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 УЧЕБНО-ТЕХНОЛОГИЧЕСКОЙ ПРАКТИКЕ

Студент Боренко Анастасия Денисовна
фамилия, имя, отчество

Группа ИУ7-12Б

Тип практики Распределенная практика

Название предприятия МГТУ имени Н. Э. Баумана

Студент _____ Боренко А.Д.
подпись, дата *фамилия, и.о.*

Руководитель практики _____ Борисов С. В.
подпись, дата *фамилия, и.о.*

Оценка _____

2019 г.

Оглавление

Введение

Исследование реабилитационного оборудования

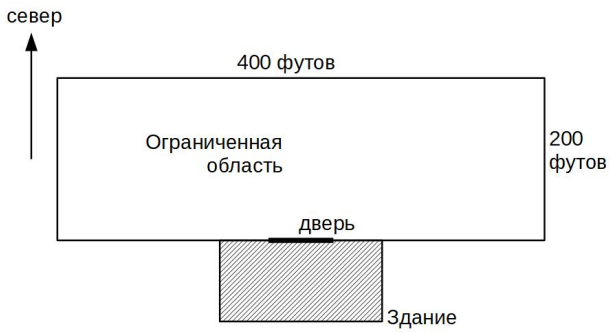
Цели и задачи:

Из полученных данных о передвижении кресла, получить весь путь пройденный креслом, либо координату точки, в которой пациент покинул поле в первый раз, и время, когда это произошло, либо если пациент не покидал поле, то расстояние от наиболее удаленной точки до двери

Условие задачи

Управление движением реабилитационного оборудования

Исследователь области реабилитационного оборудования занимается изучением использования пациентами инвалидного кресла, снабженного мотором, которое он проводит на ограниченном участке. Мотор этого кресла соединен с осью при помощи цепного привода. Поэтому оба колеса крутятся с одинаковой скоростью, что позволяет креслу передвигаться только строго по прямой. Больной может останавливать коляску, поворачивать колеса и таким образом может менять направление только тогда, когда кресло неподвижно. Для того чтобы облегчить контроль за его использованием, кресло снабжено компасом, часами и спидометром. Записывающее устройство фиксирует каждый раз интервал времени, пока кресло находится в движении, среднюю скорость и азимут на протяжении этого временного интервала. Компас представляет собой стандартный компас, у которого 0o соответствует северному направлению, а 90o — восточному и т.д.



На рисунке представлена карта ограниченной области. Этот участок представляет собой газон прямоугольной формы размером 200 футов на 400 футов. Пациент попадает в ограниченную область через дверь здания, расположенного в южной части этой области. Дверь находится в центре четырехсотфутовой южной границы так как показано на чертеже.

Записывающее устройство включается автоматически и фиксирует все передвижения пациента приблизительно в течение часа, когда пациент через дверь попадает в ограниченную область. Время измеряется в секундах от 0 до 3600, где 0 — начало отсчета времени — стартовый момент времени, когда

пациент пересекает границу этой области в районе двери. Устройство записывает показания в виде четырех чисел описывающих движение кресла в течение каждого интервала времени, когда мотор находится в действии. Первые два числа — это время начала и конца движения; третье число показывает скорость в течение этого временного периода, а четвертое число показывает азимут компаса в течение этого интервала.

(В продолжение каждого интервала времени кресло сохраняет постоянные скорость и направление.)

Например, зафиксированная строка показаний:

10.6 15.9 2.8 274

означает, что между t1= 10.6 и t2=15.9 секунд кресло двигалось со скоростью 2.8 футов в секунду и с азимутом 274°. Время регистрируется каждые 0.1 секунды, скорость — каждые 0.1 футов в секунду, а азимут — по всему диапазону шкалы компаса.

Ваша задача состоит в том, чтобы проанализировать информацию, полученную от записывающего устройства кресла. Особенно подробно вы должны определить следующее:

1) Покидал ли пациент ограниченную область? Если да, то определите время, когда пациент покинул область в первый раз и также определите, в какой точке периметра ограниченной области кресло пересекло ее границы. Если же пациент не покидал ограниченный участок, то каково было расстояние от двери до наиболее удаленной точки, которой достиг пациент, не выходя за пределы этого участка.

2) Какая общая дистанция, которую преодолел пациент?

Для того, чтобы ответить на поставленные вопросы, используйте координаты, определяющие местоположение точки (0,0), которые соответствуют юго-западному углу ограниченного участка, и координаты (400,200) соответствующие северо-восточному углу. В тот момент, когда записывающее устройство начинает работать, когда пациент проходит через

дверь, положение пациента в момент времени t=0.0 всегда соответствует координатам (200,0). Когда он попадает в ограниченную область, то двигается на сервер.

Входные данные

Входные данные состоят из нескольких блоков информации. Первая строка каждого блока представляет собой целое число, соответствующее количеству строк показаний, записанных устройством в течение определенного временного интервала. Конец информации соответствует тому блоку, у которого в первой строке стоит цифра 0.

Выходные данные

Вывод информации для каждого блока начинается с идентификации набора данных. Форма входных данных показывает, пересекал ли пациент границы участка, и, если да, то еще и время и место по примеру. Если нет, то должно быть представлено максимальное расстояние, которое пациент преодолел. Выходные данные следует оформить так, чтобы одинаковая маркированная информация включала в себя все параметры, указанные в примере; блоки данных отделять друг от друга с помощью звездочек.

Пример входных данных

```
4
0.0 5.0 3.0 0
7.0 9.0 2.0 30
10.0 100.0 4.0 60
110.0 200.0 2.0 0
3
0.0 20.0 2.0 0
500.0 600.0 1.0 270
3000.0 3100.0 1.0 0
7
0.0 5.3 2.1 0
```

19.8 35.6 2.7 346

42.0 78.4 2.3 15

1181.4 1192.1 1.7 117

2107.0 2193.6 2.1 295

2196.3 2201.2 2.0 298

2704.3 2709.2 1.5 208

Пример выходных данных

Первый случай

Пациент покинул ограниченный участок в точке (400.0,132.8) в момент времени 67.2 секунд.

Общая пройденная дистанция составляет 559.0 футов.

★★

Второй случай

Пациент не пересекал границы области

Максимальное расстояние, на которое пациент удалился от двери, равно 172.0 футов

Общая пройденная дистанция составляет 240.0 футов

★★

Третий случай

Пациент покинул ограниченный участок в точке (67.0,200.0) в момент времени 2191.4 секунд

Общая пройденная дистанция составляет 354.7 футов

★★

Допущения и требования:

1 В пределах каждого блока информации должны быть перечисленны в хронографическом порядке; аервый временной интервал всегда содержит время 0.0, соответствующее моменту времени, когда пациент пересекает границу области в районе двери. Все показания времени будут даны с точностью до

одного знака после запятой и будут находиться в интервале от 0.0 3600.0 включительно. Продолжительность каждого определенного временного интервала положительна, то есть конечное значение всегда больше, чем начальное.

