Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра "Вычислительная техника"

Отчет

по лабораторной работе №10

по курсу “Л и ОА в ИЗ”

на тему “Поиск расстояний во взвешенном графе”

Выполнили студенты группы 22ВВС1:

Агапов И.Е.

Братчиков А.А.

Приняли:

Юрова О.В.

Акифьев И.В.

Пенза 2023

### Лабораторное задание

**Задание 1**

1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного взвешенного графа *G*. Выведите матрицу на экран.
2. Для сгенерированного графа осуществите процедуру поиска расстояний, реализованную в соответствии с приведенным выше описанием. При  реализации алгоритма в качестве очереди используйте класс **queue** из стандартной библиотеки С++.

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

**Задание 2**

1. Для каждого из вариантов сгенерированных графов (ориентированного и не ориентированного) определите радиус и диаметр.
2. Определите подмножества периферийных и центральных вершин.

**Задание 3\***

Модернизируйте программу так, чтобы получить возможность запуска программы с параметрами командной строки.  В качестве параметра должны указываться тип графа (взвешенный или нет) и наличие ориентации его ребер (есть ориентация или нет).

**Листинг**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include <iostream>

#include <queue>

#include < string.h >

using namespace std;

void BFS(queue <int> Q, int size, int\* vis, int\*\* mas, int S) {

Q.push(S);

vis[S] = 0;

while (!Q.empty()) {

S = Q.front();

printf("%d\t", S + 1);

Q.pop();

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

if (mas[S][i] > 0 && vis[i] > vis[S] + mas[S][i]) {

vis[i] = vis[S] + mas[S][i];

Q.push(i);

}

}

}

return;

}

void zero(int\* vis, int size) {

for (int k = 0; k < size; k++) {

vis[k] = 1000;

}

return;

}

void zerok(int\* vis, int size) {

for (int k = 0; k < size; k++) {

vis[k] = 0;

}

return;

}

int main(int wz,char \*m[]) {

setlocale(LC\_ALL, "");

queue <int> SomeQ;

printf("Кол-во переменных %d\n", wz);

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

std::cout << i << "-й аргумент: " << m[i] << '\n';

}

int\*\* mas;

int size;

int\* Num;

int\* EXC;

int\* ZEN;

int w;

srand(2413254);

scanf("%d", &size);

w = size;

Num = (int\*)malloc(sizeof(int) \* size);

EXC = (int\*)malloc(sizeof(int) \* size);

ZEN = (int\*)malloc(sizeof(int) \* size);

mas = (int\*\*)malloc(sizeof(int\*) \* size);

int j = 0;

int i = 0;

setlocale(LC\_ALL, "");

zero(Num, size);

zero(EXC, size);

zerok(ZEN, size);

while (size > i) {

j = 0;

mas[i] = (int\*)malloc(sizeof(int) \* w);

while (w > j) {

mas[i][j] = 0;

j++;

}

i++;

}

i = 0;

j = 0;

for (int i = 0;size > i;i++) {

if (!strcmp(m[1], "0")) {

j = i;

}

for (int j = 0;size > j;j++) {

if (i == j) {

mas[i][j] = 0;

}

else {

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

if (!strcmp(m[2], "1")) {

if (mas[i][j] == 1) {

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

}

}

if (!strcmp(m[1], "0")) {

mas[j][i] = mas[i][j];

}

}

}

}

zero(Num, size);

printf("\n\t");

for (int o = 0; o < size; o++) {

printf("%d\t", o + 1);

}

printf("\n");

for (int o = 0; o < size; o++) {

printf("%d\t", o + 1);

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

printf("%d\t", mas[o][n]);

}

printf("\n");

}

int ex=0;

int deam = 0;

int rad = 9999;

int zen = 9999;

printf("\n");

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

zero(Num, size);

printf("\n");

BFS(SomeQ, size, Num, mas, i);

printf("\n");

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

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

if (Num[j] > ex) { ex = Num[j]; }

ZEN[i]=ZEN[i]+Num[j];

}

if (ZEN[i]<zen){zen= ZEN[i];}

EXC[i] = ex;

printf("Эксцентриситет %d\n", EXC[i]);

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

if (ex > deam) { deam = ex; }

if (ex < rad && ex!=0) { rad = ex; }

}

printf("\n");

ex = 0;

}

int kkkk;

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

if(ZEN[i]==zen){kkkk=i;}

}

printf("Деаметр %d\n", deam);

printf("Радиус %d\n", rad);

printf("Центральные вершины:\n");

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

if (EXC[j] == rad) {printf("Вершина %d\n", j+1); }

}

printf("Переферийные вершины:\n");

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

if (EXC[j] == deam) {printf("Вершина %d\n", j+1); }

}

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

printf("Вес вершины %d %d\n",i+1,ZEN[i]);

