

vis[st] = 0;

while (!Q.empty())

{

t = Q.front();

Q.pop();

printf("%d ", t + 1);

for (int i = 0; i < num\_of\_elem; i++)

{

if (b[t][i]>0 && vis[i]==1000)

{

vis[i] = vis[t] + b[t][i];

Q.push(i);

}

}

}

}

void no\_orient\_graph()

{

int\*\* a = new int\* [max\_size];

for (i = 0; i < max\_size; i++)

{

a[i] = new int[max\_size];

}

for (i = 0; i < max\_size; i++)

{

for (j = i + 1; j < max\_size; j++)

{

num = rand() % 7;

if (num <= 3)

{

a[i][j] = 0;

a[j][i] = a[i][j];

}

else

{

a[i][j] = (rand() % 6) + 1;

a[j][i] = a[i][j];

}

}

}

printf("\nНеориентированный граф:\n");

for (i = 0; i < max\_size; i++)

{

DIST[i] = 1000;

for (j = 0; j < max\_size; j++)

{

if (i == j)

a[i][j] = 0;

printf("%2d", a[i][j]);

}

printf("\n");

}

printf("\nПоиск расстояний в НЕориентированном графе \ncтартовая вершина - ");

scanf\_s("%d", &start);

BSFD(start - 1, a, max\_size, DIST);

printf("\nРасстояние из первой:\n");

for (i = 0; i < max\_size; i++) {

printf("до %d = %d\n", i + 1, DIST[i]);

DIST[i] = 1000;

}

\_getch();

}

void orient\_graph(){

int\*\* a = new int\* [max\_size];

for (i = 0; i < max\_size; i++)

{

a[i] = new int[max\_size];

}

for (i = 0; i < max\_size; i++)

{

for (j = 0; j < max\_size; j++)

{

num = rand() % 7;

if (num <= 3)

{

a[i][j] = 0;

}

else

{

a[i][j] = (rand() % 6) + 1;

}

}

}

printf("\n Ориентированный граф:\n");

for (i = 0; i < max\_size; i++)

{

DIST[i] = 1000;

for (j = 0; j < max\_size; j++)

{

if (i == j)

a[i][j] = 0;

printf("%2d", a[i][j]);

}

printf("\n");

}

printf("\nПоиск расстояний в Ориентированном графе \ncтартовая вершина - ");

scanf\_s("%d", &start);

BSFD(start - 1, a, max\_size, DIST);

printf("\nРасстояние из первой:\n");

for (i = 0; i < max\_size; i++) {

printf("до %d = %d\n", i + 1, DIST[i]);

DIST[i] = 1000;

}

\_getch();

}

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

setlocale(LC\_ALL, "Rus");

srand(time(NULL));

if (argc == 2)

{

if (strcmp(argv[1], "Neor") == 0)

no\_orient\_graph();

else if (strcmp(argv[1], "Or") == 0)

orient\_graph();

}

else if (argc == 3)

{

if (strcmp(argv[1], "Neor") == 0 and strcmp(argv[2], "Or") == 0)

{

no\_orient\_graph();

orient\_graph();

}

else if (strcmp(argv[1], "Or") == 0 and strcmp(argv[2], "Neor") == 0)

{

orient\_graph();

no\_orient\_graph();

