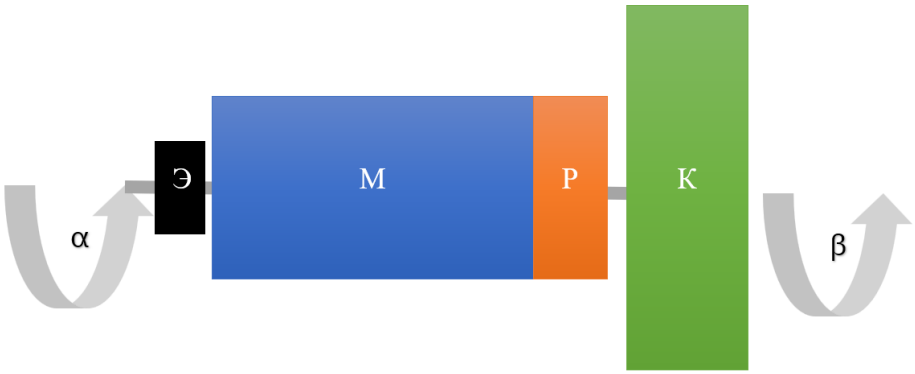
Возрастная категория 7–8 классы

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

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

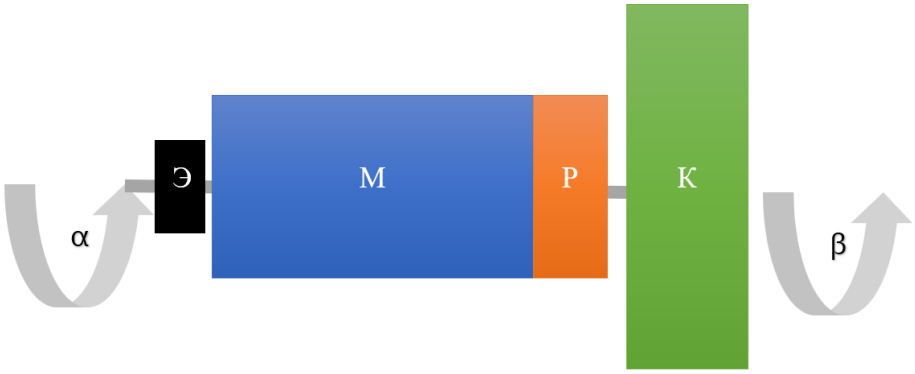
Считывание входных данных, вычисление и вывод данных могут быть реализованы следующей программой на ЯП Python:

```
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:

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

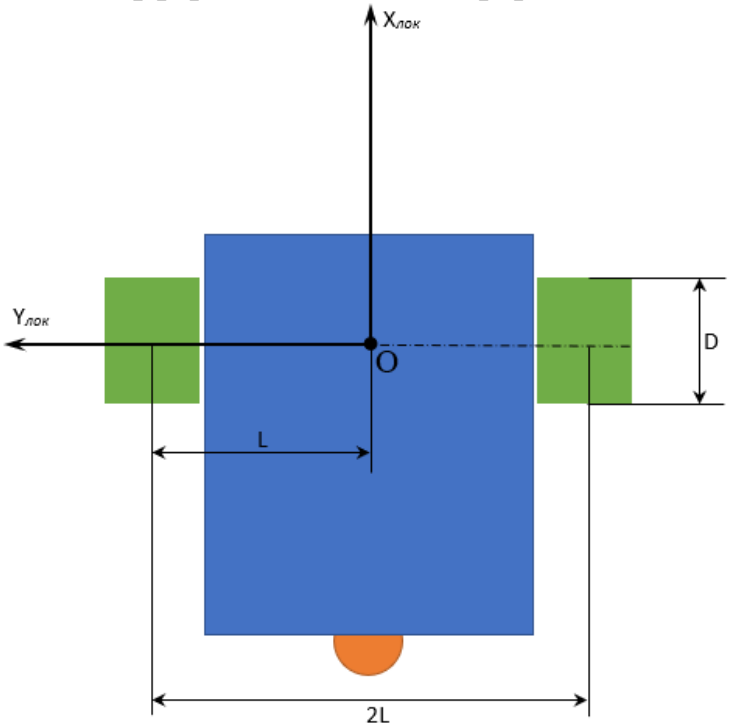
Считывание входных данных, вычисление и вывод данных могут быть реализованы следующей программой на ЯП Python:

```
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.

