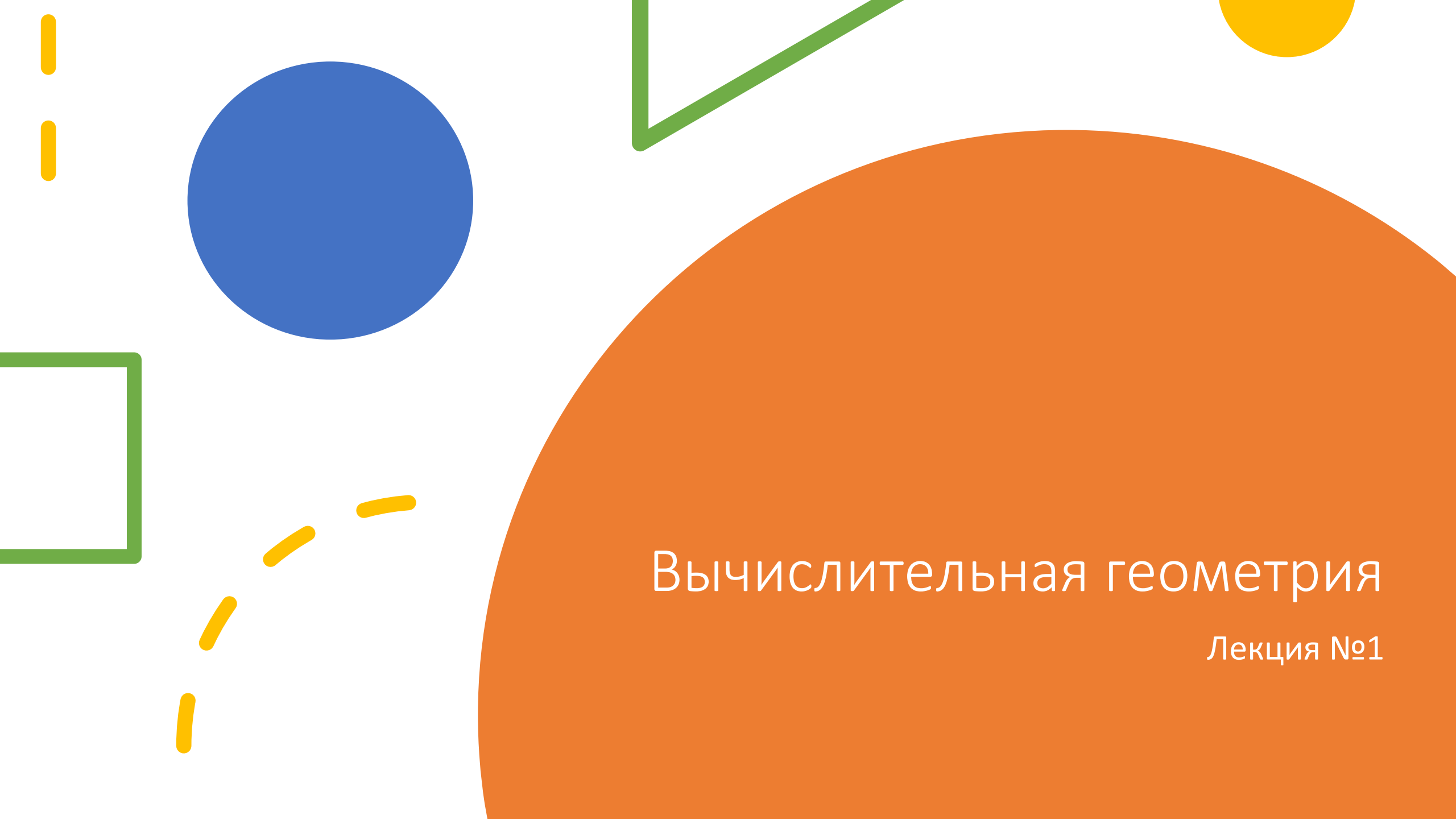


Вычислительная геометрия

Лекция №1

Вычислительная геометрия

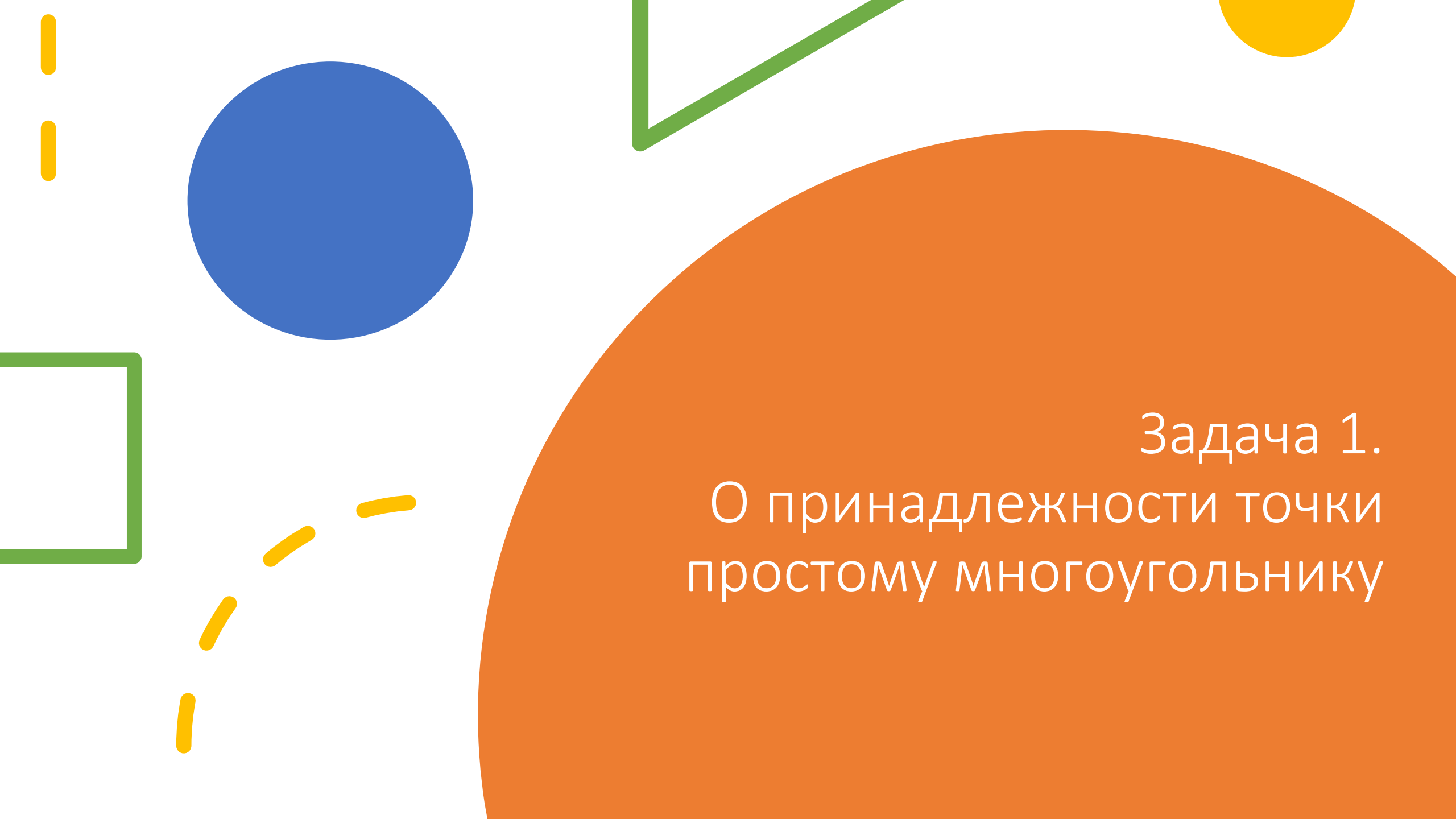
Раздел теории алгоритмов, который занимается построением эффективных алгоритмов входными и выходными данными которого являются геометрические объекты (точки, прямые, окружности и т.д.).

Вещественнозначная PAM (равнодоступная адресная машина)

Элементарные операции имеют единичную стоимость, то есть выполняются за единицу времени.

Элементарные операции:

1. Арифметические операции (сложение, вычитание, умножение, деление).
2. Операции сравнения ($<$, $>$, $=$, $\leq$, $\geq$, $\neq$).
3. Косвенная адресация памяти: индексирование массивов. Допускаются только целочисленные индексы и за единицу времени можно получить доступ к произвольному элементу массива.
4. Извлечение корней к-ой степени, тригонометрические функции, логарифмы, аналитические функции.
5. Любая ячейка памяти может хранить только одно число.