}

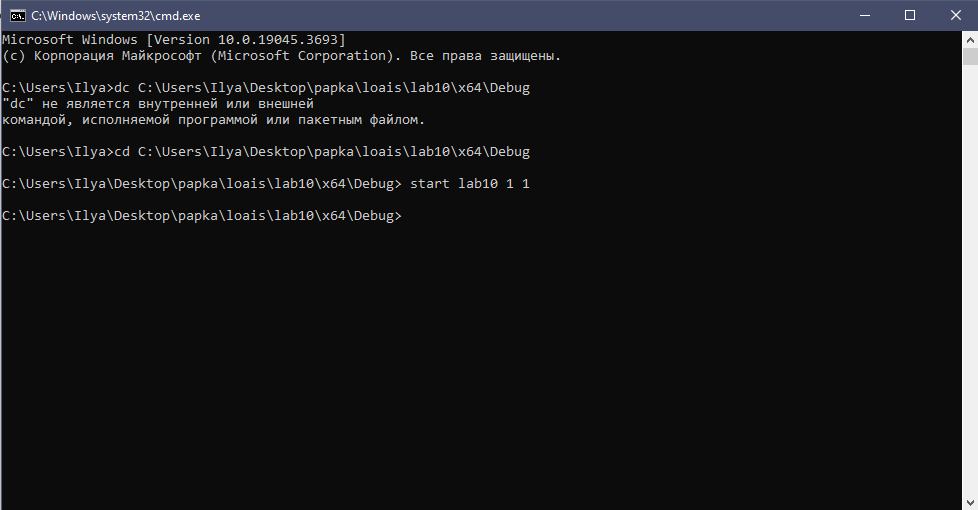
printf("Центры тяжести %d\n",kkkk+1);

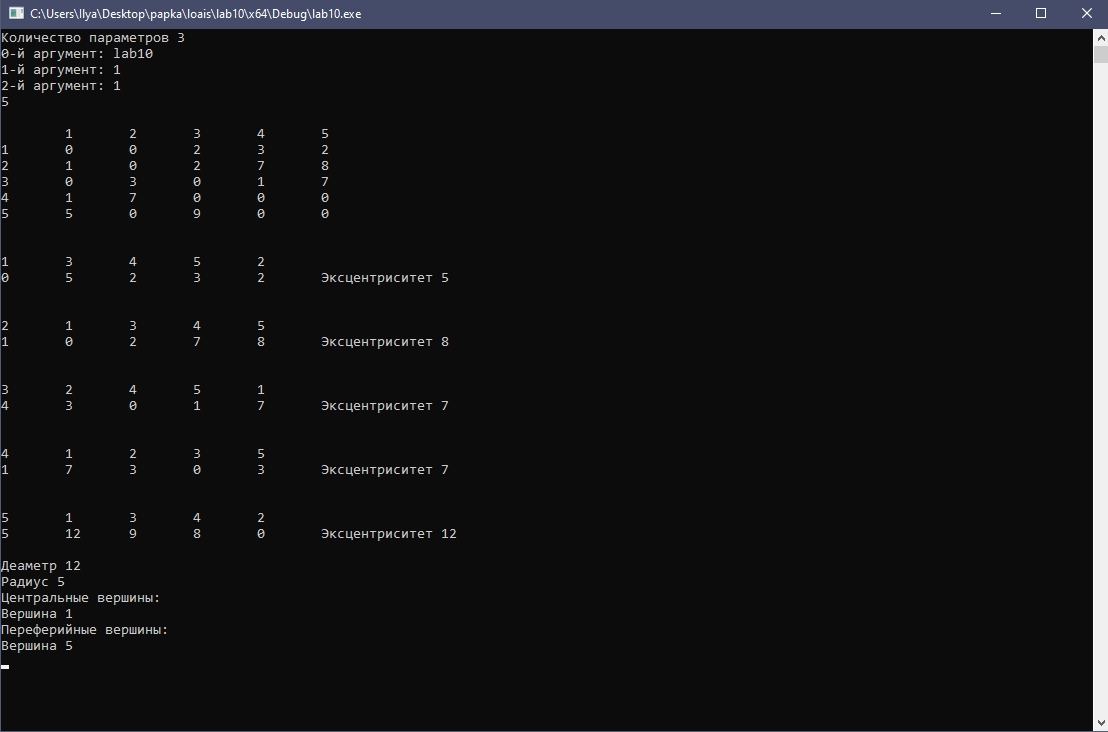
int a = 0;

scanf("%d", &a);

}

**Результат работы программы**

****

****

**Вывод**

### В ходе выполнения лабораторной работы были реализованы функции поиска расстояний во взвешенном графе. Данная задача была выполнена при помощи поиска в глубину. Так же выполнен поиск расстояний в ориентированном и неориентированном графах.

### Были выполнены измерения скорости выполнения алгоритмов, результаты измерений представлены в данном отчете, мы изучили зависимость скорости каждой функции от размера графа. Для каждого из вариантов сгенерированных графов определили радиус и диаметр. Определили подмножества периферийных и центральных вершин.

### Модернизировали программу так, чтобы получить возможность запуска программы с параметрами командной строки. В качестве параметров указываются тип графа (взвешенный или нет) и наличие ориентации его ребер (есть ориентация или нет).