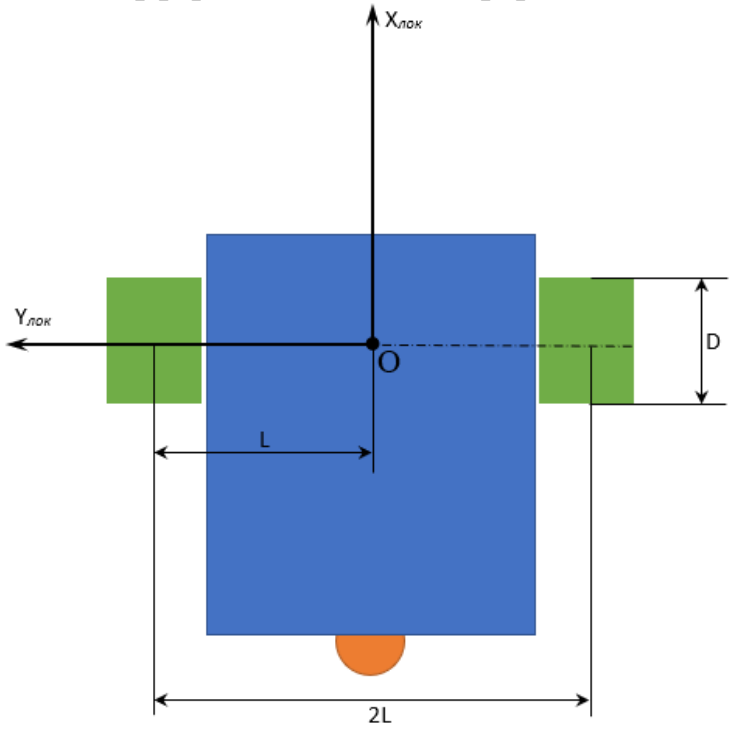
Итоговая программа на ЯП Python может иметь следующий вид:

```
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.

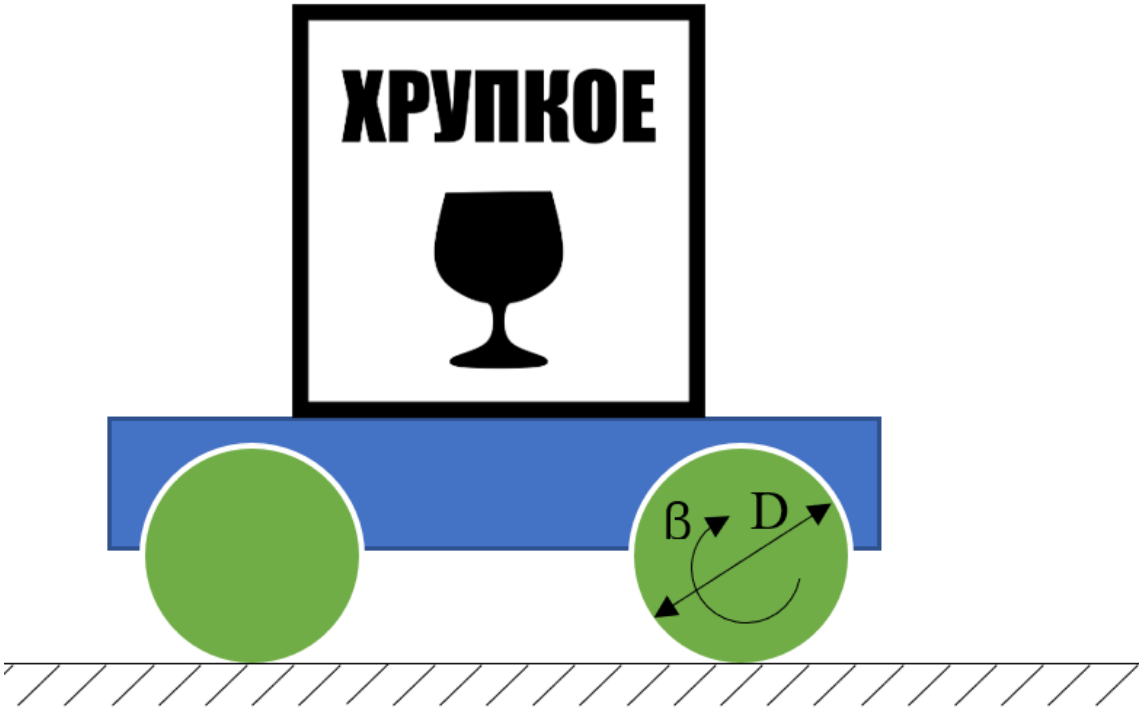
Итоговая программа на ЯП Python может иметь следующий вид:

```
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.

