Пензенский государственный университет

Кафедра «Вычислительная техника»

**Отчёт**

По лабораторной работе №6

по дисциплине: «Логика и основы оптимизации в инженерных задачах»

## на тему «Поиск расстояний в графе»

Выполнил студент группы 19ВВ4:

Шахов К.М

Приняли:

Митрохин М. А.

Юрова О.В.

Пенза 2020

**Цель работы:** освоить алгоритм поиска расстояний в графе.

**Общие сведения:**

Поиск расстояний – довольно распространенная задача анализа графов.

Для поиска расстояний можно использовать процедуры обхода графа. Для этого при каждом переходе в новую вершину необходимо запоминать, сколько шагов до нее мы сделали. При этом вектор, который хранил информацию о посещении вершин становится вектором расстояний. Довольно просто модернизировать для поиска расстояний в графе алгоритм обхода в ширину, т.к. этот алгоритм проходит вершины по уровням удаленности, то для не ориентированного графа для вершин каждого следующего уровня глубины расстояние от исходной вершины увеличивается на 1. Удалённость в данном случае понимается как количество ребер, по которым необходимо прейти до достижения вершины.

**Практическая часть**

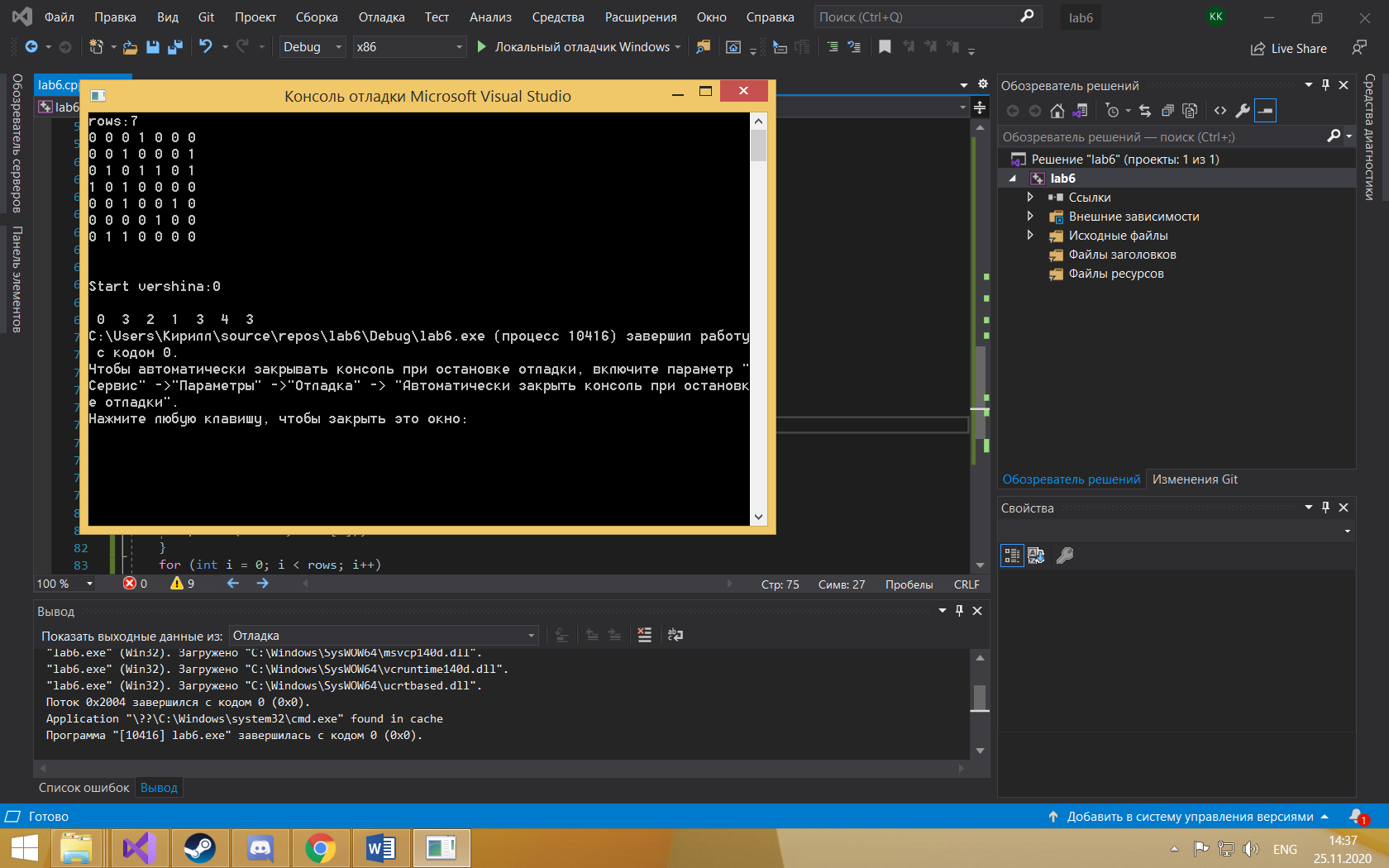
1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного графа *G*. Выведите матрицу на экран.

