|  |
| --- |
| 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 взвешанном графе» |
|  |
|  |
| Выполнил студент группы 19ВВ3:  Татаршев А.А  Принял:  Митрохин М. А. |
| Пенза  2020 |

**Цель работы**

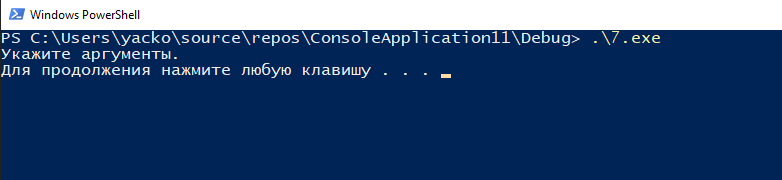
Реализовать алгоритм поиска расстояний во взвешенном/невзвешенном и ориентированном/неориентированном графе, используя алгоритм обхода в ширину.

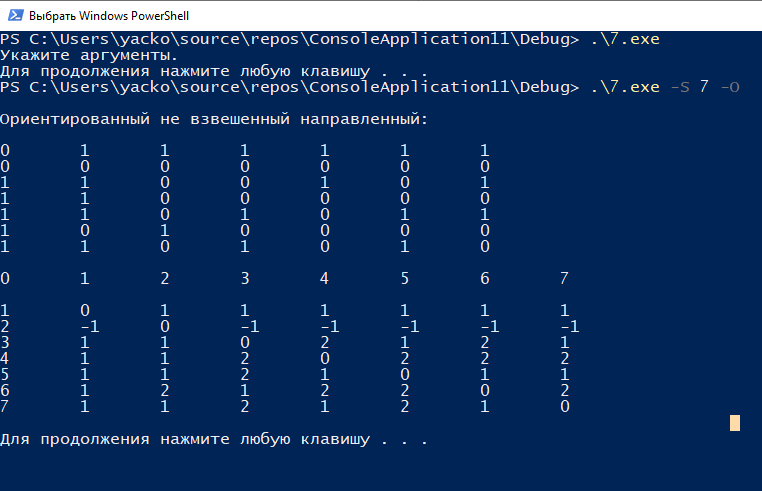
**Листинг программы**

[#include](https://vk.com/im?sel=144302641&st=%23include) <stdio.h>  
[#include](https://vk.com/im?sel=144302641&st=%23include) <time.h>  
[#include](https://vk.com/im?sel=144302641&st=%23include) <windows.h>  
[#include](https://vk.com/im?sel=144302641&st=%23include) <conio.h>  
[#include](https://vk.com/im?sel=144302641&st=%23include) <queue>  
[#include](https://vk.com/im?sel=144302641&st=%23include) <string.h>  
[#include](https://vk.com/im?sel=144302641&st=%23include) <locale.h>  
  
[#define](https://vk.com/im?sel=144302641&st=%23define) \_CRT\_SECURE\_NO\_WARNINGS  
void output(int\*\* Array, int N) {  
printf("\n");  
for (int i = 0; i < N; i++)  
{  
for (int j = 0; j < N; j++)  
{  
printf("%d\t", Array[i][j]);  
}  
printf("\n");  
}  
printf("\n");  
}  
void GenWeighed(int\*\* Array, int N) {  
for (int i = 0; i < N; i++)  
{  
for (int j = 0; j < N; j++)  
{  
int z = rand() % 100;  
if (z < 80) {  
Array[i][j] = rand() % N;  
}  
else {  
Array[i][j] = 0;  
}  
Array[j][i] = Array[i][j];  
if (i == j) {  
Array[i][j] = 0;  
}  
}  
}  
output(Array, N);  
}  
void GenWeighedOriented(int\*\* Array, int N) {  
for (int i = 0; i < N; i++)  
{  
for (int j = 0; j < N; j++)  
{  
int z = rand() % 100;  
if (z < 80) {  
Array[i][j] = rand() % N;  
}  
else {  
Array[i][j] = 0;  
}  
if (i == j) {  
Array[i][j] = 0;  
}  
}  
}  
output(Array, N);  
}  
void Gen(int\*\* Array, int N) {  
  
  
for (int i = 0; i < N; i++)  
{  
for (int j = 0; j < N; j++)  
{  
Array[i][j] = rand() % 2;  
Array[j][i] = Array[i][j];  
if (i == j) {  
Array[i][j] = 0;  
}  
}  
  
}  
  
output(Array, N);  
}  
void GenOriented(int\*\* Array, int N) {  
  
  
for (int i = 0; i < N; i++)  
{  
for (int j = 0; j < N; j++)  
{  
Array[i][j] = rand() % 2;  
if (i == j) {  
Array[i][j] = 0;  
}  
}  
  
}  
  
output(Array, N);  
}  
void ClearDistanceWeighed(int\* Dist, int N) {  
for (int i = 0; i < N; i++) {  
Dist[i] = 1000;  
}  
printf("\n");  
}  
void ClearDistance(int\* Dist, int N) {  
for (int i = 0; i < N; i++) {  
Dist[i] = -1;  
}  
printf("\n");  
}  
void DBFSWeighed(int\*\* Array, int v, int\* DIST, int N) {  
std::queue <int> Q;  
Q.push(v);  
DIST[v] = 0;  
while (!Q.empty()) {  
v = Q.front();  
Q.pop();  
for (int i = 0; i < N; i++) {  
if (Array[v][i] > 0 && DIST[i] > DIST[v] + Array[v][i]) {  
Q.push(i);  
DIST[i] = DIST[v] + Array[v][i];  
}  
}  
}  
  
