РЕШЕНИЕ ЗАДАНИЙ
второго этапа
Международной олимпиады
Innopolis Open
по профилю Робототехника

2023/2024 учебный год

19.02.2024

Оглавление

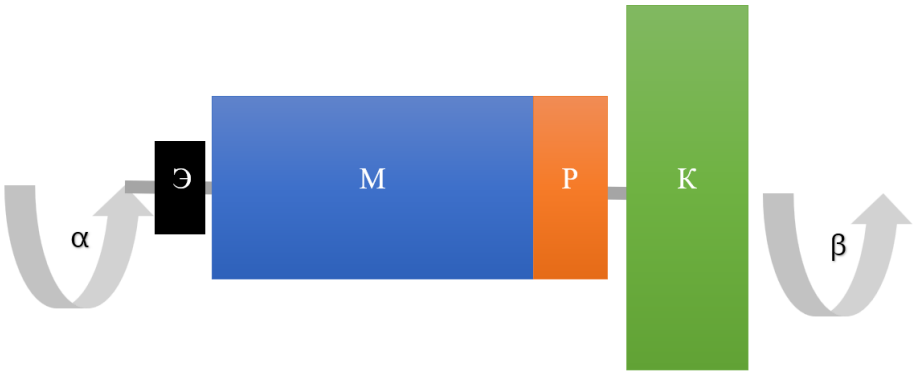
Возрастная категория 7–8 классы 3
Задача A. Энкодер из пути. 3
Задача B. Путь по энкодеру 5
Задача C. Движение по дуге. Скорости 7
Задача D. Движение по дуге. Градусы 9
Задача E. Ускорение робота 12
Задача F. Движение по энкодерам ТРИК 16
Задача G. Движение по линии ТРИК 20
Возрастная категория 9–11 классы 22
Задача A. Вычисление скоростей mecanum-платформы 22
Задача B. LIDAR 25
Задача C. Одометрия материальной точки 27
Задача D. Одометрия дифф. платформы 29
Задача E. Движение по линии ТРИК 33
Задача F. Граф 35
Задача G. Одометрия дифф. платформы ТРИК 37
Задача H. Фигуры 41

Возрастная категория 7–8 классы

Задача A. Энкодер из пути.

Ограничения: по времени 1 сек, по памяти 256 Мб.

Описание робота: привод робота.



Э – энкодер. Основная характеристика – количество «тиков» на один оборот (n).

М – мотор. Энкодер вращается на одном валу с мотором и измеряет его угол поворота α.

Р – редуктор. Основная характеристика – передаточное число (i), показывает во сколько раз выходной вал вращается медленнее, чем входной.

К – колесо. Основная характеристика – диаметр (D). Вращается в i раз медленнее, чем мотор. Пока мотор и энкодер делают поворот на α градусов, колесо поворачивается на β градусов.

Задание:

Необходимо определить, на сколько градусов должно повернуться колесо робота диаметром D, чтобы оно проехало требуемое расстояние S.

Система оценки

В задаче три подгруппы тестов, каждая подгруппа оценивается в 10 баллов.

Формат входных данных

Входные данные состоят из двух строк.

Первая строка содержит одно целое положительное число D от 1 до 1000 – диаметр колеса в миллиметрах.

Вторая строка содержит одно целое число S от -10^6 до 10^6 – расстояние движения в миллиметрах.

Формат выходных данных

Выведите дробное число (с точностью до 4 знаков после запятой) – угол поворота колеса в градусах.

Пример 1

Входные данные:	Выходные данные:
3 123	4698.2539

Пример 2

Входные данные:	Выходные данные:
100 100001	114592.7049

Решение:

Решение сводится к вычислению угла по формуле

$$\beta = \frac{360S}{\pi D}$$

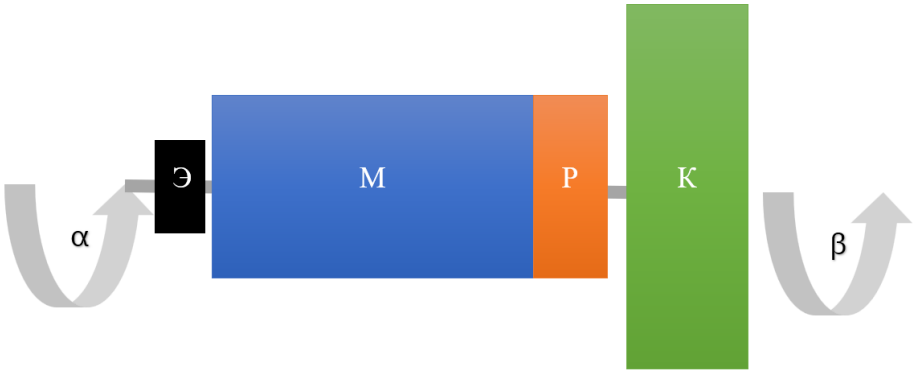
Считывание входных данных, вычисление и вывод данных могут быть реализованы следующей программой на ЯП Python:

```
from sys import stdin
import sys
import math

lines = []
for line in stdin:
    lines.append(line)

D = int(lines[0]) #в мм
S = int(lines[1]) #в мм
Betta = (360*S)/(math.pi*D)

print(Betta)
```

Задача B. Путь по энкодеру

Ограничения: по времени 1 сек, по памяти 256 Мб.

Описание робота: привод робота.



Э – энкодер. Основная характеристика – количество «тиков» на один оборот (n).

М – мотор. Энкодер вращается на одном валу с мотором и измеряет его угол поворота α.

Р – редуктор. Основная характеристика – передаточное число (i), показывает во сколько раз выходной вал вращается медленнее, чем входной.

К – колесо. Основная характеристика – диаметр (D). Вращается в i раз медленнее, чем мотор. Пока мотор и энкодер делают поворот на α градусов, колесо поворачивается на β градусов.

Задание:

Робот начал движение с нулевым показанием энкодера. Он совершил движение, известны показания энкодера на момент окончания движения (N). Известно, что во время движения колесо вращалось только в одну строну. Необходимо определить, какой путь в миллиметрах преодолело колесо (S).

Формат входных данных

Первая строка содержит целое положительное число от 1 до 1000 – диаметр колеса в миллиметрах (D).

Вторая строка содержит дробное положительное число от 0.0001 до 1000.0 (с точностью до 4 знаков после запятой) – редукция, передаточное число между мотором и колесом (i). Показывает, во сколько раз колесо вращается медленнее чем мотор.

Третья строка содержит целое положительное четное число от 2 до 1440 – количество «тиков» энкодера на один оборот мотора (n).

Четвертая строка содержит целое число от -10^9 до 10^9 – показания энкодера после окончания движения (N).

Формат выходных данных

Выведите дробное число с точностью до 4 знаков после запятой – расстояние в миллиметрах, пройденное колесом S.

Пример 1

Входные данные:

721
1.0000
360
-957078

Выходные данные:

-6021850.5086

Решение:

Усовершенствуем формулу из предыдущей задачи с учетом редуктора и показаний в «тиках» энкодера, а не в миллиметрах. Итоговая формула выглядит так:

$$S = \frac{\pi DN}{ni}$$

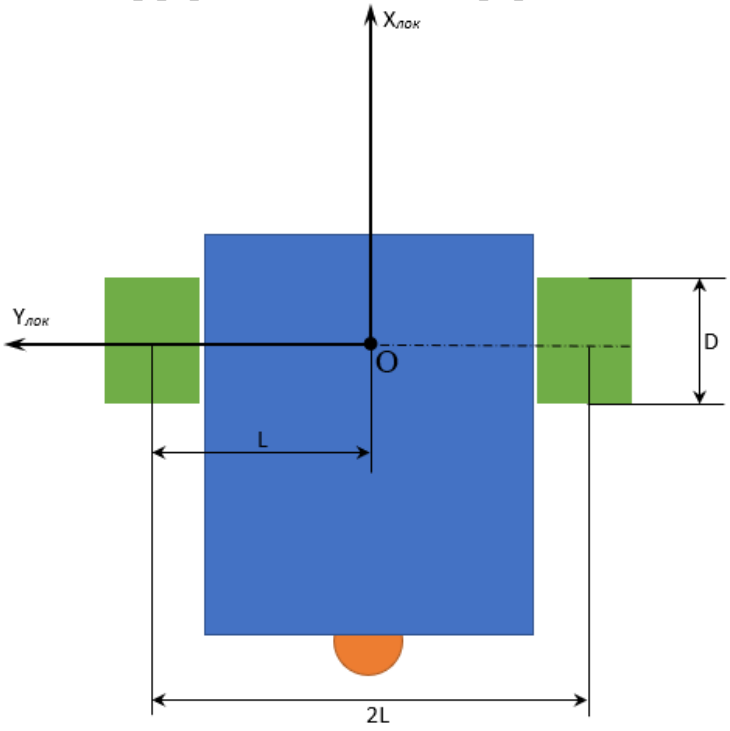
Считывание входных данных, вычисление и вывод данных могут быть реализованы следующей программой на ЯП Python:

```
from sys import stdin
import sys
import math

lines = []
for line in stdin:
    lines.append(line)

D = int(lines[0])
i=float(lines[1])
n=int(lines[2])
N=int(lines[3])
S=(math.pi*D*N)/(n*i)

print(S)

print(Betta)
```

Задача C. Движение по дуге. Скорости

Ограничения: по времени 1 сек, по памяти 256 Мб.

Описание робота: дифференциальная платформа.



D – диаметр колес робота

L – полуколея робота, расстояние от геометрического центра робота до центра колеса.

Точка *O* – геометрический центр робота, начало локальной системы координат робота. Точка является серединой оси, соединяющей колеса.

Задание:

Робот должен совершить движение по дуге с максимально возможной скоростью. Необходимо вычислить скорости вращения колес, требуемые для совершения поворота робота с заданным радиусом.

Формат входных данных

Первая строка содержит дробное положительное число *L* от 0.0001 до 1.0 (с точностью до 4 знаков после запятой) – расстояние от центра робота до центра колеса (*L*) в метрах. "Половина колеи робота".

Вторая строка содержит дробное число *R* от -10^3 до 10^3 (с точностью до 4 знаков после запятой) – радиус поворота робота в метрах (*r*). Положительное или нулевое значение означает движение по дуге против часовой стрелки, отрицательное – по часовой стрелке.

Формат выходных данных

Выведите два числа, содержащие дробные числа (с точностью до 4 знаков после запятой), причем одно из них равно 100.0 или -100.0 (максимальная возможная скорость), а второе в диапазоне от -100.0 до 100.0.

В первой строке скорость левого мотора робота.

Во второй строке скорость правого мотора робота.

