

Санкт-Петербург 2021

Оглавление

Оглавление	2
Постановка задачи	3
1. Алгоритм.....	4
1.1 Сравнение алгоритмов	4
1.2 Описание алгоритма	4
1.3 Реализация алгоритма	7
2. Инструкция пользователя.....	8
3. Тестовые примеры	9
Заключение.....	11
Список источников.....	12

Постановка задачи

Задачей данной курсовой работы является разработка программы, которая проверяет, принадлежит ли заданная координатами точка многоугольнику. Описание задачи можно найти в книге [1] *Препарата Ф., Шеймос М.* Раздел 2.2: Задачи локализации точек. Вычислительная геометрия: введение. Москва: Мир, 1989.

Обычно предполагается, что многоугольник без самопересечений и его углы меньше 180 градусов.

На вход программы подается координата точки и координаты точек, из которых состоит многоугольник. Требуется определить, находится ли точка внутри многоугольника. Например, для многоугольника, приведенного на рис.1, ответом является то, что точка лежит вне многоугольника.

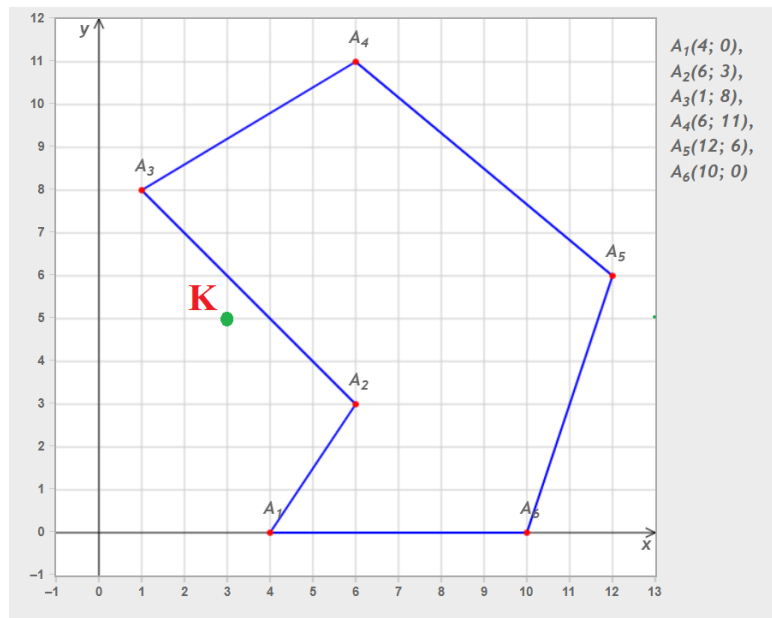


Рисунок 1 - Пример исходного многоугольника

1. Алгоритм

1.1 Сравнение алгоритмов

Самыми популярными алгоритмами для решения данной задачи являются методы суммирования углов, трассировки луча и двоичного поиска.

- Метод суммирования работает за $O(n)$, где n - число вершин многоугольника, но данный метод весьма непрактичен, так как требует вычисления дорогостоящих операций для каждого ребра (обратных тригонометрических функций, квадратных корней), и был даже назван «худшим в мире алгоритмом» для данной задачи.
- Метод трассировки луча работает за $O(n)$, где n – число рёбер многоугольника, и он прост в своей реализации. Так же данный метод наиболее популярен в мире графики и игр.
- Метод двоичного поиска является самым быстрым из всех, он работает за $O(\log n)$, где n – число вершин многоугольника, однако он является сложным с точки зрения реализации.

Поэтому был выбран самый оптимальный вариант для поставленной задачи: метод трассировки луча.

1.2 Описание алгоритма

Структуры данных:

Для создания точки используется класс `Point`, содержащий в себе две переменные типа `double` `x` и `y`. Класс `Edge` хранит в себе координаты двух точек, концов отрезка. Класс `Polygon` хранит в себе вектор точек, из которых состоит многоугольник.

