|  |
| --- |
| Министерство образования 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» |
|  |
|  |
| Выполнила студентка группы 19ВВ2:  Серов Д.В.  Принял:  Митрохин М. А  Юрова О.В. |
| Пенза  2020 |

**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, реализованную в соответствии с приведенным описанием. При реализации алгоритма в качестве очереди использовать класс **queue** из стандартной библиотеки С++.

**3.**\* Сгенерировать (используя генератор случайных чисел) матрицу смежности для ориентированного взвешенного графа *G*. Вывести матрицу на экран и осуществить процедуру поиска расстояний, реализованную в соответствии с приведенным описанием.

**Результат работы программы:**

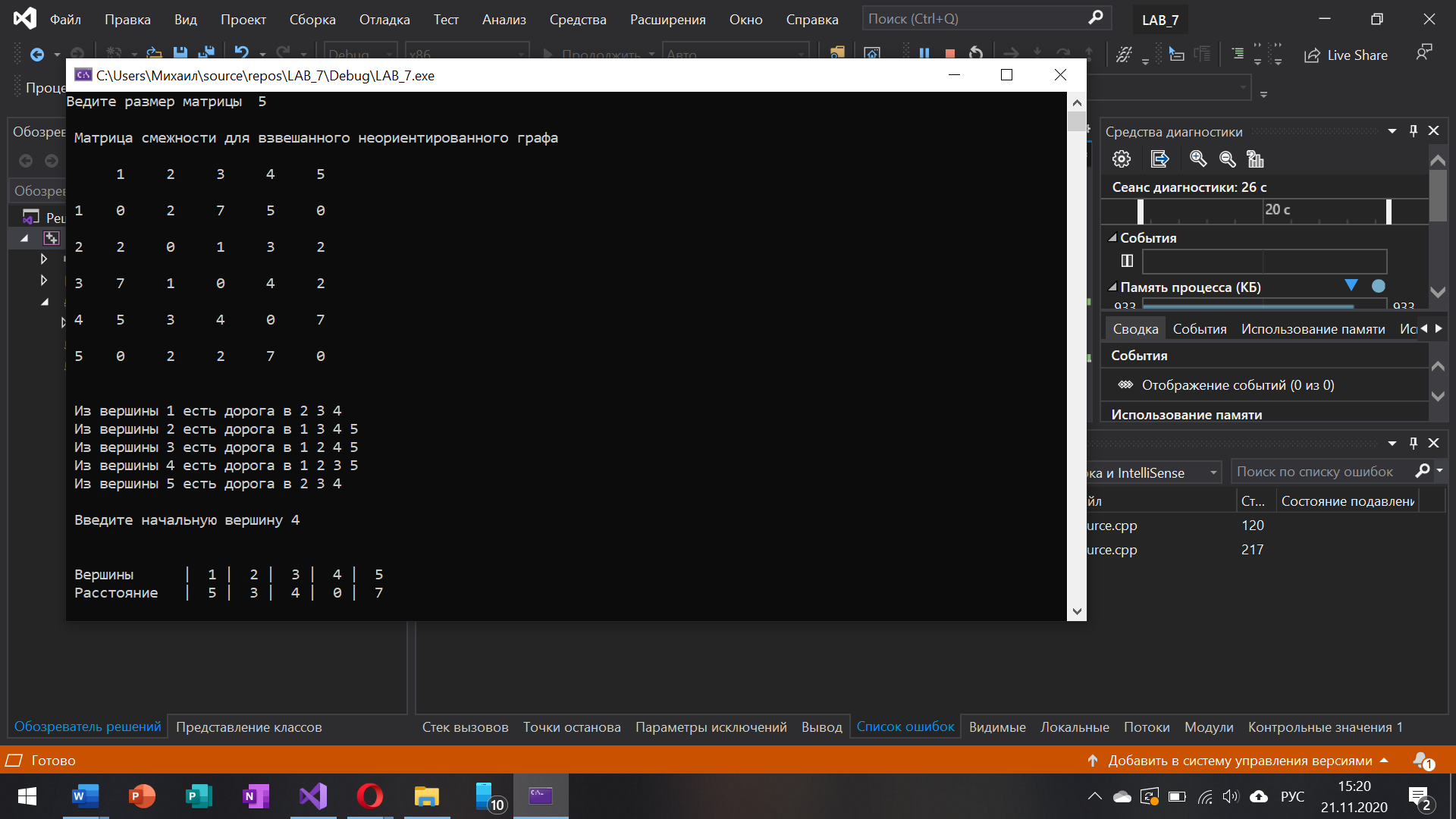


Рис. 1 Поиск расстояний в неориентированном взвешенном графе

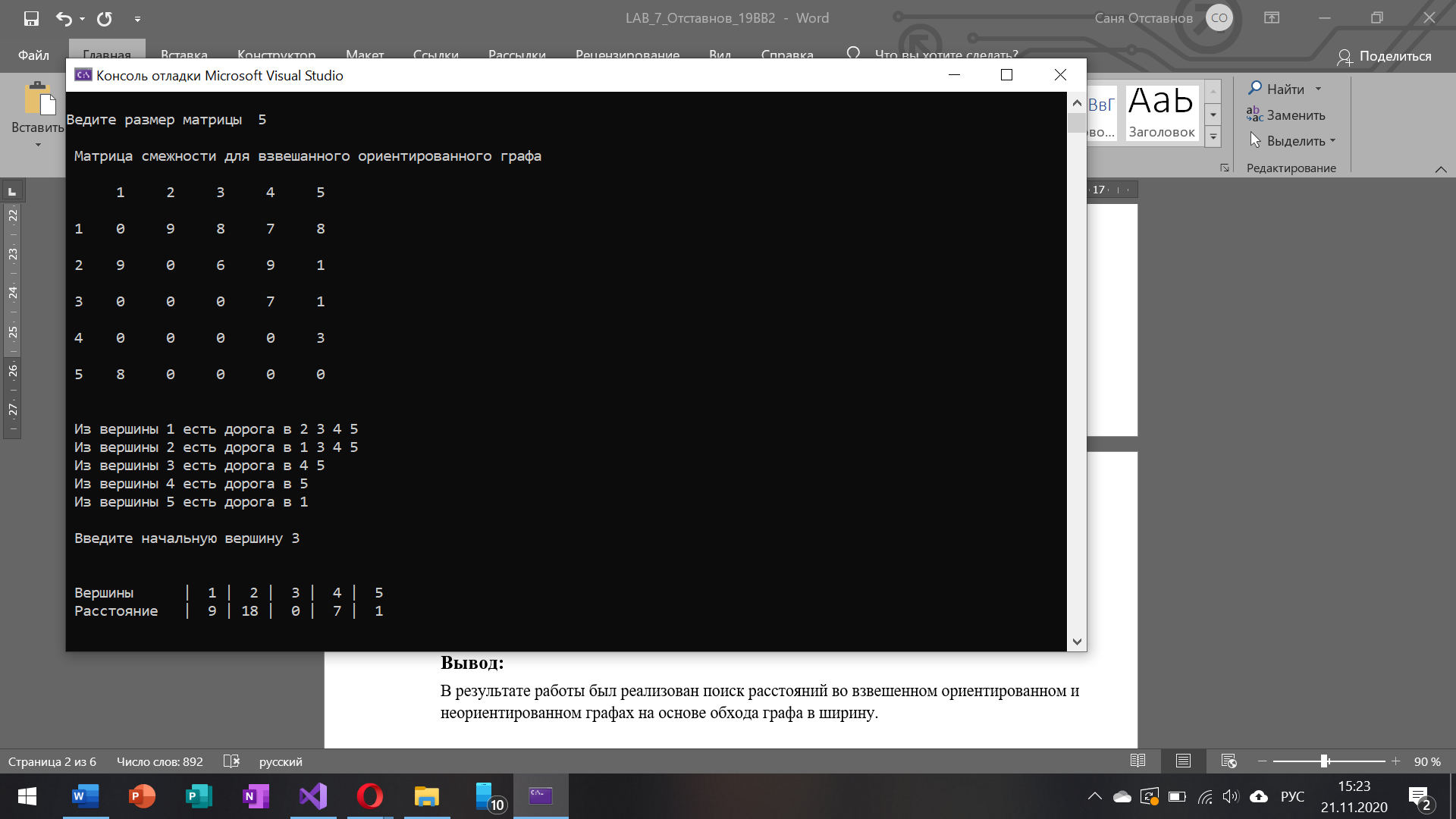


Рис. 2 Поиск расстояний в ориентированном взвешенном графе

**Описание работы программы:**

При запуске программы нас просят ввести размер матрицы смежности для неориентированного взвешенного графа. После чего выводится сама матрица. Для упрощения проверки работы алгоритма на экране появляется вершина и смежная(-ые) ей, если такие есть. Далее нужно ввести вершину, с которой мы хотим начать обход графа в ширину. Если мы введем не существующую вершину, то появится сообщение с просьбой выбрать вершину заново. Следом за ним каждая вершина графа, а под ней расстояние от выбранной нами ранее вершины. Потом происходит все то же самое, но уже для ориентированного взвешенного графа.

**Вывод:**

В результате работы был реализован поиск расстояний во взвешенном ориентированном и неориентированном графах на основе обхода графа в ширину.

