|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Информатика и системы управления (ИУ)

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ7)

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7  
«Обработка графов»**

Студент, группа **Пронин А.С. ИУ7-32Б**

*2020 г.*

# Условия задачи

Обработать графовую структуру в соответствии с заданным вариантом. Обосновать выбор необходимого алгоритма и выбор структуры для представления графов. Ввод данных осуществить на усмотрение программиста. Результат выдать в графической форме.

Найти все вершины графа, к которым от заданной вершины можно добраться по пути не длиннее А.

# Техническое задание

## Входные данные

1) Строка для ввода имени файла

2) Целое число от 0 до n-1 (для ввода номера начальной вершины, где n кол-во вершин)

3) Целое число >= 0 (для ввода максимальной длины A)

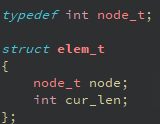
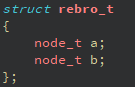
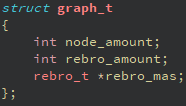
## Выходные данные

Вывод всех вершин графа, к которым от заданной начальной вершины можно добраться по пути не длиннее А.

## Функции программы

При запуске программы предлагается ввести имя файла из которого будет читаться граф, затем начальную вершину и максимальную длину пути(A). После чего выводится список достижимых вершин, и программа завершается.

## Структуры данных

**node\_t** обозначает что это переменная для хранения значения (номера) вершины  
 **rebro\_t** хранит 2 номера вершин (из a выходит в b входит => граф ориентированный, вес каждого ребра = 1)

**graph\_t** хранит кол-во вершин, кол-во рёбер и динамический массив рёбер

**elem\_t** это структура данных сделана для использования в очереди в поиске по графу в ширину, которая хранит текущую длину пройденного пути и номер вершины в которой мы сейчас находимся

Также была использована очередь из библиотеки <queue>

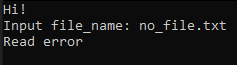
# Описание алгоритма

Для нахождения всех вершин графа, к которым от заданной начальной вершины можно добраться по пути не длиннее А, использовался алгоритм поиска в ширину. Этот алгоритм был выбран т.к. нам нужно определить дину пути для каждой вершины. Для этого алгоритма мы используем очередь и начинаем с того что помещаем в неё заданную начальную вершину с текущей длиной пути 0 (тип **elem\_t**). Далее пока очередь не опустеет: вытаскиваем передний элемент из очереди и если текущая длина не превышает максимальную (А) И если мы еще не уверены в том, что эта вершина достижима проходим по массиву рёбер, добавляем в очередь все вершины в которые введёт текущая с текущей длиной + 1 (не более A) и обозначаем то что передняя вершина достижима. После чего удаляем передний элемент из вершины.

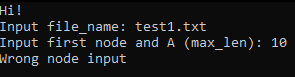
# Тесты

### Некорректный ввод:

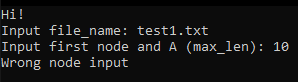
1. Несуществующий файл



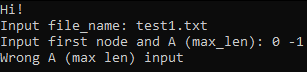
1. Некорректный файл



1. Несуществующая вершина



1. Отрицательная длина пути



# Контрольные вопросы

**1. Что такое граф?**

Граф – конечное множество вершин и соединяющих их ребер;

G = <V, E>. Если пары Е (ребра) имеют направление, то граф называется ориентированным; если ребро имеет вес, то граф называется взвешенным.

**2. Как представляются графы в памяти?**

С помощью матрицы смежности или списков смежности.

**3. Какие операции возможны над графами?**

Обход вершин, поиск различных путей, исключение и включение вершин.

**4. Какие способы обхода графов существуют?**

Обход в ширину (BFS – Breadth First Search), обход в глубину (DFS – Depth First Search).

**5. Где используются графовые структуры?**  
 Графовые структуры могут использоваться в задачах, в которых между элементами могут быть установлены произвольные связи, необязательно иерархические.

**6. Какие пути в графе Вы знаете?**

Эйлеров путь, простой путь, сложный путь, гамильтонов путь.

**7. Что такое каркасы графа?**

Каркас графа – дерево, в которое входят все вершины графа, и некоторые (необязательно все) его рёбра.

# Вывод

Для решения данной задачи используется алгоритм поиска в ширину, следовательно, если **V** – количество вершин графа, **Vд** – количество достижимых вершин (зависит от **A** – макс. длины пути и от начальной вершины), **E** – количество ребер, то алгоритм поиска всех вершин графа, к которым от заданной начальной вершины можно добраться по пути не длиннее **А** имеет сложность **O(Vд)**.

В реальности этот алгоритм может быть использован в создании игры: граф данном случае, является квадратным полем, ребра которого соединяют соседние клетки (как в шахматах), и разные фигуры ходят на разное расстояние. Здесь можно использовать наш алгоритм для нахождения всех клеток на которые может походить фигура в зависимости от её положения и характеристик передвижения (максимальная длина).