Пример 1

Входные данные:	Выходные данные:
0.1220	0.0000
0.1220	100.0000

Решение:

Описанное движение по дуге может иметь разные направления, то есть скорость левого колеса может быть как больше, так и меньше скорости правого. На первом этапе определяем знак радиуса и от него выбираем, скорость какого колеса будет максимальной по модулю. Выбор колеса с максимальной скоростью можно реализовать с помощью ветвления с условием, в каждой ветке которого вычисляется скорость более медленного колеса. То есть, при неотрицательном радиусе скорости правого колеса присваивается значение 100, а скорость левого вычисляется; при отрицательном радиусе скорости левого колеса присваивается значение 100, а скорость правого вычисляется.

Вторым этапом происходит вычисление скорости вращения более медленного колеса. Об этом подробно описано в видеоролике «Дифференциальная платформа роботов». Там же приводятся итоговые формулы для вычисления скоростей.

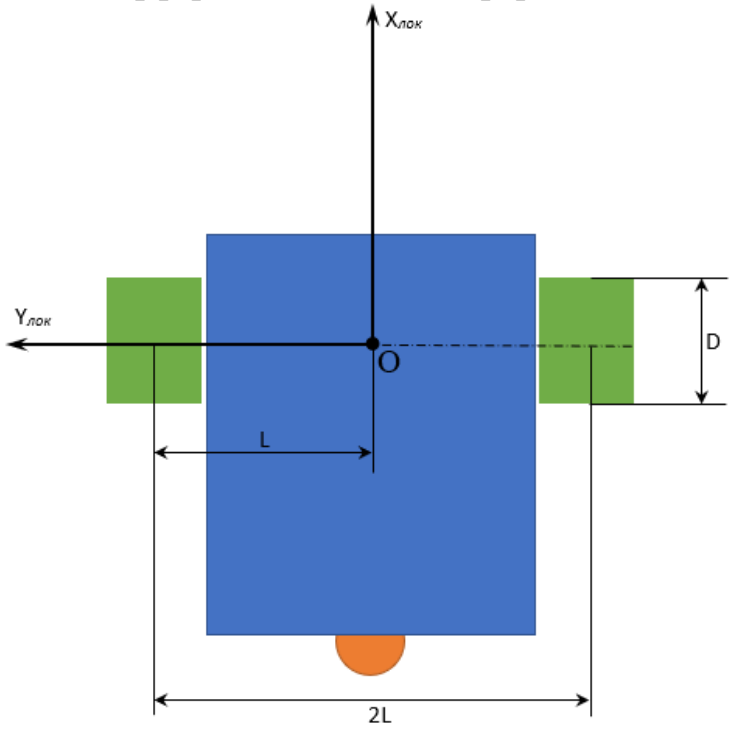
Итоговая программа на ЯП Python может иметь следующий вид:

```
from sys import stdin
import sys
import math

lines = []
for line in stdin:
    lines.append(line)

L = float(lines[0])
R=float(lines[1])
if R>=0:
    Vr = 100.0
    Vl = Vr*(R-L)/(R+L)
else:
    Vl = 100.0
    Vr = Vl*(R+L)/(R-L)

print(Vl)
print(Vr)
```

Задача D. Движение по дуге. Градусы

Ограничения: по времени 1 сек, по памяти 256 Мб.

Описание робота: дифференциальная платформа.



D – диаметр колес робота

L – полуколея робота, расстояние от геометрического центра робота до центра колеса.

Точка *O* – геометрический центр робота, начало локальной системы координат робота. Точка является серединой оси, соединяющей колеса.

Задание:

Робот совершает движение по дуге заданного радиуса. В момент начала движения его направление совпадает с осью *X* глобальной системы координат, то есть угол его поворота равен нулю. Необходимо определить, на сколько градусов надо повернуть колеса робота, чтобы весь робот совершил поворот на заданный угол по дуге заданного радиуса. Причем если угол поворота больше 360 градусов, то совершать полный оборот не нужно, требуется совершить поворот на количество градусов, превышающих 360.

Формат входных данных

Первая строка содержит дробное положительное число *D* от 0.0001 до 1.0 (с точностью до 4 знаков после запятой) – диаметр колес робота в метрах.

Вторая строка содержит дробное положительное число *L* от 0.0001 до 1.0 (с точностью до 4 знаков после запятой) – расстояние от центра робота до центра колеса в метрах. "Половина колеи робота".

Третья строка содержит дробное число *r* от -10^3 до 10^3 (с точностью до 4 знаков после запятой) – радиус поворота робота в метрах. Положительное или нулевое

значение означает движение по дуге против часовой стрелки, отрицательное – по часовой стрелке.

Четвертая строка содержит целое число φ от -10^6 до 10^6 – угол поворота робота в градусах. Положительное число означает поворот робота против часовой стрелки, отрицательное – по часовой стрелке.

Формат выходных данных

Выведите на первой строке дробное число с точностью до 4 знаков после запятой – угол поворота левого колеса. Положительное число означает вращение, обеспечивающее движение робота вперед.

Выведите во второй строке дробное число с точностью до 4 знаков после запятой – угол поворота правого колеса. Положительное число означает вращение, обеспечивающее движение робота вперед.

Пример 1

Входные данные:	Выходные данные:
1 1 3 12	48.0 96.0

Пример 2

Входные данные:	Выходные данные:
1 1 3 -12	1392.0 2784.0

Решение:

Задача компилирует в себе решения из предыдущих. Рекомендуется предварительно изучить их решения.

Угол поворота робота при движении по дуге определяется по следующей формуле:

$$\varphi = \frac{360S}{2\pi r}$$

где r – радиус поворота робота, в метрах,

S – путь, пройденный центром робота, в метрах,

φ – угол поворота робота, в градусах.

При совершении поворота роботом его колеса проходят разный путь, что связано с разным радиусом поворота самих колес. Так как за положительный поворот принято движение робота влево (против часовой стрелки), то радиус левого колеса вычисляется как $R_{лев} = r - L$, а правого как $R_{прав} = r + L$.

Из задачи A известно, как вычислять угол β поворота колеса. Совместим все эти формулы и получим выражения для вычисления угла поворота каждого из колес:

$$\beta_{\text{лев}} = \frac{2\varphi(r - L)}{D}$$

$$\beta_{\text{прав}} = \frac{2\varphi(r + L)}{D}$$

По условию задачи совершать поворот более 360 градусов не требуется, поэтому в формулах следует использовать не исходный угол φ, а его остаток от деления на 360.

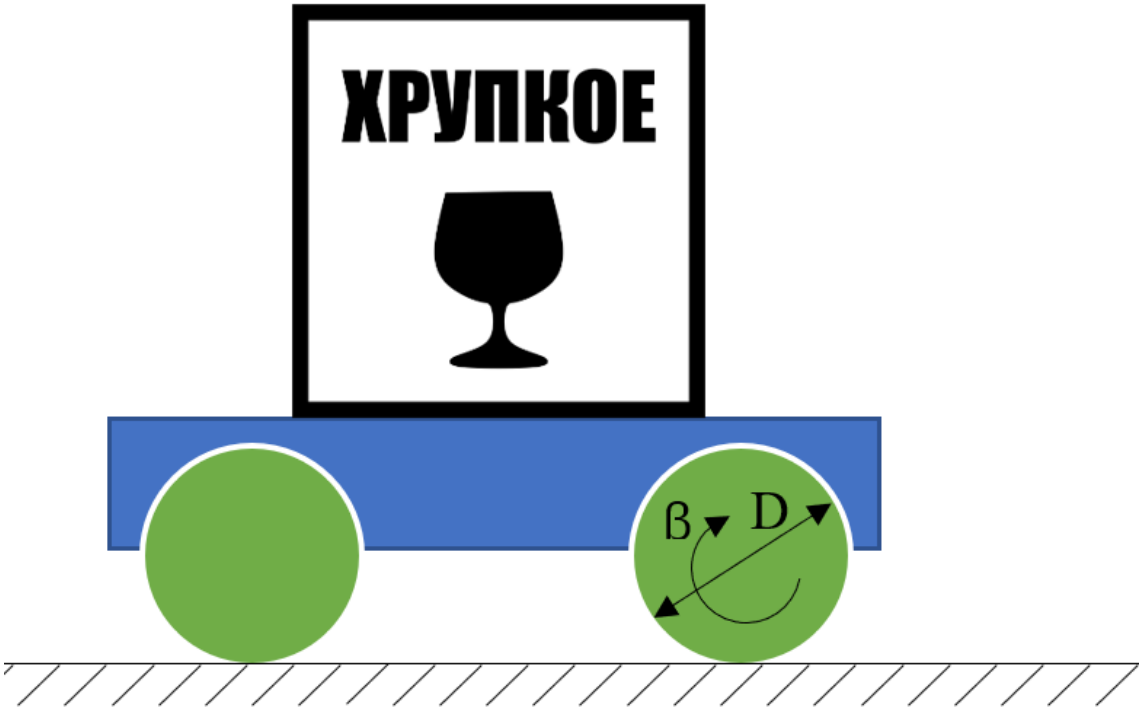
Итоговая программа на ЯП Python может иметь следующий вид:

```
from sys import stdin
import sys
import math

lines = []
for line in stdin:
    lines.append(line)

D = float(lines[0])
L = float(lines[1])
R = float(lines[2])
F = int(lines[3])
F = F % 360
Bl = (2*F*(R-L))/D
Br = (2*F*(R+L))/D

print(Bl)
print(Br)
```

Задача E. Ускорение робота

Ограничения: по времени 1 сек, по памяти 256 Мб.

Описание робота: колесный робот.



Робот оборудован колесами диаметром D мм.

Каждое колесо имеет привод с редукцией i между мотором и выходным валом, энкодером с количеством тиков n на один оборот мотора.

Робот перевозит хрупкий груз, который разрушается при ускорении или торможении выше $a_{доп}$ м/с^2.

Задание:

Робот перевозит хрупкий груз, который разрушается при слишком высоком ускорении / торможении. Робот совершил движение с неравномерной скоростью, но при этом в любой момент времени скорости всех моторов были равны между собой. Каждые 0.01 секунды робот записывал показания энкодеров. По показаниям энкодера необходимо вычислить скорость (в м/с) и ускорение (в м/с^2) робота в каждый момент времени. И определить, в какой момент времени у робота была самая высокая скорость (если таких моментов несколько – то назвать самый первый по времени) и не повредил ли он груз.

Система оценки

Каждый тест оценивается 3-я баллами.

Формат входных данных

Входные данные состоят из четырех строк.

Первая строка содержит три разделенных пробелами числа, описывающих параметры привода робота:

Первое дробное число – диаметр колес робота в метрах D ($0.0001 \le D \le 100.0$), в метрах.

Второе дробное число – редукция, передаточное число между мотором и колесом i ($0.0001 \le i \le 1000.0$). Показывает, во сколько раз колесо вращается медленнее, чем мотор.

Третье целое четное число – количество "тиков" энкодера на один оборот мотора n ($2 \le n \le 1440$).

Вторая строка содержит дробное положительное число A от 0.0001 до 100.0 (с точностью до 4 знаков после запятой) – предельно допустимое ускорение/торможение в м/с^2.

