Министерство высшего образования и науки Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

Отчет

По лабораторной работе №7

По курсу «Логика и основы алгоритмизации в инженерных задачах»

На тему «Поиск расстояний во взвешенном графе»

Работу выполнила студентка группы 20ВВ2:

Войнова Д.А.

Приняли:

Юрова О.В.

Митрохин М.А.

Пенза 2021

Ход работы:

Создала матрицу смежности для неориентированного взвешенного графа, осуществила процедуру обхода в ширину для данного графа.

На вход программа получается число вершин в графе, генерирует матрицу смежности, осуществляет обход в ширину с учетом веса ребер.

Результат работы программы представлен на рисунке 1.

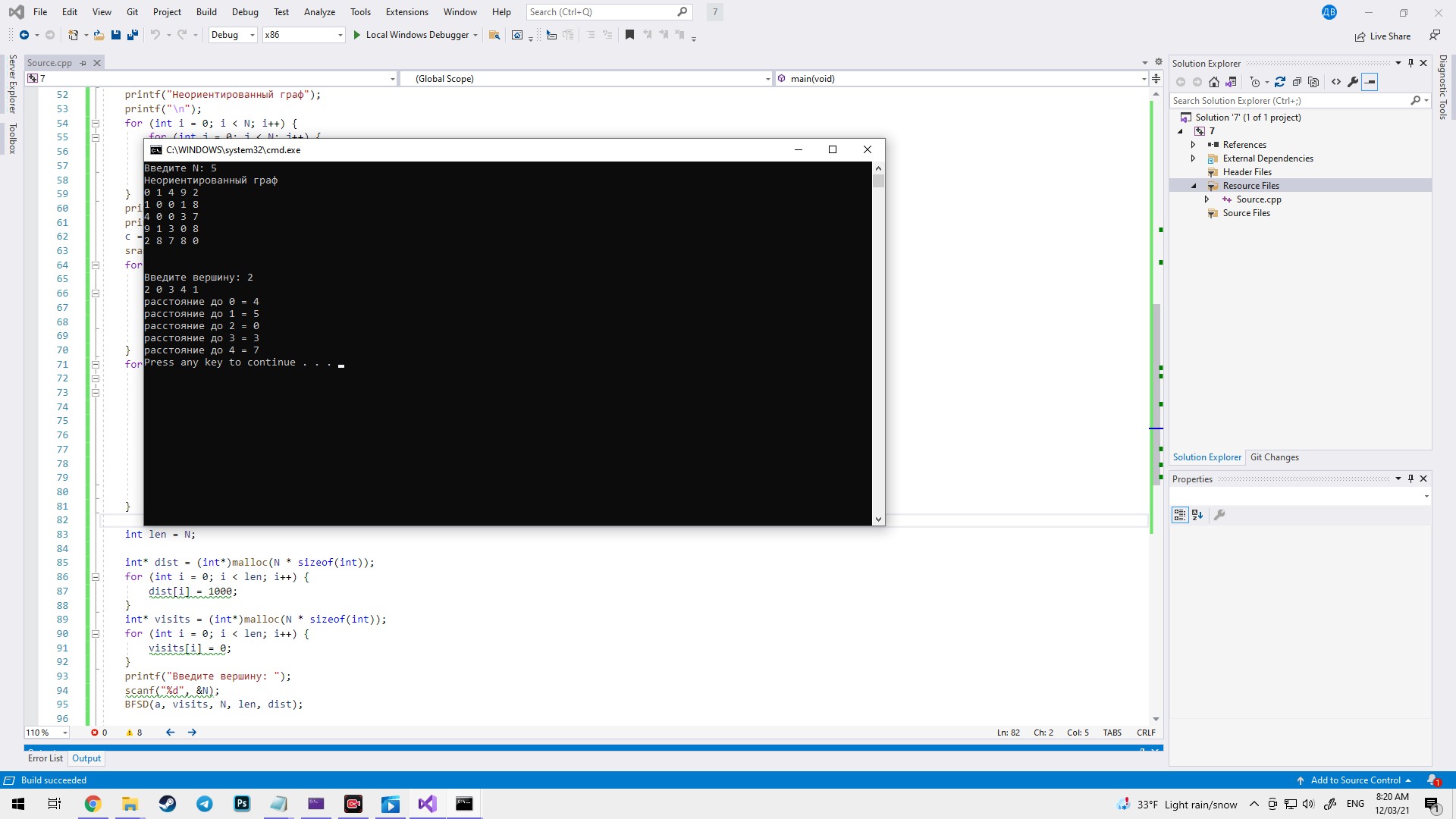


Рис.1 - Результат работы программы

Сгенерировала матрицу смежности для ориентированного взвешенного графа, аналогично осуществила поиск расстояний в ширину.

Результат работы программы представлен на рисунке 2.

