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 взвешенном графе»

Выполнил:

студент группы 20ВВ2

Кочергин В.П.

Принял:

к.т.н. доцент Юрова О.В.

д.т.н. профессор Митрохин М. А.

Пенза 2021

**Цель работы:** Написать алгоритм для поиска кратчайших расстояний во взвешенном ориентированном и неориентированном графах.

**Задание 1** Сгенерировать матрицу смежности ориентированного и неориентированного графов и найти расстояния

Листинг:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <windows.h>

#include <limits.h>

#include <queue>

using namespace std;

void BFS(int\* dist, int\*\* a, int size, int num)

{

queue <int> q;

int i, v;

q.push(num);

dist[num] = 0;

while (!q.empty())

{

num = q.front();

q.pop();

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

{

if (dist[i] > dist[num] + a[num][i] && a[num][i] != 0)

{

q.push(i);

dist[i] = dist[num] + a[num][i];

}

}

}

}

int main()

{

int i, j, size, \*\* a, \* dist, ran, num, type;

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setvbuf(stdin, NULL, \_IONBF, 0);

setvbuf(stdout, NULL, \_IONBF, 0);

printf("Введите размеры массива ");

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

printf("Выберите тип матрицы:\n1-неориентированный\n2-ориентированный\n");

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

printf("Введите начальную строку ");

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

printf("Матрица смежности:\n");

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

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

srand(time(NULL));

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

{

dist[i] = INT\_MAX;

}

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

{

a[i] = (int\*)malloc(size \* sizeof(int));

}

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

{

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

{

a[i][j] = 0;

}

}

switch (type)

{

case 1:

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

{

for (j = 1 + i; j < size; j++)

{

ran = rand() % 101;

if (ran > 40) a[i][j] = 1;

else a[i][j] = 0;

}

}

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

{

for (j = 1 + i; j < size; j++)

{

if (a[i][j] == 1) a[i][j] = rand() % 8;

}

}

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

{

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

{

a[j][i] = a[i][j];

}

}

break;

case 2:

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

{

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

{

ran = rand() % 101;

if (ran > 40) a[i][j] = 1;

else a[i][j] = 0;

}

}

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

{

j = i;

a[i][j] = 0;

}

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

{

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

{

if (a[i][j] == 1) a[i][j] = rand() % 8;

}

}

}

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

{

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

{

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

}

printf("\n");

}

BFS(dist, a, size, num);

printf("\nРасстояния до вершин: ");

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

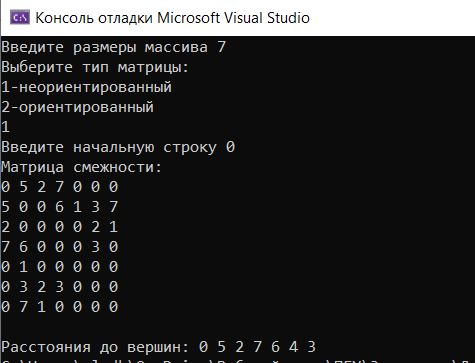
{

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

}

}

Результат работы программы



**Вывод:** мне удалось написать алгоритм поиска расстояний в списке и матрице смежности