Задача 1.
О принадлежности точки
простому многоугольнику

Простой многоугольник

Область плоскости, ограниченная замкнутой не самопересекающейся ломаной.

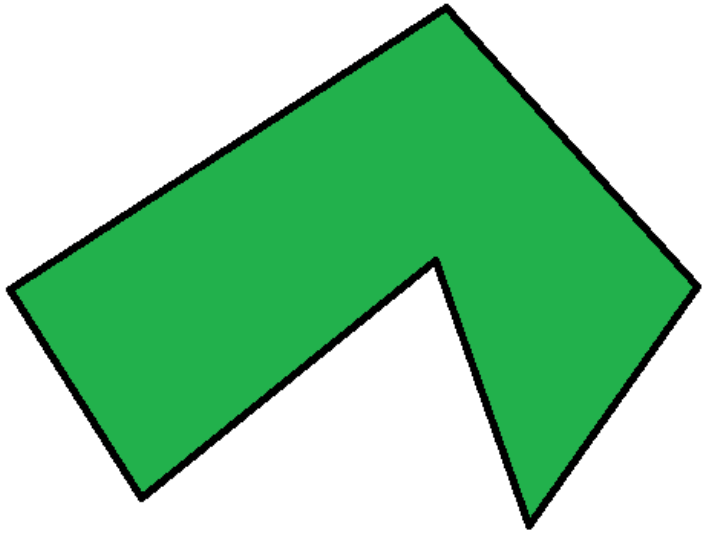


Рис. 1. Простой многоугольник

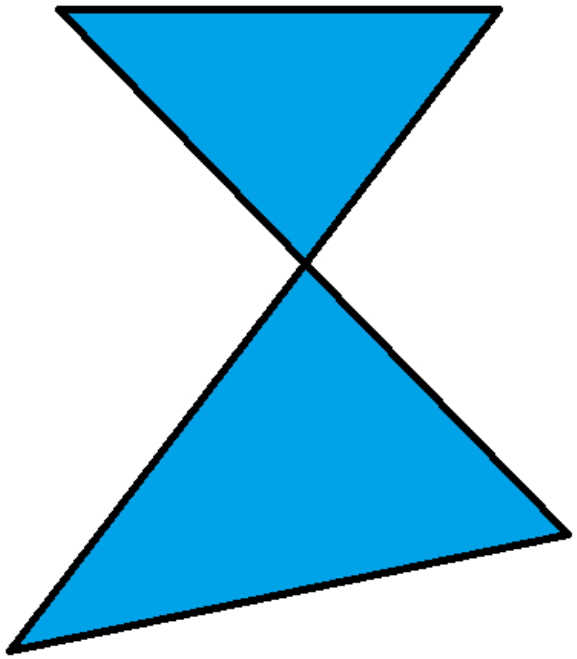


Рис. 2. Не простой многоугольник

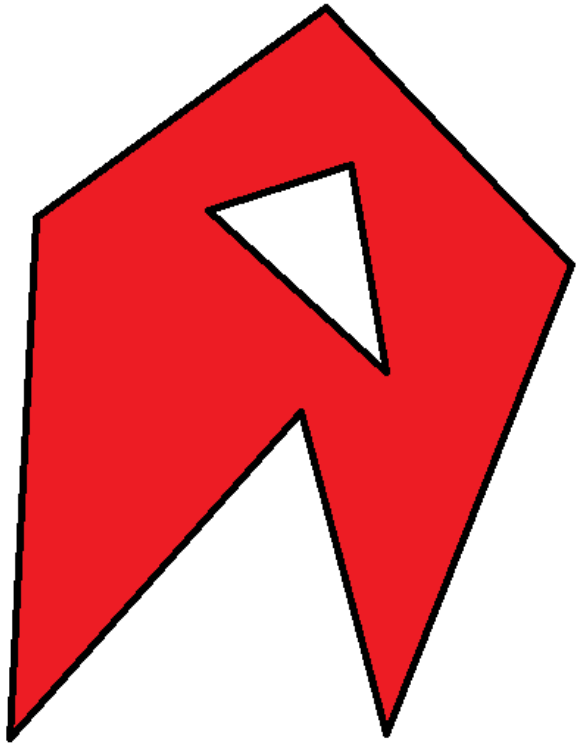
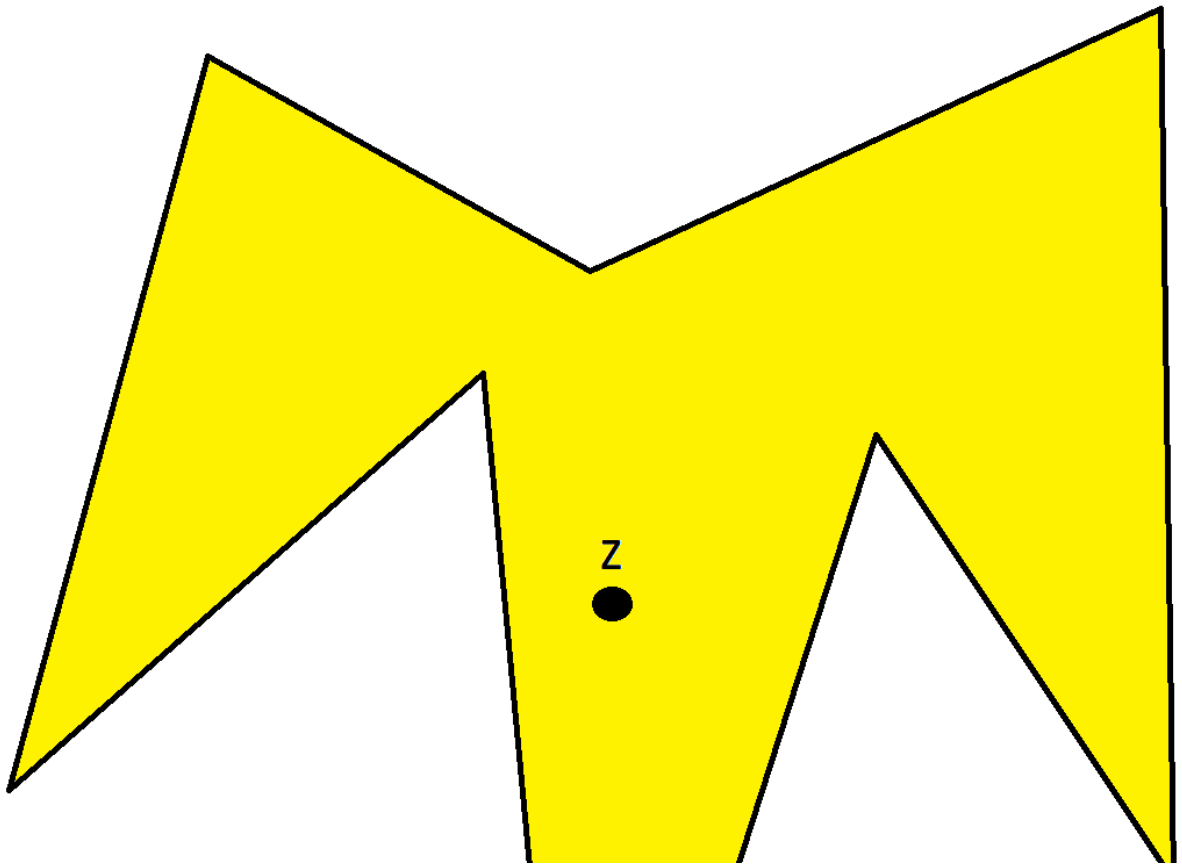


Рис. 3. Многоугольник с дыркой

ЗАДАЧА 1.

Дан простой многоугольник P и точка Z.

Выяснить находится ли точка Z внутри многоугольника P.