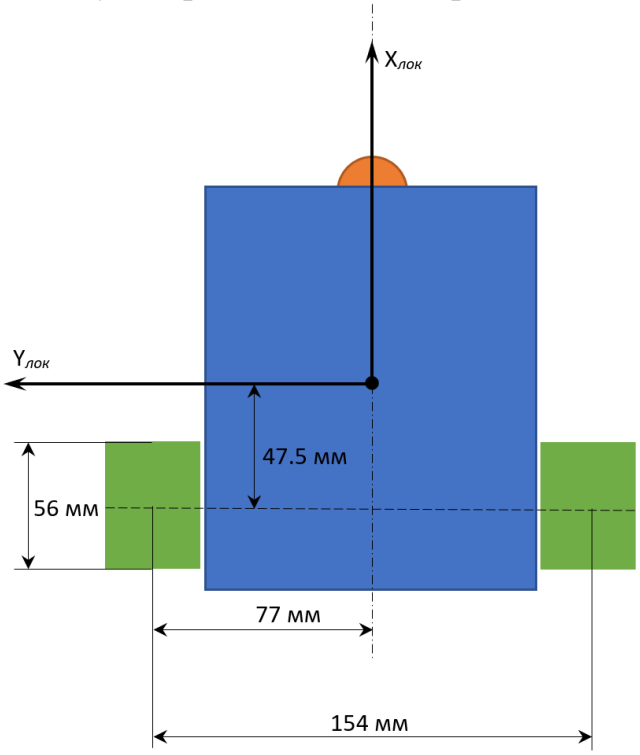
Управляющая программа проверяется на нескольких полях, имеющих разные входные данные. За каждое поле начисляется от 4% до 6% баллов, в зависимости от сложности. Поле засчитывается при совпадении следующих условий:

- 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.

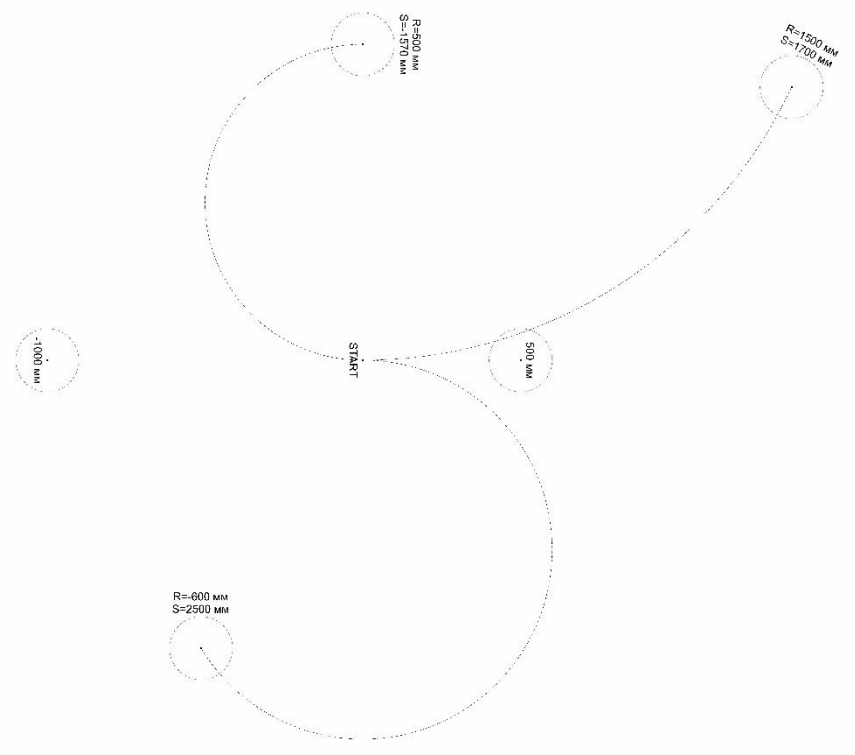
Энкодеры робота возвращают угол поворота колес в градусах. Положительные числа обозначают вращение колеса, обеспечивающего движение робота вперед, отрицательные - назад.

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.

