

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет Информатика и вычислительная техника

Кафедра Кибербезопасность информационных систем

**ЛАБОРАТОРНАЯ РАБОТА № 3**

на тему «Компоненты связности графа»

Выполнил обучающийся гр. ВКБ31

Деев Дмитрий Андреевич

Проверил

Доцент, Савельев Василий Александрович

Ростов-на-Дону

2022

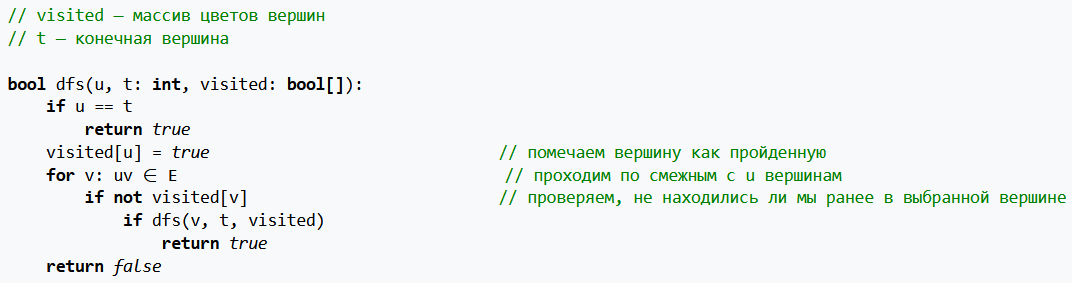
Понятие компоненты связности вытекает из понятия связности графа. Попросту говоря, компонента связности - часть графа (подграф), являющаяся связной. Формально, компонента связности - набор вершин графа, между любой парой которых существует путь.

**Задание 1.**

Проверить, что две вершины принадлежат одной компоненте связности.

**Алгоритм.**

Смысл алгоритма заключается в том, чтобы запустить обход в глубину из вершины s и проверять при каждом посещении вершины, не является ли она искомой вершиной t. Так как в первый момент времени все пути в графе "белые", то если вершина t и была достижима из s, то по лемме о белых путях в какой-то момент времени мы зайдём в вершину t, чтобы её покрасить.

**Реализация.**

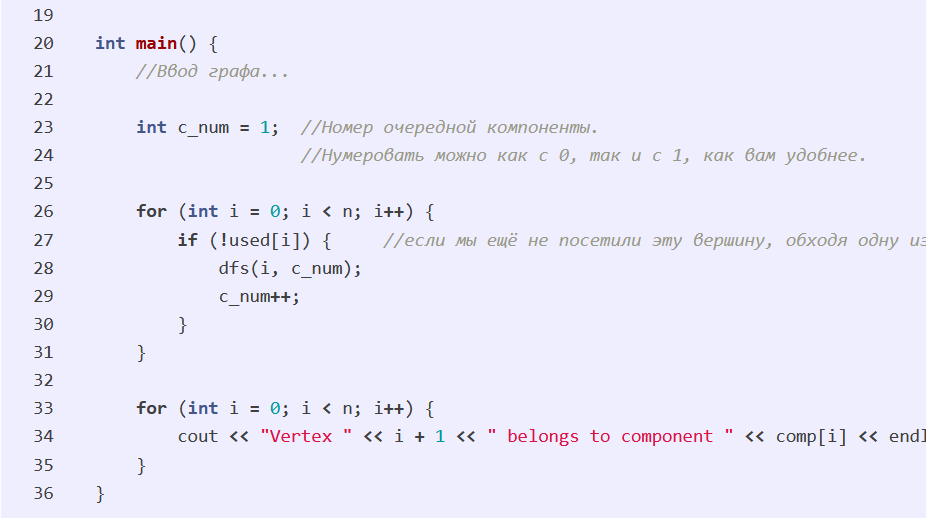
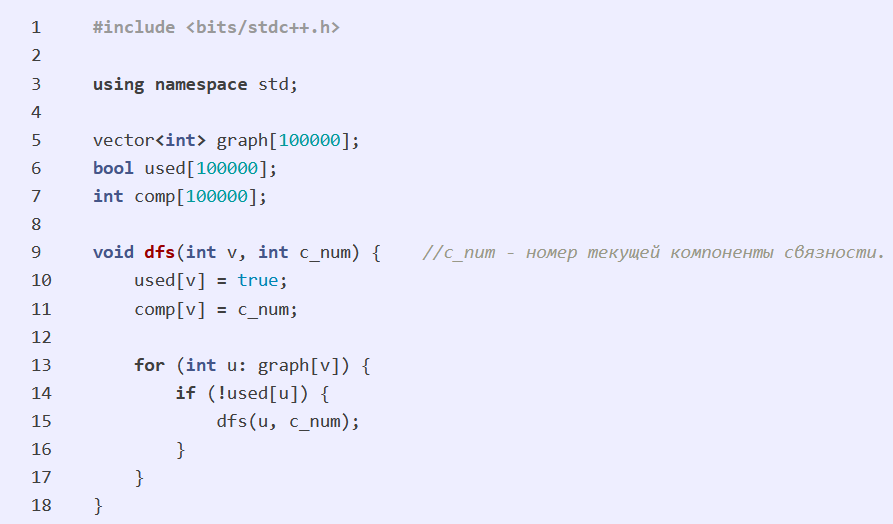
**Задание 2.**

Выделить компоненту связности, к которой принадлежит данная вершина.

**Алгоритм.**

Для поиска компонент связности используется обычный DFS практически без модификаций (можно использовать и BFS). При запуске обхода из одной вершины, он гарантированно посетит все вершины, до которых возможно добраться, то есть, всю компоненту связности, к которой принадлежит начальная вершина. Для нахождения всех компонент просто попытаемся запустить обход из каждой вершины по очереди, если мы ещё не обошли её компоненту ранее.

**Реализация.**



**Задание 3.**

Разбить граф на составляющие его компоненты связности.

**Алгоритм.**

Мы будем производить **серию обходов**: сначала запустим обход из первой вершины, и все вершины, которые он при этом обошёл — образуют первую компоненту связности. Затем найдём первую из оставшихся вершин, которые ещё не были посещены, и запустим обход из неё, найдя тем самым вторую компоненту связности. И так далее, пока все вершины не станут помеченными.

Итоговая **асимптотика** составит O(n + m): в самом деле, такой алгоритм не будет запускаться от одной и той же вершины дважды, а, значит, каждое ребро будет просмотрено ровно два раза (с одного конца и с другого конца).

**Реализация.**

