Пензенский государственный университет  
Кафедра «Вычислительной техники»

**Отчет**по лабораторной работе №8  
по дисциплине: «Логика и основы алгоритмизации в инженерных задачах»  
на тему: «Определение характеристик графов»

**Выполнил студент группы 19ВВ1:**

Тебнев Р.

**Приняли:**

д.т.н. профессор Митрохин М. А.

Юрова О.В.

Пенза 2020

**Цель работы:** изучить и определить характеристики графа.

**Теория**

Если G - граф, содержащий непустое множество n вершин V и

множество ребер E и d(v i , v j ) – расстояние между двумя произвольными

вершинами v i  и v j , тогда для фиксированной вершины v величина

),(max)(jvvdve

где v, v j V и j = 1…n называется эксцентриситетом вершины v i .

Другими словами эксцентриситет вершины – расстояние до наиболее

удаленной вершины графа.

Максимальный эксцентриситет среди эксцентриситетов всех вершин

графа называется диаметром графа G и обозначается через D(G).

Вершина v i называется периферийной, если её эксцентриситет равен

диаметру графа e(v i ) = d(G).

Минимальный из эксцентриситетов вершин графа называется его

радиусом и обозначается через r(G).

Вершина v i называется центральной, если её эксцентриситет равен

радиусу графа e(v i ) = r(G).

Множество всех центральных вершин графа называется его центром.

Граф G может иметь единственную центральную вершину или несколько

центральных вершин.

Степенью вершины графа G называется число инцидентных ей ребер.

Степень вершины v i обозначается через deg(v i ).

Вершина v i со степенью 0 называется изолированной, со степенью 1 –

концевой.

Вершина графа, смежная с каждой другой его вершиной, называется

доминирующей.

**Код программы**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <time.h>

#include <locale.h>

#include <queue>

using namespace std;

void NewGraphNeOr(int\*\* G, int n) {

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

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

if (i == j) {

G[i][j] = 0;

}

if (i < j) {

if (rand() % 100 > 50) {

G[i][j] = rand() % 15;

}

else {

G[i][j] = 0;

}

G[j][i] = G[i][j];

}

}

}

}

void NewGraphOr(int\*\* G, int n) {

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

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

if (i == j) {

G[i][j] = 0;

}

if (i < j) {

if (rand() % 100 > 50) {

G[i][j] = rand() % 15;

}

else {

G[j][i] = rand() % 15;

}

}

}

}

}

void PrintGraph(int\*\* G, int n) {

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

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

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

}

printf("\n");

}

printf("\n");

}

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

queue <int> q;

q.push(s);

dist[s] = 0;

while (!q.empty()) {

s = q.front();

q.pop();

printf("%d ", s);

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

if ((G[s][i] > 0) && (dist[i] == 1000)) {

q.push(i);

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

}

}

}

printf("\n");

}

int main() {

int\*\* G; // указатель на указатель на строку элементов

int\* dist;

int i, j, n;

setlocale(LC\_ALL, "rus");

srand(time(NULL));

printf("Введите размер графа: ");

scanf("%d", &n);

printf("\n");

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

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

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

}

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

printf("Неориентированный взвешенный граф \n");

NewGraphNeOr(G, n);

PrintGraph(G, n);

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

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

dist[i] = 1000;

}

printf("от вершины %d: ", j);

BFSD(j, G, dist, n);

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

if (dist[i] != 1000)

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

else

printf("нет прохода\t");

}

printf("\n\n");

}

printf("Ориентированный взвешенный граф \n");

NewGraphOr(G, n);

PrintGraph(G, n);

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

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

dist[i] = 1000;

}

printf("от вершины %d: ", j);

BFSD(j, G, dist, n);

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

if (dist[i] != 1000)

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

else

printf("нет прохода\t");

}

printf("\n\n");

}

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

free(G[i]);

}

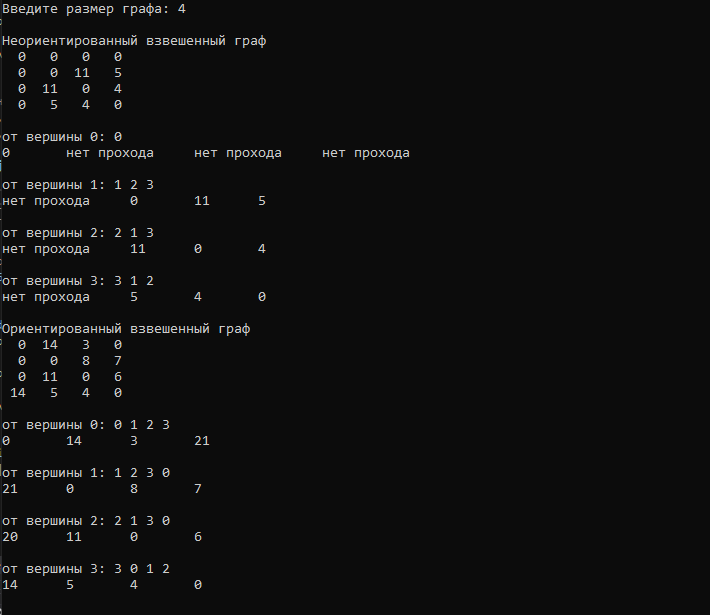
free(G);

free(dist);

getchar();

getchar();

}**Результат программы**



**Вывод:** научились искать расстояния во взвешенном графе между вершинами.