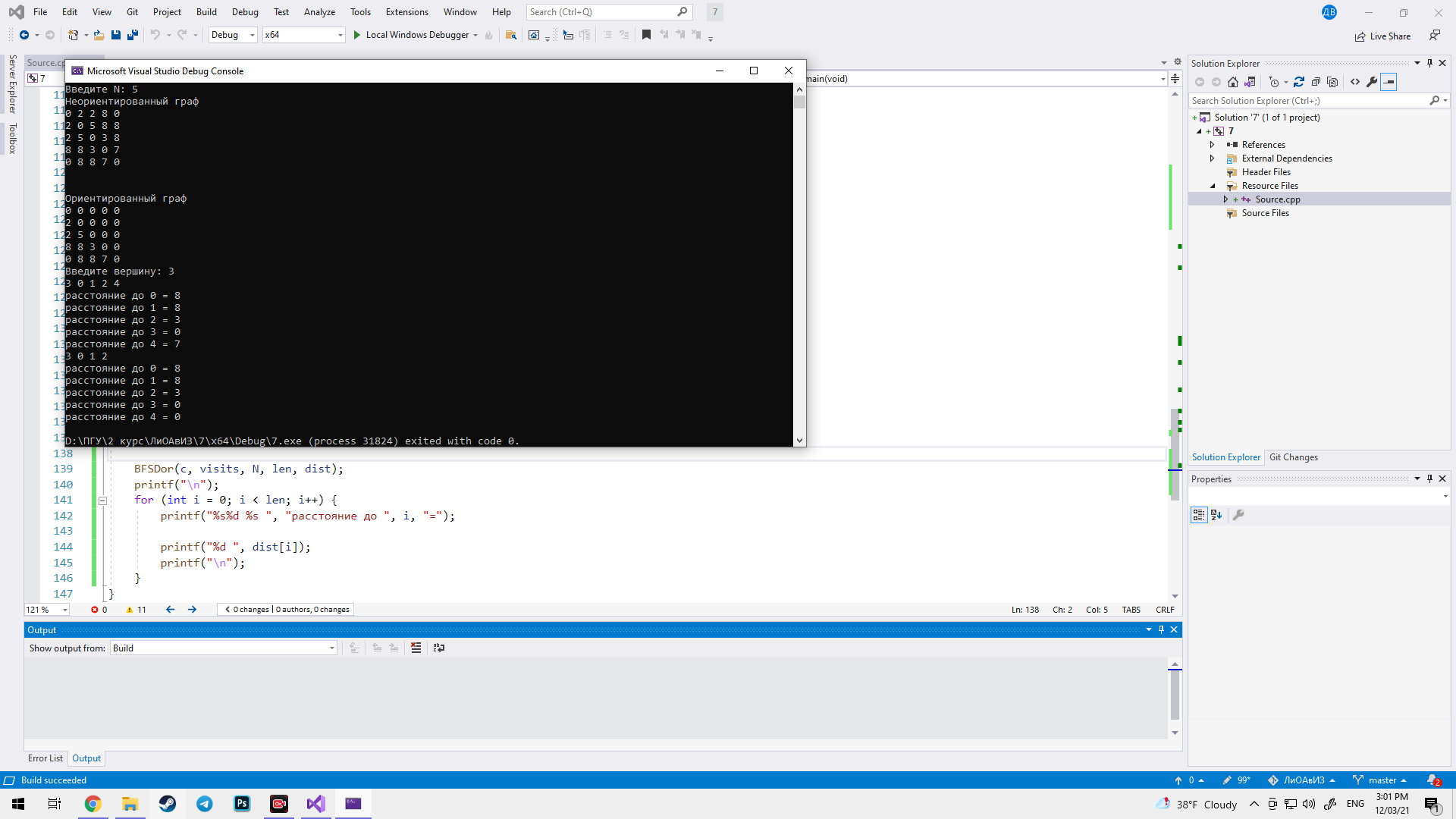


Рис.2 - Результат работы программы

Листинг:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <queue>

#include <locale.h>

using namespace std;

void BFSD(int\*\* G, int\* vis, int num, int n, int\* dist) {

queue <int> q;

int i;

q.push(num);

vis[num] = 0;

dist[num] = 0;

while (!q.empty()) {

num = q.front();

q.pop();

printf("%d ", num);

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

if (vis[i] == 0 && G[num][i] > 0 && dist[i] > dist[num] + G[num][i]) {

q.push(i);

vis[i] = 1;

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

}

}

}

}

void BFSDor(int\*\* G, int\* vis, int num, int n, int\* dist) {

queue <int> q;

int i;

q.push(num);

vis[num] = 0;

dist[num] = 0;

while (!q.empty()) {

num = q.front();

q.pop();

printf("%d ", num);

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

if (vis[i] == 0 && G[num][i] > 0 && dist[i] > dist[num] + G[num][i]) {

q.push(i);

vis[i] = 1;

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

}

if (vis[i] == 0 && G[num][i] == 0) {

dist[i] = 0;

}

}

}

}

int main(void) {

setlocale(LC\_ALL, "Rus");

int\*\* a;

int\*\* c;

int N;

printf("Введите N: ");

scanf("%d", &N);

a = (int\*\*)malloc(N \* sizeof(int));

srand(time(NULL));

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

int\* b = (int\*)malloc(N \* sizeof(int));

for (int j = 0; j < N; j++) {

b[j] = 0;

}

a[i] = b;

}

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

for (int j = 0; j < N; j++) {

if (i != j) {

int num = rand() % 10;

a[i][j] = num;

a[j][i] = num;

}

}

}

printf("Неориентированный граф");

printf("\n");

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

for (int j = 0; j < N; j++) {

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