**Листинг:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <locale>

#include <iostream>

#include <conio.h>

#include <time.h>

#include <stack>

#include <queue>

#include <stdio.h>

#include <Windows.h>

#include <stdlib.h>

using namespace std;

int\* dist;

int\* dist\_2;

int\* visited;

int\*\* graph; // взвешанный ориентированный

int\*\* graph\_2; // взвешанный неориентированный

queue<int>Q;

queue<int>QQ;

int start, start\_2;

int N, N\_2, j, i, R;

void BFSD(int v, int N, int\*\* graph) {

Q.push(v);

dist[v] = 0;

while (!Q.empty()) {

v = Q.front();

Q.pop();

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

if ((graph[v][i] > 0) && (dist[i] == -1)) {

Q.push(i);

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

}

}

}

}

void BFSD\_2(int v, int N\_2, int\*\* graph\_2) {

QQ.push(v);

dist\_2[v] = 0;

while (!QQ.empty()) {

v = QQ.front();

QQ.pop();

for (int i = 0; i < N\_2; i++) {

if ((graph\_2[v][i] > 0) && (dist\_2[i] == -1)) {

QQ.push(i);

dist\_2[i] = dist\_2[v] + graph\_2[v][i];

}

}

}

}

int main() {

srand(time(NULL));

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

printf("Ведите размер матрицы ");

scanf\_s("%d", &N\_2);

srand(time(NULL));

graph\_2 = new int\* [N\_2];

dist\_2 = new int[N\_2];

visited = new int[N\_2];

for (int i = 0; i < N\_2; i++)

{

graph\_2[i] = new int[N\_2];

}

for (int i = 0; i < N\_2; i++)

{

dist\_2[i] = -1;

}

printf("\n Матрица смежности для взвешенного неориентированного графа \n\n");

for (i = 0; i < N\_2; i++)

{

for (j = i; j < N\_2; ++j)

{

graph\_2[i][j] = graph\_2[j][i] = rand() % 10;

graph\_2[i][i] = graph\_2[j][j] = 0; // чтобы петля(узел) не создавалась

}

}

printf(" ");

for (j = 0; j < N\_2; j++)

{

printf("%4d ", j + 1); //горизонт

}

printf("\n\n");

for (i = 0; i < N\_2; i++) {

printf(" %d ", i + 1); // вертик

for (j = 0; j < N\_2; j++)

{

printf("%4d ", graph\_2[i][j]);

}

printf("\n\n");

}

for (i = 0; i < N\_2; i++)

{

printf("\n Из вершины %d есть дорога в ", i + 1);

for (j = 0; j < N\_2; j++)

{

if (graph\_2[i][j] >= 1)

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

}

}

printf("\n\n Введите начальную вершину ");

scanf("%d", &start\_2);

BFSD\_2(start\_2 - 1, N\_2, graph\_2);

printf("\n\n Вершины ");

for (int i = 1; i <= N\_2; i++) {

printf("|%3d ", i);

}

printf("\n Расстояние ");

for (int i = 0; i < N\_2; i++) {

printf("|%3d ", dist\_2[i]);

}

printf("\n\n");

printf("Ведите размер матрицы ");

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

graph = new int\* [N];

dist = new int[N];

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

{

graph[i] = new int[N];

}

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

{

dist[i] = -1;

}

printf("\n Матрица смежности для взвешенного ориентированного графа\n\n");

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

{

for (j = i; j < N; ++j)

{

R = rand() % 100;

if (R > 30)

{

graph[i][j] = rand() % 10;

graph[j][i] = 0;

graph[i][i] = graph[j][j] = 0; // чтобы петля(узел) не создавалась

}

else

graph[i][j] = graph[j][i] = rand() % 10;

graph[i][i] = graph[j][j] = 0; // чтобы петля(узел) не создавалась

}

}

printf(" ");

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

{

printf("%4d ", j + 1); //горизонт

}

printf("\n\n");

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

printf(" %d ", i + 1); // вертик

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

{

printf("%4d ", graph[i][j]);

}

printf("\n\n");

}

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

{

printf("\n Из вершины %d есть дорога в ", i + 1);

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

{

if (graph[i][j] >= 1)

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

}

}

printf("\n\n Введите начальную вершину ");

scanf("%d", &start);

BFSD(start - 1, N, graph);

printf("\n\n Вершины ");

for (int i = 1; i <= N; i++) {

printf("|%3d ", i);

}

printf("\n Расстояние ");

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

printf("|%3d ", dist[i]);

}

printf("\n\n");

}

**Доп. Задание**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <locale>

#include <iostream>

#include <conio.h>

#include <time.h>

#include <stack>

#include <queue>

#include <stdio.h>

#include <Windows.h>

#include <stdlib.h>

using namespace std;

int\* dist;

int\* dist\_2;

int\* visited;

int\*\* graph; // взвешанный ориентированный

int\*\* graph\_2; // взвешанный неориентированный

queue<int>Q;

queue<int>QQ;

int start, start\_2;

int N, N\_2, j, i, R;

void BFSD(int v, int N, int\*\* graph) {

Q.push(v);

dist[v] = 0;

while (!Q.empty()) {

v = Q.front();

