Пензенский государственный университет   
Кафедра «Вычислительная техника»

**Отчет**о выполнении лабораторной работы №7 по дисциплине “Логика и основа алгоритмизации в инженерных задачах” на тему**:**

**Поиск расстояний во взвешенном графе**

Выполнили студенты гр. 19ВВ4:   
Хлыстов А. Ю.  
Привалов А. Э.

Проверили:   
Юрова О. В.  
Митрохин М. А.

Пенза, 2020 г.

**Название**Поиск расстояний во взвешенном графе

**Цель работы**Научиться реализовывать алгоритм поиска расстояний в графе.

**Лабораторное задание**  
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++.

**Код программы**

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <time.h>

#include <queue>

using namespace std;

struct graph {

int\*\* matrix;

int size;

}m;

int\* generate\_list(int size, int value) {

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

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

NUM[i] = value;

}

return NUM;

}

void outputMatrix(graph m) {

printf\_s(" ");

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

printf\_s("%2d ", i);

}

printf\_s("\n");

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

printf\_s("%2d ", i);

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

printf\_s("%2d ", m.matrix[i][j]);

}

printf\_s("\n");

}

}

//Заполнение массива

void fillMatrix(graph m) {

int value;

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

for (int j = i + 1; j < m.size; j++) {

value = rand() % 10;

if (value < 3)

value = 0;

else

value = rand() % 8;

m.matrix[i][j] = value;

m.matrix[j][i] = m.matrix[i][j];

}

}

}

//Создание двумерного массива

int\*\* createMatrix(int size) {

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

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

matrix[i] = (int\*)calloc(size, sizeof(int));

}

return matrix;

}

void BFSD(int v, graph m, queue <int> q, int\* DIST) {

q.push(v);

DIST[v] = 0;

while (q.empty() != true) {

v = q.front();

q.pop();

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

if (m.matrix[v][i] > 0 && DIST[i] > DIST[v] + m.matrix[v][i]) {

q.push(i);

DIST[i] = DIST[v] + m.matrix[v][i];

}

}

}

}

bool checkNewVertex(int\* NUM) {

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

if (NUM[m.size - i - 1] == 0) {

return true;

break;

}

}

return false;

}

int main()

{

setlocale(LC\_ALL, "Rus");

srand(time(NULL));

queue <int> q;

int\* DIST;

int v;

printf\_s("Количество вершин графа:");

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

DIST = generate\_list(m.size, 1000);

m.matrix = createMatrix(m.size);

fillMatrix(m);

outputMatrix(m);

printf\_s("\n");

printf\_s("Расстояния до вершин: \n\n");

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

printf\_s("%2d: ", i);

BFSD(i, m, q, DIST);

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

printf\_s("%d ", DIST[i]);

DIST[i] = 1000;

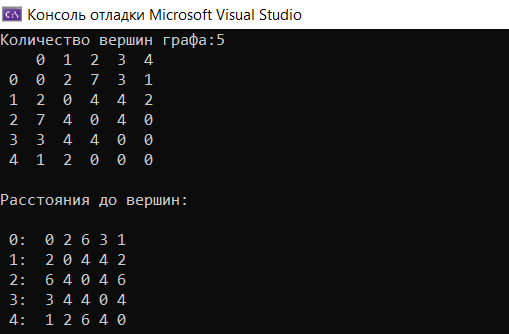
}

printf\_s("\n");

}

}

**Результаты работы программы**



**Выводы**В ходе выполнения лабораторной работы была разработана программа, выполняющая алгоритм обхода графа в ширину.