Пензенский Государственный университет

Кафедра «Вычислительной техники»

Отчёт

По лабораторной работе №7

По дисциплине «Л и ОА в ИЗ»

На тему: «Поиск расстояний во взвешенном графе»

***Выполнил студент группы 19ВВ1:***

Гусев В

**Приняли:**

Митрохин М.А.

Юрова О.В.

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

**Задание 1**

1. Сгенерировать (используя генератор случайных чисел) матрицу смежности для неориентированного взвешенного графа *G*. Вывести матрицу на экран.
2. Для сгенерированного графа осуществить процедуру поиска расстояний. При  реализации алгоритма в качестве очереди использовать класс **queue** из стандартной библиотеки С++.

**Ход работы**

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.

**Листинг**

#include "stdafx.h"

#include "iostream"

#include "conio.h"

#include "locale.h"

#include <queue>

using namespace std;

int \*dist; // указатель на вектор

int \*\*matr;//указатель для массива указателей

void BFSD(int s , int n)

{

queue <int> Q;

Q.push(s);

dist[s] = 0;

printf("Порядок обхода:");

while(!Q.empty())

{

s = Q.front();

Q.pop();

printf("%d", s);

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

{

if((matr[s][i] > 0) && dist[i] == -1)

{

Q.push(i);

dist[i] = dist[s] + matr[s][i];

}

}

}

printf("\nРасстояние: ");

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

{

if(dist[i] != -1)

{

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

}

}

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

system("chcp 1251");

system("cls");

int n,i,j,x, Nach;

printf("Введите размер матририцы: ");

scanf("%d", &n);

printf("Введите размер вектора расстояния: ");

scanf("%d", &x);

printf("Введите начальную точку обхода:\n");

scanf("%d", &Nach);

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

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

//выделение памяти под массив указателей

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

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

}

//выделение памяти для массива значений

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

{

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

{

if(i == j)

{

matr[i][j] = 0;

}

if (i < j)

{

matr[i][j] = rand()%6;

matr[j][i] = matr[i][j];

}

}

}

for(i = 0; i < x; i++)

{

dist[i] = -1;

}

printf( "Result: ");

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

{

cout << "\n";

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

printf("%d\t",matr[i][j]);

}

printf( " \n ");

printf( " \n ");

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

{

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

}

printf( " \n ");

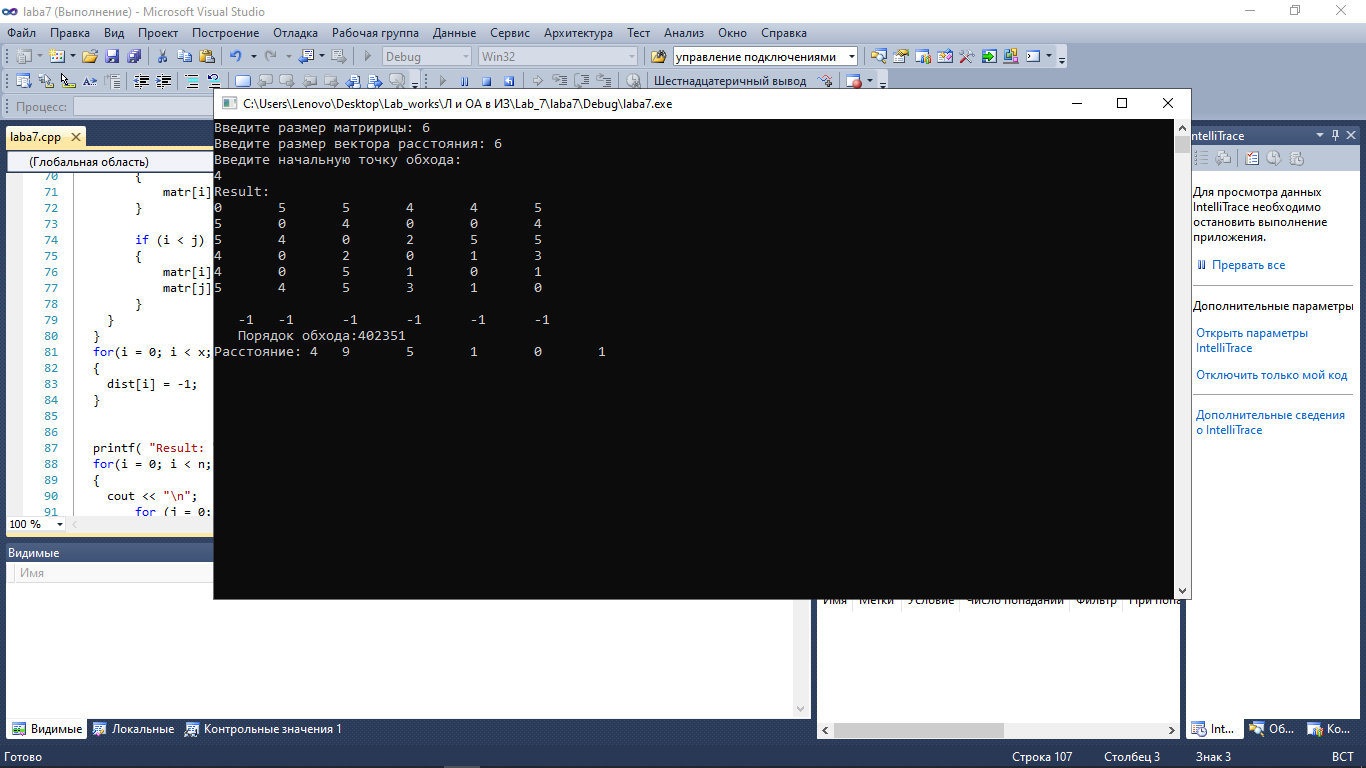
BFSD(Nach, n);