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++.

**Листинг.**

[#define](https://vk.com/im?sel=394368468&st=%23define) \_CRT\_SECURE\_NO\_WARNINGS  
[#include](https://vk.com/im?sel=394368468&st=%23include) <iostream>  
[#include](https://vk.com/im?sel=394368468&st=%23include) <stdio.h>  
[#include](https://vk.com/im?sel=394368468&st=%23include) <conio.h>  
[#include](https://vk.com/im?sel=394368468&st=%23include) <malloc.h>  
[#include](https://vk.com/im?sel=394368468&st=%23include) <stdlib.h>  
[#include](https://vk.com/im?sel=394368468&st=%23include) <time.h>  
[#include](https://vk.com/im?sel=394368468&st=%23include) <queue>  
using namespace std;  
int\*\* G;  
int\* dist;  
int ver;  
  
void BFS(int s, int rows) {  
int v;  
queue <int> q;  
q.push(s);  
dist[s] = 0;  
while (!q.empty()) {  
v = q.front();  
//printf(" %d", v);  
q.pop();  
for (int i = 0; i < rows; i++) {  
if (G[v][i] == 1 && dist[i] == -1) {  
dist[i] =dist[v]+1;  
q.push(i);  
}  
}  
}  
}  
  
int main()  
{  
int rows;  
int n = 0;  
int k = 0;  
system("cls");  
printf("rows:");  
scanf("%d", &rows);  
srand(time(NULL));  
G = (int\*\*)malloc(rows \* sizeof(int\*));  
  
for (int i = 0; i < rows; i++)  
{  
G[i] = (int\*)malloc(rows \* sizeof(int));  
}  
for (int i = 0; i < rows; i++)  
{  
for (int j = 0; j < rows; j++)  
{  
G[i][j] = rand() % 2;  
if (i == j) {  
G[i][j] = 0;  
}  
G[j][i] = G[i][j];  
}  
}  
  
for (int i = 0; i < rows; i++)  
{  
for (int j = 0; j < rows; j++)  
{  
printf("%d ", G[i][j]);  
}  
printf("\n");  
}  
printf("\n");  
  
dist = (int\*)malloc(rows \* sizeof(int));  
  
for (int i = 0; i < rows; i++) {  
dist[i] = -1;  
}  
printf("\n");  
printf("Start Vershina:");  
scanf("%d", &ver);  
printf("\n");  
BFS(ver, rows);  
for (int i = 0; i < rows; i++) {  
printf("%d ", dist[i]);  
}  
for (int i = 0; i < rows; i++)  
free(G[i]);  
free(G);  
getchar();  
return 0;  
}

## Результат работы.



**Вывод:** Научились находить расстояние в графе с помощью процедур обхода графа.