2 Величина скорости принимает значение от 0.1 до 9.9 футов в секунду

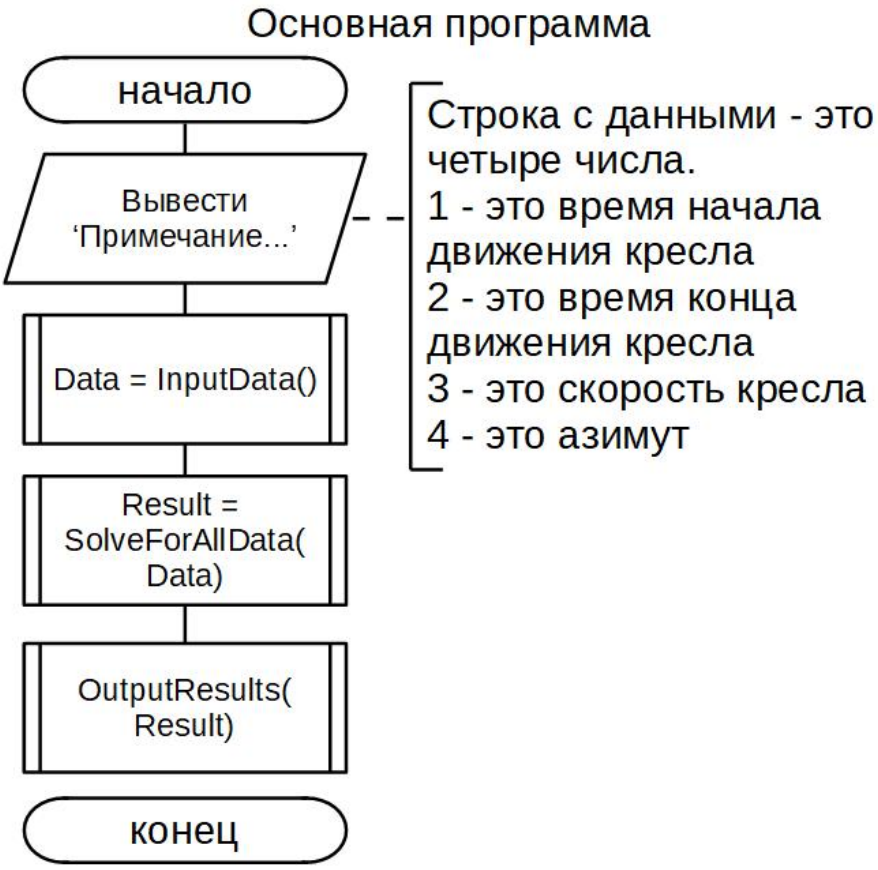
3 Величина азимута может варьироваться в пределах всего диапазона шкалы компаса от 0о до 359о включительно. Начальный азимут для первой строки данных в каждом блоке информации будет равен 0

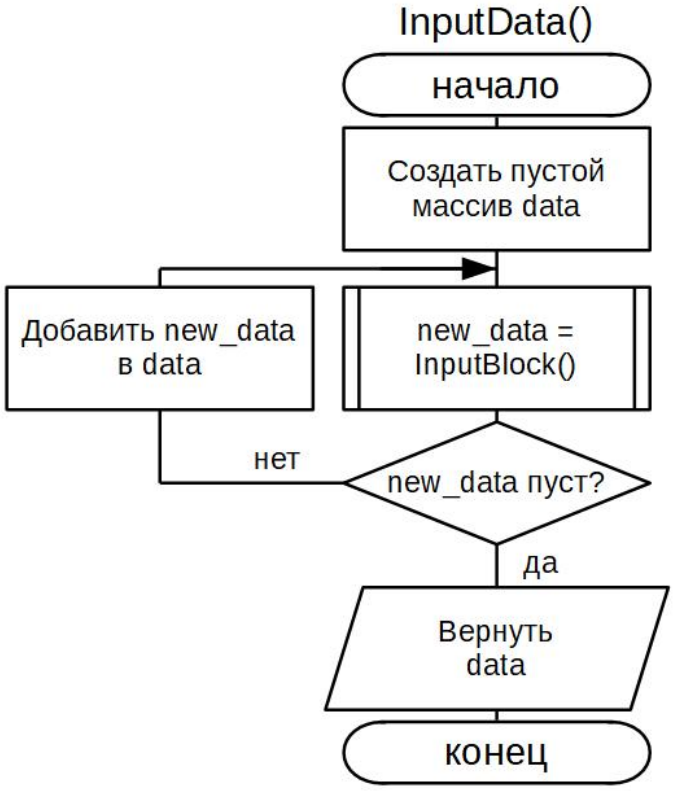
4 В каждой строке данных расстояние между числами должно быть по крайней мере один пробел.

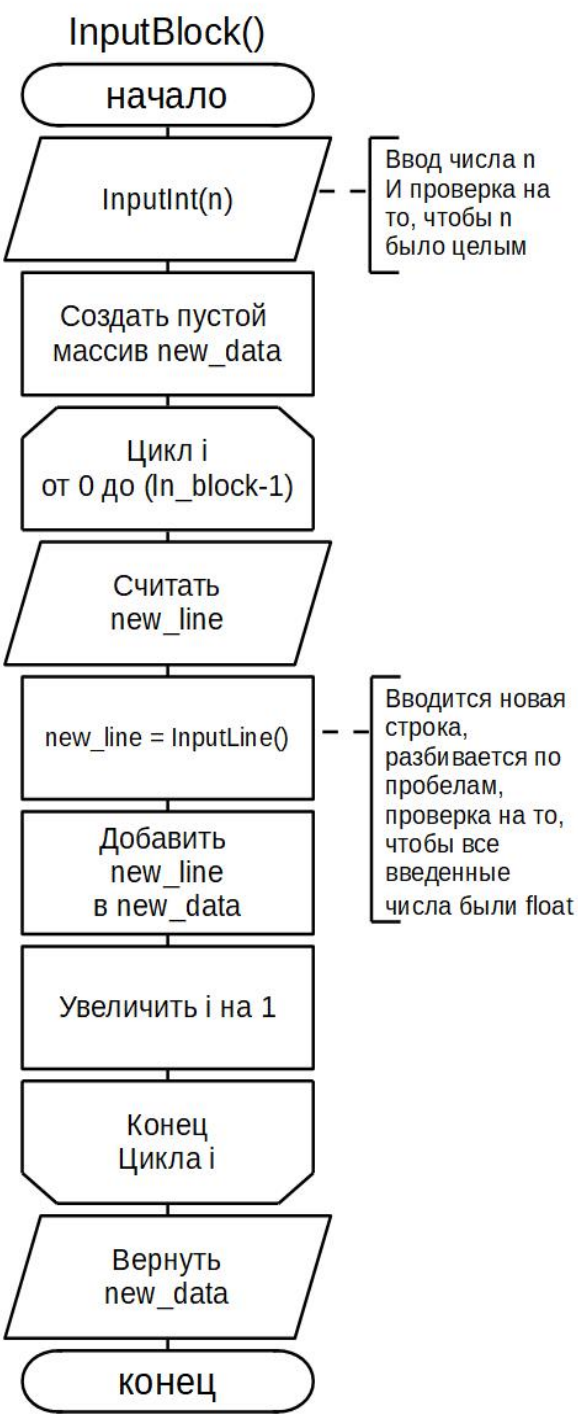
5 Все численные результаты должны быть выведенны с точностью до одного знака после запятой так, как показано в примере.

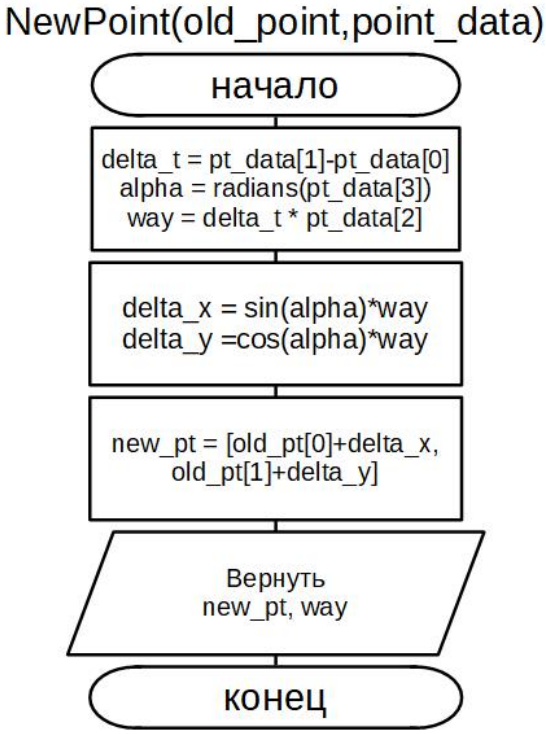
6 Если пациент выходит за пределы ограниченной области, его местоположение может характеризоваться отрицательными координатами. Тем не менее, Вы не должны беспокоиться, о том, что кресло может врезаться в стены здания.

