Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №6

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Поиск расстояний в графе»

Выполнили:

студенты группы 19ВВ2

Аниськин Н.В.

Приняли:

старший преподаватель

### Название

Программа, содержащая в себе функцию поиска расстояний между вершинами графа.

### Цель работы

Научиться создавать функцию нахождения расстояний между вершинами невзвешенного графа.

**Лабораторное задание**

### **Задание 1**

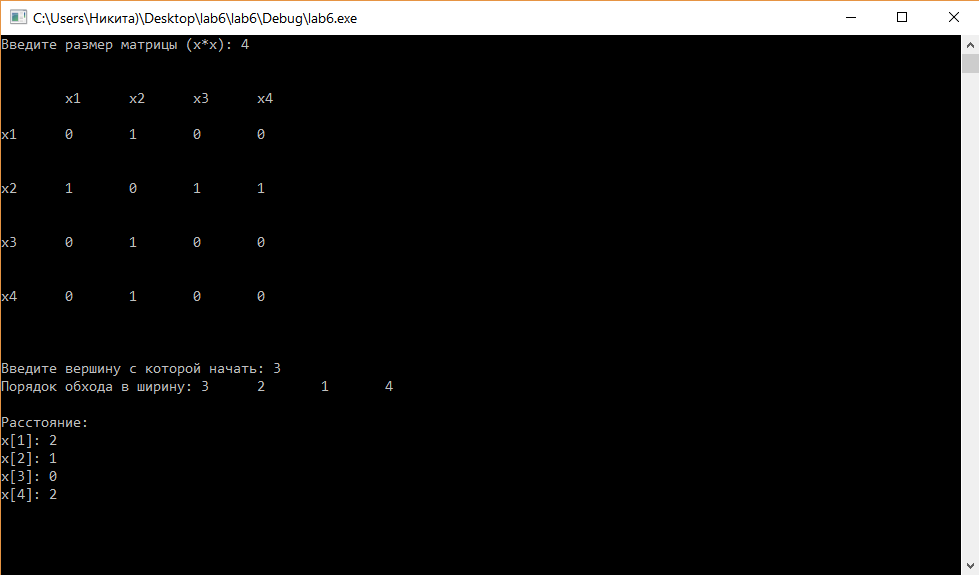
1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного графа *G*. Выведите матрицу на экран.
2. Для сгенерированного графа осуществите процедуру поиска расстояний, реализованную в соответствии с приведенным выше описанием. При реализации алгоритма в качестве очереди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

### **Пояснительный текст к программе**

Программа представляет из код, состоящий из основной функции и функции обхода графа в ширину.

1. Генерируется матрица смежности (с псевдослучайными значениями), выводится в виде графа.
2. С клавиатуры вводится вершина, с которой будет выполнен обход графа.
3. Все вершины помечаются как непосещённые ( при помощи вектора dist[i]=-1).
4. Выполняется функция обхода графа, которая вычисляет расстояния между вершинами графа и выводит их на экран .

### **Результаты работы программы**

****

**Рисунок 1 — Результаты работы программы**

### **Выводы**

В ходе выполнения лабораторной работы была разработана программа, выполняющая поиск расстояний между вершинами графа.

Получили опыт в создании проектов в среде Microsoft Visual Studio, научились писать и отлаживать линейные программы на языке Си.

### **Листинг программы**

### Файл «Source.cpp»

#include <stdio.h>

#include <conio.h>

#include <time.h>

#include <locale.h>

#include <stdlib.h>

#include <queue>

#include <vector>

using namespace std;

int i, j, m;

bool\* visited = new bool[m];

int\*\* graph;

vector<int> dist;

void BFS(int st)

{

queue<int> Queue = {}; // инициализация очереди

Queue.push(st); // помещаем в очередь вершину

dist[st] = 0;

while (!Queue.empty()){

// пока очередь не пустая

st = Queue.front(); // извлекаем вершину

Queue.pop(); // извлекаем элемент из очереди

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

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

{ // проверяем для нее все смежные вершины

if ((graph[st][i] != 0) && (dist[i]==-1))

{ // если вершина смежная и не обнаружена

Queue.push(i); // добавляем ее в очередь

dist[i]=dist[st]+1; // складываем расстояния }

}

}

}

void main()

{

setlocale(LC\_ALL, "Rus");

printf("Введите размер матрицы (x\*x): ");

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

printf("\n\n");

////создание и генерация графа

graph = new int\*[m];

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

graph[i] = new int[m];

}

srand(time(NULL));

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

for (j = i; j < m; ++j){

/\* if ((i % 2>0) && (j % 2>0))

{\*/

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

/\*}

else

{

graph[i][j] = graph[j][i] = 0;

}\*/

}

}

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

for (j = i; j < m; ++j){

if (j == i)

{

graph[j][i] = 0;

}

}

}

////////////////////////// ВЫВОД графа ////////

printf(" \t");

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

{

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

}

printf("\n\n");

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

{

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

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

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

printf("\n\n\n");

}

/////////////////// BFS

int vershina;

printf("\nВведите вершину с которой начать: ");

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

while (vershina > m) {

printf("\nТакой вершины не существует\n");

printf("\nВведите вершину с которой начать: ");

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

}

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

dist.push\_back(-1);

}

printf("Порядок обхода в ширину: ");

BFS(vershina-1);

printf("\n\nРасстояние:\n");

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

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

printf("]: ");

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

}

\_getch();

}