Ситуация 1 (не вырожденная)

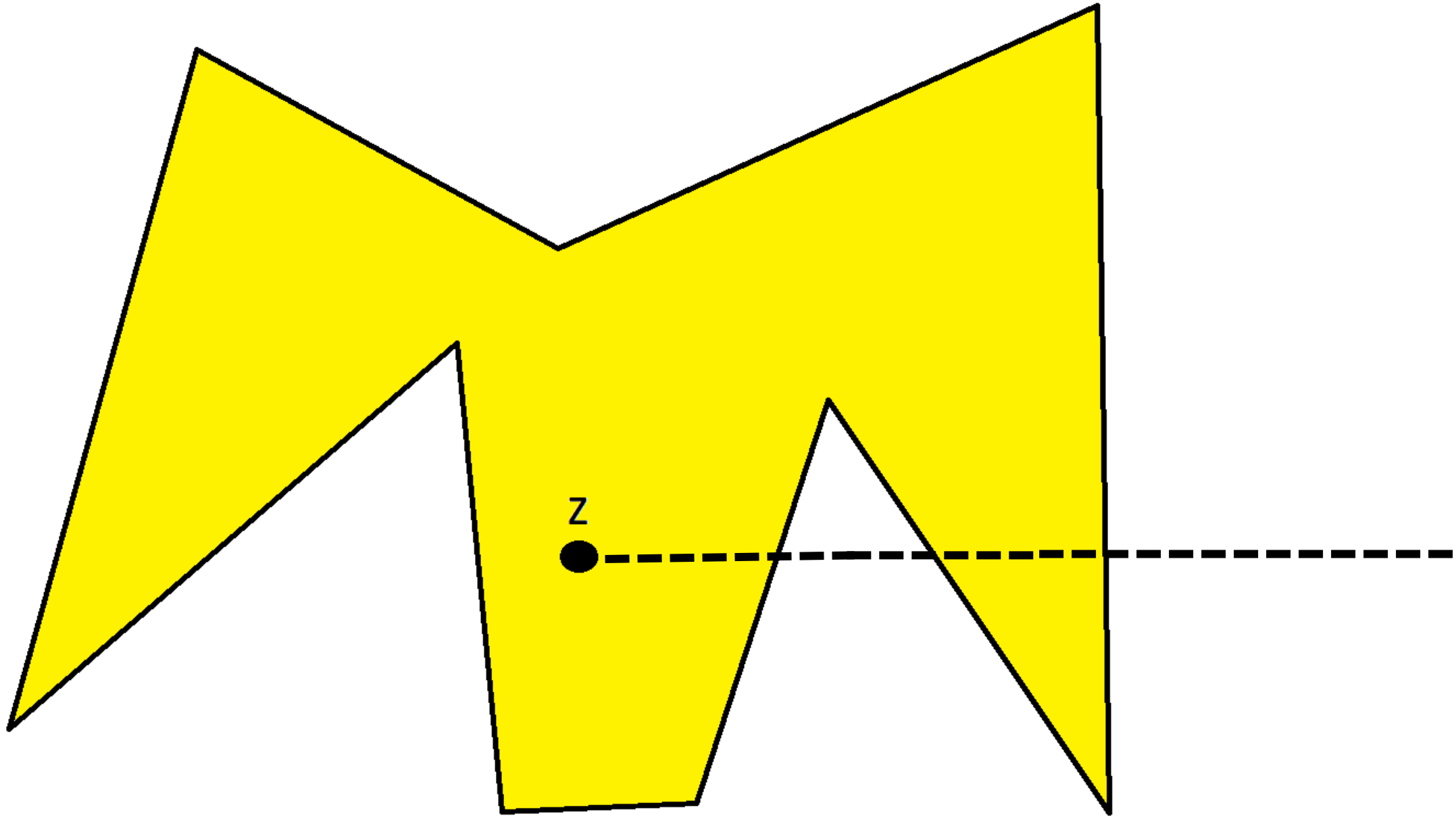
Точка Z может лежать как внутри так и снаружи многоугольника

Алгоритм решения ситуации 1

Выпустить луч из точки z.

Подсчитать сколько раз он пересекает границу.

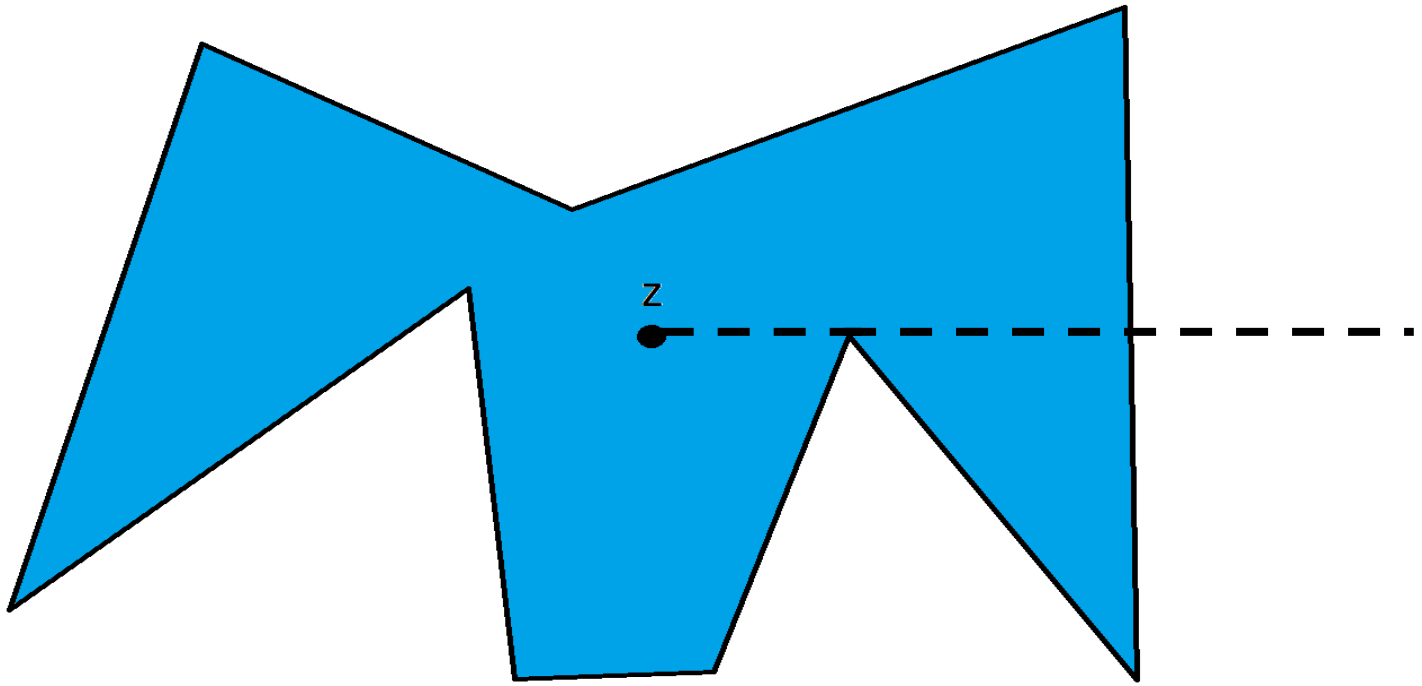
Если луч пересекает нечетное количество раз, то точка z находится внутри многоугольника.