Шаги для реализации метода трассировки луча следующие:

1. Если точка лежит на вершине или на ребре многоугольника, об этом выводится информация.
2. Если первый пункт не выполнен, то на каждой итерации цикла с помощью формулы пресечения двух отрезков проверяется, пересекает ли луч, проходящий через точку в фиксированном положении параллельно оси `OX`, каждое ребро многоугольника.
3. Если луч проходит через вершину многоугольника, то учитывается прохождение только через нижнюю точку отрезка.
4. Если пересечение с ребром существует, то увеличивается счетчик на единицу.
5. Если счетчик – нечетное число, то точка лежит внутри многоугольника, если четное, то вне его границ.

Более подробное описание алгоритма можно найти в книгах:

[1] *Препарата Ф., Шеймос М.* Раздел 2.2: Задачи локализации точек. Вычислительная геометрия: введение. Москва: Мир, 1989.,

[2] *Hormann K., Agathos A.* The point in polygon problem for arbitrary polygons. Comput. Geom. Theory Appl., 2001,

а также на сайтах:

[3] *Point in Polygon Strategies* <https://erich.realtimerendering.com/ptinpoly/>, 2001 и

[4] *Принадлежность точки выпуклому и невыпуклому многоугольникам*, https://neerc.ifmo.ru/wiki/index.php?title=Принадлежность_точки_выпуклому_и_невыпуклому_многоугольникам, 2015.

Пошаговое выполнение алгоритма:

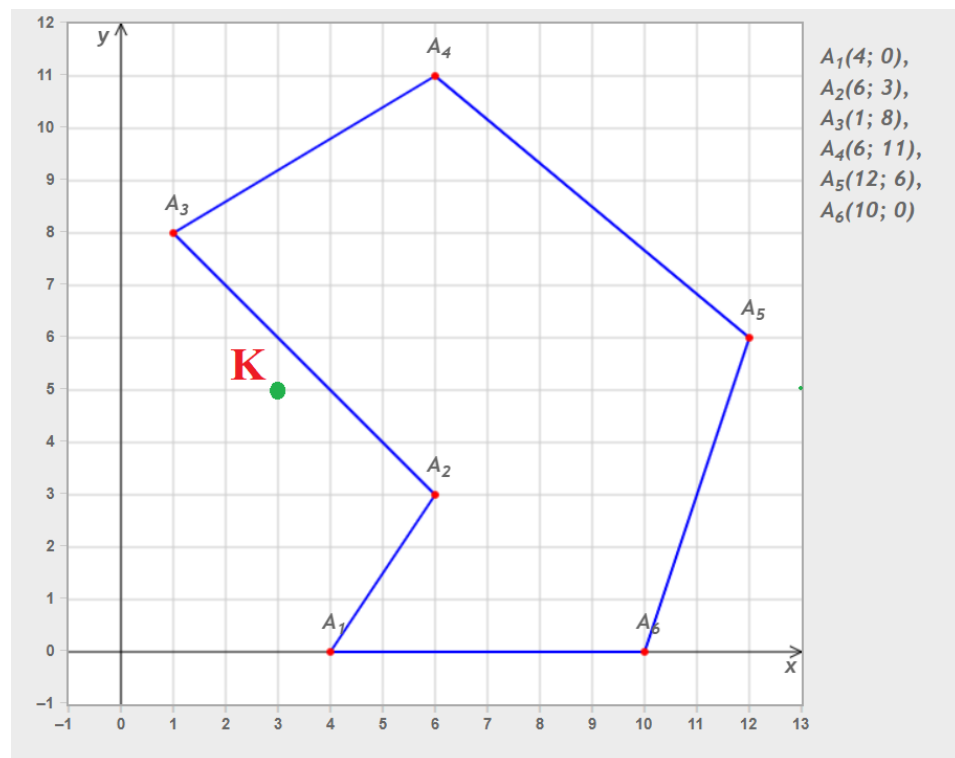


Рисунок 2 – исходное состояние точки.

