Пензенский государственный университет  
Кафедра «Вычислительной техники»

**Отчет**по лабораторной работе №7  
по дисциплине: «Логика и основы алгоритмизации в инженерных задачах»  
на тему: «Поиск расстояний во взвешенном графе»

**Выполнил студент группы 19ВВ1:**

Балалаев А.А.

**Приняли:**

д.т.н. профессор Митрохин М. А.

к.т.н. Юрова О.В.

Пенза 2020.

**Цель работы:** изучить поиск расстояний в взвешенном графе, научится реализовывать его используя обход в ширину, реализовать ориентированный граф, познакомится с командной строкой и несколькими ее командами, реализовать запуск программы с параметрами командной строки.

**Теория**

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.

Алгоритм обхода в ширину.

Вход: G – матрица смежности графа, v – исходная вершина.

Выход: DIST – вектор расстояний до всех вершин от исходной.

Алгоритм ПОШ

1.1. для всех i положим DIST [i] = 1000 пометим как &quot;не посещенную&quot;;

1.2. ВЫПОЛНЯТЬ BFSD (v).

1.3 для всех i вывести DIST [i] на экран;

Алгоритм BFSD(v):

2.1. Создать пустую очередь Q = {};

2.2. Поместить v в очередь Q.push(v);

2.3. Обновить вектор расстояний DIST [ x ] = 0;

2.4. ПОКА Q != ∅ очередь не пуста ВЫПОЛНЯТЬ

2.5. v = Q.front() установить текущую вершину;

2.6. Удалить первый элемент из очереди Q.pop();

2.7. вывести на экран v;

2.8. ДЛЯ i = 1 ДО size\_G ВЫПОЛНЯТЬ

2.9. ЕСЛИ G(v,i) &gt; 0 И DIST [ i ] &gt; DIST [ v ] + G(v,i)

2.10. ТО

2.11. Поместить i в очередь Q.push(i);

2.12. Обновить вектор расстояний DIST [ i ] = DIST [ v ] +

G(v,i);

*С*и - компилируемый язык. После сборки программа представляет собой исполняемый файл (мы не рассматриваем создание динамических библиотек, драйверов и т.д.). Наши программы очень простые и не содержат библиотек времени выполнения (Runtime libraries), поэтому могут быть перенесены на компьютер с такой же операционной системой (и подобной архитектурой) и там запущены.

Программа во время запуска может принимать параметры. Они являются аргументами функции main. Общий вид функции main следующий

[?](https://learnc.info/c/cmd.html)

|  |  |
| --- | --- |
| 1  2  3 | void main(int argc, char \*\*argv) {      ...  } |

Первым аргументом argc является число переданных функции параметров. Второй аргумент – массив строк – собственно сами параметры. Так как параметры у функции могут быть любыми, то они передаются как строки, и уже сама программа должна их разбирать и приводить к нужному типу.

Первым аргументом (argv[0]) всегда является имя программы. При этом имя выводится в зависимости от того, откуда была запущена программа.

[?](https://learnc.info/c/cmd.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | #include <conio.h>  #include <stdio.h>    void main(int argc, char \*\*argv) {      printf("%s", argv[0]);  } |

Если же требуется не только поменять диск, но и сразу же перейти в конкретную папку, тогда команду cd следует использовать с ключом /d. По окончании выполнения команды, текущим каталогом станет указанная папка. Вывести текущую директорию можно той же командой cd без указания параметров.

Теперь, после того, как мы перешли в нашу папку, можно выполнить нашу программу. Для этого наберите её имя.

оберите проект. Перед сборкой убедитесь, что программа закрыта. Теперь вызовите программу, передав ей разные аргументы. Для этого напишите имя программы и через пробел аргументы

**Код программы**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <time.h>

#include <queue>

#include <locale.h>

#include <malloc.h>

#include <windows.h>

using namespace std;

void rand\_Zap\_vz\_or(int\*\* Matrix, int n)

{

srand(time(NULL));

printf("Взвешаный орентированный граф\n");

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

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

{

if (i == j)

{

Matrix[i][j] = 0;

}

if (i < j)

{

if (rand() % 100 > 50)

{

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

if (rand() % 100 > 50)

Matrix[j][i] = 0;

else

Matrix[j][i] = Matrix[i][j];

}

else

if (rand() % 100 > 50)

{

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

Matrix[i][j] = 0;

}

else

{

Matrix[i][j] = 0;

Matrix[j][i] = 0;

}

}

}

}

void rand\_Zap\_vz\_nor(int\*\* Matrix, int n)

{

srand(time(NULL));

printf("Взвешаный неорентированный граф\n");

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

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

{

if (i == j)

{

Matrix[i][j] = 0;

}

if (i < j)

{

if (rand() % 100 > 50)

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

else

Matrix[i][j] = 0;

Matrix[j][i] = Matrix[i][j];

}

}

}

void rand\_Zap\_nor(int\*\* Matrix, int n)

{

srand(time(NULL));

printf("Не взвешаный неорентированный граф\n");

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

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

if (i == j) {

Matrix[i][j] = 0;

}

if (i < j) {

Matrix[i][j] = rand() % 2;

Matrix[j][i] = Matrix[i][j];

}

}

}

void rand\_Zap\_or(int\*\* Matrix, int n)

{

srand(time(NULL));

printf("Не взвешаный орентированный граф\n");

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

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

if (i == j) {

Matrix[i][j] = 0;

}

if (i < j)

{

if (rand() % 100 > 50)

{

Matrix[i][j] = rand() % 2;

if (rand() % 100 > 50)

Matrix[j][i] = 0;

else

Matrix[j][i] = Matrix[i][j];