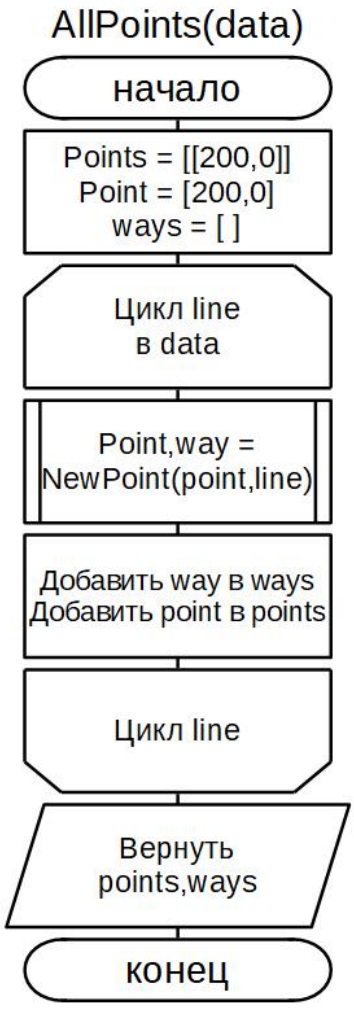
Схемы программы



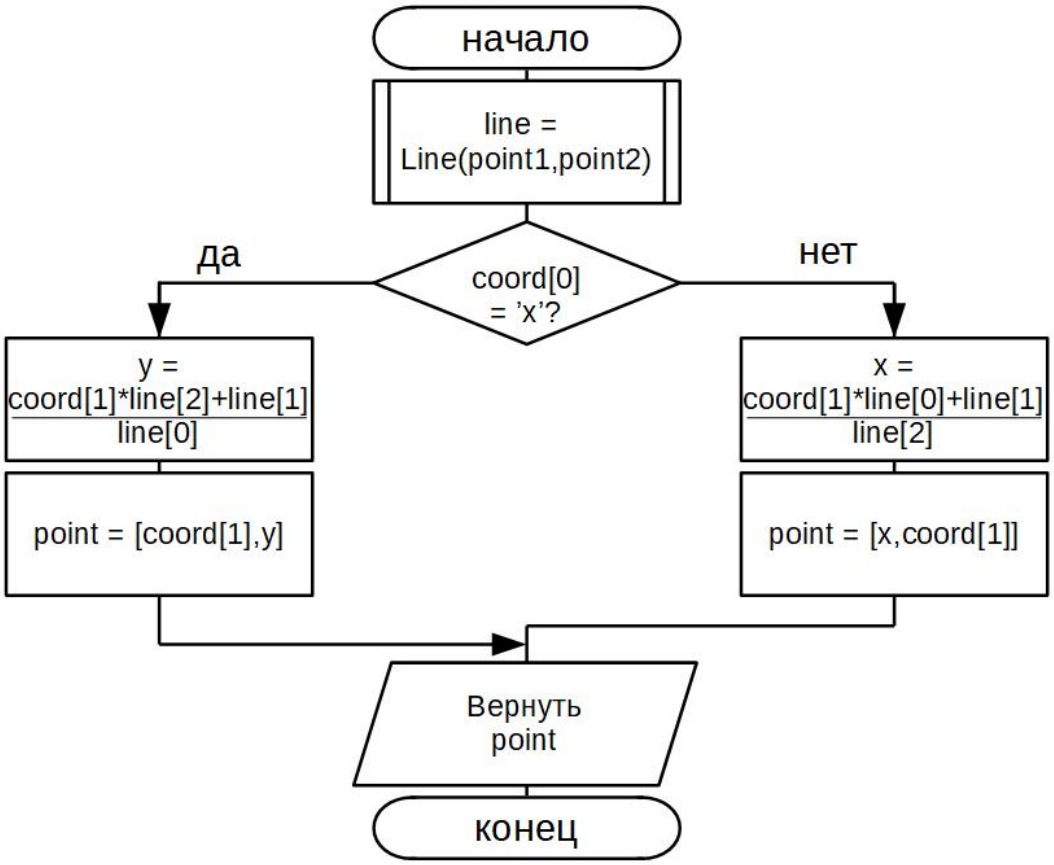



InputBlock()
начало
InputInt(n)
Ввод числа n
И проверка на то, чтобы n было целым
Создать пустой массив new_data
Цикл i от 0 до (ln_block-1)
Считать new_line
new_line = InputLine()
Вводится новая строка, разбивается по пробелам, проверка на то, чтобы все введенные числа были float
Добавить new_line в new_data
Увеличить i на 1
Конец Цикла i
Вернуть new_data
конец

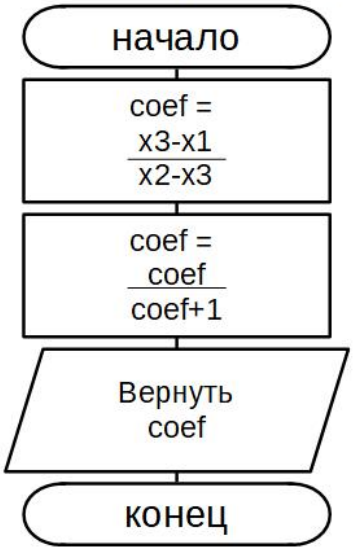

NewPoint(old_point,point_data)
начало
delta_t = pt_data[1]-pt_data[0]
alpha = radians(pt_data[3])
way = delta_t * pt_data[2]
delta_x = sin(alpha)*way
delta_y =cos(alpha)*way
new_pt = [old_pt[0]+delta_x, old_pt[1]+delta_y]
Вернуть new_pt, way
конец


AllPoints(data)
начало
Points = [[200,0]]
Point = [200,0]
ways = [ ]
Цикл line в data
Point,way = NewPoint(point,line)
Добавить way в ways
Добавить point в points
Цикл line
Вернуть points,ways
конец

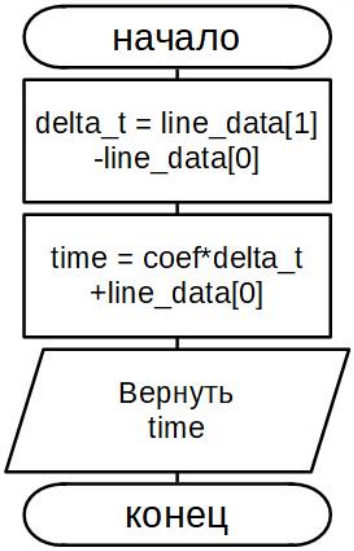
FoundIntersect(point1,point2,coord)