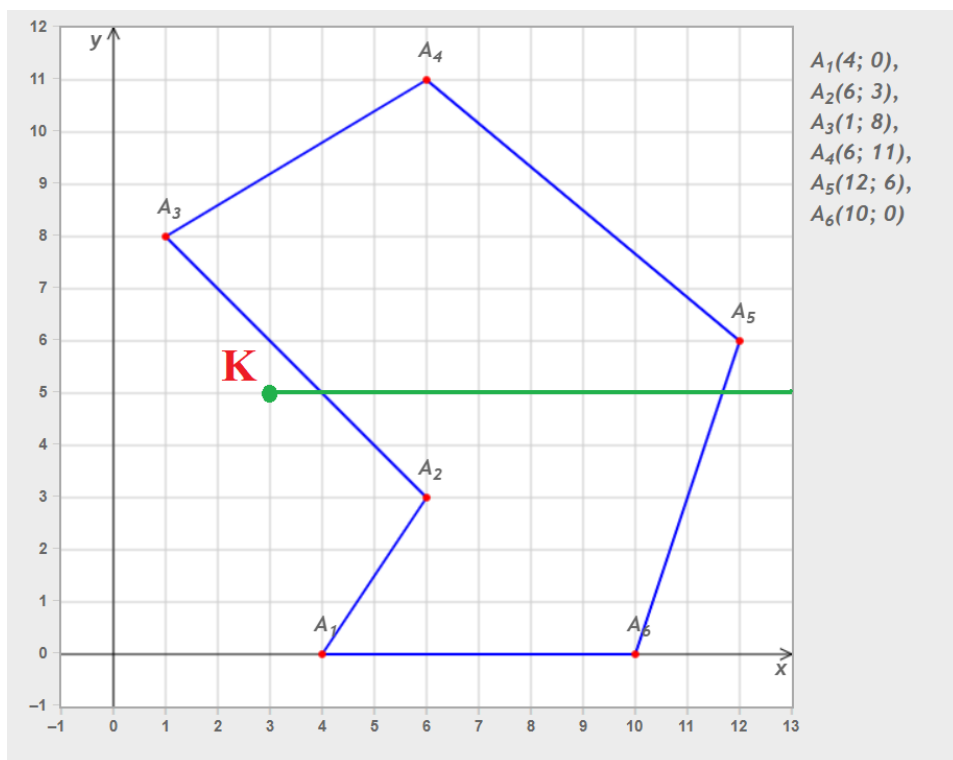


Рисунок 3 – Пускаем луч вправо.

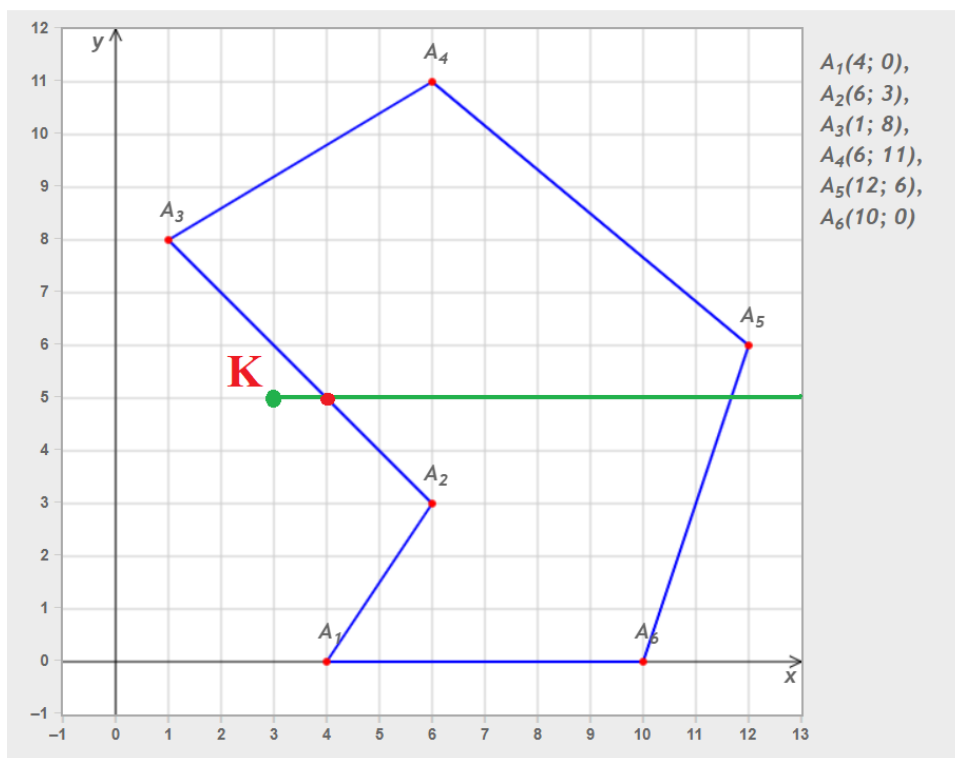


Рисунок 4 – Находим первое пересечение с ребром, увеличиваем счетчик на единицу.