}

printf("\n");

}

printf("\n");

printf("\n");

c = (int\*\*)malloc(N \* sizeof(int));

srand(time(NULL));

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

int\* d = (int\*)malloc(N \* sizeof(int));

for (int j = 0; j < N; j++) {

d[j] = 0;

}

c[i] = d;

}

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

for (int j = 0; j < N; j++) {

if (i != j) {

int num1 = rand() % 10;

c[i][j] = num1;

c[j][i] = 0;

}

}

}

printf("Ориентированный граф");

printf("\n");

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

for (int j = 0; j < N; j++) {

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

}

printf("\n");

}

int len = N;

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

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

dist[i] = 1000;

}

int\* visits = (int\*)malloc(N \* sizeof(int));

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

visits[i] = 0;

}

printf("Введите вершину: ");

scanf("%d", &N);

BFSD(a, visits, N, len, dist);

printf("\n");

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

printf("%s%d %s ","расстояние до ",i,"=");

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

printf("\n");

} //проход для ориентированного графа

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

dist[i] = 1000;

}

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

visits[i] = 0;

}

BFSDor(c, visits, N, len, dist);

printf("\n");

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

printf("%s%d %s ", "расстояние до ", i, "=");

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

printf("\n");

}

}

Вывод:

В ходе работы была написана программа по поиску расстояний . Поиск осуществляется в ширину для ориентированного и неориентированного взвешенного графа.