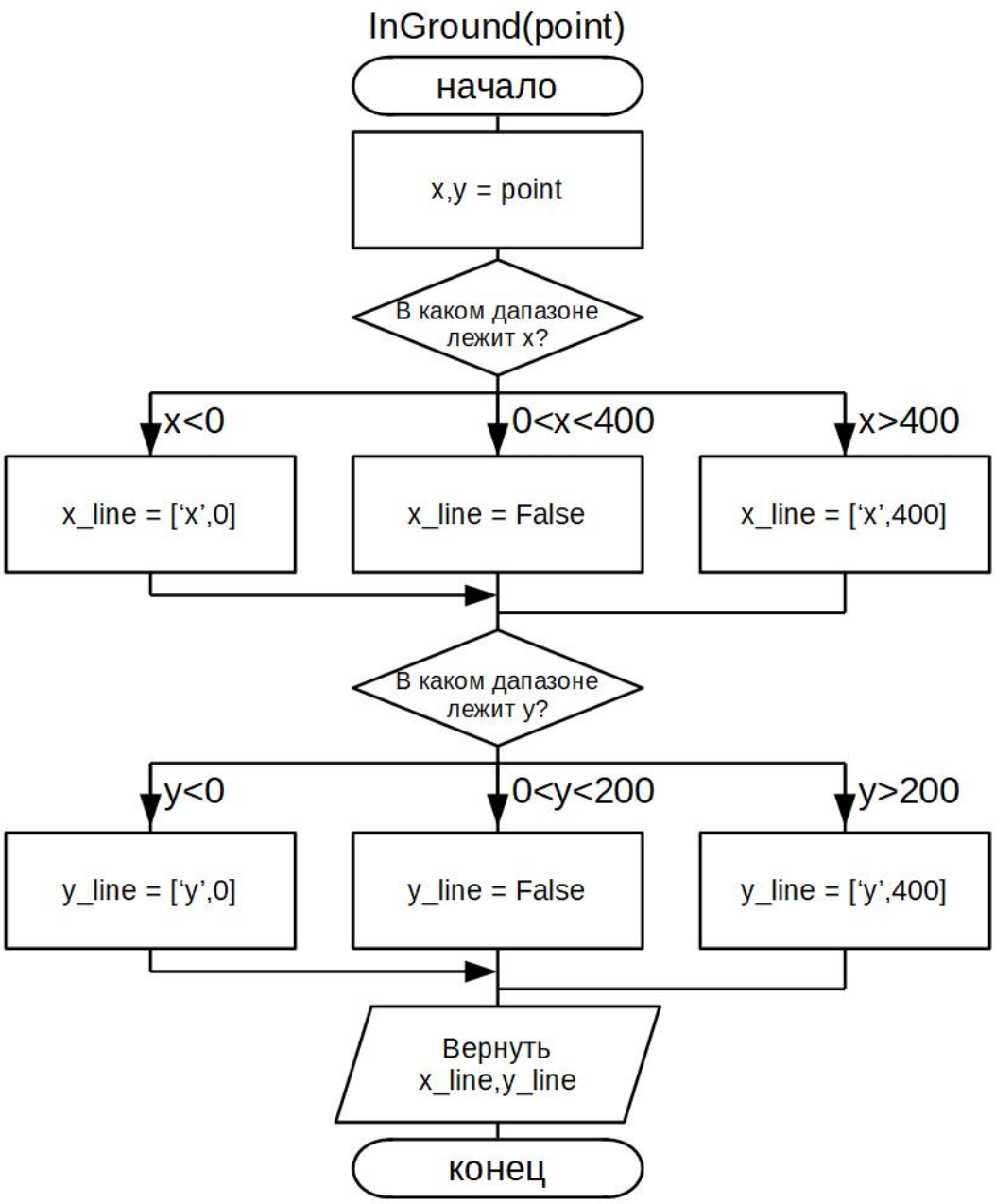


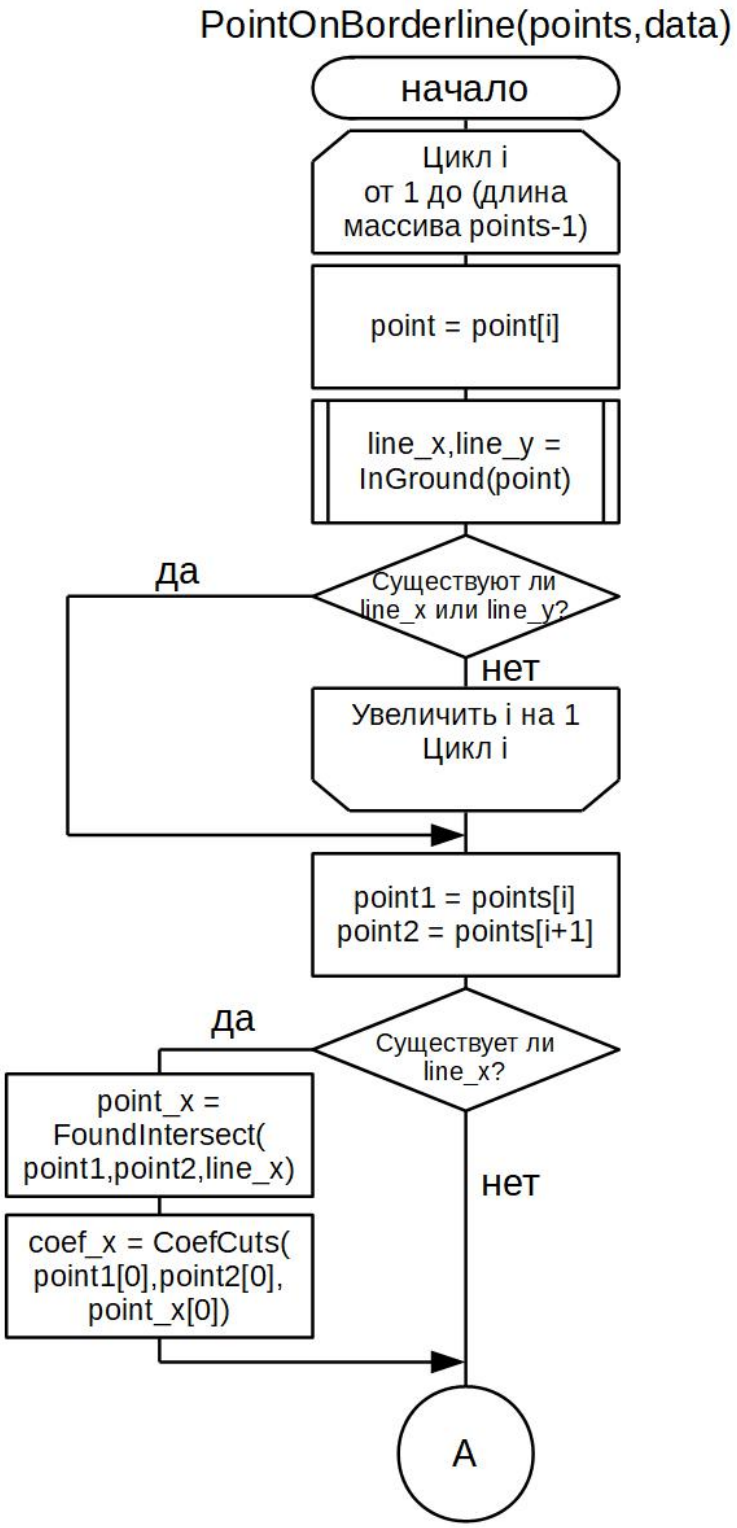
CoefCuts(x1,x2,x3)