Третья строка содержит целое положительное число E от 1 до $6*10^4$ – количество показаний энкодеров в последующем массиве.

Четвертая строка содержит E целых чисел от -10^9 до 10^9, разделенных пробелами – показания энкодеров в каждый момент времени.

Формат выходных данных

Выведите 3 строки с числами.

Строка 1 возвращает дробное положительное число – время, в которое была зафиксирована максимальная по модулю скорость (если таких моментов несколько – первое встреченное), в секундах.

Строка 2 возвращает дробное положительное число – время, в которое было зафиксировано максимальное ускорение / торможение робота (если таких моментов несколько – первое встреченное), в секундах.

Строка 3 возвращает: 7 если во время движения ускорение не превышало допустимого, 1 если выше допустимого было ускорение, -1 если выше допустимого было торможение, и 0 если выше допустимого ускорения и торможения.

Пример 1

Входные данные:	Выходные данные:
13.9802 263.6 112 54.7715 6 -445896924 25059452 916183550 -514759757 -886923682 -175817540	0.03000 0.03000 0

Пример 2

Входные данные:	Выходные данные:
0.082 263.6 16 4.8900 6 6 13 20 28 36 42	0.03000 0.00000 7

Решение:

Из школьного курса физики известно, что скорость – это изменение положения тела за единицу времени:

$$\vartheta = \frac{\Delta x}{\Delta t} = \frac{S}{t_{\text{кон}} - t_{\text{нач}}},$$

а ускорение – это изменение скорости за единицу времени:

$$a = \frac{\Delta \vartheta}{\Delta t} = \frac{\vartheta_{\text{кон}} - \vartheta_{\text{нач}}}{t_{\text{кон}} - t_{\text{нач}}}.$$

Из условия задачи известно, что dt=0,01 с, а из входных данных известны показания энкодеров на каждом шаге. Учитывая, что разница в показаниях энкодеров между двумя шагами определяет пройденный роботом путь и используя формулу из решения задачи А можно вывести формулу для вычисления скорости между двумя считывания энкодера:

$$\vartheta = \frac{\pi D(Enc_{\text{кон}} - Enc_{\text{нач}}}{in(t_{\text{кон}} - t_{\text{нач}})}$$

где $Enc_{нач}$ и $Enc_{кон}$ – показания энкодеров на предыдущем и текущем шаге, $t_{нач}$ и $t_{кон}$ – моменты начала и окончания измерения, $t_{кон} - t_{нач} = dt = 0.01$ с, остальные величины описаны в условиях задачи.

Проверка на превышение допустимого значения ускорения и выбор значения для последней выходной строки реализуется с помощью ветвления или операций с логическими значениями.

Итоговая программа на ЯП Python может иметь следующий вид:

```
from sys import stdin
import sys
import math

dt = 0.01
Vmax=0.0
Amax=0.0
tVmax=0.0
tAmax=0.0
t=0.0
enc_old=0
Vold=0
overAcc=0
overBrak=0

D,i,n=map(float,input().split())
n=int(n)
Aacc = float(input())
E = int(input())
enc_list = list( map( int, input().split() ) )

for enc in enc_list:
    t=t+dt
    dEnc=enc - enc_old
    V = (math.pi*D*dEnc)/(i*n*dt)
    A=(V-Vold)/dt
    if math.fabs(V)>Vmax:
        Vmax=math.fabs(V)
        tVmax=t
    if math.fabs(A)>Amax:
        Amax=math.fabs(A)
        tAmax=t
    if A > Aacc:
```

```
            overAcc=1
        if -A > Aacc:
            overBrak=1
        enc_old=enc
        Vold=V

print(tVmax)
print(tAmax)

res=7
if overAcc>0 or overBrak>0:
    res=overAcc-overBrak

print(res)
```

Задача F. Движение по энкодерам ТРИК

Задача:

В симуляторе TRIK Studio подготовить управляющую программу, перемещающую робота по окружности с заданным радиусом на заданное расстояние.

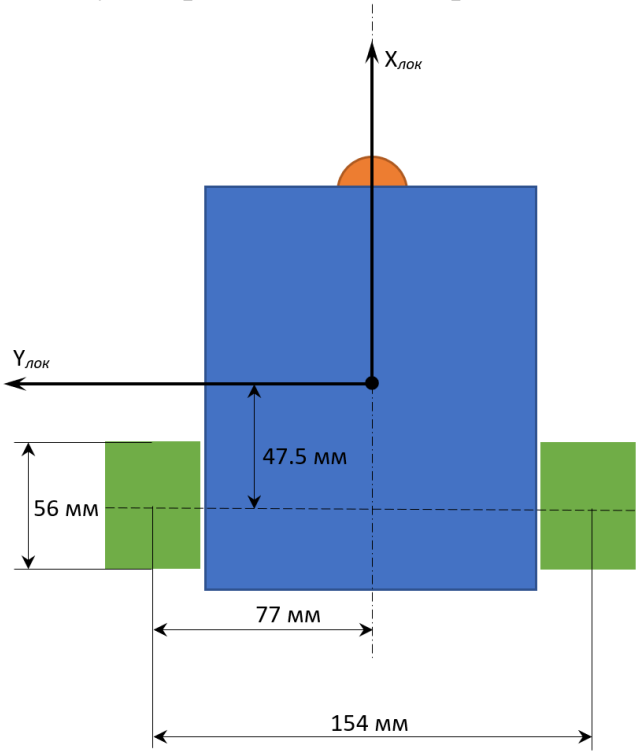
Управляющая программа проверяется на нескольких полях, имеющих разные входные данные. За каждое поле начисляется от 4% до 6% баллов, в зависимости от сложности. Поле засчитывается при совпадении следующих условий:

- любая точка робота находится в зоне радиусом 200 мм, центр которой совпадает с центром эталонной точки остановки робота (координаты эталонной точки вычислялись в миллиметрах с точностью до третьего знака после запятой);
- скорости обоих моторов робота равны нулю.

Робот:

Дифференциальная платформа.

Размеры модели в симуляторе TRIK Studio представлены на рисунке.



Левый мотор робота подключен к порту M4, правый мотор - к порту M3.

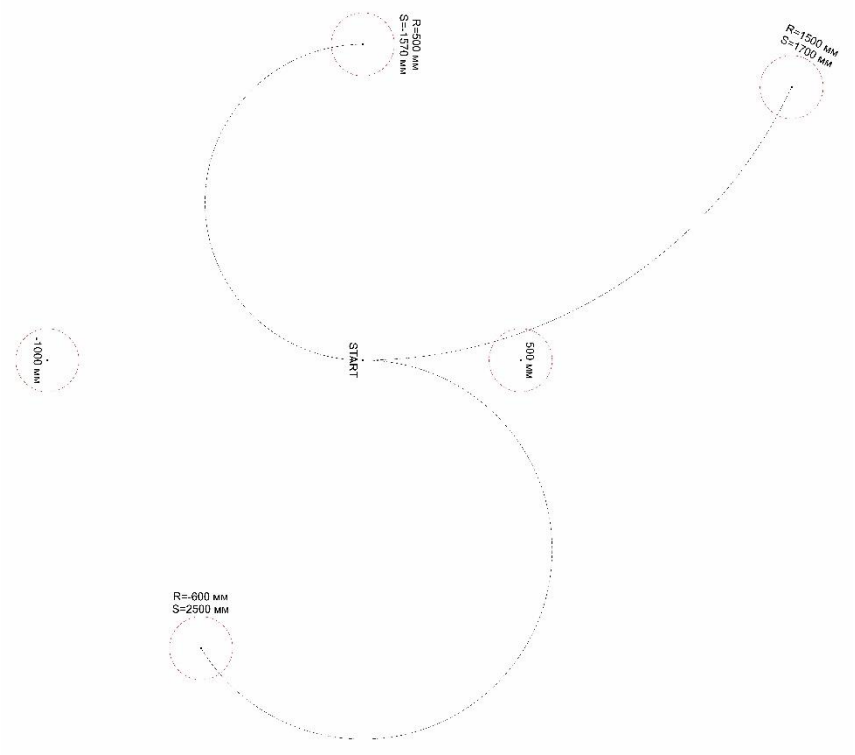
Энкодеры робота возвращают угол поворота колес в градусах. Положительные числа обозначают вращение колеса, обеспечивающего движение робота вперед, отрицательные - назад.

Полигон:

Общий размер полигона составляет 4 x 4 метра.

Стартовая секция и стартовое положение робота - точно в центре полигона.

Пример полей:



Формат входных данных:

Параметры движения содержатся в файле *input.txt*

В первой строке файла указан радиус поворота робота в виде дробного числа с точностью до 4 знаков после запятой. Положительное число означает поворот налево (при взгляде сверху движение по часовой стрелке) с радиусом, равным указанному числу (в метрах). Отрицательное число означает поворот направо (при взгляде сверху движение против часовой стрелки) с радиусом, равным указанному числу (в метрах). Число 9999 означает прямолинейное движение.

Во второй строке файла указано расстояние движения в виде дробного числа с точностью до 4 знаков после запятой. Положительное число означает движение вперед на расстояние, равное указанному числу (в метрах). Отрицательное число означает движение назад на расстояние, равное указанному числу (в метрах).

Пример 1 (прямолинейное движение вперед на 0,5 м):

9999

0.5

Пример 2 (прямолинейное движение назад на 1 м):

9999

-1

Пример 3 (движение направо по дуге с радиусом 0,6 м на 2,5 м):

-0,6

2.5

Пример 4 (движение назад налево по дуге с радиусом 0,5 м на 1,57 м):
0.5
-1.57

Примеры заданий и тестовый проект:
В приложенном архиве находится следующие файлы:

- *Test_field_easy_exercise.qrs* - файл-упражнение с настроенным тренировочным полем и отключенными реалистичными физикой и поведением моторов;
- *Test_field_hard_exercise.qrs* - файл-упражнение с настроенным тренировочным полем и включенными реалистичными физикой и поведением моторов;
- *input.txt* - текстовый файл, структура которого описана в разделе "Формат входных данных";
- *task_6.py* - файл-программа с исходным кодом участника.
- Файлы-упражнения могут использоваться для отладки программ. После отладки код программы необходимо перенести в файл *task_6.py* и отправить в качестве решения задачи.

Данные: solutions.zip

Решение:
Задача компилирует в себе решения из предыдущих. Рекомендуется предварительно изучить их решения.

Для прохождения всех тестовых полей на полный балл требуется не только вычислить скорости колес и расстояние проезда из показаний энкодеров, но и добиться стабильного движения робота с синхронизацией колес. О синхронизации подробнее описывается в видео «Синхронизация моторов двухколесного робота».

