Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**Отчёт**

по лабораторной работе №6

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Поиск расстояний в графе»

Выполнил:

 студент группы 20ВВ2

Лазутин Д.Д.

 Принял:

 к.т.н., доцент Юрова О.В.

Пенза 2020

### **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++.

**3.**\* Реализуйте процедуру поиска расстояний для графа, представленного списками смежности. (В разработке)

### **Задание 2\*** (В разработке)

1. Реализуйте процедуру поиска расстояний на основе обхода в глубину.
2. Реализуйте процедуру поиска расстояний на основе обхода в глубину для графа, представленного списками смежности.
3. Оцените время работы реализаций алгоритмов поиска расстояний на основе обхода в глубину и обхода в ширину для графов разных порядков.

**Описание метода решения:**

Объявлены переменные: матрица G, размер матрицы S, массив посещённых вершин графа Visited, счётчики i и j, глобальная переменная – очередь q, номер стартовой вершины N и датчик повторного посещения стартовой вершины ReVis, чтобы вовремя прекратить процедуру обхода.  
Создаём и заполняем матрицу смежности, выводим её на экран. Далее выполняем обход графа в ширину посредством вызова функции BFS до тех пор, пока все вершины графа не станут посещёнными (т.е. все элементы массива Visited должны стать больше, чем -1).

Элемент массива Visited[i], равный -1, считается номером изолированной вершины, а элемент, равный 0 – номером стартовой вершины (т.е. откуда начался обход). Остальные элементы будут больше 0, т.е. достижимы из стартовой вершины через 1 или несколько рёбер.

**Листинг** LZ\_6.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <queue>

#include <iostream>

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include <locale.h>

using namespace std;

queue <int> q;

void BFS(int v, int \*\*M, int \*Vis, int Size)

{

int i;

q.push(v); Vis[v]=0;

while (!q.empty())

{

v=q.front();

q.pop();

printf("%3d", v+1);

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

{

if((Vis[i]==-1)&&(M[v][i]==1))

{

q.push(i);

Vis[i]=Vis[v]+1;

}

}

}

}

void main()

{

int \*\*G=NULL, i=0, j=0, S=0, \*Visited=NULL, N;

char ReVis=0;

setlocale(LC\_ALL, "Rus");

srand(time(NULL));

printf("Введите размерность матрицы смежности: ");

scanf("%d", &S);

while(S<=0)

{

printf("Некорректно введённое число! Введите целое положительное число: ");

scanf("%d", &S);

}

G=(int\*\*)malloc(S\*sizeof(int));

Visited=(int\*)malloc(S\*sizeof(int));

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

Visited[i]=-1;

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

{

G[i]=(int\*)malloc(S\*sizeof(int));

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

{

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

if(G[i][i]!=0)

G[i][j]=0;

}

}

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

{

printf("\n%4d|", i+1);

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

{

G[j][i]=G[i][j];

printf("%2d", G[i][j]);

}

}

printf("\nС какой вершины начать обход в ширину? "); scanf("%d", &N);

while (N<1 || N>S)

{

printf("Некорректно введённое число! Введите целое положительное число в пределах размера матрицы смежности: ");

scanf("%d", &N);

}

printf("\nПорядок обхода графа в ширину: ");

for(i=N-1;Visited[i]<0;i++)

{

if(ReVis==1 && i==N-1)

break;

if(i==S-1)

i=0;

BFS(i, G, Visited, S);

ReVis=1;

}

printf("\nВектор расстояний от выбранной вершины до всех остальных: ");

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

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

printf("\n");

free(Visited);

free(G);

}

**Результаты работы программ**

Результаты работы программы для 1-го задания представлены на рис.1-2.

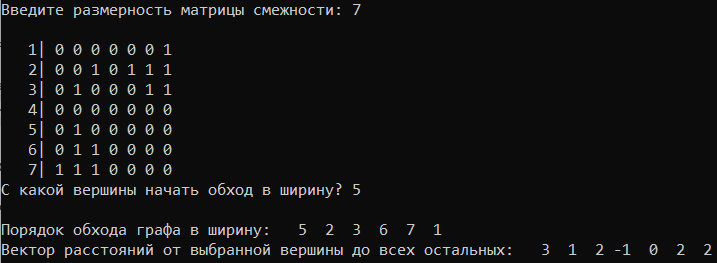


Рис.1.

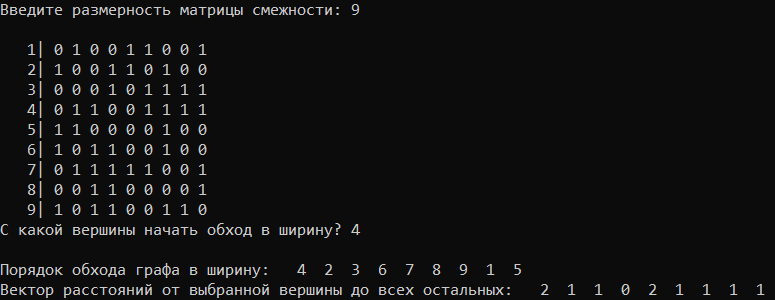


Рис.2.