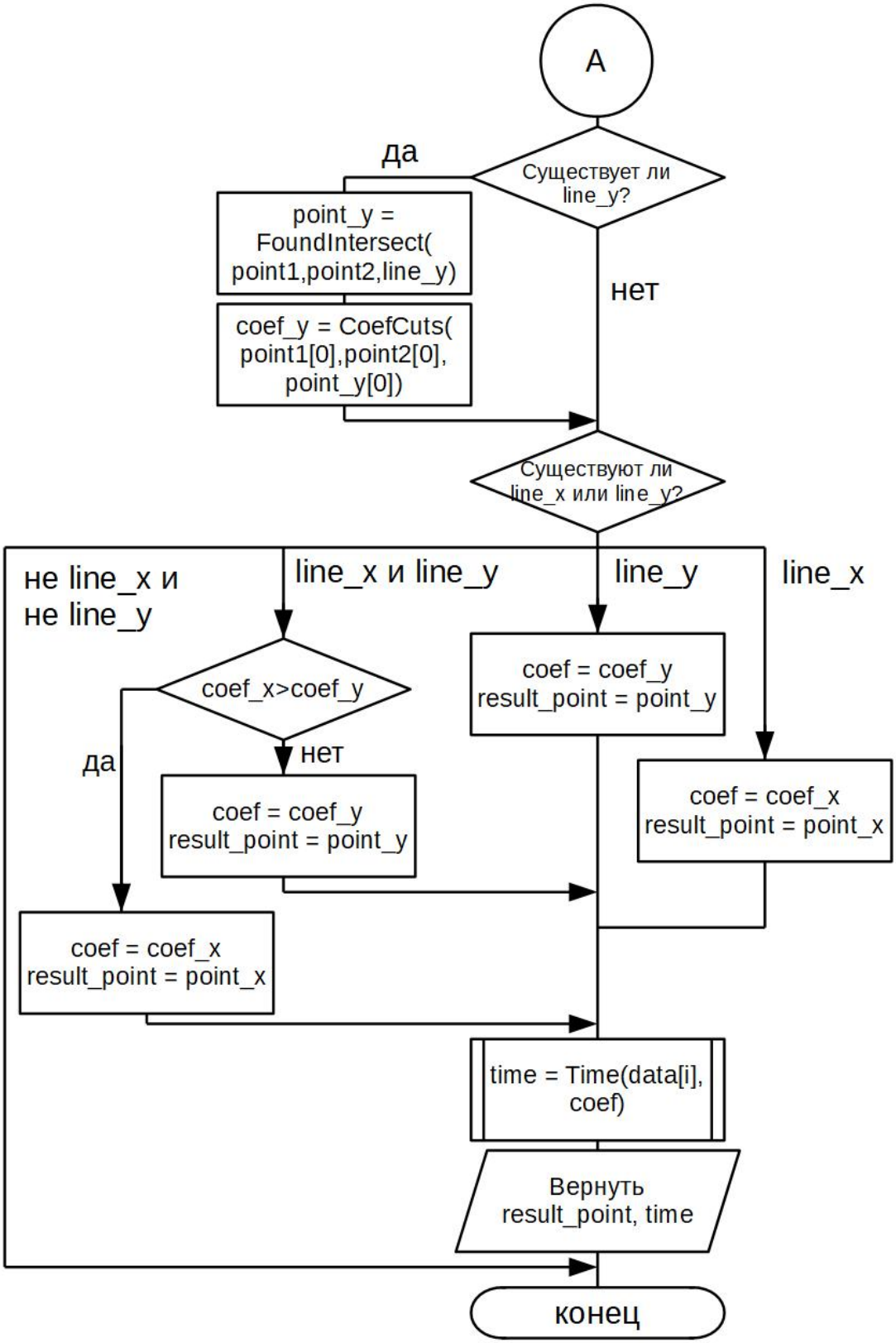


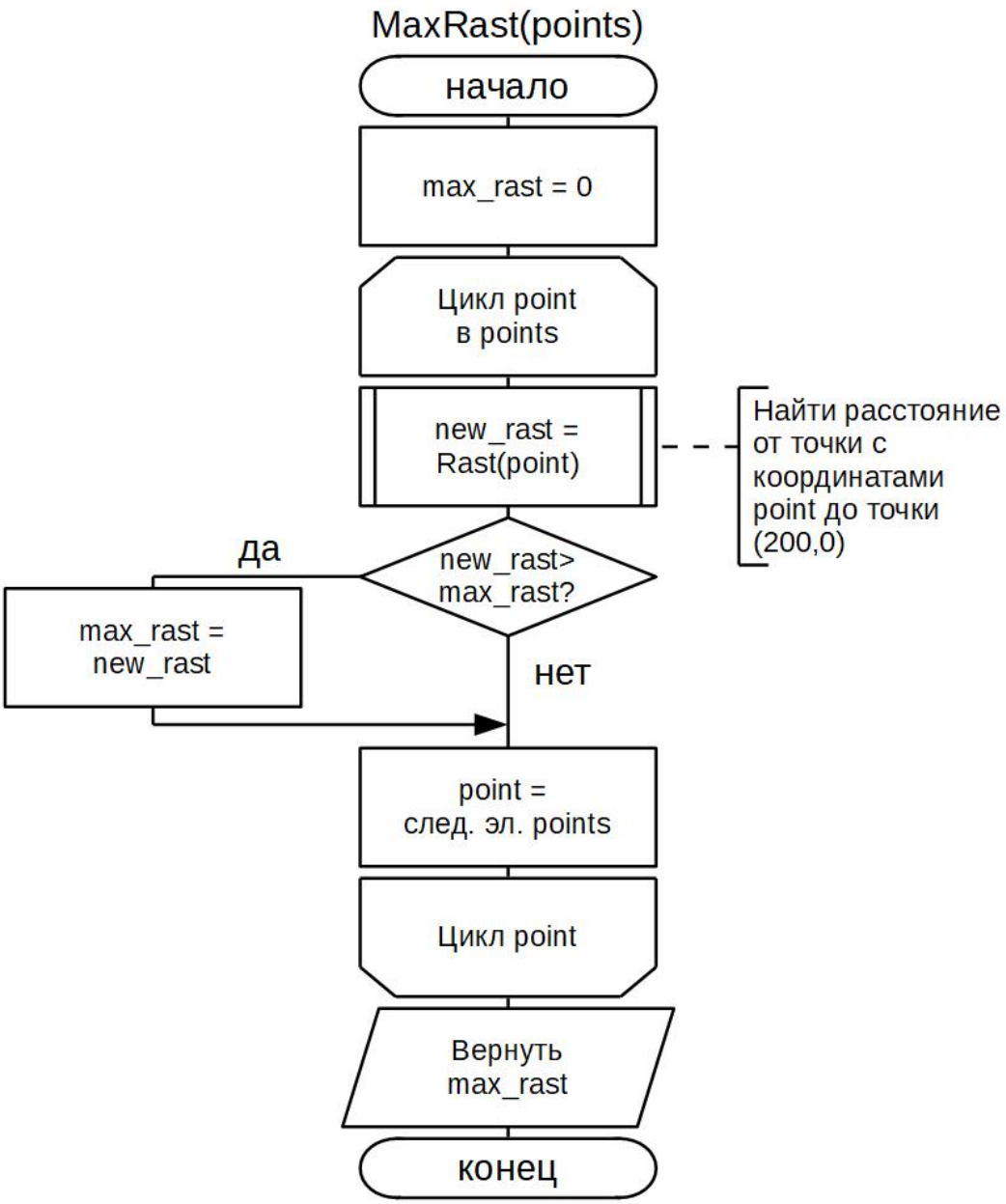
Time(line_data,coef)