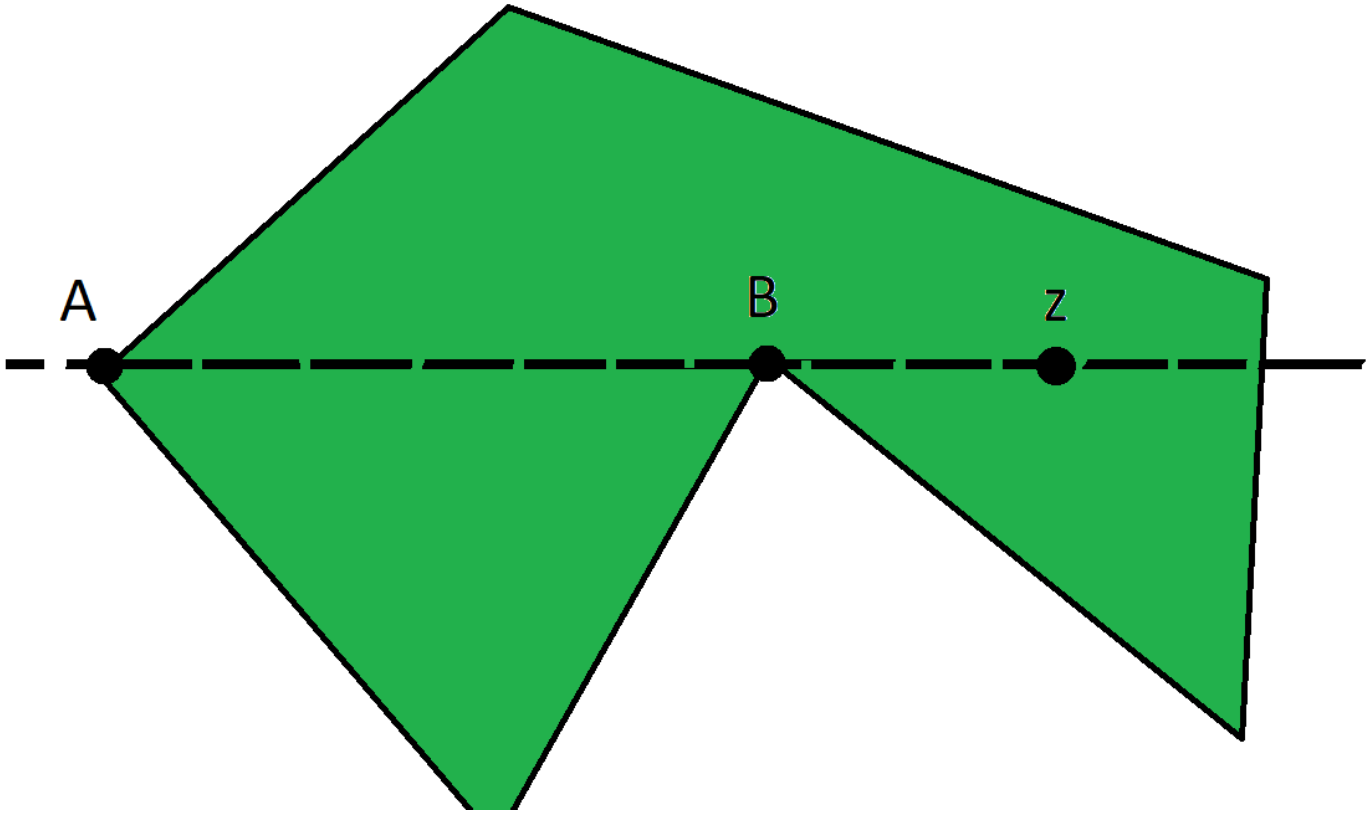

z

Ситуация 2 (вырожденная)

Точка z лежит внутри, но луч попал в вершину многоугольника.

В этом случае мы получаем пересечение с 2 ребрами.

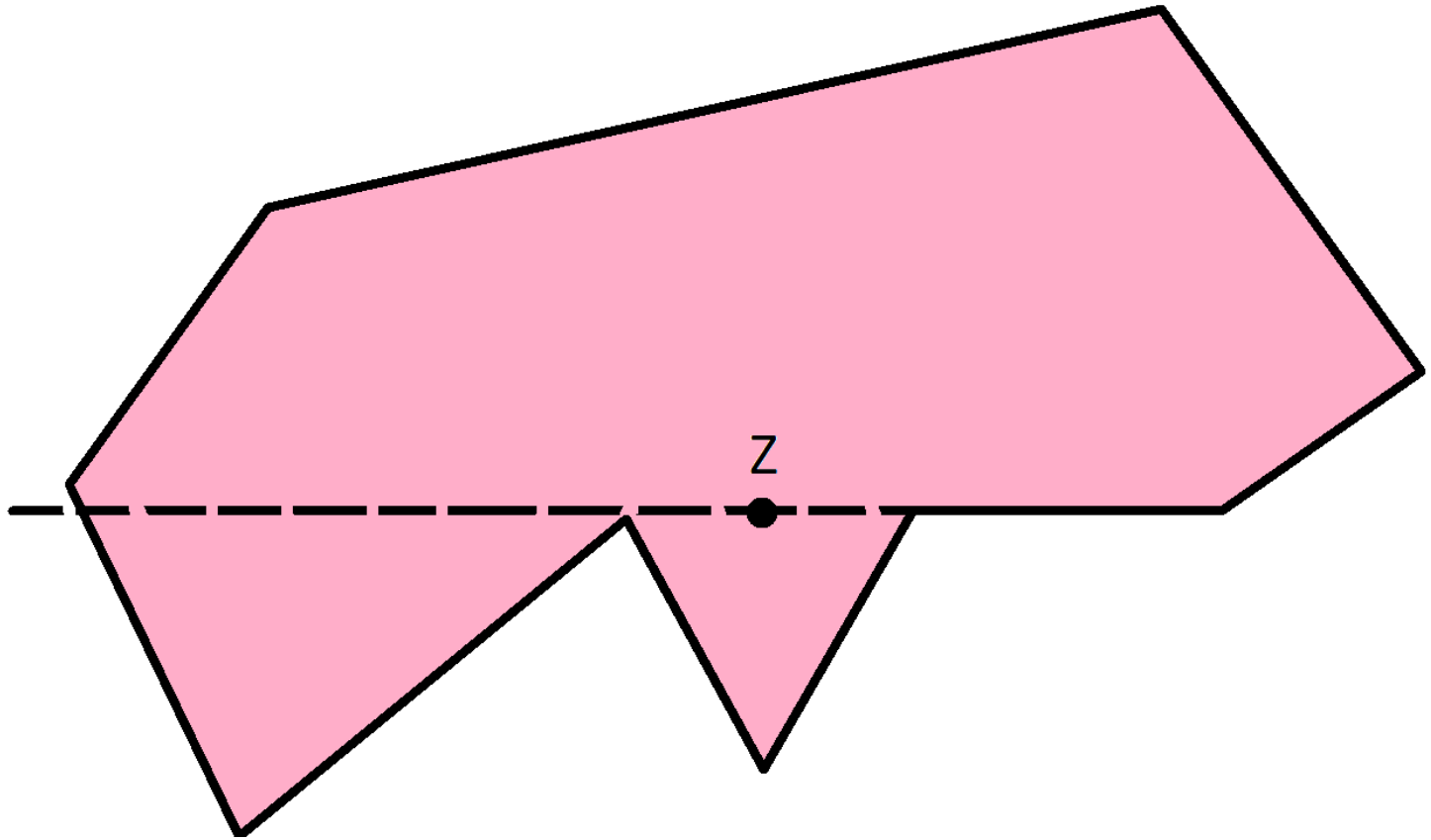




Алгоритм решения ситуации 2

Мы учитываем пересечение для того ребра, для которого точка является нижним концом и не учитываем для ребра, для которого точка является верхним концом.

Точка A будет учтена 1 раз, а точка B – 0 раз.



Ситуация 3 (горизонтальные ребра)

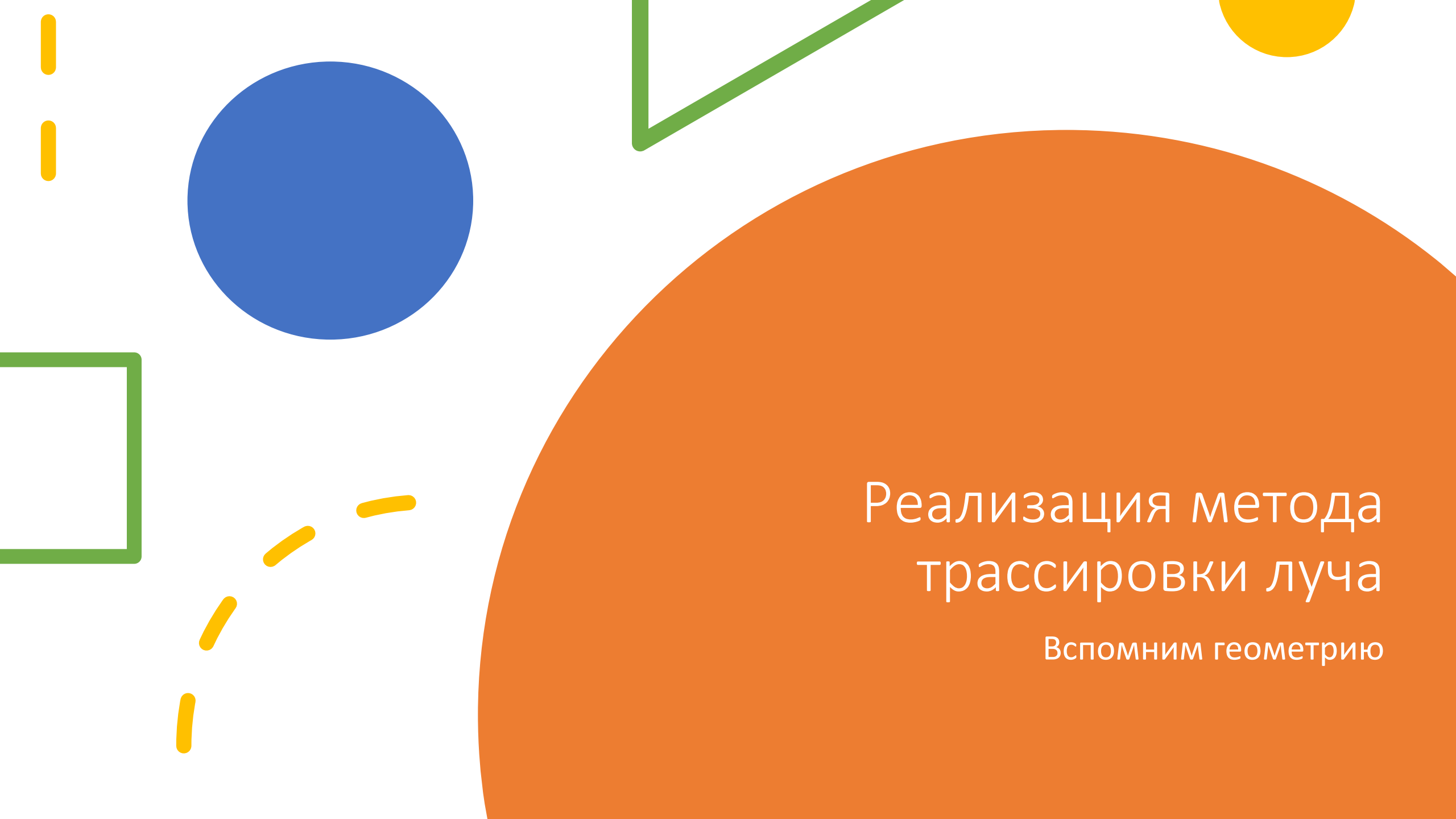
Горизонтальные ребра можно не учитывать, но необходимо проверить не оказалась ли точка z на горизонтальном ребре.