}

else

if (rand() % 100 > 50)

{

Matrix[j][i] = rand() % 2;

Matrix[i][j] = 0;

}

else

{

Matrix[i][j] = 0;

Matrix[j][i] = 0;

}

}

}

}

void print\_G(int\*\* Matrix, int n)

{

printf(" ");

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

printf("%3d", i + 1);

printf("\n\n");

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

printf("%2d", i + 1);

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

printf("%3d", Matrix[i][j]);

}

printf("\n");

}

printf("\n");

}

void dsit\_0(int\* Matrix, int n)

{

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

Matrix[i] = -1;

}

void BFSD\_vz(int\*\* Matrix, int\* dist, int n, int v)

{

queue <int> Q;

Q.push(v);

dist[v] = 0;

while (Q.empty() == false)

{

v = Q.front();

Q.pop();

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

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

{

if ((Matrix[v][i] != 0) && (dist[i] > (dist[v] + Matrix[v][i])))

{

Q.push(i);

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

}

}

}

printf("\n");

}

void BFSD(int\*\* Matrix, int\* dist, int n, int v)

{

queue <int> Q;

Q.push(v);

dist[v] = 0;

while (Q.empty() == false)

{

v = Q.front();

Q.pop();

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

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

{

if ((Matrix[v][i] == 1) && dist[i] == -1)

{

Q.push(i);

dist[i] = dist[v] + 1;

}

}

}

printf("\n");

}

int main(int argc, char\*\* argv)

{

setlocale(LC\_ALL, "Rus");

if (argc != 3)

{

printf("Error: found %d arguments. Needs exactly 2", argc - 1);

exit(1);

}

int N = 1;

printf("введите кол-во вершин\n");

scanf("%d", &N);

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

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

G[i] = (int\*)malloc(N \* sizeof(int));

}

int\* dist;

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

switch (atoi(argv[1]))

{

case 0: switch (atoi(argv[2]))

{

case 0:rand\_Zap\_nor(G, N);

break;

case 1:rand\_Zap\_or(G, N);

break;

}

break;

case 1:switch (atoi(argv[2]))

{

case 0:rand\_Zap\_vz\_nor(G, N);

break;

case 1:rand\_Zap\_vz\_or(G, N);

break;

}

break;

}

print\_G(G, N);

switch (atoi(argv[1]))

{

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

{

dsit\_0(dist, N);

printf("от вершины %d\n", i + 1);

BFSD(G, dist, N, i);

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

{

if (dist[j] != -1)

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

else

printf("нет прохода \t");

}

printf("\n");

}

break;

case 1:for (int i = 0; i < N; i++)

{

dsit\_0(dist, N);

printf("от вершины %d\n", i + 1);

BFSD\_vz(G, dist, N, i);

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

{

if (dist[j] != -1)

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

else

printf("нет прохода \t");

}

printf("\n");

}

break;

}

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

free(G[i]);

}

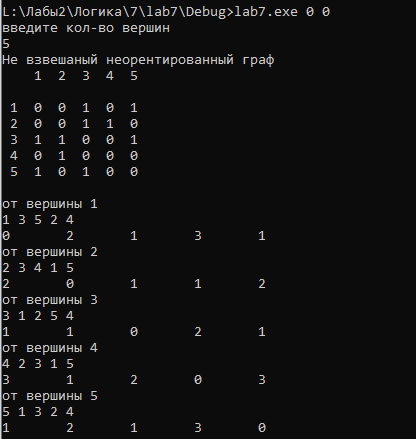
free(G);

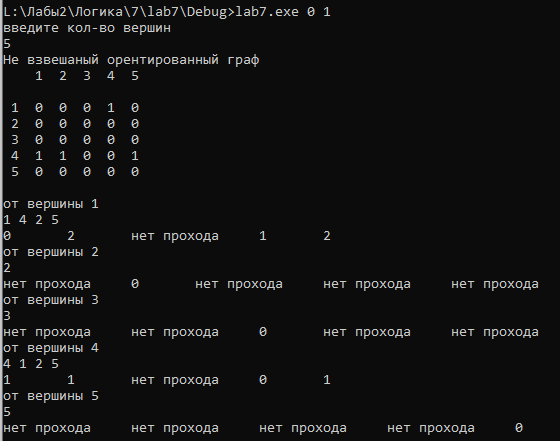
free(dist);

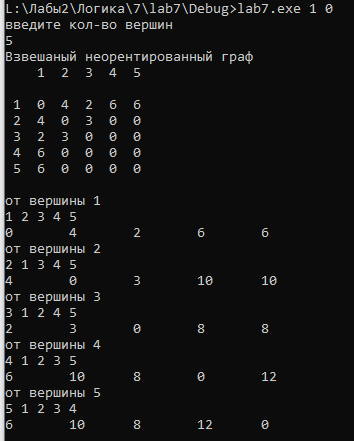
getchar();

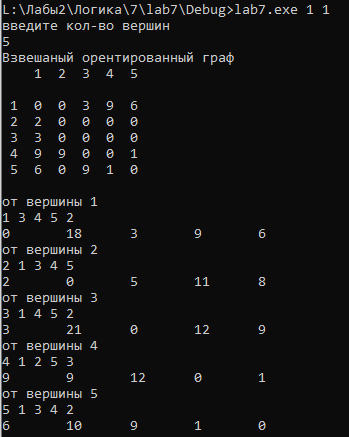
}

**Результат работы кода**









**Вывод:** изучил поиск расстояний в взвешенном графе, научился реализовывать его используя обход в ширину, также реализовать ориентированный граф, познакомился с командной строкой и несколькими ее командами, реализовал запуск программы с параметрами командной строки.