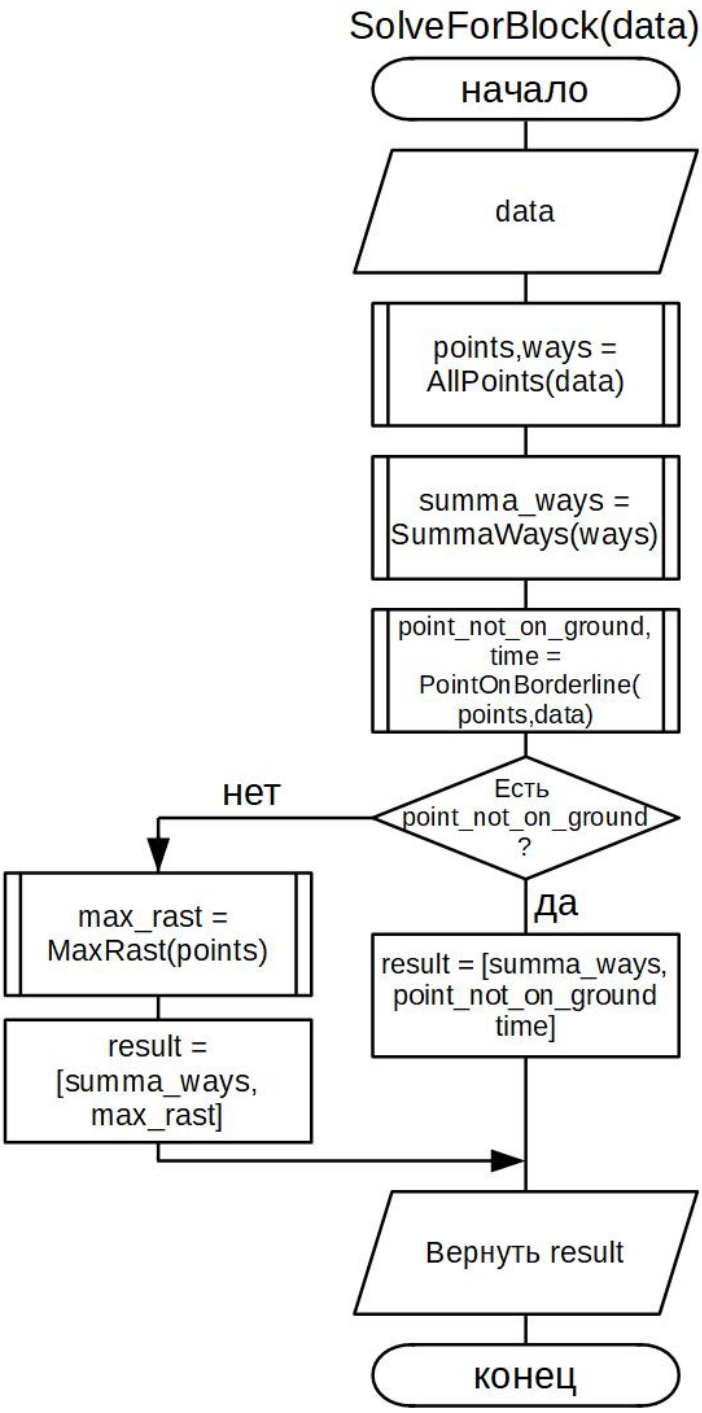


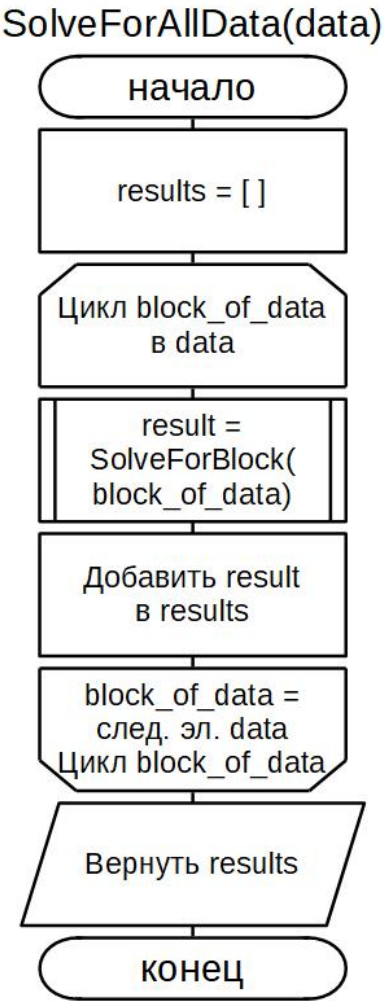

InGround(point)
начало
x,y = point
В каком дапазоне лежит x?
x<0
0<x<400
x>400
x_line = [‘x’,0]
x_line = False
x_line = [‘x’,400]
В каком дапазоне лежит y?
y<0
0<y<200
y>200
y_line = [‘y’,0]
y_line = False
y_line = [‘y’,400]
Вернуть x_line,y_line
конец