Итоговая программа на ЯП Python может иметь следующий вид:

```
import sys
import time
import random
import math

class Program():
  __interpretation_started_timestamp__ = time.time() * 1000

  pi = 3.141592653589793

  def execMain(self):
    Diam = 0.056 #диаметр колес, в метрах
    L = 0.077 #полуколея робота, в метрах
    L2 = 0.154 #колея робота, в метрах
    #добавьте свой код сюда

    R=9999
    S=0.5
    leftSpd=0
    rightSpd=0
    maxSpd=75
```

```
        enc = 0
        encL=0

        encR=0
        lines = script.readAll("solutions/input.txt")
        R=float(lines[0])
        S=float(lines[1])
        alpha=(360*S)/(math.pi*Diam)
        kp=1.5
        ki=0
        kd=0.5
        I=0
        err=0
        errold=0
        if R==9999:
          leftSpd=maxSpd * S / math.fabs(S)
          rightSpd=maxSpd* S / math.fabs(S)
        else:
          if R>=0:
            rightSpd = maxSpd * S / math.fabs(S)
            leftSpd = rightSpd*(R-L)/(R+L)
          else:
            leftSpd = maxSpd * S / math.fabs(S)
            rightSpd = leftSpd*(R+L)/(R-L)
        brick.motor("M3").setPower(rightSpd)
        brick.motor("M4").setPower(leftSpd)
        while not (math.fabs(enc) > math.fabs(alpha)):
          if rightSpd != 0 and leftSpd != 0:
            err=encR*leftSpd/rightSpd - encL
            P=kp*err
            I=I+ki*err
            D=kd*(err-errold)
            U=P+I+D
            errold=err
            brick.motor("M3").setPower(rightSpd-
U*rightSpd/math.fabs(rightSpd)*leftSpd/math.fabs(leftSpd) )
            brick.motor("M4").setPower(leftSpd+U)

          encL=brick.encoder("E4").read()

          encR=brick.encoder("E3").read()
          enc=(encL+encR)/2
        brick.motor("M3").powerOff()
        brick.motor("M4").powerOff()
        brick.stop()
        return

    def main():
      program = Program()
      program.execMain()

    if __name__ == '__main__':
      main()
```

Задача G. Движение по линии ТРИК

Задача:

В симуляторе TRIK Studio подготовить управляющую программу, перемещающую робота вдоль линии и останавливающую его на третьем встреченном перекрестке.

Управляющая программа проверяется на нескольких полях, имеющих разную конфигурацию поворотов, прямых участков и перекрестков. За каждое поле начисляется до 10% баллов. Поле засчитывается при совпадении следующих условий:

- центр робота находится в зоне 300x300 мм, центр которой совпадает с центром третьего по ходу движения робота перекрестка;
- скорости обоих моторов робота равны нулю.

Робот:

Дифференциальная платформа. Робот оснащен четырьмя датчиками отраженного света, направленными вниз:

К порту A1 подключен левый боковой датчик.

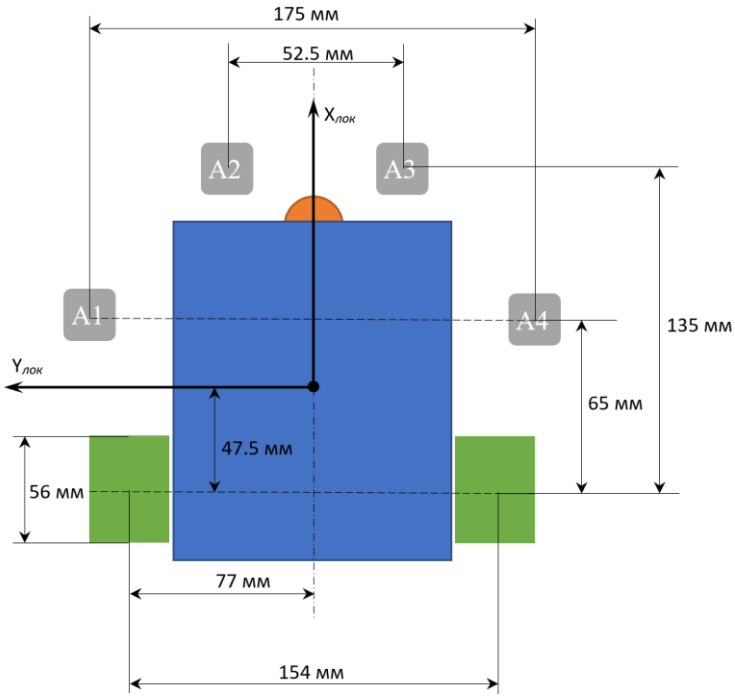
К порту A2 подключен левый передний датчик.

К порту A3 подключен правый передний датчик.

К порту A4 подключен правый боковой датчик.

Размеры модели в симуляторе TRIK Studio представлены на рисунке.

Описание робота ТРИК



Левый мотор робота подключен к порту M4, правый мотор - к порту M3.

Энкодеры робота возвращают угол поворота колес в градусах. Положительные числа обозначают вращение колеса, обеспечивающего движение робота вперед, отрицательные - назад.

Полигон:

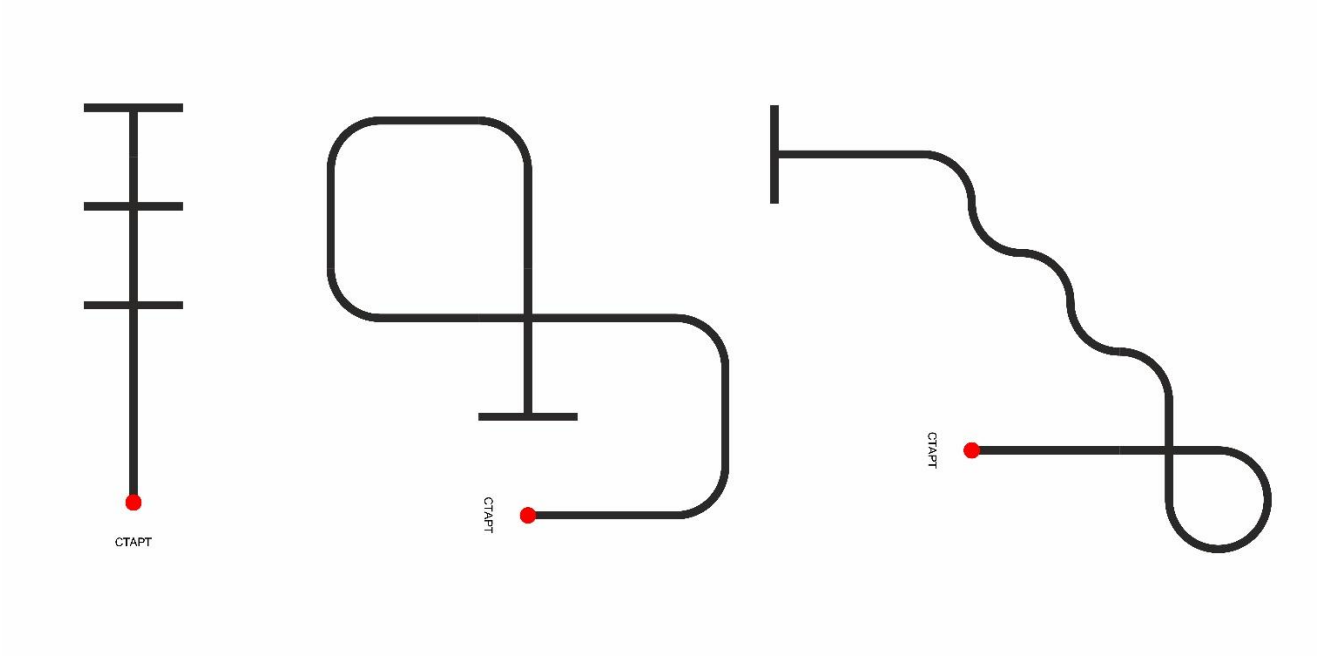
Общий размер полигона составляет 4 х 4 метра.

Стартовая секция и стартовое положение робота - точно в центре полигона.

Ширина линии не менее 25 мм.

Гарантируется, что на старте линия расположена между датчиками A2 и A3. Гарантируется. что первые 300 мм линии - прямой участок. Далее следуют повороты с радиусом не менее 150 мм. Гарантируется, что боковые линии на перекрестках отстоят от основной линии не менее чем на 125 мм и имеют ширину не менее 25 мм.

Пример полей:



Примеры заданий и тестовый проект:

В приложенном архиве находится следующие файлы:

- *Test_field_1_exercise.qrs* / *Test_field_2_exercise.qrs* / *Test_field_3_exercise.qrs* – файлы-упражнения с настроенными тренировочными полями;
- *task_7.py* - файл-программа с исходным кодом участника.
- Файлы-упражнения могут использоваться для отладки программ. После отладки код программы необходимо перенести в файл *task_7.py* и отправить в качестве решения задачи.

Данные: solutions.zip

Решение:

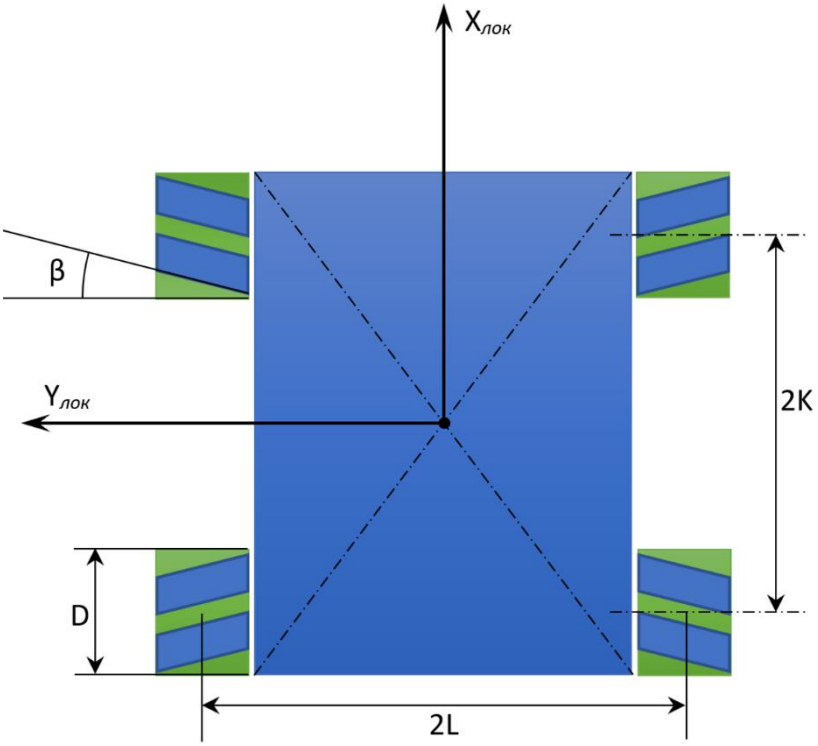
См. решение задачи G «Одометрия дифф. платформы ТРИК» для возрастной группы 9–11 класс, оно подходит.

Возрастная категория 9–11 классы

Задача A. Вычисление скоростей mecanum-платформы

Ограничения: по времени 1 сек, по памяти 256 Мб.

