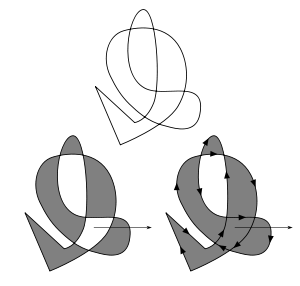
***Вычислительная геометрия. Принадлежность точки многоугольнику.***

В вычислительной геометрии известна **задача об определении принадлежности точки многоугольнику**. На плоскости даны [многоугольник](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D1%83%D0%B3%D0%BE%D0%BB%D1%8C%D0%BD%D0%B8%D0%BA) и [точка](https://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D1%87%D0%BA%D0%B0_(%D0%B3%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%8F)).

Многоугольник может быть как [выпуклым](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D0%BF%D1%83%D0%BA%D0%BB%D1%8B%D0%B9_%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE%D1%83%D0%B3%D0%BE%D0%BB%D1%8C%D0%BD%D0%B8%D0%BA), так и невыпуклым. Обычно предполагается, что многоугольник простой, то есть без самопересечений; но задачу рассматривают и для не-простых многоугольников. В последнем случае разные способы определения принадлежности точки многоугольнику могут привести к разным результатам. Различают алгоритмы без предварительной обработки; и алгоритмы с предварительной обработкой, в ходе которой создаются некоторые структуры данных, позволяющие в дальнейшем быстрее отвечать на множество запросов о принадлежности разных точек одному и тому же многоугольнику.

Метод трассировки луча

### Учёт числа пересечений



Один из стандартных методов определения принадлежности точки произвольному простому многоугольнику заключается в следующем. Выпустим луч из данной точки в произвольном направлении (например в положительном направлении горизонтальной оси), и посчитаем сколько раз луч пересекает рёбра многоугольника. Для этого достаточно пройтись в цикле по рёбрам многоугольника и определить, пересекает ли луч каждое ребро. Если число пересечений нечётно, то объявляется, что точка лежит внутри многоугольника, если чётно — то снаружи. Это основано на том простом наблюдении, что при движении по лучу с каждым пересечением границы точка попеременно оказывается то внутри, то снаружи многоугольника. Алгоритм известен под такими названиями, как *crossing number (count) algorithm* или *even-odd rule*.

В алгоритме возникает затруднение в вырожденном случае, когда луч пересекает вершину многоугольника. Один из приёмов для его преодоления заключается в том, чтобы считать, что такие вершины многоугольника лежат на бесконечно малую величину выше (или ниже) прямой луча, и стало быть пересечения на самом деле и нет.[[1]](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D0%BF%D1%80%D0%B8%D0%BD%D0%B0%D0%B4%D0%BB%D0%B5%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D0%B8_%D1%82%D0%BE%D1%87%D0%BA%D0%B8_%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE%D1%83%D0%B3%D0%BE%D0%BB%D1%8C%D0%BD%D0%B8%D0%BA%D1%83#cite_note-haines-1) Таким образом, пересечение луча с ребром засчитывается, если один из концов ребра лежит строго ниже луча, а другой конец — выше или лежит на луче.

Алгоритм работает за время O(*N*) для *N*-угольника.

