|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования 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\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_Программное обеспечение ЭВМ и информационные технологии\_\_\_\_\_\_\_\_

**ОТЧЕТ ПО ПРОЕКТНО-ТЕХНОЛОГИЧЕСКОЙ ПРАКТИКЕ**

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_Татаринова Дарья Алексеевна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*фамилия, имя, отчество*

Группа\_\_\_\_\_\_ИУ7-14Б\_\_\_\_\_\_\_\_\_

Тип практики \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Учебная\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Название предприятия\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_Татаринова Д.А.\_\_\_

*подпись, дата фамилия, и.о.*

Руководитель практики \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_Кузнецова О.В.\_\_\_

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2020 г.*

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 2](#_Toc57650343)

[УСЛОВИЕ ЗАДАЧИ 3](#_Toc57650344)

[Оригинальный текст задачи 3](#_Toc57650345)

[Переведенный текст задачи 4](#_Toc57650346)

[СХЕМА АЛГОРИТМА 5](#_Toc57650347)

[Функция isInside 5](#_Toc57650348)

[Функция linesIntersection 6](#_Toc57650349)

[Функция belongToPiece 7](#_Toc57650350)

[Основная программа 8](#_Toc57650351)

[ОПИСАНИЕ ПРОГРАММЫ 10](#_Toc57650352)

[ТЕКСТ ПРОГРАММЫ 11](#_Toc57650353)

[ЗАКЛЮЧЕНИЕ 13](#_Toc57650354)

[СПИСОК ЛИТЕРАТУРЫ 14](#_Toc57650355)

# ВВЕДЕНИЕ

Цель практики – расширение, углубление и обобщение знаний в области написания кода, овладение методами оформления документации программы.

Задачи – сформировать алгоритм решения задачи, написать код, протестировать программу, составить схему алгоритма.

# УСЛОВИЕ ЗАДАЧИ

## Оригинальный текст задачи

Intersection

You are to write a program that has to decide whether a given line segment intersects a given rectangle.

The line is said to intersect the rectangle if the line and the rectangle have at least one point in common. The rectangle consists of four straight lines and the area in between. Although all input values are integer numbers^ valid intersection points do not have to lay on the integer grid.

INPUT

The input consists of n test cases. The first line of the input file contains the number n. Each following line contains one test case of the format:

xstart ystart xend yend xleft ytop xright ybottom

where (xstart, ystart) is the start and (xend, yend) the end point of the line and (xleft, ytop) the top left and (xright, ybottom) the bottom right corner of the rectangle. The eight numbers are separated by a blank. The terms top left and bottom right do not imply any ordering of coordinates.

OUTPUT

For each test case in the input file, the output file should contain a line consisting either of the letter “T” if the line segment intersects the rectangle or the letter “F” if the line segment does not intersect the rectangle.

SAMPLE INPUT

1

4 9 11 2 1 5 7 1

SAMPLE OUTPUT

F

## Переведенный текст задачи

Пересечение

Вы должны написать программу, которая определяет пересекаются ли данная линия и данный прямоугольник.

Линия пересекается с прямоугольником, если линия и прямоугольник имеют хотя бы одну общую точку. Прямоугольник состоит из четырех линий и пространства между ними. Все вводимые величины являются целыми числами, координаты точки пересечения на сетке не обязательно должны быть целочисленными.

ВВОД

Ввод состоит из n тестов. Первая строка состоит из числа n. Каждая следующая строка содержит информацию следующего вида:

xstart ystart xend yend xleft ytop xright ybottom

где (xstart, ystart) – начальная и (xend, yend) – конечная точки линии, (xleft, ytop) – левая верхняя и (xright, ybottom) – правая нижняя точки углов прямоугольника. Восемь чисел отделены друг от друга пробелами.

ВЫВОД

Для каждой вводимой записи вывод должен состоять из буквы «Т», если линия пересекается с прямоугольником, или буквы «F», если линия не пересекается с прямоугольником.