Описание робота: mecanum-платформа.



Мобильный робот построен по кинематической схеме с четырьмя колесами Илона (mecanum-wheel) диаметром *D* мм. Ролики колес отклонены от нормали на β градусов (при β=90 градусов ролики установлены поперек плоскости колеса, как у омни-колес). Обратите внимание, что показан вид робота сверху, то есть в точках касания колес с поверхностью ролики каждого колеса повернуты зеркально.

Расстояние между центрами передних и задних колес *2K* миллиметров. Между центром левых и правых колес *2L* миллиметров.

Центр локальной системы координат ($X_{лок}$, $Y_{лок}$) совпадает с геометрическим центром робота и центром между колесами. При этом ось $X_{лок}$ совпадает с направлением робота “вперед”.

Задание:

Робот начинает перемещение с заданными управляющими координатами *X*, *Y* и *M*. Необходимо вычислить скорости вращения колес в рад/с для совершения этого движения.

Формат входных данных

Входные данные состоят из пяти строк.

Первая строка содержит три разделенных пробелами числа, описывающих параметры mecanum-платформы:

Первое число – половина расстояния между центрами левого и правого колес (L), в метрах.

Второе число – половина расстояния между центрами переднего и заднего колес (K), в метрах.

Третье число – угол поворота роликов относительно нормали (β) в градусах.

Вторая строка содержит дробное положительное число от 0.0001 до 1.0 (с точностью до 4 знаков после запятой) – диаметр колес робота (D) в метрах.

Третья строка содержит дробное число от -1.0 до 1.0 (с точностью до 4 знаков после запятой) – составляющая X вектора движения в м/с; положительное значение – движение робота вперед, отрицательное – назад.

Четвертая строка содержит дробное число от -1.0 до 1.0 (с точностью до 4 знаков после запятой) – составляющая Y вектора движения в м/с; положительное значение – движение робота влево, отрицательное – вправо.

Пятая строка содержит дробное число от -1.0 до 1.0 (с точностью до 4 знаков после запятой) – составляющая M (вращение) движения в рад/с; положительное значение – поворот робота против часовой стрелки при взгляде сверху, отрицательное – по часовой стрелке.

Формат выходных данных

Выведите четыре строки с числами, обозначающими скорости вращения колес робота, от -1.0 до 1.0 с точностью до 4 знаков после запятой, положительное значение – вращение колеса вперед, отрицательное – назад.

Первая строка содержит скорость вращение левого переднего колеса в рад/с.

Вторая строка содержит скорость вращения левого заднего колеса в рад/с.

Третья строка содержит скорость вращения правого заднего колеса в рад/с.

Четвертая строка содержит скорость вращения правого переднего колеса в рад/с.

Пример 1

Входные данные:	Выходные данные:
4 4 45	-16.0000
1.0	-14.0000
0.5	16.0000
0.5	18.0000
1.0	

Решение:

Учитывая материалы из видео «Плоскопараллельное движение роботов» и «Роботы на роликовых колесах» можно вывести формулы для вычисления колес:

$$F_{xi} = X - \sqrt{K^2 + L^2} M \sin \alpha_i$$
$$F_{yi} = Y + \sqrt{K^2 + L^2} M \cos \alpha_i$$

$$F_i = \sqrt{F_{xi}^2 + F_{yi}^2}$$

$$\varphi_i = arctan\frac{F_{yi}}{F_{xi}}$$

$$\vartheta_i = F_i \cos\varphi_i - \frac{F_i \sin\varphi_i}{\tan\beta_i}$$

$$\omega_i = \frac{2\vartheta_i}{D}$$

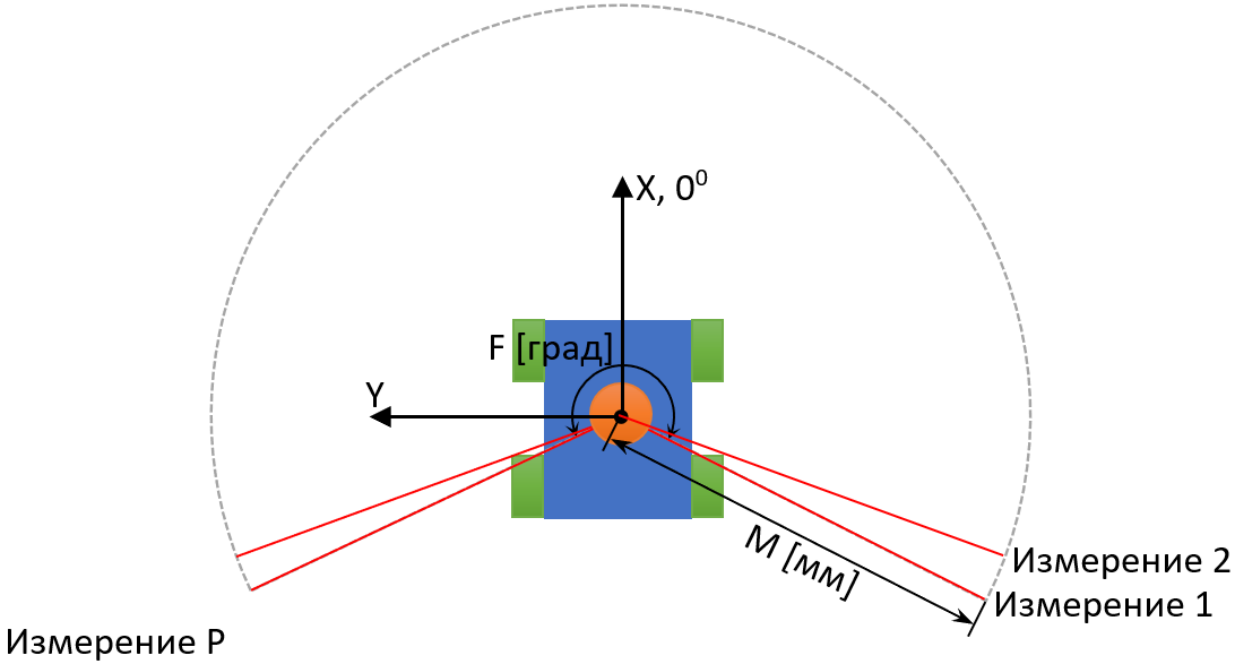
Во всех этих формулах индекс i означает номер колеса, для которого отличаются углы α и β.

Итоговая программа на ЯП Python может иметь следующий вид:

```
from sys import stdin
import sys
import math
import array

L,K,B=map(float,input().split())
B=int(B)
D = float(input())
X = float(input())
Y = float(input())
M = float(input())

alpha=[0 for i in range(4)]
betta=[0 for i in range(4)]
Fx=[0 for i in range(4)]
Fy=[0 for i in range(4)]
Phi=[0 for i in range(4)]
Vel=[0 for i in range(4)]
Ang=[0 for i in range(4)]

alpha[0]=math.atan2(L, K)
alpha[1]=math.atan2(L, -K)
alpha[2]=math.atan2(-L, -K)
alpha[3]=math.atan2(-L, K)
betta[0]=math.radians(B)
betta[1]=math.radians(-B)
betta[2]=math.radians(B)
betta[3]=math.radians(-B)

for i in range(0,4):
    Fx[i] = X - math.sqrt(K*K+L*L)*M*math.sin( alpha[i] )
    Fy[i] = Y + math.sqrt(K*K+L*L)*M*math.cos( alpha[i] )
    F = math.sqrt(Fx[i]**2+Fy[i]**2)
    Phi[i] = math.atan2(Fy[i],Fx[i])
    Vel[i] = F*math.cos(Phi[i]) - F*math.sin(Phi[i]) / math.tan( betta[i] )
    Ang[i]=Vel[i]*2/D

for i in range(0,4):
    print(  float( '{:.4f}'.format(Ang[i]) ) )
```

Задача B. LIDAR

Ограничения: по времени 1 сек, по памяти 256 Мб.



Робот оборудован LIDAR'ом. Угол его сканирования F градусов, на эти F градусов приходится P измерений LIDAR'а.

Максимальное расстояние измерения – до M миллиметров. Если LIDAR не видит объекта, то соответствующее измерение имеет максимальное значение.

Нулевые координаты декартовой системы координат OXY совпадают с центром LIDAR'а, от которого он измеряет расстояние. Ось X этой системы координат совпадает с серединой сектора сканирования LIDAR'а.

Кроме того, в системе присутствует полярная система координат.

Ее полюс совпадает с началом декартовой системы координат и центром LIDAR'а.

А полярная ось (нулевой луч) совпадает с осью X декартовой системы координат.

Задание:

Необходимо перевести координаты каждой найденной точки объекта в декартовы.

При этом измерения LIDAR'а, которые не обнаружили объект, переводить в декартовы координаты не требуется.

Замечание

Готовые библиотеки для решения задачи использовать запрещено.

Примеры входных данных представлены по ссылке и не выводятся в самом задании. Примеры тестовых примеров представлены по ссылке.

Формат входных данных

Первая строка содержит целое число F –угол в градусах ($F \in [1; 360]$).

Вторая строка содержит целое число P – количество измерений ($P \in [1; 4096]$).

Третья строка содержит целое число M – дистанция измерения LIDAR'a в миллиметрах ($M \in [1; 4000]$).

Четвертая строка содержит P целых чисел – показания LIDAR'a.

Формат выходных данных

Первая строка содержит целое число – количество измерений LIDAR'a, в которых обнаружен объект.

Если ни в одном измерении объекты не обнаружены, то строка содержит число 0.

Следующие строки присутствуют только если в первой строке число больше нуля; они содержат через пробел пары чисел с плавающей точкой – координаты X и Y точки обнаруженного объекта с точностью до 0.1 мм.

Вторая строка содержит координаты из первого измерения LIDAR'a, в котором обнаружен объект, третья – из второго измерения, в котором обнаружен объект и т.д.

Решение:

Задача сводится к последовательному переводу координат из полярной в прямоугольную систем координат. Дополнительно вводится условие на проверку не максимальных значений, координаты для которых и преобразуются.

Итоговая программа на ЯП Python может иметь следующий вид:

```
from sys import stdin
import sys
import math

#F,P,M=map(int,input().split())
F=int(input())
P=int(input())
M=int(input())
values = list( map( int, input().split() ) )
xyValues = []
n=0

for i in range(P):
    #print(i)
    if values[i]<M:
        step=P/F
        alpha = i*F/(P-1)-F/2
        #print(alpha)
        X= values[i]*math.cos(math.radians(alpha))
        Y= values[i]*math.sin(math.radians(alpha))
        xyValues.append([0]*2)
        xyValues[n]=[X,Y]
        n+=1

print(n)
for val in xyValues:
    print( float('{:.1f}'.format(val[0])), float( '{:.1f}'.format(val[1])))
```

Задача C. Одометрия материальной точки

Ограничения: по времени 1 сек, по памяти 256 Мб.