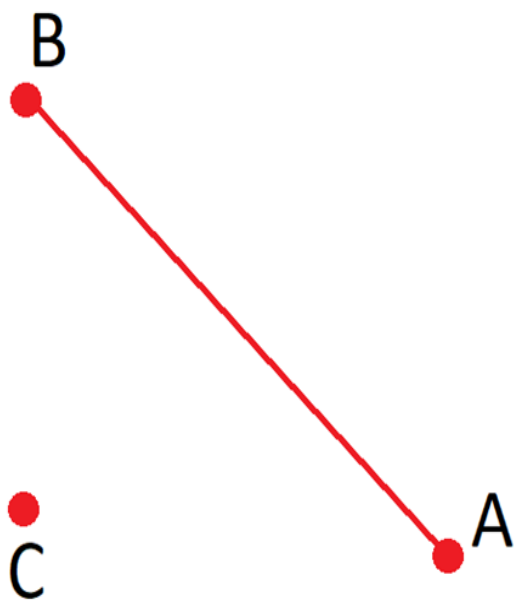
Метод трассировки луча

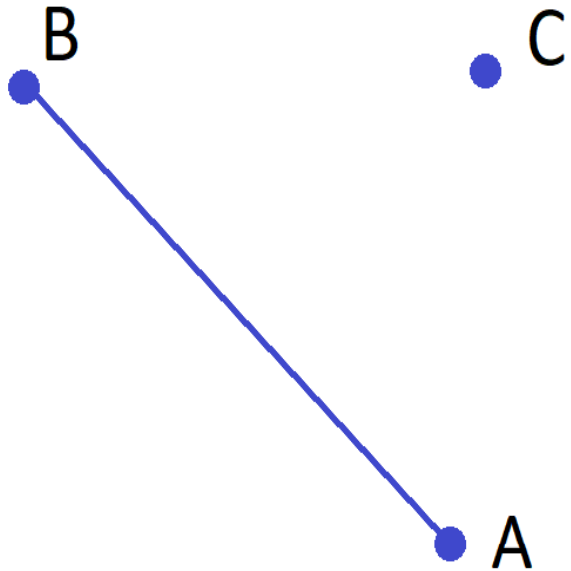
Рассмотренный алгоритм называется **методом трассировки луча.**

Сложность алгоритма составляет O(n) для n-угольника.

Вырожденные случаи, которые возникают у данного алгоритма осложняют написание кода.


Реализация метода трассировки луча
Вспомним геометрию





1. Положение точки относительно отрезка

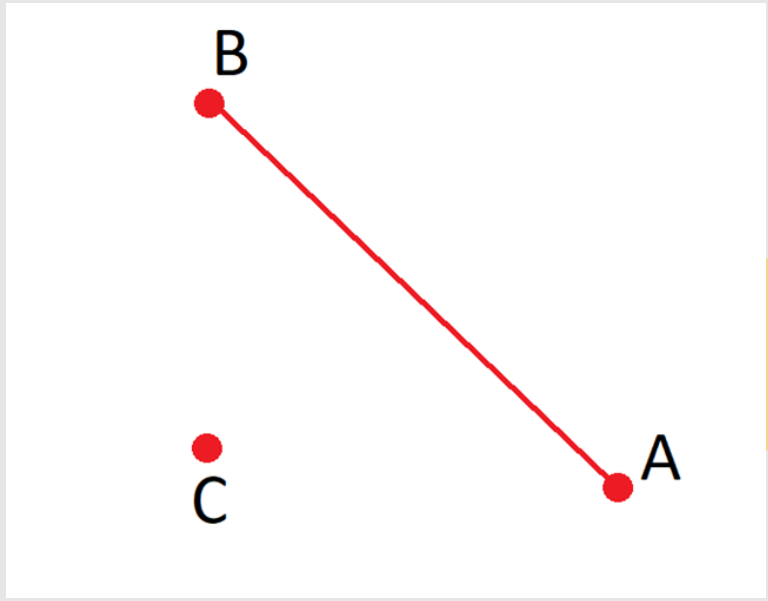
Пусть на плоскости заданы три точки:

A(a_x, a_y), B (b_x, b_y), C (c_x, c_y).

Предположим, что мы смотрим из точки A в точку B. Где при этом окажется точка C — справа или слева относительно направления нашего взгляда?

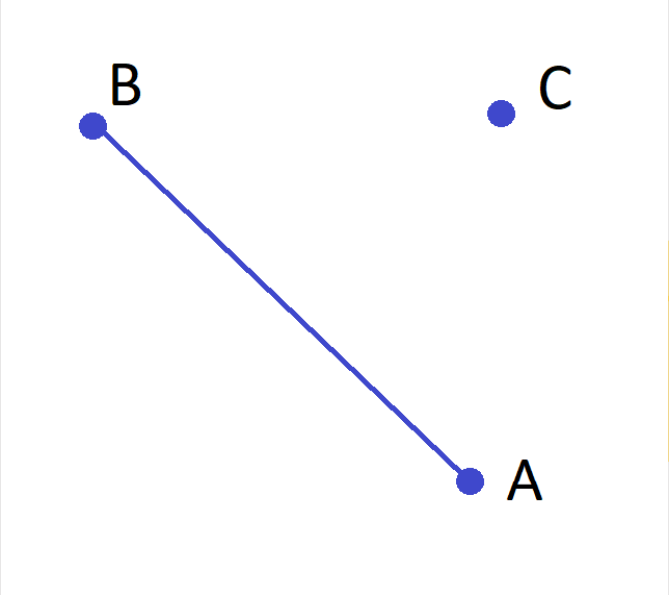
$$\Delta = \begin{vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{vmatrix}$$

Точка C лежит слева от отрезка AB (левый поворот)



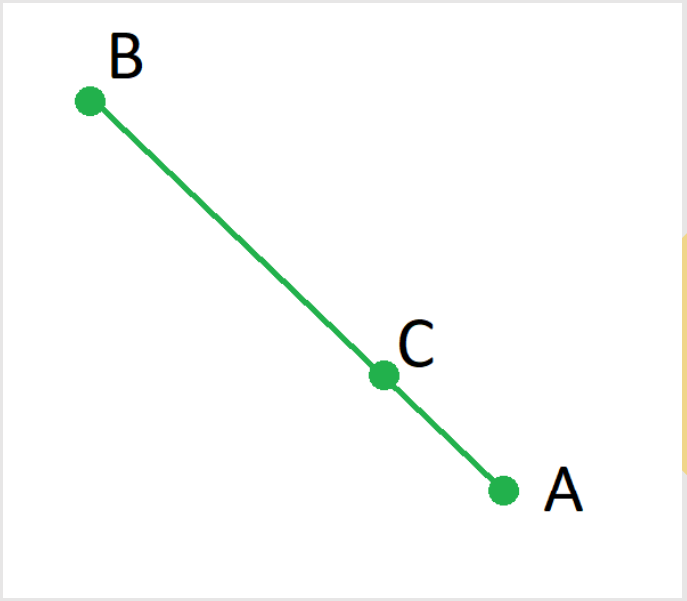
$$\Delta = \begin{vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{vmatrix} > 0$$