}  
void DBFS(int\*\* Array, int v, int\* DIST, int N) {  
std::queue <int> Q;  
Q.push(v);  
DIST[v] = 0;  
while (!Q.empty()) {  
v = Q.front();  
Q.pop();  
for (int i = 0; i < N; i++) {  
if (Array[v][i] == 1 && DIST[i] == -1) {  
Q.push(i);  
DIST[i] = DIST[v] + 1;  
}  
}  
}  
}  
int main(int argc, char\* argv[])  
{  
srand(time(NULL));  
setlocale(LC\_ALL, "RUS");  
int N = 0;  
int\*\* ArrayM1 = NULL;  
int\*\* ArrayM2 = NULL;  
int\*\* ArrayM3 = NULL;  
int\*\* ArrayM4 = NULL;  
int\* DIST = NULL;  
if (argc > 1) {  
for (int j = 0; j < argc; j++) {  
if (strcmp(argv[j], "-S") == 0) {  
N = atoi(argv[j + 1]);  
}  
}  
for (int i = 0; i < argc; i++) {  
  
//ориентированный не взвешенный направленный//  
if (strcmp(argv[i], "-O") == 0) {  
  
//N = atoi(argv[i]);  
DIST = (int\*)malloc(N \* sizeof(int));  
ClearDistance(DIST, N);  
ArrayM4 = (int\*\*)malloc(N \* sizeof(int\*));  
for (int i = 0; i < N; i++)  
{  
ArrayM4[i] = (int\*)malloc(N \* sizeof(int));  
}  
printf("Ориентированный не взвешенный направленный: \n");  
  
GenOriented(ArrayM4, N);  
for (int i = 0; i < N+1; i++) {  
printf("%d\t", i);  
}  
printf("\n\n");  
for (int i = 0; i < N; i++) {  
  
DBFS(ArrayM4, i, DIST, N);  
printf("%d\t", i + 1);  
for (int j = 0; j < N; j++) {  
  
printf("%d\t", DIST[j]);  
}  
for (int j = 0; j < N; j++) {  
DIST[j] = -1;  
  
}  
printf("\n");  
}  
}  
//ориентированный не взвешенный не направленный//  
if (strcmp(argv[i], "-N") == 0) {  
  
DIST = (int\*)malloc(N \* sizeof(int));  
ClearDistance(DIST, N);  
ArrayM3 = (int\*\*)malloc(N \* sizeof(int\*));  
for (int i = 0; i < N; i++)  
{  
ArrayM3[i] = (int\*)malloc(N \* sizeof(int));  
}  
printf("Ориентированный не взвешенный не направленный: \n");  
Gen(ArrayM3, N);  
for (int i = 0; i < N + 1; i++) {  
printf("%d\t", i);  
}  
printf("\n\n");  
for (int i = 0; i < N; i++) {

DBFS(ArrayM3, i, DIST, N);  
printf("%d\t", i + 1);  
for (int j = 0; j < N; j++) {  
printf("%d\t", DIST[j]);  
  
}  
for (int j = 0; j < N; j++) {  
DIST[j] = -1;  
  
}  
printf("\n");  
}  
}  
//ориентированный взвешенный направленный//  
if (strcmp(argv[i], "-OW") == 0) {  
//N = atoi(argv[i+1]);  
DIST = (int\*)malloc(N \* sizeof(int));  
ArrayM2 = (int\*\*)malloc(N \* sizeof(int\*));  
for (int i = 0; i < N; i++)  
{  
ArrayM2[i] = (int\*)malloc(N \* sizeof(int));  
}  
printf("Взвешенный ориентированный:\n");  
GenWeighedOriented(ArrayM2, N);  
ClearDistanceWeighed(DIST, N);  
for (int i = 0; i < N + 1; i++) {  
printf("%d\t", i);  
}  
printf("\n\n");  
for (int i = 0; i < N; i++) {  
DBFSWeighed(ArrayM2, i, DIST, N);  
printf("%d\t", i + 1);  
for (int j = 0; j < N; j++) {  
if (DIST[j] == 1000) {  
printf("-\t");  
}  
else  
printf("%d\t", DIST[j]);  
}  
for (int j = 0; j < N; j++) {  
DIST[j] = 1000;  
}  
printf("\n");  
}  
}  
//ориентированный взвешенный не направленный//  
if (strcmp(argv[i], "-NW") == 0) {  
//N = atoi(argv[2]);  
DIST = (int\*)malloc(N \* sizeof(int));  
ArrayM1 = (int\*\*)malloc(N \* sizeof(int\*));  
for (int i = 0; i < N; i++)  
{  
ArrayM1[i] = (int\*)malloc(N \* sizeof(int));  
}  
printf("Взвешенный:\n");  
GenWeighed(ArrayM1, N);  
DIST = (int\*)malloc(N \* sizeof(int));  
ClearDistanceWeighed(DIST, N);  
for (int i = 0; i < N + 1; i++) {  
printf("%d\t", i);  
}  
printf("\n\n");  
for (int i = 0; i < N; i++) {  
printf("%d\t", i + 1);  
DBFSWeighed(ArrayM1, i, DIST, N);  
for (int j = 0; j < N; j++) {  
if (DIST[j] == 1000) {  
printf("-\t");  
}  
else  
printf("%d\t", DIST[j]);  
}  
for (int j = 0; j < N; j++) {  
DIST[j] = 1000;  
}  
printf("\n");  
}  
}  
  
  
}  
}  
else  
{  
printf("Укажите аргументы.");  
}  
printf("\n");  
system("pause");  
return 0;  
}

### **Результат работы программы:**





**Вывод:** Реализовал алгоритм поиска матрицы кратчайших расстояний во взвешенном/невзвешенном и ориентированном/неориентированном графе, используя алгоритм обхода в ширину.