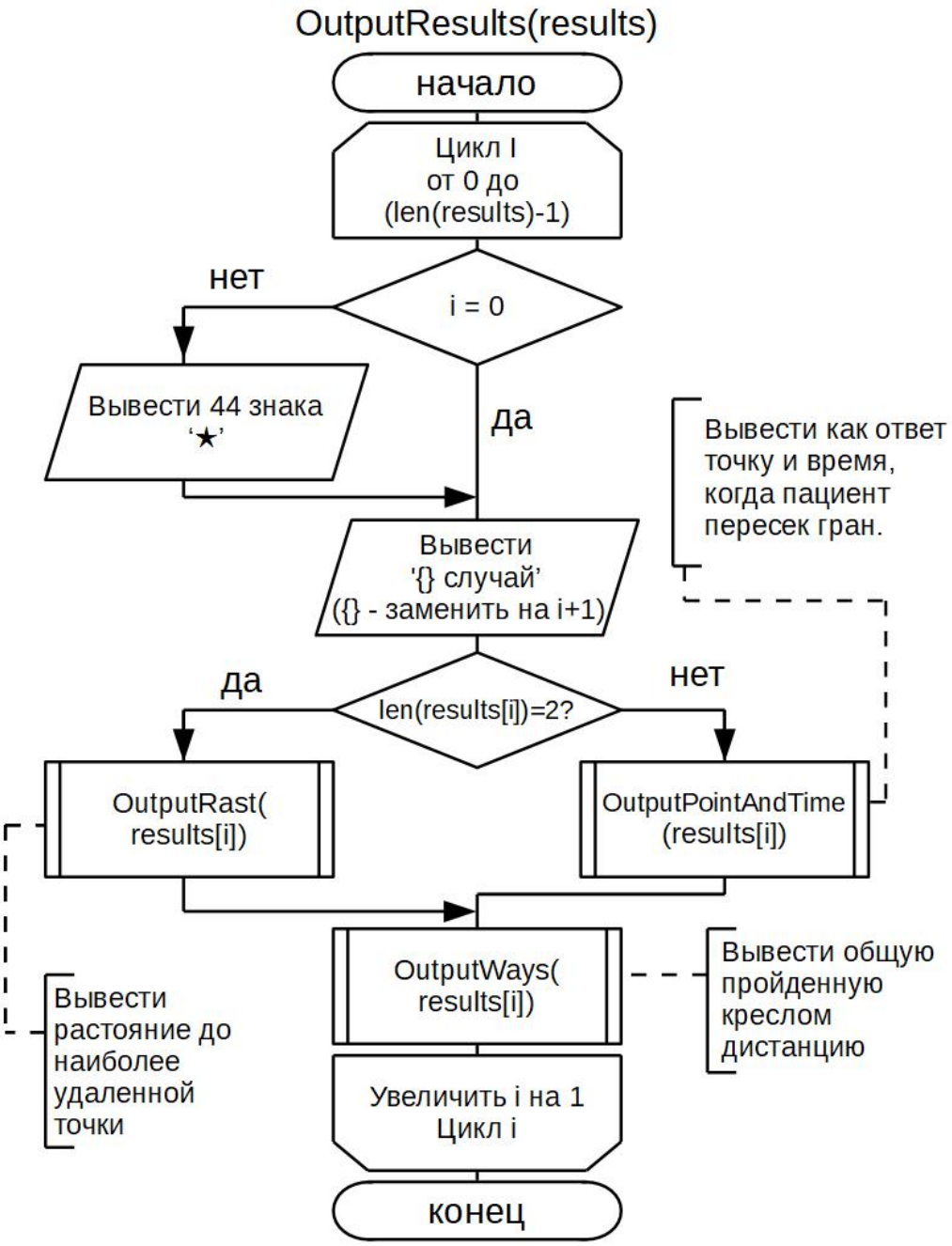

PointOnBorderline(points,data)
начало
Цикл i
от 1 до (длина
массива points-1)
point = point[i]
line_x,line_y =
InGround(point)
да
Существуют ли
line_x или line_y?
нет
Увеличить i на 1
Цикл i
point1 = points[i]
point2 = points[i+1]
да
Существует ли
line_x?
point_x =
FoundIntersect(
point1,point2,line_x)
нет
coef_x = CoefCuts(
point1[0],point2[0],
point_x[0])
A


A
Существует ли line_y?
да
нет
point_y = FoundIntersect(point1,point2,line_y)
coef_y = CoefCuts(point1[0],point2[0], point_y[0])
Существуют ли line_x или line_y?
не line_x и не line_y
line_x и line_y
line_y
line_x
coef_x>coef_y
да
нет
coef = coef_y result_point = point_y
coef = coef_y result_point = point_y
coef = coef_x result_point = point_x
coef = coef_x result_point = point_x
time = Time(data[i], coef)
Вернуть result_point, time
конец


MaxRast(points)
начало
max_rast = 0
Цикл point
в points
new_rast =
Rast(point)
Найти расстояние
от точки с
координатами
point до точки
(200,0)
да
new_rast>
max_rast?
max_rast =
new_rast
нет
point =
след. эл. points
Цикл point
Вернуть
max_rast
конец


SolveForBlock(data)
начало
data
points,ways = AllPoints(data)
summa_ways = SummaWays(ways)
point_not_on_ground, time = PointOnBorderline(points,data)
Есть point_not_on_ground ?
нет
да
max_rast = MaxRast(points)
result = [summa_ways, max_rast]
result = [summa_ways, point_not_on_ground time]
Вернуть result
конец


SolveForAllData(data)
начало
results = [ ]
Цикл block_of_data в data
result = SolveForBlock(block_of_data)
Добавить result в results
block_of_data = след. эл. data Цикл block_of_data
Вернуть results
конец


OutputResults(results)
начало
Цикл I
от 0 до
(len(results)-1)
нет
i = 0
да
Вывести 44 знака
‘★’
Вывести как ответ точку и время, когда пациент пересек гран.
Вывести
'{} случай’
({} - заменить на i+1)
да
len(results[i])=2?
нет
OutputRast(
results[i])
OutputPointAndTime
(results[i])
Вывести растояние до наиболее удаленной точки
OutputWays(
results[i])
Вывести общую пройденную креслом дистанцию
Увеличить i на 1
Цикл i
конец

Описание программы

Исследование реабилитационного оборудования

Алгоритм

Ввод данных с проверкой

Решение:

1 Нахождение массива точек, через заданные время, скорость и угол

2 Поиск первой точки, лежащей вне прямоугольника, окруженного x = 0, x = 400, y = 0, y = 200

3а.1 Если такая точка есть, необходимо определить с какой из сторон прямоугольника пересекается прямая, содержащая эту (M2) и пердыдущую точки (M1),

3а.2 Найти точку (M3) пересечения этой прямой и найденной стороны.

3а.3 Найти отношение M1M3/M2M3, в котором делится отрезок

3а.4 Найти время, когда пациент оказался в точке M3, учитывая, что t_{M1M3}/t_{M2M3} = M1M3/M2M3

3а.5 Записать в результат координаты M3 и найденное время t

3б.1 Если такой точки нет, найти расстояние до наиболее удаленной от двери точки и записать в результата

4 Найти весь путь, пройденный креслом, дозаписать в результат

Вывести результат.