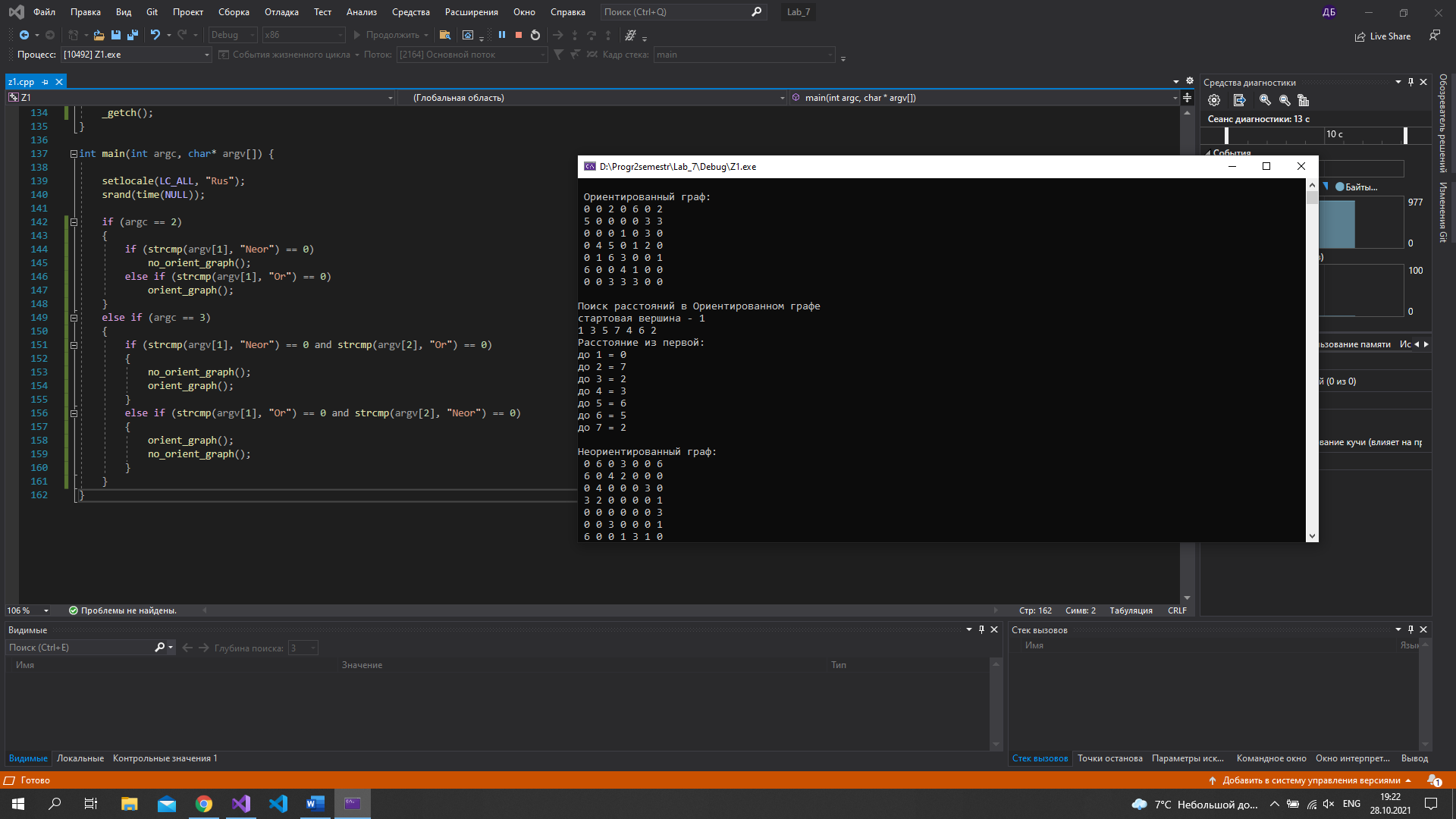
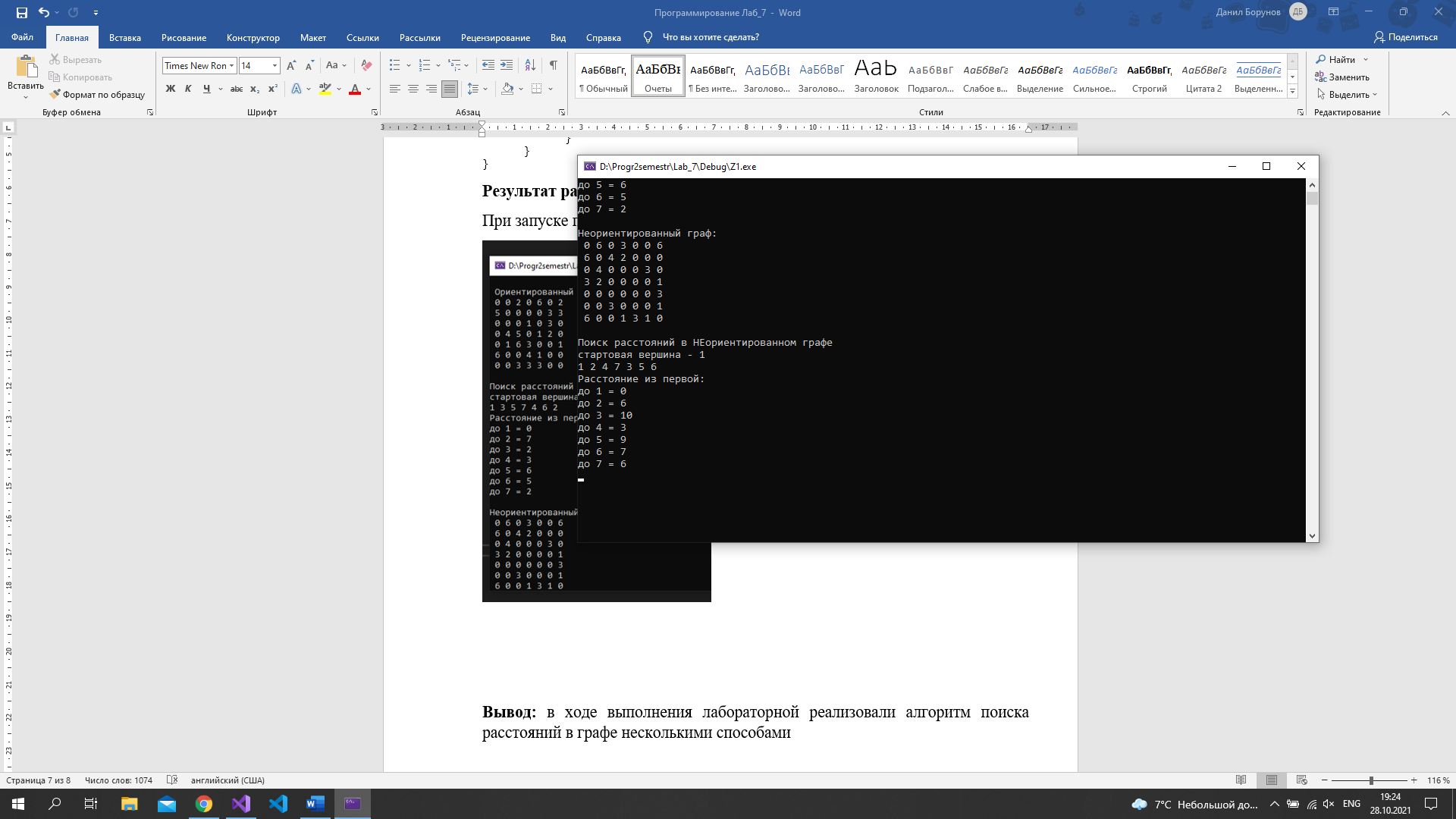
}

}

}

**Результат работы программы**

При запуске передались следующие аргументы: **Or Neor**

**Вывод:** в ходе выполнения лабораторной реализовали алгоритм поиска расстояний во взвешенном ориентированном и НЕориентированном графах. Модернизировали программу для ее запуска с передачей аргументов.