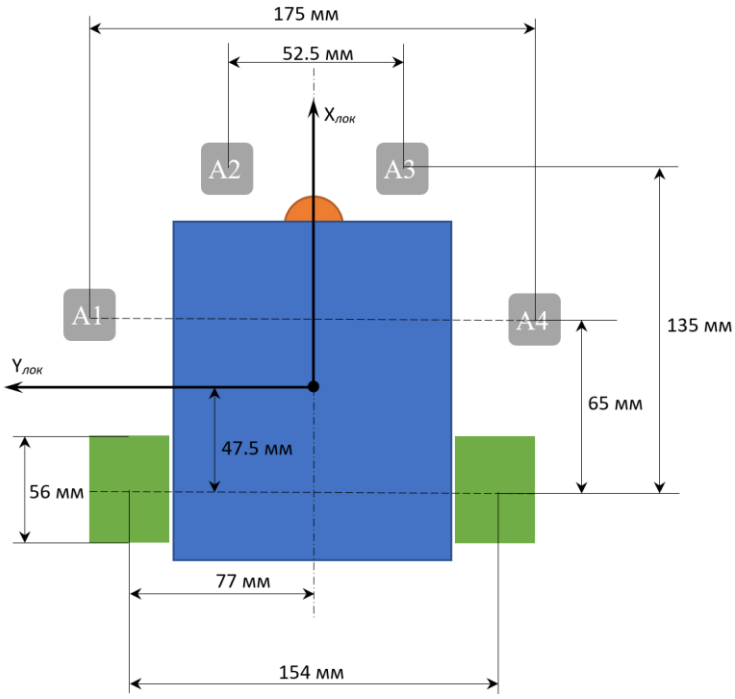
К порту A2 подключен левый передний датчик.

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:

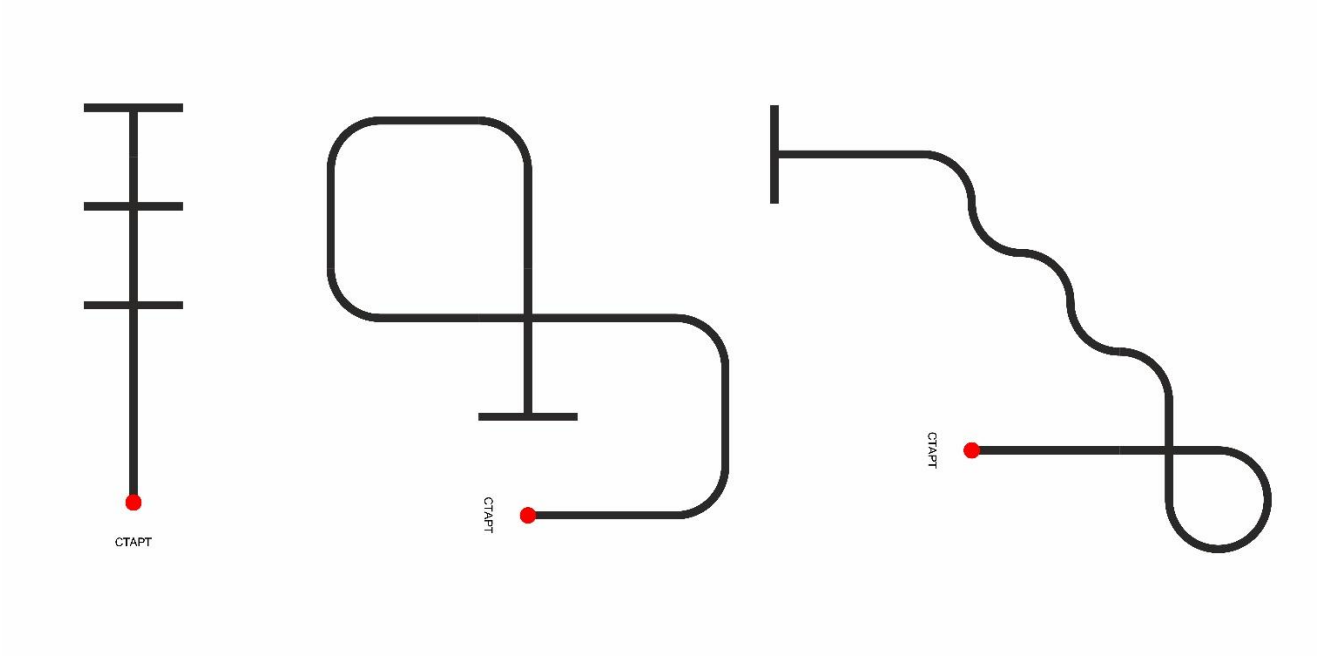
Общий размер полигона составляет 4 х 4 метра.

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.