Текст программы

```
#Импортирование необходимых функций из модуля math
from math import sin,cos,radians,sqrt
#Функция для ввода одного целого числа -
#количества строк с данными в новом блоке
#n - кол-во строк с данными
def InputInt():
    while True:
        n = input('Введите кол-во строк с данными: ')
        try:
            n = int(n)
        except ValueError:
            print('Число должно быть целым, повторите ввод:')
        else:
            break
    return n
#Функция для ввода одной строки данных
#Строка с данными - это четыре числа.
#1 - это время начала движения кресла
#2 - это время конца движения кресла
#3 - это скорость кресла
#4 - это азимут
#n - строка с данными
def InputLine():
    while True:
        n = input('Введите строку с данными: ')
        try:
            n = list(map(float, n.split()))
```

```
        except ValueError:
            print('Некорректный ввод, введите еще раз: ')
        else:
            if len(n)==4:
                break
            print('Должно быть 4 числа в строке. Повторите ввод:')
    return n
#Функция ввода блока данных
#Блок данных -
#кол-во строк с данными
#строки с данными
#ln_block - кол-во строк с данными
#new_line - строка с данными
#new_data - блок данных содержащий строки с данными
def InputBlock():
    ln_block = InputInt()
    new_data = []
    for i in range(ln_block):
        new_line = InputLine()
        new_data.append(new_line)
    return new_data
#Функция ввода всех данных
#new_data - Блок данных
#data - все введенные данные
def InputData():
    data = []
    while True:
        new_data = InputBlock()
        if new_data == []:
```

```
            break
        data.append(new_data)
    return data
#Функция высчитывания координат новой точки
#old_pt - координаты старой точки
#pt_data - строка данных для высчитывания этой точки
#delta_t - время движения кресла
#alpha - азимут движения
#way - путь, пройденный креслом
#delta_x,delta_y - смещение координат новой точки относительно старой точки
#new_pt - координаты новой точки
def NewPoint(old_pt, pt_data):
    delta_t = pt_data[1] - pt_data[0]
    alpha = radians(pt_data[3])
    way = delta_t*pt_data[2]
    delta_x = sin(alpha)*way
    delta_y = cos(alpha)*way
    new_pt = [old_pt[0]+delta_x, old_pt[1]+delta_y]
    return new_pt,way
#Функция высчитывания координат
#points - массив с массивами координат для всех точек
#point - координаты текущей (обрабатываемой точки) точки
#ways - массив с путями
#way - путь до текущей точки
#line - строка данных
def AllPoints(data):
    points = [[200,0]]
    point = [200,0]
    ways = []
```

```
    for line in data:
        point,way = NewPoint(point, line)
        ways.append(way)
        points.append(point)
    return points,ways
#Функция для подсчета расстояния от введенной точки до двери
#Координаты двери (200,0)
#point1 - координаты введенной точки
#x1,y1,x2,y2 - координаты x и y введенной точки и двери соответственно
#rast - растояние от введенной точки до двери
def Rast(point1):
    x1,y1 = point1
    x2,y2 = 200,0
    rast = sqrt((x1-x2)**2+(y1-y2)**2)
    return rast
#Функция для подсчета коэффицентов
#point1, point2 - координаты двух точек прямой
#k,m,b - коэффиценты прямой в уравнении m*y=k*x+b
#line - массив с коэффицентами прямой
def Line(point1,point2):
    try:
        k = (point1[1]-point2[1])/(point1[0]-point2[0])
    except ZeroDivisionError:
        k = 1
        m = 0
    else:
        m = 1
    b = point1[1] - k*(point1[0])
    line = [k, b, m]
```

```
    return line
#Функция для определения: находится ли точка в ограниченном поле
#point - координаты точки
#x,y - координаты точки
#x_line - вылет за границы по оси x, и указание за какую (0 или 400)
#y_line - вылет за границы по оси y, и указание за какую (0 или 200)
def InGround(point):
    x_line = False
    y_line = False
    x,y = point
    if x<0:
        x_line = ['x',0]
    elif x>400:
        x_line = ['x',400]
    if y<0:
        y_line = ['y',0]
    elif y>200:
        y_line = ['y',200]
    return x_line,y_line
#Функция для поиска пересечения прямой, заданной двумя точками,
#и прямой, заданной одним из уравнений x=0, x=400, y=0, y=200
#point1 - первая точка первой прямой
#point2 - вторая точка первой прямой
#coord - данные о второй прямой
#x,y - координата точки пересечения x или y (соответственно)
#point - точка пересечения
def FoundIntersect(point1,point2,coord):
    line = Line(point1,point2)
    if coord[0] == 'x':
```

```
        y = (coord[1]*line[0] + line[1])/line[2]
        point = [coord[1], y]
    else:
        x = (coord[1]*line[2] - line[1])/line[0]
        point = [x, coord[1]]
    return point
#Функция считает коэффицент соотношения отрезков M1M3 и M2M3,
#где M3 принадлежит M1M2
#x1,x2,x3 - координаты точек соответсвенно M1M3 и M2M3
#coef - искомый коэффицент
def CoefCuts(x1,x2,x3):
    coef = (x3-x1)/(x2-x3)
    coef = coef/(coef+1)
    return coef
#Функция, которая считает время до момента, когда пациент покинул
#ограниченную область в первый раз
#line_data - строка данных
#coef - коэффицент отношения, в котором разбивается время
#delta_t - изменение времени
#time - время, когда покинул ограниченную область в первую
def Time(line_data,coef):
    delta_t = line_data[1]-line_data[0]
    time = coef*delta_t + line_data[0]
    return time
#Функция для подсчета расстояния от двери до наиболее удаленной точки
#points - все точки
#point - текущая точка
#max_rast - максимальное расстояние
#new_rast - текущее расстояние
```

```
def MaxRast(points):
    max_rast = 0
    for point in points:
        new_rast = Rast(point)
        if new_rast > max_rast:
            max_rast = new_rast
    return max_rast
#Функция для поиска точки первого выезда пациента за ограниченную область
#points – массив точек
#data – массив со всеми введенными данными
#line_x, line_y – данные о пересечении верт. и гор. границ
#point1,point2 – первая точка за границей и предыдущая
#coef_x, point_x – данные о пересечении с гориз. границами
#coef_y, point_y – данные о пересечении с верт. границами
#coef, result_point – данные о пересечении с нужной границей
#time – время пересечения границы
def PointOnBorderline(points,data):
    for i in range(1,len(points)):
        point = points[i]
        line_x,line_y = InGround(point)
        if (line_x or line_y):
            break
    if not (line_x or line_y):
        return False,False
    point1 = points[i]
    point2 = points[i + 1]
    if line_x:
        point_x = FoundIntersect(point1,point2,line_x)
        coef_x = CoefCuts(point1[0],point2[0],point_x[0])
```

```
    if line_y:
        point_y = FoundIntersect(point1,point2,line_y)
        coef_y = CoefCuts(point1[0],point2[0],point_y[0])
    if line_x and line_y:
        if coef_x>coef_y:
            coef = coef_y
            result_point = point_y
        else:
            coef = coef_x
            result_point = point_x
    elif line_x:
        coef = coef_x
        result_point = point_x
    else:
        coef = coef_y
        result_point = point_y
    time = Time(data[i-1], coef)
    return result_point, time
#Функция для подсчета суммы всего пути, пройденного креслом
#ways - все пути
#summa - сумма путей
def SummaWays(ways):
    summa = 0
    for way in ways:
        summa += way
    return summa
#Функция решает поставленную задачу для одного блока
#data - данные для блока
#points - точки в этом блоке
```

```
#ways - пути в этом блоке
#summa_ways - путь, пройденный креслом за этот блок наблюдений
#point_not_on_ground - точка первого выезда пациента из ограниченной зоны
#time - момент времени, в который пациент пересек границу в первый раз
#max_rast - расстояние от двери до наиболее удаленной от нее точки
#result - массив с ответом для заданного блока
def SolveForBlock(data):
    points, ways = AllPoints(data)
    summa_ways = SummaWays(ways)
    point_not_on_ground, time = PointOnBorderline(points,data)
    if not point_not_on_ground:
        max_rast = MaxRast(points)
        result = [summa_ways, max_rast]
    else:
        result = [summa_ways, point_not_on_ground, time]
    return result
#Функция для решения для всех блоков
#data - все введенные данные
#block_of_data - блок данных
#results - результаты по всем блокам
#result - результат по текущему блоку
def SolveForAllData(data):
    results = []
    for block_of_data in data:
        result = SolveForBlock(block_of_data)
        results.append(result)
    return results
#Функция
#i - счетчик цикла
```

```
#results – результаты работы программы
def OutputResults(results):
    for i in range(len(results)):
        if i != 0:
            print('★'*44)
        print('{} случай'.format(i+1))
        if len(results[i]) == 2:
            OutputRast(results[i])
        else:
            OutputPointAndTime(results[i])
        OutputWays(results[i][0])
#Функция для вывода результата - расстояния
#result – результат, который надо вывести
#l1, l2 - строки  для вывода
def OutputRast(result):
    l1 = 'Пациент не пересекал границы ограниченнной области.\n'
    l2 = 'Наибольшее расстояние от точки до двери равно {:.1f}.'.format(
        result[1])
    print(l1+l2)
#Функция для вывода результата - точки пересечения с границами поля
#l1, l2 - строка для вывода
#result - результат, который надо вывести
def OutputPointAndTime(result):
    l1 = 'Пациент пересек ограниченное поле в точке ({:.1f},{:.1f})'.format(
        result[1][0], result[1][1])
    l2 = 'в момент времени {:.1f}.'.format(result[2])
    print(l1+l2)
#Функция для вывода результата - всего пути пройденного креслом за #этот
блок
```

```
#l3 - строка для вывода
#result - результат, который надо вывести
def OutputWays(result):
    l3 = 'Путь, который преодолел пациент, равен {:.1f}'.format(result)
    print(l3)
#Data - все введенные данные
#Result - все полученные результаты
print('''Примечание: Строка с данными - это четыре числа.
1 - это время начала движения кресла
2 - это время конца движения кресла
3 - это скорость кресла
4 - это азимут''')
Data = InputData()
Result = SolveForAllData(Data)
OutputResults(Result)
```

Заключение

Я научилась работать с координатами в python, обрабатывать многомерные массивы, используя функции и методы python.

Список использованной литературы:

1)Доусон М. Програмируем на Python. - Спб.: Питер, 2016. - 416 с.: ил.

2010 Course Technology, a part of Cengage Learning

Перевод на русский язык ООО Издательство «Питер», 2016

Издание на русском языке, оформление

ОООИздательство «Питер», 2016

2)Бизли Д., Джонс Б. К. Python. Книга рецептов / пер. с анг. Б. В. Уварова. - М.: ДМК Пресс, 2019. - 648 с.: ил.