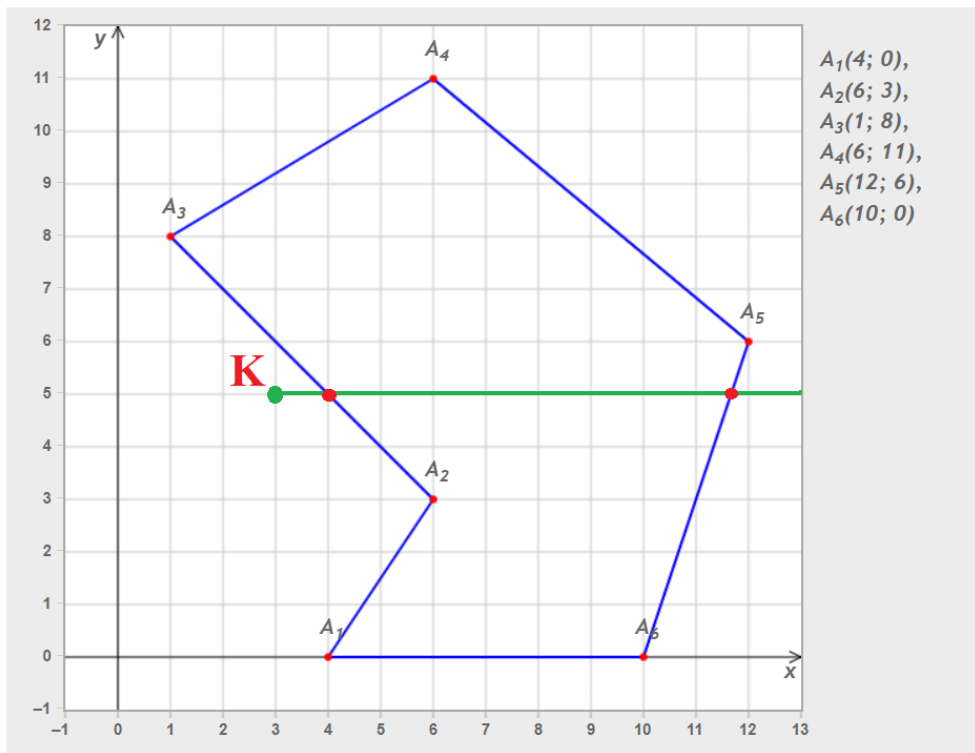


Рисунок 5 – Находим второе пересечение с ребром, увеличиваем счетчик на единицу еще раз. Пересечений четное количество, значит точка вне многоугольника.

1.3 Реализация алгоритма

```

CheckpointResponse Polygon::CheckPoint(Point const& p)
{
    Edge line (p, p1); // строим луч из точки
    int count = 0;
    for (size_t i = 0; i < Edges.size(); i++)
        if (есть пересечение с ребром)
            count ++;
    return (Если счетчик делится на 2, то OUT, иначе IN) ;
}

```

- enum class CheckPointResponse // класс, содержащий символьные константы: VERTEX, EDGE, OUT, IN.
- Edges // вектор ребер.
- int count // счетчик пересечений луча с ребрами

Алгоритм проверяет, есть ли пересечение луча с каждым из ребер многоугольника, следовательно, его сложность $O(n)$, где n – число ребер многоугольника.

2. Инструкция пользователя

Пользователю необходимо создать текстовый документ с расширением .txt, в котором он укажет координаты точки и с новой строки укажет координаты вершин многоугольника.

Запустив программу, в консоли нужно указать путь файла с исходными данными и путь, по которому создастся <выходной_файл>.txt, в котором будет сообщение следующего вида: Point [X, Y] is IN/ is OUT/ on the Edge/ on the Vertex в зависимости от результата выполнения.