В этой задаче робот представлен абстрактной точкой.

Задание:

Робот начинает движение в точке с нулевыми координатами и совершает сложное движение. Каждые *dt* секунд фиксируется смещение робота *dS* и изменение угла поворота *dM*. По известным на каждом шаге движения изменению времени *dt*, пройденному роботом пути *dS* и изменению угла поворота робота *dM* необходимо определить конечные координаты робота.

Формат входных данных

В первой строке записано натуральное число QT ($1 \le QT \le 6 * 10^4$) -- общее количество сделанных измерений.

Следующие QT строк содержат по три числа, разделенных пробелами:

Первое число (дробное, от 0.0001 до 1.0 с точностью до 4 знаков после запятой) – *dt* текущего шага, в секундах.

Второе число (дробное, от -2.0 до 2.0 с точностью до 4 знаков после запятой) – *dS* текущего шага, в метрах.

Третье число (дробное, от -π до π с точностью до 4 знаков после запятой) – *dM* текущего шага, в радианах.

Формат выходных данных

Выведите две строки, содержащих дробные числа, с точностью до четырех знаков после запятой.

В первой строке должна быть указана результирующая *X*-координата робота.

Во второй строке должна быть указана результирующая *Y*-координата робота.

Пример 1

Входные данные:	Выходные данные:
1	-0.0000
1 2 1.5708	2.0000

Решение:

Одометрию материальной точки можно описать формулами

$$\begin{cases} X_n = X_{n-1} + \Delta X \\ Y_n = Y_{n-1} + \Delta Y \\ M_n = M_{n-1} + \Delta M \end{cases}$$

где:

X, *Y* – координаты центра робота в Декартовой системе координат,

M – угол поворота робота, то есть его направление,

n – обозначение текущего шага,

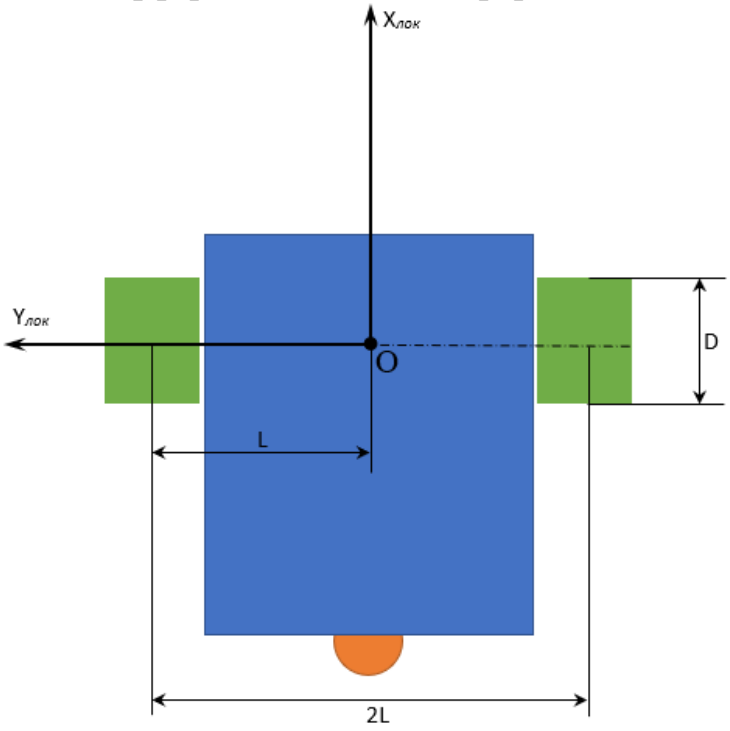
n-1 – обозначение предыдущего шага,
Δ – «приращение», изменение координаты за текущий шаг.
Так как все эти данные поступают в программу как входные, то задача сводится к многократному аккуратному вычислению по формулам.
Итоговая программа на ЯП Python может иметь следующий вид:

```
from sys import stdin
import sys
import math

QT=int(input())
X=0
Y=0
M=0

for i in range(QT):
    dt, dS, dM=map(float,input().split())
    dX = dS*math.cos(M+dM)
    dY = dS*math.sin(M+dM)
    X = X + dX
    Y = Y + dY
    M = M + dM

print(float( '{:.4f}'.format(X) ))
print(float( '{:.4f}'.format(Y) ))
```

Задача D. Одометрия дифф. платформы

Ограничения: по времени 1 сек, по памяти 256 Мб.

Описание робота: дифференциальная платформа.



D – диаметр колес робота

L – полуколея робота, расстояние от геометрического центра робота до центра колеса.

Точка *O* – геометрический центр робота, начало локальной системы координат робота. Точка является серединой оси, соединяющей колеса.

Задание:

Робот начинает движение в точке с нулевыми координатами и совершает сложное движение. Каждые *dt* секунд фиксируется показания энкодеров обоих моторов робота. По известным на каждом шаге движения изменению времени *dt* и показаниям энкодеров необходимо определить конечные координаты робота.

Формат входных данных

Первая строка содержит три разделенных пробелами числа, описывающих параметры привода робота:

Первое число (дробное, положительное, от 0.0001 до 1.0 с точностью до 4 знаков после запятой) – диаметр колес робота (*D*) в метрах.

Второе число (дробное, положительное, от 0.0001 до 1000.0 с точностью до 4 знаков после запятой) – редукция, передаточное число между мотором и колесом (*i*). Показывает, во сколько раз колесо вращается медленнее чем мотор.

Третье число (целое, положительное, четное, от 2 до 1440) – количество «тиков» энкодера на один оборот мотора (*n*).

Вторая строка содержит дробное положительное число от 0.0001 до 1.0 (с точностью до 4 знаков после запятой) – расстояние от центра робота до центра колеса (L) в метрах. «Половина колеи робота».

Третья строка содержит целое число от 1 до 6 * 10^4 – количество измерений (QT).

Следующие QT строк содержат по три разделенных пробелами числа:

Первое число (дробное, положительное, от 0.0001 до 1.0 с точностью до 4 знаков после запятой) – dt текущего шага, в секундах.

Второе число (целое, от -10^6 до 10^6) – показания энкодера левого мотора на текущем шаге (N_{left}).

Третье число (целое, от -10^6 до 10^6) – показания энкодера правого мотора на текущем шаге (N_{right}).

Формат выходных данных

Выведите две строки, содержащих дробные числа, с точностью до четырех знаков после запятой.

Первая строка должна содержать X-координату робота в момент окончания движения.

Вторая строка должна содержать Y-координату робота в момент окончания движения.

Пример 1

Входные данные:	Выходные данные:
5.0 1.0 1 1.0 1 1.0 1 -1	0.0000 0.0000

Пример 2

Входные данные:	Выходные данные:
0.1571 738.7856 360 0.2985 5 0.4176 659 711 0.9776 1364 1688 0.7325 1915 2852 0.2752 2517 3466 0.3579 2943 3779	0.0062 0.0000

Пример 3

Входные данные:

0.7456 177.6791 1056
0.7913
10
0.4560 399 740
0.0605 1001 1205
0.7352 1316 1377
0.1687 1327 1387
0.1482 1699 2059
0.0264 1980 2236
0.5002 2455 2710
0.3754 2688 3231
0.8237 3325 4228
0.1459 3607 4595

Выходные данные:

0.0512
0.0002

Решение:

Одометрия дифференциальной платформы описана в видео «Дифференциальная платформа роботов». В текущей задаче можно воспользоваться полными формулами из видео или упрощенными, предполагая, что при малых «шагах» (перемещениях робота) длина хорды примерно равна длине дуги. В этом случае формулы примут вид:

$$\begin{cases} X_n \\ Y_n \\ M_n \end{cases} = \begin{cases} X_{n-1} + \Delta X \\ Y_{n-1} + \Delta Y \\ M_{n-1} + \Delta M \end{cases} = \begin{cases} X_{n-1} + \dfrac{\Delta S_{\text{прав}} + \Delta S_{\text{лев}}}{2} \cos\left(M_{n-1} + \dfrac{\Delta S_{\text{прав}} - \Delta S_{\text{лев}}}{4L}\right) \\ Y_{n-1} + \dfrac{\Delta S_{\text{прав}} + \Delta S_{\text{лев}}}{2} \sin\left(M_{n-1} + \dfrac{\Delta S_{\text{прав}} - \Delta S_{\text{лев}}}{4L}\right) \\ M_{n-1} + \dfrac{\Delta S_{\text{прав}} - \Delta S_{\text{лев}}}{2L} \end{cases}$$

Итоговая программа на ЯП Python может иметь следующий вид:

```
from sys import stdin
import sys
import math

D,i,n=map(float,input().split())
n=int(n)
L=float(input())
QT=int(input())

X=0
Y=0
M=0
Nleft_old=0
Nright_old=0

for stp in range(QT):
    dt, Nleft, Nright=map(float,input().split())
    Nleft=int(Nleft)
```

```
    Nright=int(Nright)
    dSleft = math.pi * D *(Nleft - Nleft_old)/(n*i)
    dSright = math.pi * D *(Nright - Nright_old)/(n*i)
    X = X+(dSright+dSleft)/2*math.cos( M + (dSright-dSleft)/(4*L) )
    Y = Y+(dSright+dSleft)/2*math.sin( M + (dSright-dSleft)/(4*L) )
    M=M+(dSright-dSleft)/(2*L)
    Nleft_old = Nleft
    Nright_old = Nright

print(float( '{:.4f}'.format(X) ))
print(float( '{:.4f}'.format(Y) ))
```

Задача E. Движение по линии ТРИК

Задача:

В симуляторе TRIK Studio подготовить управляющую программу, перемещающую робота вдоль линии и останавливающую его на третьем встреченном перекрестке.

Управляющая программа проверяется на нескольких полях, имеющих разную конфигурацию поворотов, прямых участков и перекрестков. За каждое поле начисляется до 10% баллов. Поле засчитывается при совпадении следующих условий:

- центр робота находится в зоне 300x300 мм, центр которой совпадает с центром третьего по ходу движения робота перекрестка;
- скорости обоих моторов робота равны нулю.

Робот:

Дифференциальная платформа. Робот оснащен четырьмя датчиками отраженного света, направленными вниз:

К порту A1 подключен левый боковой датчик.