getch();

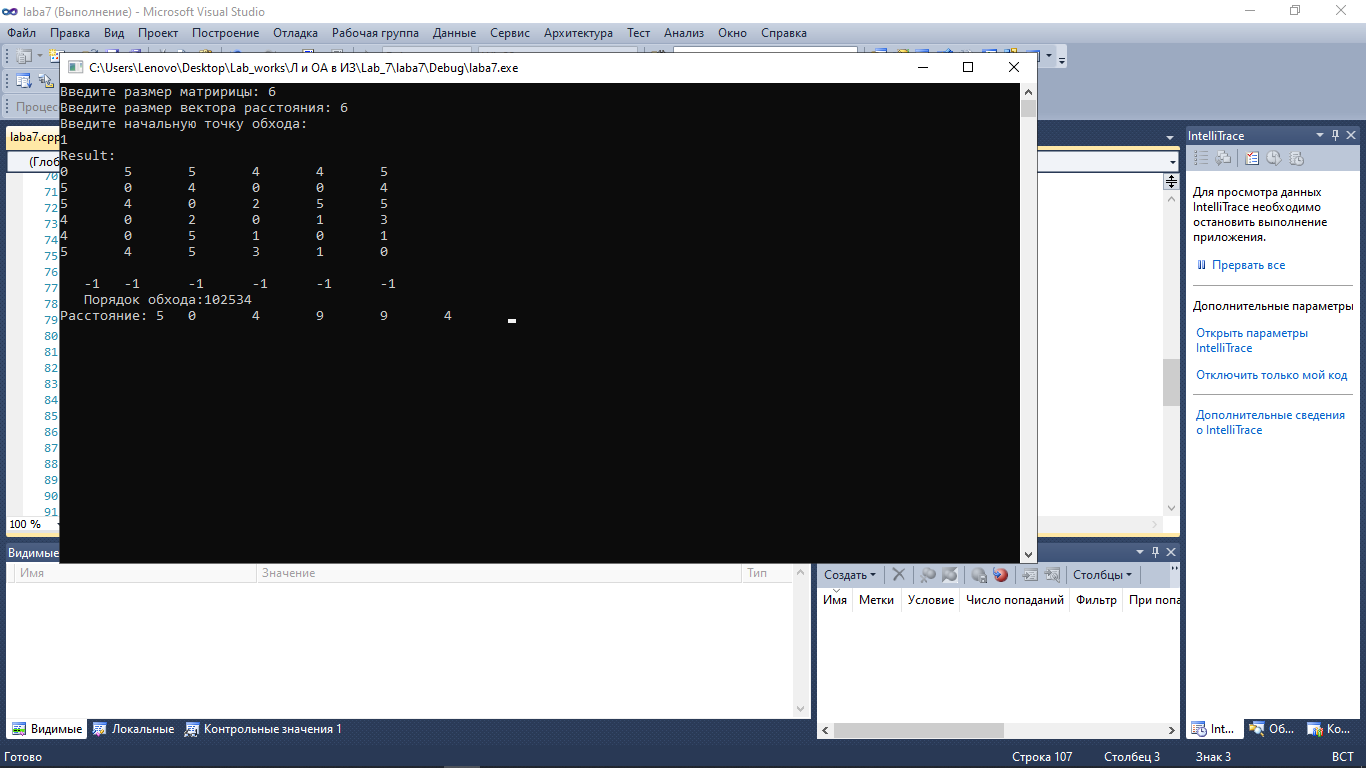
}

**Результат работы программы**

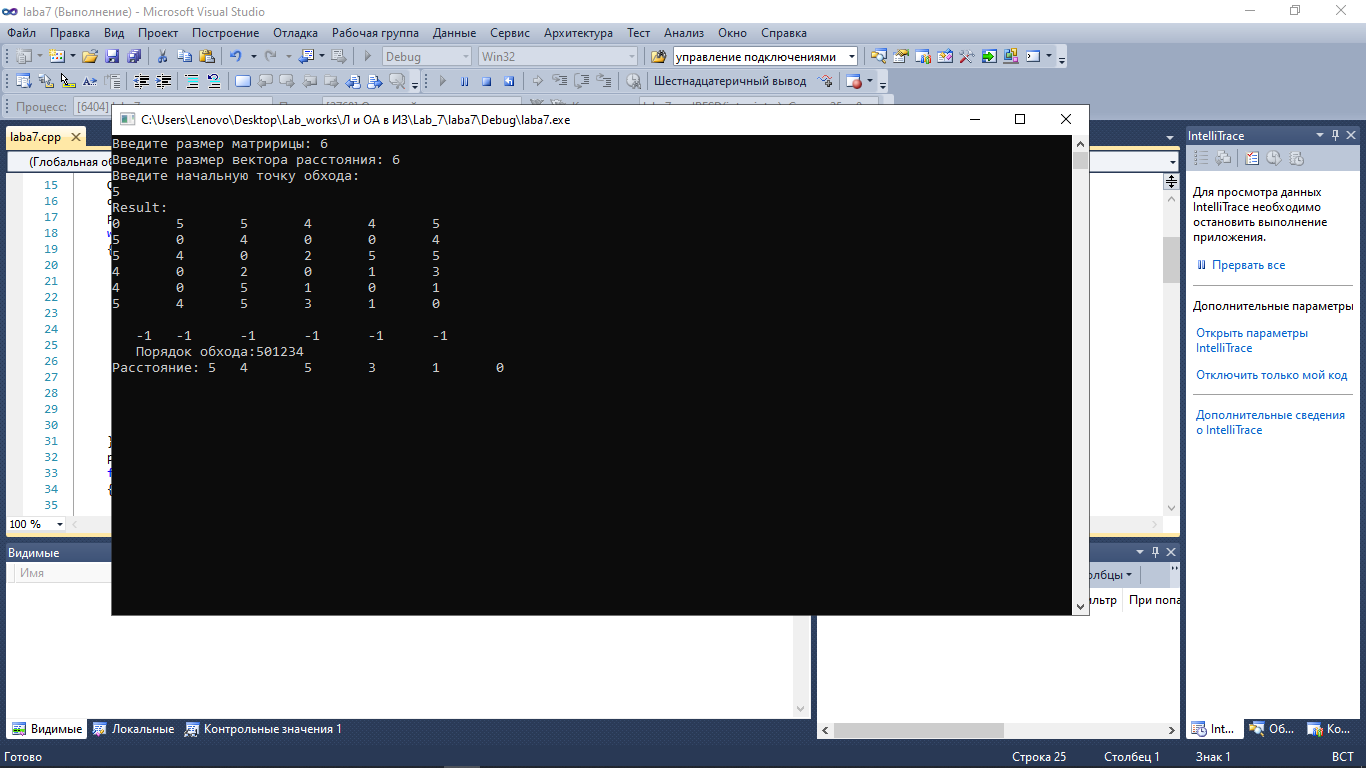
Результат поиска расстояний в графе 6х6, с вершины №4.



Результат поиска расстояний в графе 6х6, с вершины №1.



Результат поиска расстояний в графе 6х6, с вершины №5.



**Вывод:** В ходе данной работы был изучен способ поиска расстояний во взвешенном графе. Так же данный алгоритм был реализован для матрицы , размер которой пользователь может вводить с экрана.