Точка C лежит справа от отрезка AB (правый поворот)



$$\Delta = \begin{vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{vmatrix} < 0$$

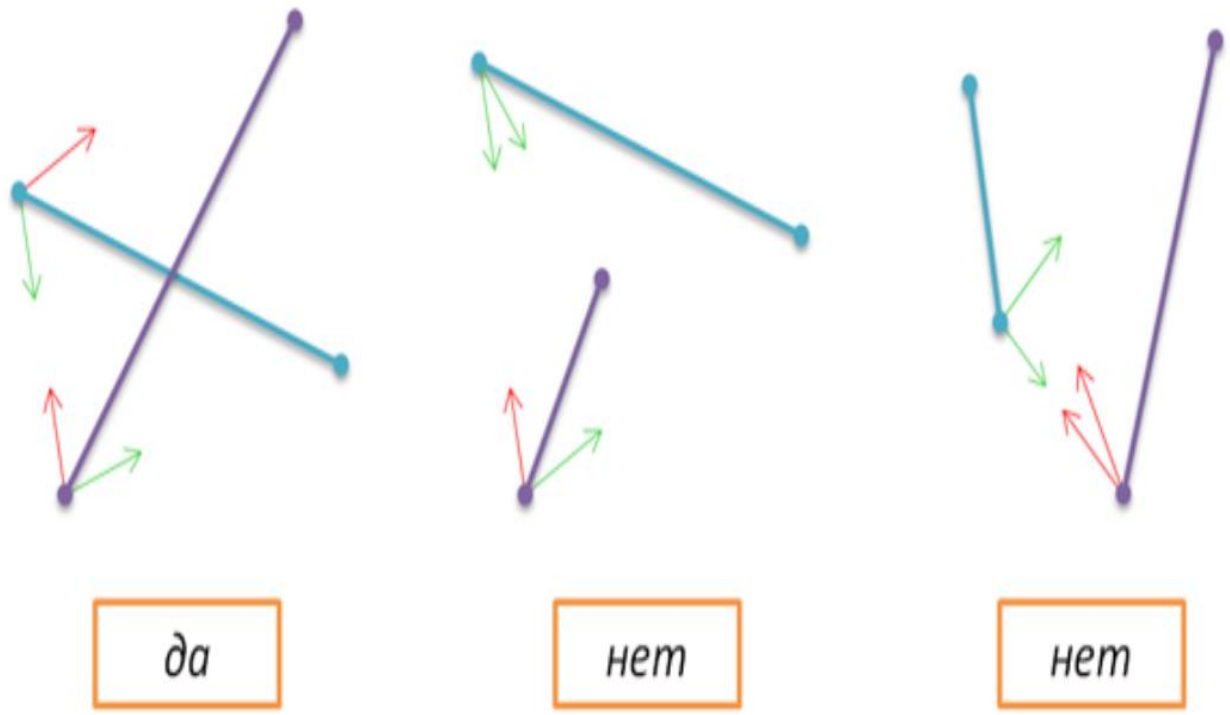
Точка C лежит на прямой AB
(как внутри, так и вне отрезка AB)



$$\Delta = \begin{vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{vmatrix} = 0$$

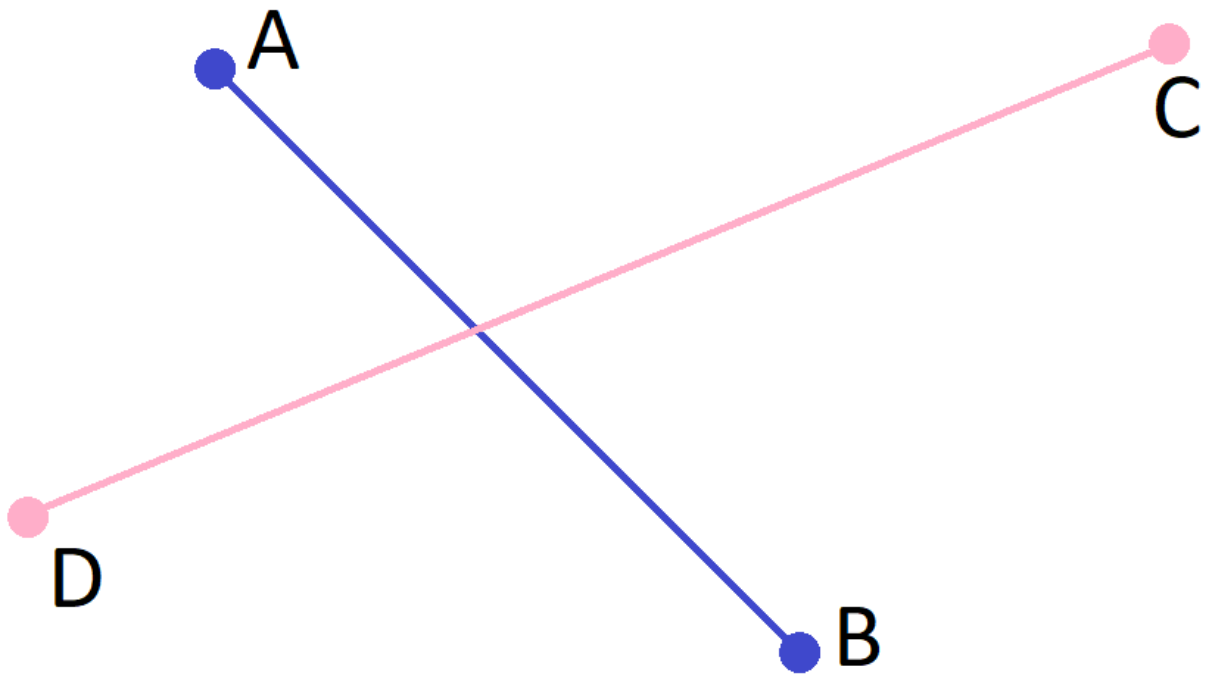
Код MatLab

```
function Delta = Rotate( A,B,C )
  M=[A(1) A(2) 1;B(1) B(2) 1;C(1) C(2) 1];
  Delta=det(M);
end
```



2. Пересечение двух отрезков

Два отрезка пересекаются тогда и только тогда, когда концы одного отрезка лежат по разные стороны другого и наоборот.



Когда отрезки пересекаются

Точки C и D должны лежать по разные стороны относительно отрезка AB

Направления поворотов rotate(A,B,C) и rotate(A,B,D).

Если знаки этих выражений различны, то прямая AB пересекает отрезок CD (причем во внутренней точке).

Знаки двух чисел различны, в том и только в том случае, когда их произведение отрицательно.

Критерий пересечения отрезков AB и CD

Критерием пересечения отрезков AB и CD является отрицательность двух выражений

Rotate(A,B,C)* Rotate(A,B,D)<0
и

Rotate(C,D,A)* Rotate(C,D,B)<0

Если заменить строгую отрицательность на неположительность, то сможем отлавливать пересечение и в концевых точках отрезков. Поэтому одно произведение будем брать

≤ 0, второе строго <0.