Вот, как это будет выглядеть в консоли:

```
//D:\Inf\Coursework\Debug>Coursework.exe ..\PointAndPolygon.txt ..\Result.txt
```

Первый параметр: *Coursework.exe*. Файл курсовой работы с расширением .exe

Второй параметр: *PointAndPolygon.txt*. Текстовый файл с расширением .txt, в котором записаны координаты точки и многоугольника

Третий параметр: *Result.txt*. Текстовый файл с расширением .txt, в который выводится результат выполнения программы.

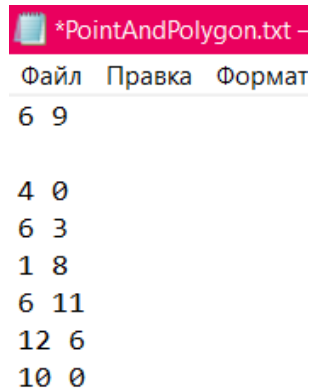
3. Тестовые примеры

Пример 1

Консоль:

```
//D:\Inf\Coursework\Debug>Coursework.exe ..\PointAndPolygon.txt ..\Result.txt
```

Входной файл:

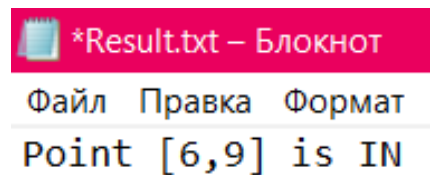


```
*PointAndPolygon.txt –
Файл  Правка  Формат
6 9

4 0
6 3
1 8
6 11
12 6
10 0
```

Рисунок 6 – входные данные

Файл на выходе:



```
*Result.txt – Блокнот
Файл  Правка  Формат  |
Point [6,9] is IN
```

Рисунок 7 – выходные данные

Пример 2

Консоль:

```
//D:\Inf\Coursework\Debug>Coursework.exe ..\PointAndPolygon.txt ..\Result.txt
```

Входной файл:

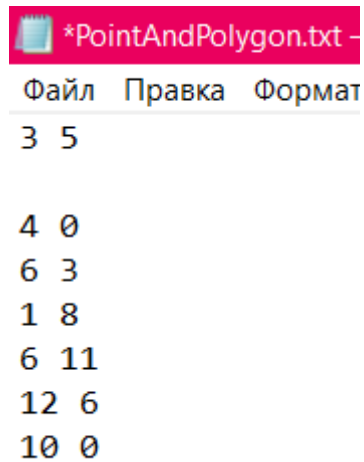


Рисунок 8 – входные данные

Файл на выходе:

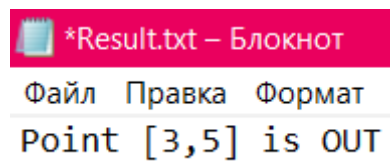


Рисунок 9 – выходные данные

Пример 3

Консоль:

```
//D:\Inf\Coursework\Debug>Coursework.exe ..\PointAndPolygon.txt
```

Выведется сообщение об ошибке в консоль:

Filename is not set

Заключение

Результатом курсовой работы является реализованный алгоритм трассировки луча. Данный метод предназначен для эффективной проверки наличия точки внутри многоугольника. На практике он часто используется при разработке игр, где используется трассировка, так как довольно прост в своей реализации и имеет сложность $O(n)$, где n – число рёбер многоугольника.

Разработанная программа реализует один из методов поставленной задачи.

Список источников

Литература:

- [1] Препарата Ф., Шеймос М. Раздел 2.2: Задачи локализации точек. Вычислительная геометрия: введение. Москва: Мир, 1989.
- [2] Hormann K., Agathos A. The point in polygon problem for arbitrary polygons. Comput. Geom. Theory Appl., 2001.

Сайты:

- [3] *Point in Polygon Strategies* <https://erich.realtimerendering.com/ptinpoly/>, 2001.
- [4] *Принадлежность точки выпуклому и невыпуклому многоугольникам*, https://neerc.ifmo.ru/wiki/index.php?title=Принадлежность_точки_выпуклому_и_невыпуклому_многоугольникам, 2015.