Q.pop();

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

if ((graph[v][i] > 0) && (dist[i] == -1)) {

Q.push(i);

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

}

}

}

}

void BFSD\_2(int v, int N\_2, int\*\* graph\_2) {

QQ.push(v);

dist\_2[v] = 0;

while (!QQ.empty()) {

v = QQ.front();

QQ.pop();

for (int i = 0; i < N\_2; i++) {

if ((graph\_2[v][i] > 0) && (dist\_2[i] == -1)) {

QQ.push(i);

dist\_2[i] = dist\_2[v] + graph\_2[v][i];

}

}

}

}

int main() {

srand(time(NULL));

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

printf("Ведите размер матрицы ");

scanf\_s("%d", &N\_2);

srand(time(NULL));

graph\_2 = new int\*[N\_2];

dist\_2 = new int[N\_2];

visited = new int[N\_2];

for (int i = 0; i < N\_2; i++)

{

graph\_2[i] = new int[N\_2];

}

for (int i = 0; i < N\_2; i++)

{

dist\_2[i] = -1;

}

printf("\n Матрица смежности для взвешенного неориентированного графа \n\n");

for (i = 0; i < N\_2; i++)

{

for (j = i; j < N\_2; ++j)

{

graph\_2[i][j] = graph\_2[j][i] = rand() % 10;

graph\_2[i][i] = graph\_2[j][j] = 0; // чтобы петля(узел) не создавалась

}

}

printf(" ");

for (j = 0; j < N\_2; j++)

{

printf("%4d ", j + 1); //горизонт

}

printf("\n\n");

for (i = 0; i < N\_2; i++) {

printf(" %d ", i + 1); // вертик

for (j = 0; j < N\_2; j++)

{

printf("%4d ", graph\_2[i][j]);

}

printf("\n\n");

}

for (i = 0; i < N\_2; i++)

{

printf("\n Из вершины %d есть дорога в ", i + 1);

for (j = 0; j < N\_2; j++)

{

if (graph\_2[i][j] >= 1)

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

}

}

/\*printf("\n\n Введите начальную вершину ");

scanf("%d", &start\_2);\*/

printf(" Результат обхода в ширину \n");

for (i = 0; i < N\_2; i++) {

BFSD\_2(i, N\_2, graph\_2);

printf("\n");

for (int l = 0; l < N\_2; l++) {

printf(" %2d ", dist\_2[l]);

}

for (int l = 0; l < N\_2; l++) {

dist\_2[l] = -1;

}

}

/\*BFSD\_2(start\_2 - 1, N\_2, graph\_2);

printf("\n\n Вершины ");

for (int i = 1; i <= N\_2; i++) {

printf("|%3d ", i);

}

printf("\n Расстояние ");

for (int i = 0; i < N\_2; i++) {

printf("|%3d ", dist\_2[i]);

}\*/

printf("\n\n");

printf("Ведите размер матрицы ");

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

graph = new int\*[N];

dist = new int[N];

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

{

graph[i] = new int[N];

}

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

{

dist[i] = -1;

}

printf("\n Матрица смежности для взвешенного ориентированного графа\n\n");

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

{

for (j = i; j < N; ++j)

{

R = rand() % 100;

if (R > 30)

{

graph[i][j] = rand() % 10;

graph[j][i] = 0;

graph[i][i] = graph[j][j] = 0; // чтобы петля(узел) не создавалась

}

else

graph[i][j] = graph[j][i] = rand() % 10;

graph[i][i] = graph[j][j] = 0; // чтобы петля(узел) не создавалась

}

}

printf(" ");

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

{

printf("%4d ", j + 1); //горизонт

}

printf("\n\n");

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

printf(" %d ", i + 1); // вертик

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

{

printf("%4d ", graph[i][j]);

}

printf("\n\n");

}

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

{

printf("\n Из вершины %d есть дорога в ", i + 1);

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

{

if (graph[i][j] >= 1)

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

}

}

/\*printf("\n\n Введите начальную вершину ");

scanf("%d", &start);\*/

printf(" Результат обхода в ширину \n");

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

BFSD(i, N, graph);

printf("\n");

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

printf(" %2d ", dist[l]);

}

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

dist[l] = -1;

}

}

/\*BFSD(start - 1, N, graph);

printf("\n\n Вершины ");

for (int i = 1; i <= N; i++) {

printf("|%3d ", i);

}

printf("\n Расстояние ");

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

printf("|%3d ", dist[i]);

}\*/

printf("\n\n");

system("pause");

}