Код MatLab

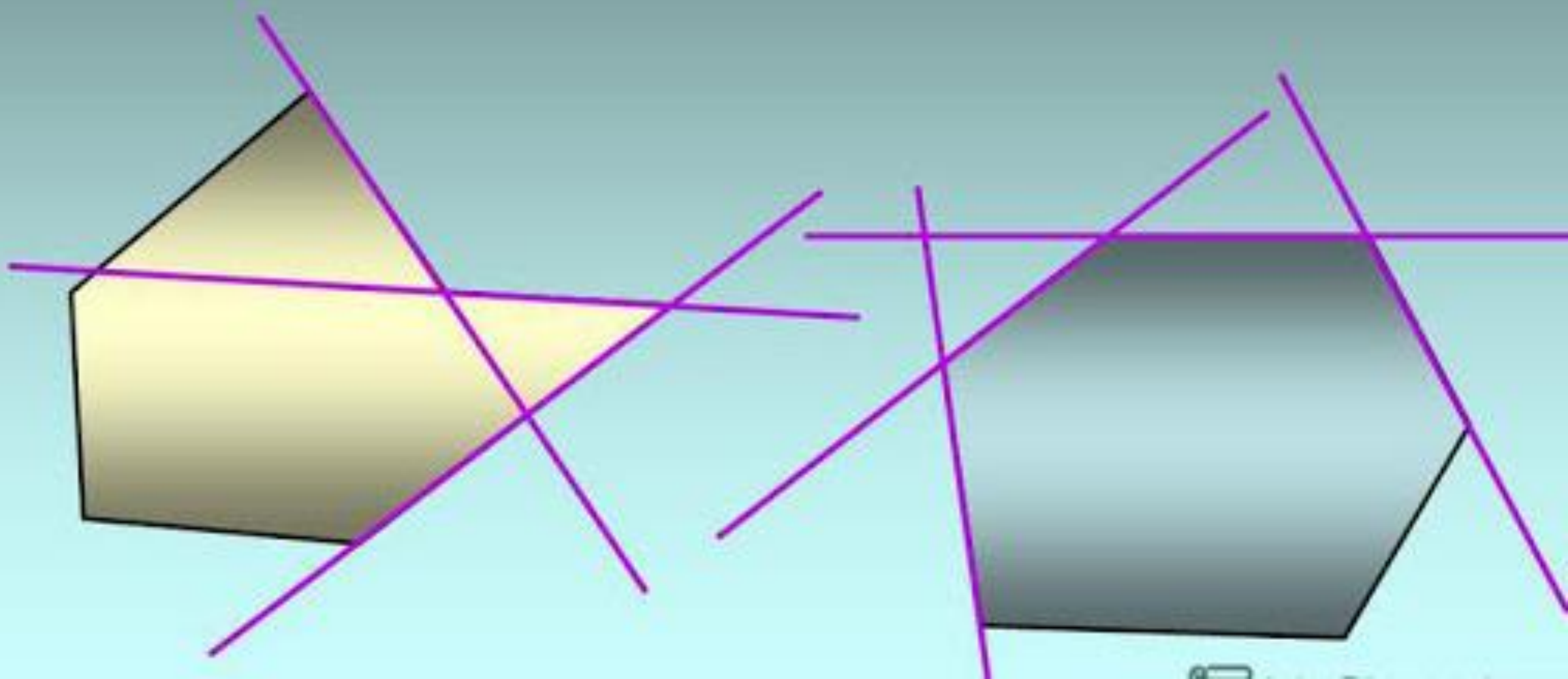
```
function val = intersect( A,B,C,D)
      val=((rotate(A,B,C)*
      rotate(A,B,D)<=0)&
     (rotate(C,D,A)*
      rotate(C,D,B)<0));
end
```

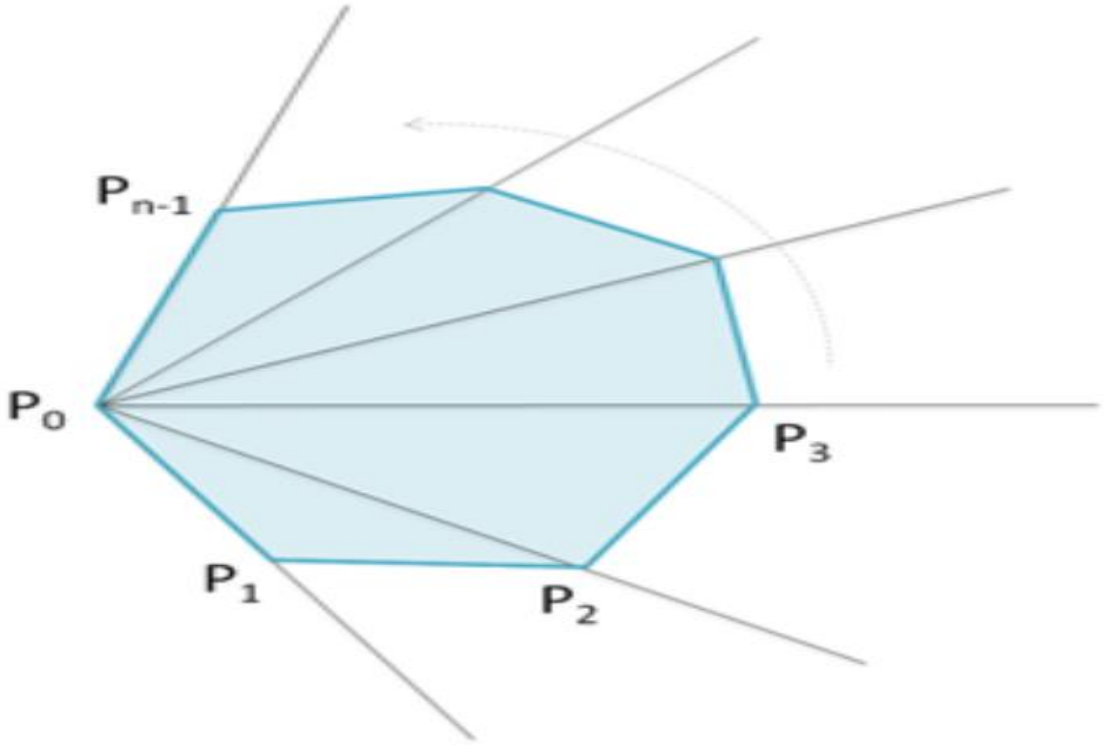
Метод трассировки луча

Вырожденные случаи, которые возникают у данного алгоритма осложняют написание кода.

Задача 2.
О принадлежности точки выпуклому многоугольнику

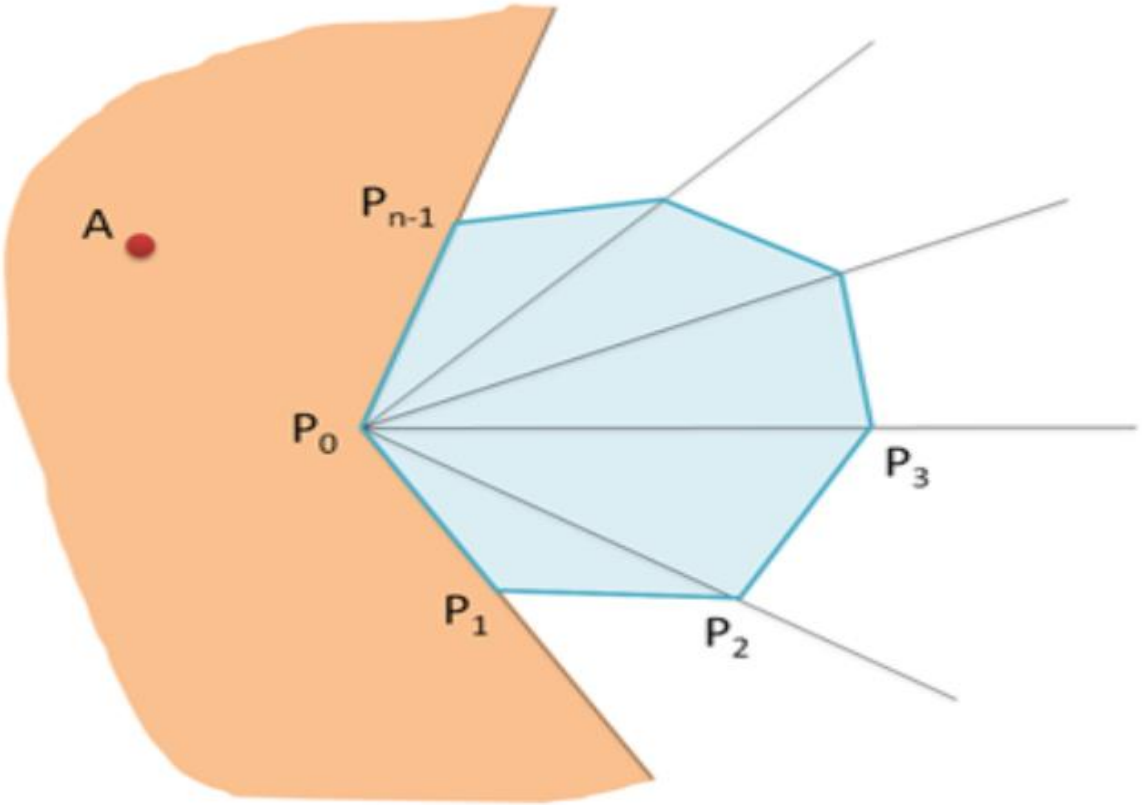
Многоугольник называется **выпуклым**, если он лежит по одну сторону от каждой прямой, проходящей через две его соседние вершины.





Постановка задачи

Заданы выпуклый многоугольник P, состоящий из n вершин и точка A. При этом предполагается, что вершины в P пронумерованы против часовой стрелки (говорят, что направление обхода по периметру P положительно).