ПРИМЕР ВВОДА

1

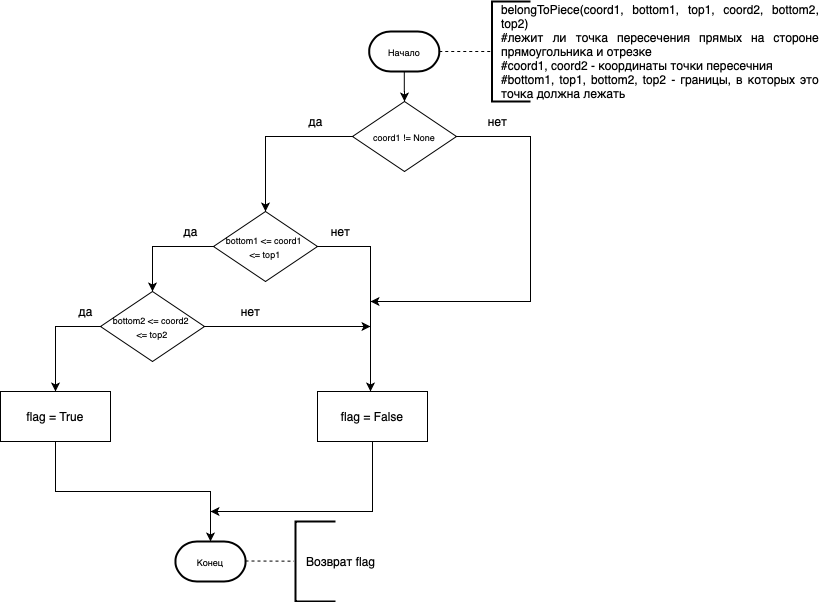
4 9 11 2 1 5 7 1

ПРИМЕР ВЫВОДА

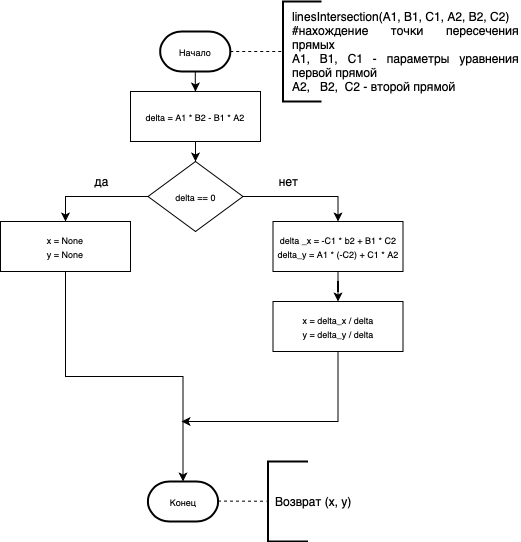
F

# СХЕМА АЛГОРИТМА

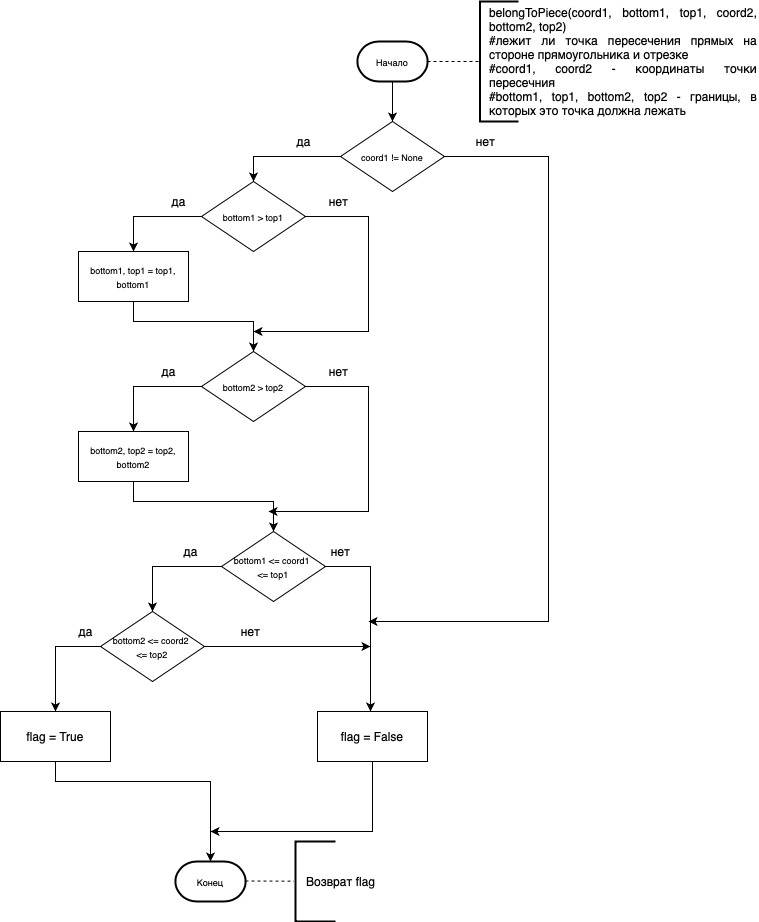
## Функция isInside



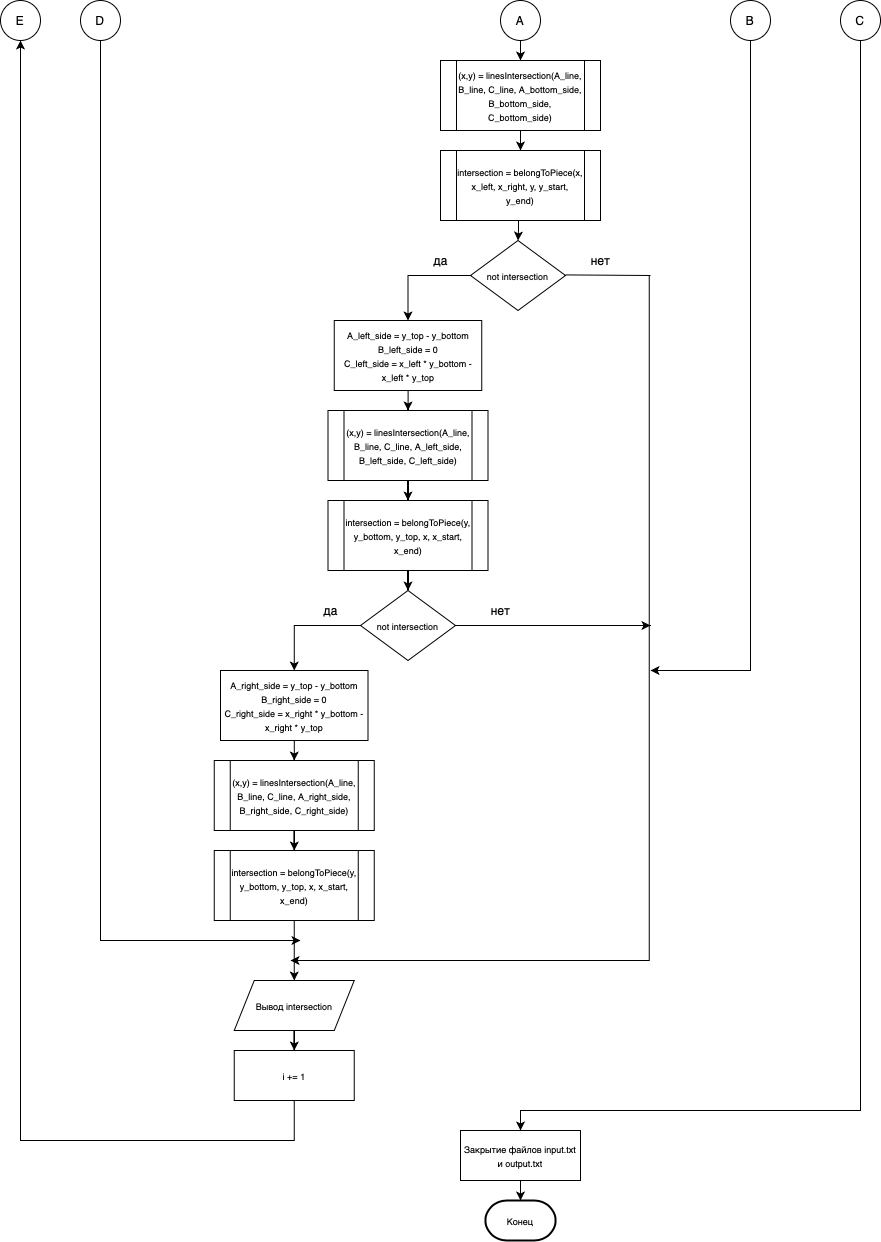
## Функция linesIntersection



## Функция belongToPiece



## Основная программа



# ОПИСАНИЕ ПРОГРАММЫ

Программа работает на N (вводится из файла) количестве тестов. Алгоритм проверки пересечения отрезка с прямоугольником разбивается на 2 случая.

Сначала программа проверяет, находится ли отрезок полностью внутри прямоугольника. Функция isInside принимает координаты объектов, и, если x-координаты начала и конца отрезка расположены между x-координатами левой и правой стороны прямоугольника или равны им, а y-координаты начала и конца отрезка расположены между y-координатами верхней и нижней стороны прямоугольника или равны им, то функция возвращает true.

Если функция isInside вернула false, то программа проверяет второй случай: если отрезок пересекается с одной из сторон прямоугольника, то он пересекается и с самим прямоугольником. Сначала составляется уравнение прямой, содержащей отрезок, затем поочередно уравнения прямых, содержащих стороны прямоугольника. Функция linesIntersection возвращает координаты точки пересечения прямой, содержащей отрезок, с прямой, содержащей сторону, а функция belongToPiece проверяет, принадлежит ли эта точка отрезку и стороне прямоугольника. Если принадлежит, то функция возвращает true, в противном случае – false, тест заканчивается, и начинается следующий.