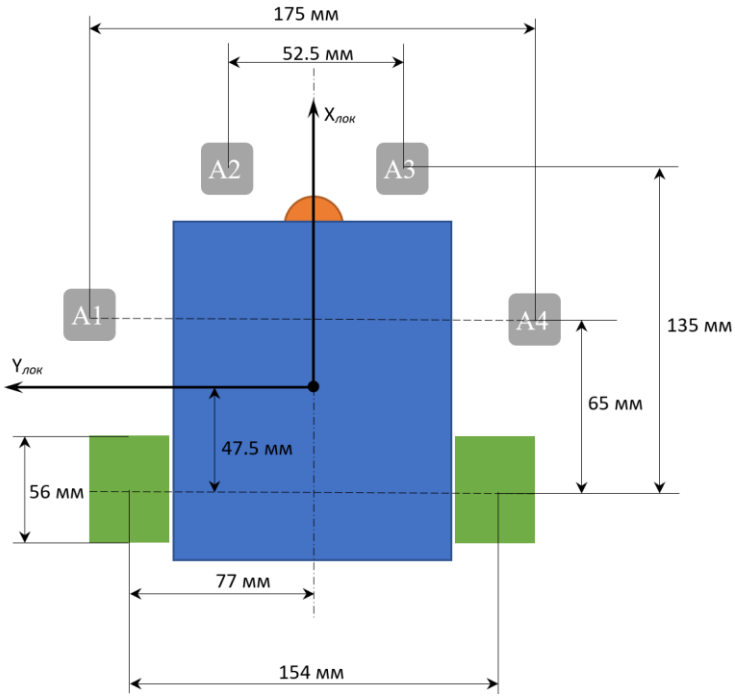
К порту A2 подключен левый передний датчик.

К порту A3 подключен правый передний датчик.

К порту A4 подключен правый боковой датчик.

Размеры модели в симуляторе TRIK Studio представлены на рисунке.

Описание робота ТРИК



Левый мотор робота подключен к порту M4, правый мотор - к порту M3.

Энкодеры робота возвращают угол поворота колес в градусах. Положительные числа обозначают вращение колеса, обеспечивающего движение робота вперед, отрицательные - назад.

Полигон:

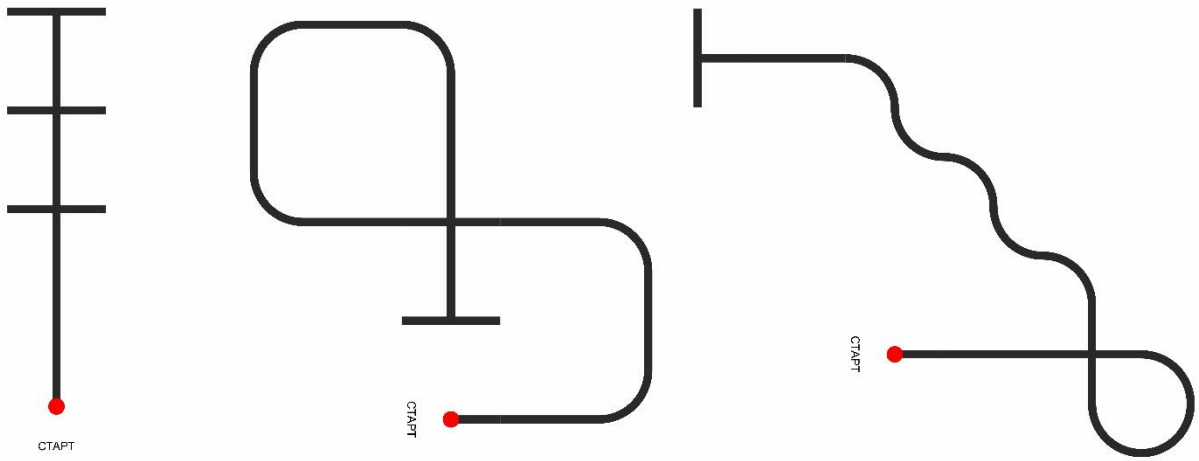
Общий размер полигона составляет 4 х 4 метра.

Стартовая секция и стартовое положение робота - точно в центре полигона.

Ширина линии не менее 25 мм.

Гарантируется, что на старте линия расположена между датчиками A2 и A3. Гарантируется. что первые 300 мм линии - прямой участок. Далее следуют повороты с радиусом не менее 150 мм. Гарантируется, что боковые линии на перекрестках отстоят от основной линии не менее чем на 125 мм и имеют ширину не менее 25 мм.

Пример полей:



Примеры заданий и тестовый проект:

В приложенном архиве находится следующие файлы:

- *Test_field_1_exercise.qrs* / *Test_field_2_exercise.qrs* / *Test_field_3_exercise.qrs* – файлы-упражнения с настроенными тренировочными полями;
- *task_7.py* - файл-программа с исходным кодом участника.
- Файлы-упражнения могут использоваться для отладки программ. После отладки код программы необходимо перенести в файл *task_7.py* и отправить в качестве решения задачи.

Данные: solutions.zip

Решение:

См. решение задачи G «Одометрия дифф. платформы ТРИК» для возрастной группы 9–11 класс, оно подходит.

Задача F. Граф

Ограничения: по времени 3 сек, по памяти 256 Мб.

Задание:

Вам дан неориентированный граф с n вершинами и m ребрами. Вершины пронумерованы от 1 до n.

Вам надо найти количество кратчайших путей из 1 до n. Так как ответ может быть большим, выведите его по модулю $10^9 + 7$.

Система оценки

Каждая группа тестов будет оцениваться отдельно и баллы начисляются в случае, если все тесты группы пройдены. Все тесты разбиты на группы со следующими ограничениями:

Подзадача	Ограничения	Баллы	Необходимые подзадачи
1	*$1 \le n \le 10$*	*30*	–
2	*$1 \le n \le 1000$*	*30*	1
3	–	*40*	1, 2

Формат входных данных

В первой строке записано два целых числа n и m ($2 \le n \le 2*10^5$ и $0 \le m \le 2*10^5$).

Следующие m строк содержат описание рёбер по одному на строке. Ребро номер i описывается двумя натуральными числами b_i, e_i – номерами концов ребра ($1 \le b_i, e_i \le n$).

Формат выходных данных

Первая строка выходного файла должна содержать одно натуральное число – количество кратчайших путей из 1 до n. Если нет пути из 1 в n, то выведите 0.

Пример 1

Входные данные:	Выходные данные:
3 1 2 1 2 3	0

Пример 2

Входные данные:	Выходные данные:
2 1 1 2	1

Пример 3

Входные данные:	Выходные данные:
4 3 1 3 2 3 2 4	1

Пример 4

Входные данные:	Выходные данные:
7 8 1 3 1 4 2 3 2 4 2 5 2 6 5 7 6 7	4

Решение:

Для подсчета кратчайшего расстояния от вершины 1 до вершины *n*, воспользуемся алгоритмом BFS (алгоритм обхода в ширину) за время $O(N + M)$.

Во время работы алгоритма BFS будем вычислять с помощью динамического программирования количество кратчайших путей.

Когда находим кратчайшее расстояние, мы заполняем массив кратчайших расстояний *dist* следующим образом:

Для каждой вершины *t*, у которой есть ребро от *v*:

• если мы еще не посещали *t*, обновляем *dist[t]* на *dist[v]* + 1;

• в противном случае ничего не делаем.

Одновременно будем обновлять массив количества кратчайших путей *dp* следующим образом:

Для каждой вершины *t*, у которой есть ребро от *v*:

• если мы еще не посещали *t*, обновляем *dist[t]* на *dist[v]* + 1, и к *dp[t]* присваиваем *dp[v]*;

• если мы посещали *t* и *dist[t]* == *dist[v]* + 1, то к *dp[t]* добавляем *dp[v]*;

• в противном случае ничего не делаем.

Задача G. Одометрия дифф. платформы ТРИК

Задача:

В симуляторе TRIK Studio подготовить управляющую программу, перемещающую робота вдоль линии и останавливающую его на третьем встреченном перекрестке, посчитав при этом финишные координаты робота.

Управляющая программа проверяется на нескольких полях, имеющих разную конфигурацию поворотов, прямых участков и перекрестков. За каждое поле начисляется до 10% баллов. Поле засчитывается при совпадении следующих условий:

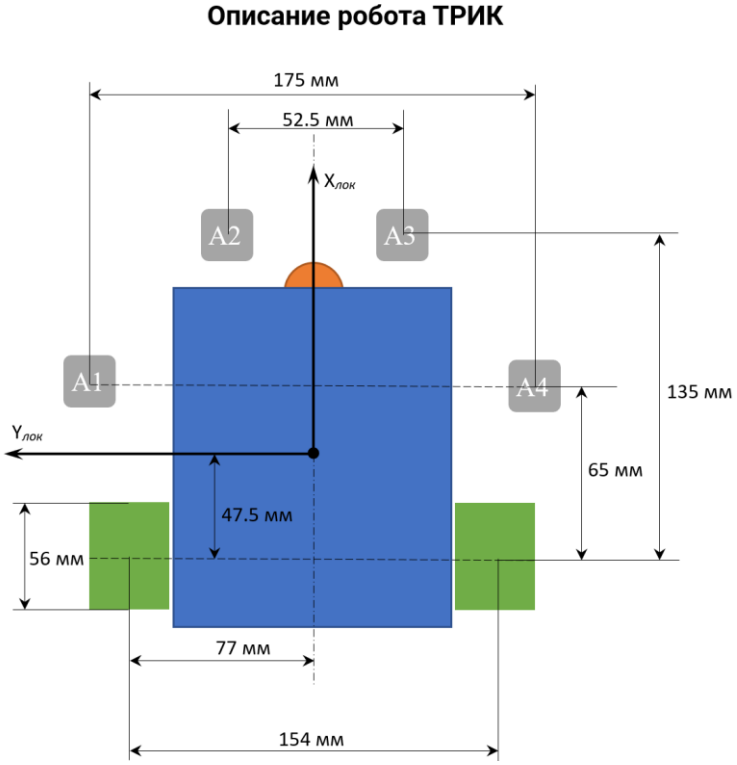
5 баллов начисляются при совпадении условий:

- центр робота находится в зоне 300x300 мм, центр которой совпадает с центром третьего по ходу движения робота перекрестка;
- скорости обоих моторов робота равны нулю.

5 баллов начисляются при совпадении условий:

- перед завершением программы робот вывел на экран не менее двух сообщений в разных строках,
- первое и последнее выведенные на экран робота сообщения содержат целые числа,
- первое выведенное число отличается от эталонной координаты X финишной зоны не более чем на 100 (ошибка вычисления координаты X не превышает 100 мм),
- последнее выведенное число отличается от эталонной координаты Y финишной зоны не более чем на 100 (ошибка вычисления координаты Y не превышает 100 мм).

Робот:



Дифференциальная платформа. Робот оснащен четырьмя датчиками отраженного света, направленными вниз:

К порту A1 подключен левый боковой датчик.

К порту A2 подключен левый передний датчик.

К порту A3 подключен правый передний датчик.

К порту A4 подключен правый боковой датчик.

Размеры модели в симуляторе TRIK Studio представлены на рисунке.

Левый мотор робота подключен к порту M4, правый мотор - к порту M3.

Энкодеры робота возвращают угол поворота колес в градусах. Положительные числа обозначают вращение колеса, обеспечивающего движение робота вперед, отрицательные - назад.