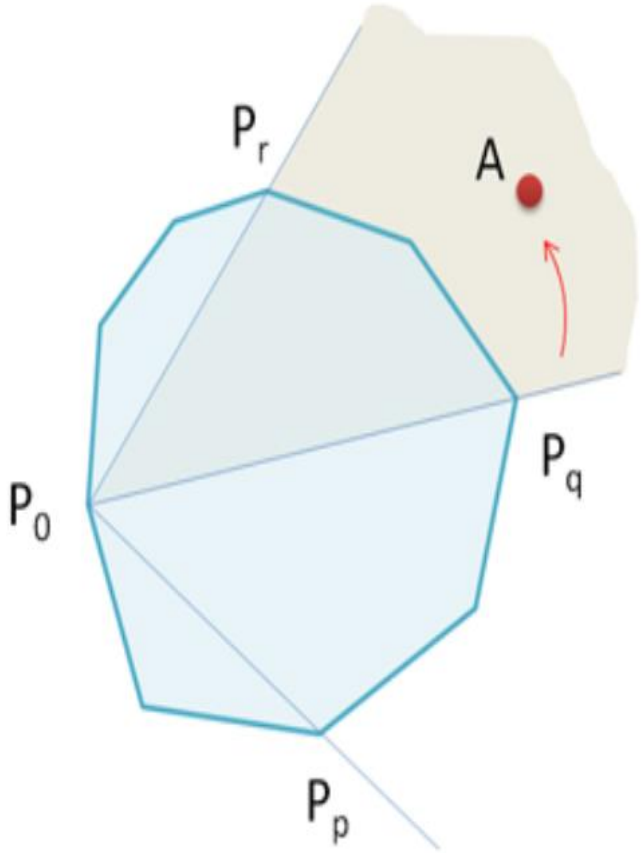
Идея алгоритма

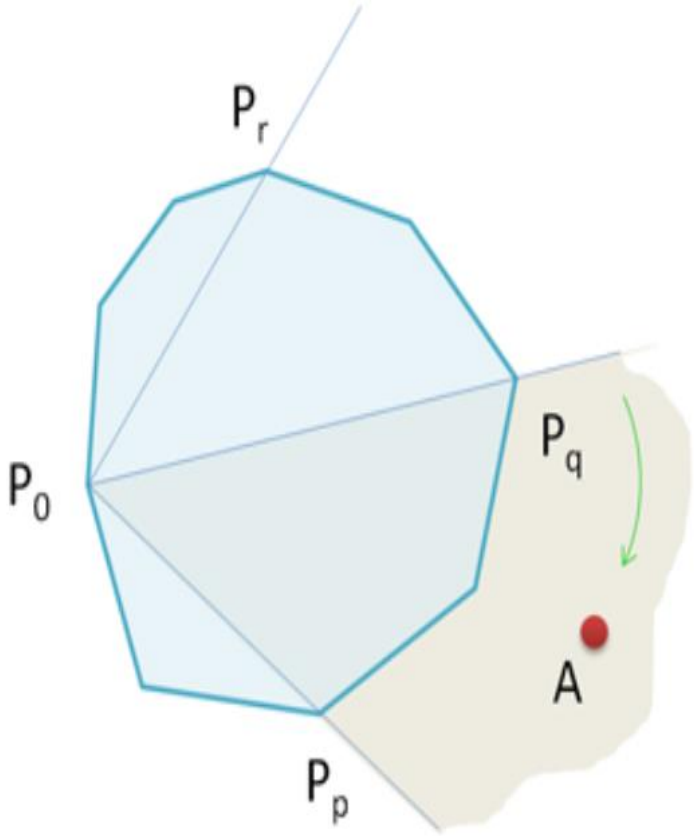
Возьмем первую вершину многоугольника P_0 и попытаемся определить, в какой сегмент $P_iP_0P_{i+1}$ попадает точка A.

Для начала проверим, что A попадает в сегмент $P_{n-1}P_0P_1$, если это не так, то A гарантированно лежит вне многоугольника.

Код MatLab

```
function  Val  = PointLoc(P,A)
   n=length(P);
   Val=true;
   if (MyRotate(P(1,:),P(2,:),A)<0)||
     (MyRotate(P(1,:),P(n,:),A)>0)
       Val=false
   end
```



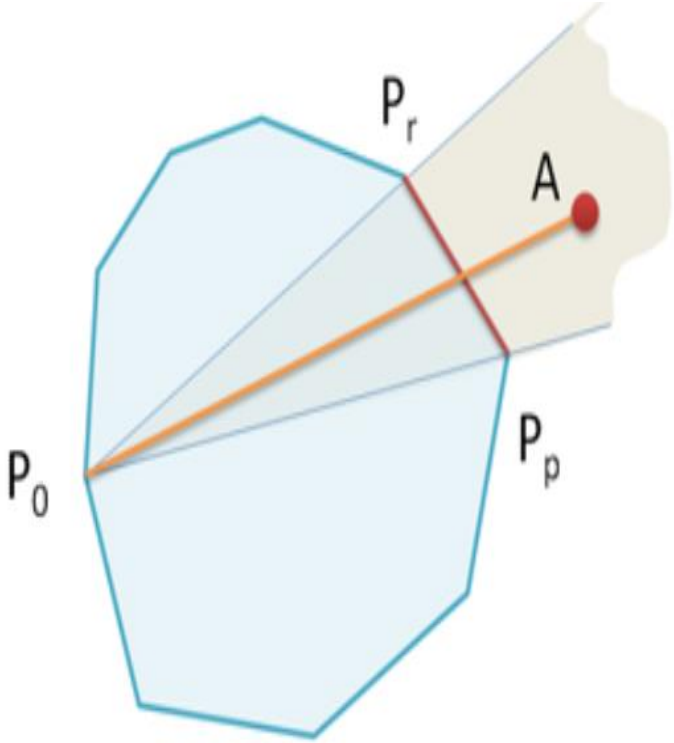


Двоичный поиск

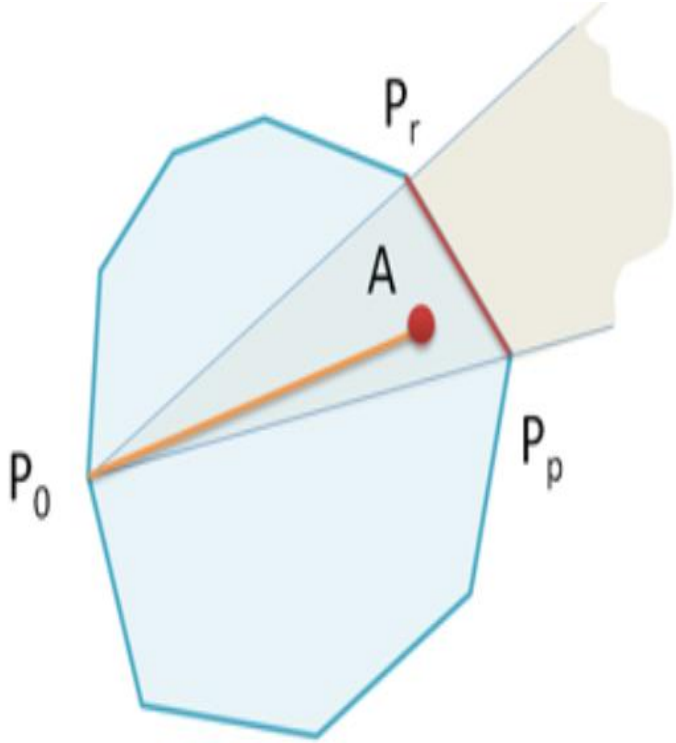
- Полагаем p=1, r=n-1 (границы текущего сегмента);
- Вычисляем среднюю вершину q=(p+r)/2;
- Смотрим, где находится A относительно вектора P_0P_q, если слева, то заменяем r на q, если справа, то заменяем p на q;
- Продолжаем этот процесс, пока не окажется, что r-p=1;

Код MatLab

```
p=1;
r=n-1;
while (r-p)>1
    q = (p+r)/2;
    if rotate(P(1,:),P(q,:),A)<0
        r=q;
    else
       p=q
    end;
end
```



пересекаются → вне



не пересекаются → внутри

Последний шаг алгоритма

Теперь осталось проверить, пересекаются ли отрезки P_0A и P_pP_r?

Если пересекаются, то точка A лежит вне, если не пересекаются, то внутри.

Код MatLab

```
if intersect(P(1,:),A,P(p,:),P(r,:))==0
    disp('лежит')
else
    disp('не лежит')
end;
```

Теорема 1

Проверка принадлежности точки выпуклому n-угольнику может быть выполнена за время $O(\log_2 n)$ при затратах памяти $O(n)$.