|  |
| --- |
| 0Министерство образования Российской Федерации  Пензенский государственный университет  Кафедра «Вычислительная техника» |
| Отчет  по лабораторной работе №8  по курсу «Логика и основы алгоритмизации в инженерных задачах»  на тему «Определение характеристик графов» |
|  |
|  |
| Выполнил студент группы 19ВВ3:  Табрисов С.А.  Принял:  Митрохин М. А. |
| Пенза  2020 |

**Цель работы**

Определить радиус и диаметр графа, подмножества периферийных и центральных вершин и найти изолированные, концевые и доминирующие вершины.

**Листинг программы**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

#include <conio.h>

#include <locale.h>

#include <String.h>

#include <time.h>

#include <queue>

void print(int\*\* Array, int N) {

printf("|№| ");

for (int k = 1; k <= N; k++) printf("|%d| ", k);

printf("\n");

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

printf("\n|%d| ", i + 1);

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

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

}

printf("\n");

}

}

int\*\* dynamic(int N) {

int\*\* Array;

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

for (int i = 0; i < N; i++) Array[i] = (int\*)malloc(N \* sizeof(int));

return Array;

}

int\* dynamic\_dist(int N) {

int\* DIST;

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

return DIST;

}

int\* dynamic\_eks(int N) {

int\* eks;

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

return eks;

}

void generate\_matrix(int\*\* Array, int N, int chance) {

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

{

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

{

int a = rand() % 100;

if (a < chance) {

Array[i][j] = rand() % N;

}

else {

Array[i][j] = 0;

}

Array[j][i] = Array[i][j];

if (i == j) {

Array[i][j] = 0;

}

}

}

print(Array, N);

}

void clear\_distance(int\* Dist, int N) {

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

Dist[i] = 1000;

}

}

void BFS(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];

}

}

}

}

int main()

{

srand(time(NULL));

setlocale(LC\_ALL, "RUS");

int count = 0;

int N = 0;

int chance = 0;

int D = -1, R = 10000;

int\*\* Array = NULL;

int\* DIST = NULL;

int\* eks = NULL;

printf("Введите порядок матрицы: ");

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

printf("Введите вероятность появления ребра между вершинами: ");

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

Array = dynamic(N);

DIST = dynamic\_dist(N);

eks = dynamic\_eks(N);

generate\_matrix(Array, N, chance);

clear\_distance(DIST, N);

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

eks[i] = -1;

}

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

BFS(Array, i, DIST, N);

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

if (eks[i] < DIST[j] && DIST[j] != 1000) {

eks[i] = DIST[j];

}

}

if (eks[i] < R && eks[i] != 0) {

R = eks[i];

}

if (eks[i] > D) {

D = eks[i];

}

clear\_distance(DIST, N);

}

printf("Вершины:");

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

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

}

printf("\nРадиус: %d", R);

printf("\nДиаметр: %d", D);

printf("\nЦентральные вершины:");

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

if (eks[i] == R) {

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

}

}

printf("\nПереферийные вершины:");

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

if (eks[i] == D) {

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

}

}

printf("\nИзолированные:");

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

if (eks[i] == 0) {

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

}

}

printf("\nКонцевые:");

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

count = 0;

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

if (Array[i][j] != 0) {

count++;

}

}

if (count == 1) {

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

}

}

printf("\nДоминирующие:");

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

count = 0;

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

if (Array[i][j] != 0) {

count++;

}

}

if (count == N - 1) {

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

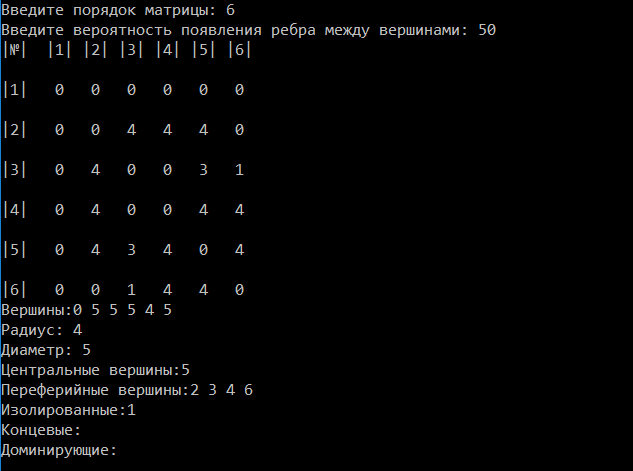
}

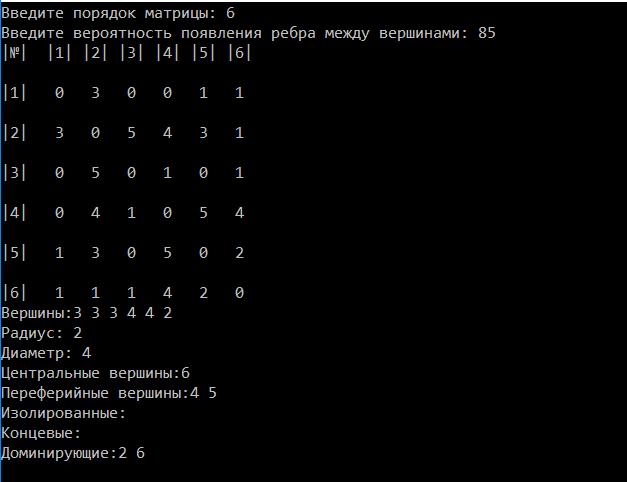
}

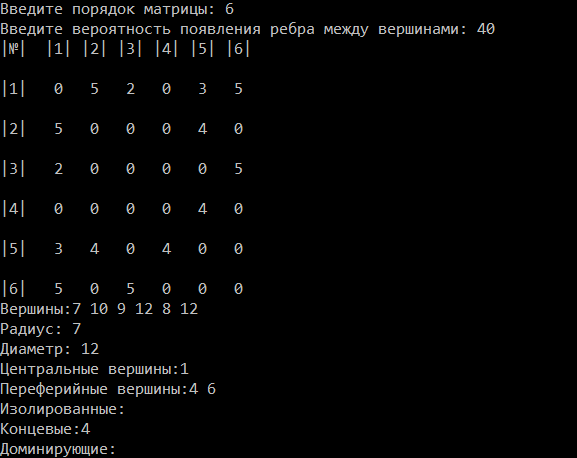
printf("\n");

### }

### **Результат работы программы:**







**Вывод:** Определил радиус и диаметр графа, подмножества периферийных и центральных вершин и нашел изолированные, концевые и доминирующие вершины.