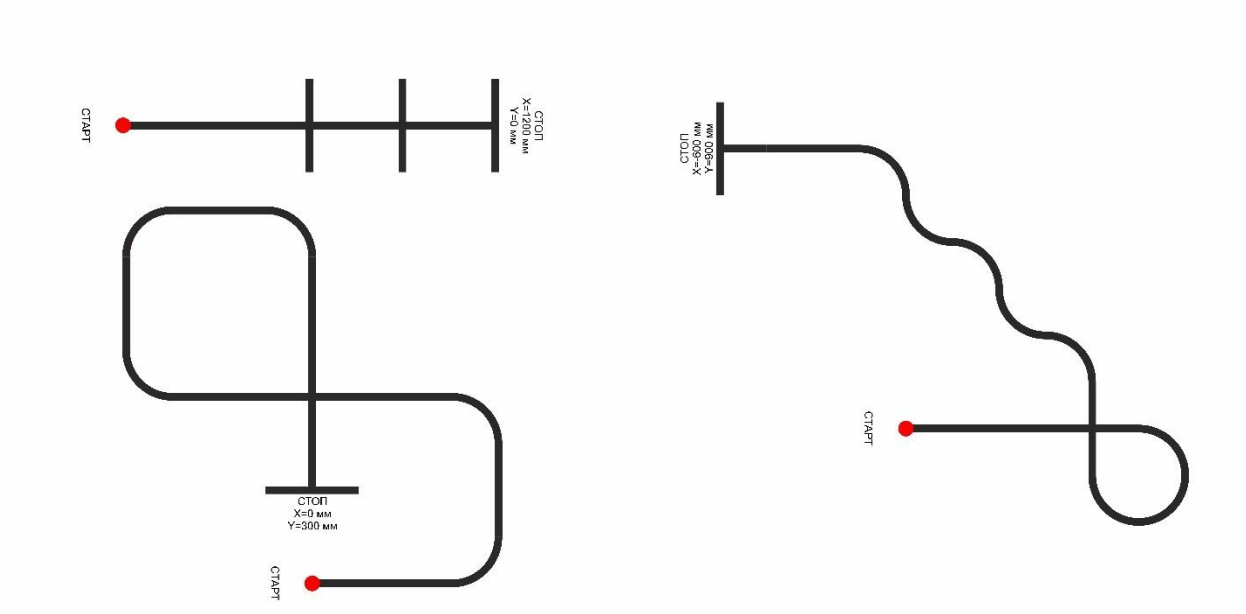
Полигон:

Общий размер полигона составляет 4 х 4 метра.

Стартовая секция и стартовое положение робота - точно в центре полигона.

Ширина линии не менее 25 мм.

Гарантируется, что на старте линия расположена между датчиками A2 и A3. Гарантируется. что первые 300 мм линии - прямой участок. Далее следуют повороты с радиусом не менее 150 мм. Гарантируется, что боковые линии на перекрестках отстоят от основной линии не менее чем на 125 мм и имеют ширину не менее 25 мм.



Примеры заданий и тестовый проект:

В приложенном архиве находится следующие файлы:

- *Test_field_1_exercise.qrs* / *Test_field_2_exercise.qrs* / *Test_field_3_exercise.qrs* – файлы упражнения с настроенными тренировочными полями;
- *task_15.py* – файл программа с исходным кодом участника.

Файлы-упражнения могут использоваться для отладки программ. После отладки код программы необходимо перенести в файл *task_15.py* и отправить в качестве решения задачи.

Данные: solutions.zip

Решение:

Движение по линии может быть реализовано с помощью алгоритма ПИД-регулятора. Для текущей задачи алгоритм можно редуцировать до ПИ- или ПД-регулятора, снизив общую скорость движения робота.

Для вычисления одометрии используются упрощенные формулы из задачи D «Одометрия дифф. платформы».

Итоговая программа на ЯП Python может иметь следующий вид:

```
import sys
import time
import random
import math

class Program():
  __interpretation_started_timestamp__ = time.time() * 1000
  pi = 3.141592653589793
  def execMain(self):
    Diam=0.056 #диаметр колес робота, в метрах
    L=0.077 #полуколея робота, в метрах
    X=0.0 #координата X робота, в метрах
    Y=0.0 #координата Y робота, в метрах
    kf=0.9
    M=0
    kp=1.5
    ki=0
    kd=0
    err=0
    errold=0
    I=0
    n=0
    leftSpd = 0
    rightSpd= 0
    Nleft_old=0
    Nright_old=0
    while n<3:
      s1 = brick.sensor("A1").read()
      s2 = brick.sensor("A2").read()
      s3 = brick.sensor("A3").read()
      s4 = brick.sensor("A4").read()
      s1f = s1
      s4f = s4
      while (s1+s4)<50:
        s1 = brick.sensor("A1").read()
        s2 = brick.sensor("A2").read()
        s3 = brick.sensor("A3").read()
        s4 = brick.sensor("A4").read()
        s1f = kf*s1f+(1-kf)*s1
        s4f = kf*s4f+(1-kf)*s4
        err = s2 - s3
        P=kp*err
        I=I+ki*err
        D=kd*(err-errold)
        U=P+I+D
        leftSpd = 50-U
```

```
            rightSpd= 50+U
            brick.motor("M3").setPower(rightSpd)
            brick.motor("M4").setPower(leftSpd)
            errold = err
            Nleft=brick.encoder("M4").read()
            Nright=brick.encoder("M3").read()
            dSleft = math.pi * Diam *(Nleft - Nleft_old)/360
            dSright = math.pi * Diam *(Nright - Nright_old)/360
            X = X+(dSright+dSleft)/2*math.cos( M + (dSright-dSleft)/(4*L) )
            Y = Y+(dSright+dSleft)/2*math.sin( M + (dSright-dSleft)/(4*L) )
            M=M+(dSright-dSleft)/(2*L)
            Nleft_old = Nleft
            Nright_old = Nright
            script.wait(10)
          n=n+1
          startTime = script.time()
          brick.playTone(200, 300)
          dt = 0
          while dt<800:
            s1 = brick.sensor("A1").read()
            s2 = brick.sensor("A2").read()
            s3 = brick.sensor("A3").read()
            s4 = brick.sensor("A4").read()
            s1f = kf*s1f+(1-kf)*s1
            s4f = kf*s4f+(1-kf)*s4
            err = s2 - s3
            P=kp*err
            I=I+ki*err
            D=kd*(err-errold)
            U=P+I+D
            leftSpd = 40-U
            rightSpd= 40+U
            brick.motor("M3").setPower(rightSpd)
            brick.motor("M4").setPower(leftSpd)
            errold = err
            Nleft=brick.encoder("M4").read()
            Nright=brick.encoder("M3").read()
            dSleft = math.pi * Diam *(Nleft - Nleft_old)/360
            dSright = math.pi * Diam *(Nright - Nright_old)/360
            X = X+(dSright+dSleft)/2*math.cos( M + (dSright-dSleft)/(4*L) )
            Y = Y+(dSright+dSleft)/2*math.sin( M + (dSright-dSleft)/(4*L) )
            M=M+(dSright-dSleft)/(2*L)
            Nleft_old = Nleft
            Nright_old = Nright
            script.wait(10)
            dt=script.time()-startTime
        X=int(X*1000) #перевод координаты в целое число, в миллиметрах
        Y=int(Y*1000) #перевод координаты в целое число, в миллиметрах
        brick.display().addLabel(X, 1, 1, 20) #вывод в первую строку координаты X
        brick.display().redraw()
        brick.display().addLabel(Y, 1, 25,20)#вывод во вторую строку координаты Y
        brick.display().redraw()
        brick.motor("M3").powerOff()
        brick.motor("M4").powerOff()
        script.wait(10)
        brick.stop()
        return

def main():
  program = Program()
  program.execMain()

if __name__ == '__main__':
  main()
```

Задача H. Фигуры

Ограничения: по времени 8 сек, по памяти 256 Мб.

Задание:

Камера робота сделала один черно-белый кадр размером W x H. На кадре зафиксировано от 1 до 30 объектов разных форм (круг, треугольник, квадрат). Необходимо определить количество квадратов, стороны которых параллельны осям координат, повернутых квадратов, кругов и треугольников.

Система оценки

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Баллы	Ограничения	Необх. подзадачи
0	0	Тесты из условия	–
1	10	только квадраты, стороны которых параллельны осям координат	–
2	20	только квадраты (могут быть повернуты)	–
3	10	только круги	–
4	15	только треугольники	–
5	45	"–	1-4

Формат входных данных

В первой строке входных данных содержит два натуральных числа W и H ($30 \le W, H \le 500$) – размеры кадра.

В следующих H строках содержится по W символов $a_{i,j} \in \{0, 1\}$ – если 0, то фон, 1 объект.

Гарантируется, что объект состоит не менее чем из 25 единиц.

Формат выходных данных

Выведите через пробел 4 числа – количество квадратов, стороны которых параллельны осям координат, повернутых квадратов, кругов и треугольников.

Пример 1

Входные данные:

40 40
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000100000000000000000000000000000
0000000001100000000000000000000000000000
0000000011110000000000000000000010000000
0000000111110000000000000000000011000000
0000001111111000000000000000000111100000
0000011111111000000000000000001111111000
0000000000001000000000000000001111111100
0000000000000000000000000000011111111111
0000000000000000000000000000111111111110
0000000000000000000000000001111111111100
0000000000000000000000000011111111111000
0000000000000000000000000001111111111000
0000000000000000001000000000011110000000
0000000000000000111111100000000001100000
0000000000000011111111111000000000100000
0000000000000111111111111100000000000000
0000000000000111111111111100000000000000
0000000000001111111111111100000000000000
0000000000001111111111111100000000000000
0000000000001111111111111100000000000000
0000000000001111111111111100000000000000
0000000000001111111111111100000000000000
0000000000001111111111111100000000000000
0000111110000111111111111100000000000000
0000111110000111111111111100000000000000
0000111110000111111111111100000000000000
0000111110000000111111100000000000000000
0000111110000000001000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000

Выходные данные:

1 1 1 1

Решение:

Разделим решение на два этапа:

1. Выделение объектов: для выделения объектов можно использовать рекурсивный обход (DFS) или BFS. Будем находить первый символ (который не был выделен) и запустим от этого символа DFS. Во время обработки выпишем позиции всех клеток этого объекта для того, чтобы создать дополнительную матрицу, только с этим объектом для дальнейшей обработки.

2. Классификация объектов:

• Для проверки на не повернутый квадрат достаточно проверить, что дополнительная матрица является квадратной и нет клеток фона.

• Для проверки на круг восстановим уравнение окружности, используя координаты центра (x_c, y_c) и радиус r, проверим, что все неповернутые точки удовлетворяют неравенству

$$(x - x_c)^2 + (y - y_c)^2 \leq r^2$$

• Для проверки на повернутый квадрат посчитаем количество компонент, состоящих только из 0. Их должно быть строго 4 (смотрите рисунок), и объект не должен удовлетворять условию, что не выполняется условие окружности.

• Для проверки на треугольник не должны выполняться условия для квадрата и окружности.