**Дейкстра(поиск кратчайшего пути) делал с разбиением на файлы**

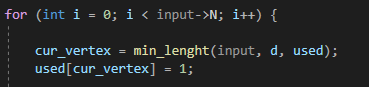
include.h - подключение нужных библиотек и define нужных значений

read.с - функции чтения данных

check.c - функции проверки данных

head\_functions - все основные функции

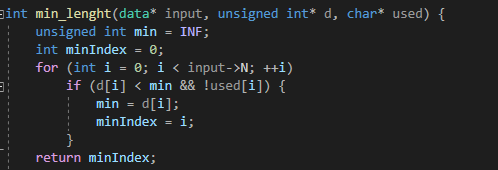
Описание алгоритма(dijkstra):



Стартуем с заданной вершины.

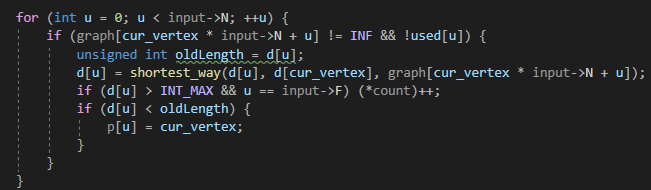
N(кол-во вершин) раз выбираем вершину, еще не рассмотренную, с минимальным путем, ведущим в начальную вершину. В cur\_vertex хранится выбранная вершина.

Работа min\_length:



В d хранятся кратчайшие пути от начальной вершины до i-ой.

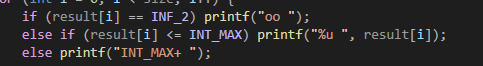
Для выбора минимальной просто пробегаем по еще не взятым вершинам и выбираем минимальное из d.



Для всех вершин, в которые есть путь из cur\_vertex(выбранная нами), смотрим, меньше ли путь до этой вершины через cur\_vertex, чем был до этого(oldLenght), и присваиваем, если меньше. Также указываем потомка, чтобы потом можно было восстанавливать путь до вершин.

Если путь через cur\_vertex > INT\_MAX и cur\_vertex - это вершина, до которой нужно восстановить путь, то count++(здесь храним количество путей больших INT\_MAX);

Вывод кратчайшего пути до каждой из вершин:



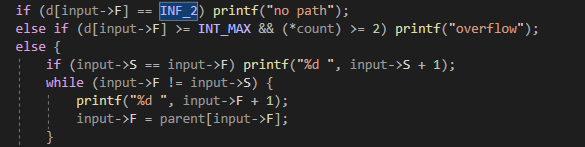
result - массив кратчайших путей(изначально заполнен INF\_2).

Если значение кратчайшего пути не изменилось, то пути нет.

Если кратчайший путь <= INT\_MAX, то выводим.

Иначе путь > INT\_MAX.

Восстановление пути:



Если значение кратчайшего пути не изменилось, то пути нет.

Если значение кратчайшего пути >= INT\_MAX и таких путей больше одного(для этого нам и нужен был count), то overflow.

Иначе восстанавливаем путь по массиву родителей. Идем вверх по дереву предков до начальной вершины.