### Учёт числа оборотов



Рассмотрим число оборотов, которое делает ориентированная граница многоугольника вокруг данной точки *P*. В алгебраической топологии это число называется [индексом точки относительно кривой](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D1%8F%D0%B4%D0%BE%D0%BA_%D1%82%D0%BE%D1%87%D0%BA%D0%B8_%D0%BE%D1%82%D0%BD%D0%BE%D1%81%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE_%D0%BA%D1%80%D0%B8%D0%B2%D0%BE%D0%B9).[[2]](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D0%BF%D1%80%D0%B8%D0%BD%D0%B0%D0%B4%D0%BB%D0%B5%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D0%B8_%D1%82%D0%BE%D1%87%D0%BA%D0%B8_%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE%D1%83%D0%B3%D0%BE%D0%BB%D1%8C%D0%BD%D0%B8%D0%BA%D1%83#cite_note-2) Оно может быть вычислено следующим образом. Как и раньше, выпустим луч из *P* в произвольном направлении и рассмотрим рёбра, которые он пересекает. Каждому пересечению присвоим число +1 или -1, в зависимости от того, как ребро пересекает луч — по часовой (слева направо) или против часовой стрелки (справа налево). Эти два случая можно различить по знаку скалярного произведения между направляющим вектором ребра и нормалью к направляющему вектору луча.[[3]](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D0%BF%D1%80%D0%B8%D0%BD%D0%B0%D0%B4%D0%BB%D0%B5%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D0%B8_%D1%82%D0%BE%D1%87%D0%BA%D0%B8_%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE%D1%83%D0%B3%D0%BE%D0%BB%D1%8C%D0%BD%D0%B8%D0%BA%D1%83#cite_note-foley-3) Сумма полученных величин и есть [индекс точки относительно кривой](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D1%8F%D0%B4%D0%BE%D0%BA_%D1%82%D0%BE%D1%87%D0%BA%D0%B8_%D0%BE%D1%82%D0%BD%D0%BE%D1%81%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE_%D0%BA%D1%80%D0%B8%D0%B2%D0%BE%D0%B9). Сумма будет положительной или отрицательной, в зависимости от ориентации границы. Если она не равна нулю, то будем считать, что точка лежит внутри многоугольника, иначе — снаружи.

Такой алгоритм известен под названием *nonzero winding rule*.[[3]](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D0%BF%D1%80%D0%B8%D0%BD%D0%B0%D0%B4%D0%BB%D0%B5%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D0%B8_%D1%82%D0%BE%D1%87%D0%BA%D0%B8_%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE%D1%83%D0%B3%D0%BE%D0%BB%D1%8C%D0%BD%D0%B8%D0%BA%D1%83#cite_note-foley-3)

Для простых многоугольников этот метод работает так же, как и метод, основанный на подсчёте числа пересечений. Разница между ними проявляется при рассмотрении многоугольников с самопересекающейся границей.

### Выпуклые и звёздные многоугольники

Принадлежность точки [выпуклому](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D0%BF%D1%83%D0%BA%D0%BB%D1%8B%D0%B9_%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE%D1%83%D0%B3%D0%BE%D0%BB%D1%8C%D0%BD%D0%B8%D0%BA) или [звёздному](https://ru.wikipedia.org/wiki/%D0%97%D0%B2%D1%91%D0%B7%D0%B4%D0%BD%D0%B0%D1%8F_%D0%BE%D0%B1%D0%BB%D0%B0%D1%81%D1%82%D1%8C) *N*-угольнику может быть определена при помощи [двоичного поиска](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA) за время O(log *N*), при затрате O(*N*) памяти и O(*N*) времени на предварительную обработку.[[6]](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D0%BF%D1%80%D0%B8%D0%BD%D0%B0%D0%B4%D0%BB%D0%B5%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D0%B8_%D1%82%D0%BE%D1%87%D0%BA%D0%B8_%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE%D1%83%D0%B3%D0%BE%D0%BB%D1%8C%D0%BD%D0%B8%D0%BA%D1%83#cite_note-6)

### Произвольный многоугольник

Задачу о принадлежности точки произвольному простому многоугольнику можно рассматривать как частный случай [задачи о локализации точки](https://ru.wikipedia.org/w/index.php?title=%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D1%82%D0%BE%D1%87%D0%BA%D0%B8&action=edit&redlink=1)[[en]](https://en.wikipedia.org/wiki/Point_location) в планарном подразбиении. Для *N*-угольника эта задача может быть решена за время O(log2 *N*) с использованием O(*N*) памяти и O(*N* log *N*) времени на предобработку.