# ТЕКСТ ПРОГРАММЫ

# Лежит ли отрезок полностью внутри прямоугольника

def isInside(x\_start, x\_end, x\_left, x\_right, y\_start, y\_end, y\_bottom, y\_top):

if x\_left <= x\_start <= x\_right and x\_left <= x\_end <= x\_right and y\_bottom <= y\_start <= y\_top and y\_bottom <= y\_end <= y\_top:

return True

return False

# Найти точку пересечения прямых

def linesIntersection(A1, B1, C1, A2, B2, C2):

delta = A1 \* B2 - B1 \* A2

if delta == 0:

return (None, None)

else:

delta\_x = -C1 \* B2 - B1 \* (-C2)

delta\_y = A1 \* (-C2) - (-C1) \* A2

x = delta\_x / delta

y = delta\_y / delta

return (x, y)

# Лежит ли точка пересечения прямых на стороне прямоугольника и отрезке

def belongToPiece(coord1, bottom1, top1, coord2, bottom2, top2):

if coord1 != None:

if bottom1 > top1:

bottom1, top1 = top1, bottom1

if bottom2 > top2:

bottom2, top2 = top2, bottom2

if bottom1 <= coord1 <= top1 and bottom2 <= coord2 <= top2:

return True

return False

# Основная программа

input\_file = open('input.txt')

output\_file = open('output.txt', 'w')

N = int(input\_file.readline())

for i in range(N):

x\_start, y\_start, x\_end, y\_end, x\_left, y\_top, x\_right, y\_bottom = map(int, input\_file.readline().split())

intersection = False

# Лежит ли отрезок полностью внутри прямоугольника

if isInside(x\_start, x\_end, x\_left, x\_right, y\_start, y\_end, y\_bottom, y\_top):

intersection = True

else:

# Уравнение прямой, содержащей отрезок

A\_line = y\_start - y\_end

B\_line = x\_end - x\_start

C\_line = x\_start \* y\_end - x\_end \* y\_start

# Уравнение прямой, содержащей верхнюю сторону прямоугольника

A\_top\_side = 0

B\_top\_side = x\_right - x\_left

C\_top\_side = x\_left \* y\_top - x\_right \* y\_top

(x, y) = linesIntersection(A\_line, B\_line, C\_line, A\_top\_side, B\_top\_side, C\_top\_side)

intersection = belongToPiece(x, x\_left, x\_right, y, y\_start, y\_end)

if not intersection:

# Уравнение прямой, содержащей нижнюю сторону прямоугольника

A\_bottom\_side = 0

B\_bottom\_side = x\_right - x\_left

C\_bottom\_side = x\_left \* y\_bottom - x\_right \* y\_bottom

(x, y) = linesIntersection(A\_line, B\_line, C\_line, A\_bottom\_side, B\_bottom\_side, C\_bottom\_side)

intersection = belongToPiece(x, x\_left, x\_right, y, y\_start, y\_end)

if not intersection:

# Уравнение прямой, содержащей левую сторону прямоугольника

A\_left\_side = y\_top - y\_bottom

B\_left\_side = 0

C\_left\_side = x\_left \* y\_bottom - x\_left \* y\_top

(x, y) = linesIntersection(A\_line, B\_line, C\_line, A\_left\_side, B\_left\_side, C\_left\_side)

intersection = belongToPiece(y, y\_bottom, y\_top, x, x\_start, x\_end)

if not intersection:

# Уравнение прямой, содержащей правую сторону прямоугольника

A\_right\_side = y\_top - y\_bottom

B\_right\_side = 0

C\_right\_side = x\_right \* y\_bottom - x\_right \* y\_top

(x, y) = linesIntersection(A\_line, B\_line, C\_line, A\_right\_side, B\_right\_side, C\_right\_side)

intersection = belongToPiece(y, y\_bottom, y\_top, x, x\_start, x\_end)

if intersection:

output\_file.write('T\n')

else:

output\_file.write('F\n')

input\_file.close()

output\_file.close()

# ЗАКЛЮЧЕНИЕ

В результате практики была написана и протестирована программа, определяющая пересекается ли отрезок с прямоугольником, построена схема алгоритма. Были получены знания в области написания понятного и читаемого кода, оформления документации программы.

# СПИСОК ЛИТЕРАТУРЫ

1. Бейдер Д. Чистый Python. Тонкости программирования для профи / Д. Бейдер . – СПб.: Питер, 2018. – 288 с.
2. ГОСТ 19.701-90 (ИСО 5807-85) Единая система программной документации (ЕСПД). Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения. – М.: Стандартинформ, 2007. - 5 с.
3. Документация Python [Электронный ресурс] – Режим доступа: https://www.python.org/doc/ (04.12.2020)