



НТО

МАТЕРИАЛЫ ЗАДАНИЙ

Всероссийской междисциплинарной олимпиады школьников

«Национальная технологическая олимпиада»

по профилю

«Беспилотные авиационные системы»

2021/22 учебный год

<http://ntcontest.ru>

Оглавление

1 Профиль «Беспилотные авиационные системы»	5
I Первый отборочный этап	7
I.1 Задачи первого этапа. Информатика	7
I.1.1 Первая попытка. Задачи 8–11 класса	7
I.1.2 Вторая попытка. Задачи 8–11 класса	16
I.1.3 Третья попытка. Задачи 8–11 класса	23
I.1.4 Четвертая попытка. Задачи 8–11 класса	32
I.2 Задачи первого этапа. Физика	43
I.2.1 Первая попытка. Задачи 8–9 класса	43
I.2.2 Первая попытка. Задачи 10–11 класса	45
I.2.3 Вторая попытка. Задачи 8–9 класса	48
I.2.4 Вторая попытка. Задачи 10–11 класса	50
I.2.5 Третья попытка. Задачи 8–9 класса	53
I.2.6 Третья попытка. Задачи 10–11 класса	55
I.2.7 Четвертая попытка. Задачи 8–9 класса	58
I.2.8 Четвертая попытка. Задачи 10–11 класса	60
II Второй отборочный этап	64
II.1 Индивидуальная часть	64
II.2 Командная часть	83
III Заключительный этап	94
III.1 Индивидуальный предметный тур	94
III.1.1 Информатика. 8–11 класс	94
III.1.2 Физика. 8–9 классы	102
III.1.3 Физика. 10–11 классы	107
III.2 Командный практический тур	114

III.2.1 Требования к команде	114
III.2.2 Оборудование и программное обеспечение	114
III.2.3 Задачи	115
IV Критерии определения победителей и призеров	133

Профиль «Беспилотные авиационные системы»

Профиль «Беспилотные авиационные системы» нацелен на вовлечение старшеклассников в техническую и инновационную деятельность в области проектирования систем автоматического управления беспилотными летательными аппаратами (БЛА). Особое внимание уделяется организации процесса разработки автопилота, при котором возможна практическая проверка разработанных программ в ходе летных испытаний на аэродроме.

Профиль направлен на решение следующих технологических барьеров НТИ:

- сенсоры и преобразующая аппаратура оптического, теплового, гиперспектрального, радиолокационного зондирования поверхности, радиолокационные станции бортового обзора, в том числе, с функцией распознавания образов людей, животных, транспортных средств и потоков, мобильных и стационарных объектов для обеспечения мониторинга, подсчета наблюдаемых объектов и выявления их характерных признаков, а также для выявления признаков чрезвычайных ситуаций;
- беспроводная система мониторинга систем и узлов беспилотных воздушных судов (БВС);
- электронно-оптическая система «улучшенного видения» для улучшения изображения в условиях тумана, плохой видимости, при наличии препятствий;
- БВС для сбора, хранения и обработки информации о характеристиках окружающего пространства.

Основной задачей заключительного этапа является разработка системы автоматического управления БЛА самолетного типа для поиска объектов в автоматическом режиме при наличии отказов в системе управления. Для решения этой основной задачи участникам необходимо было иметь соответствующие знания по физике и информатике, а также:

1. разобраться в принципах полета БЛА самолетного типа, научиться работать с файлами навигационных данных, используемых в управлении БЛА;
2. разобраться в том, что такое регуляторы, и в принципах построения системы автоматического управления;
3. используя симулятор, провести настройку регуляторов по соответствующим критериям для стабилизации бортовой фотокамеры;
4. научиться разрабатывать алгоритмы автоматического управления БЛА самолетного типа с целью полета по заданной траектории и выполнения определенного задания в условиях возможных отказов в системе измерения и управления БЛА;
5. научиться обрабатывать данные, полученные с помощью системы технического зрения;
6. проверить работоспособность созданного программного обеспечения (системы автоматического управления (САУ) БЛА самолетного типа) каждой командой путем испытаний на симуляторе.

Основной акцент при разработке командной комплексной инженерной задачи был направлен на четкое разделение основной задачи на подзадачи, при успешном решении которых участники достигали поставленной цели. Каждая подзадача на заключительном этапе имела свой уровень сложности и оценивалась соответствующим баллом. Подзадачи были распределены по соответствующим этапам.

Первый отборочный этап включает в себя решение олимпиадных задач по двум школьным предметам: информатика и физика. Хорошие знания по информатике и физике крайне необходимы участникам профиля, так как требуются для решения задач второго отборочного и заключительного этапов.

На втором отборочном этапе участники решают задачи по определению параметров полета, ориентации и навигации, моделированию полета в упрощенной форме, а также по обработке изображений. Задачи носят междисциплинарный характер и в упрощенной форме воссоздают инженерную задачу заключительного этапа.

На заключительном этапе участники используют все знания, полученные на предыдущих этапах, при работе с БЛА самолетного типа (работа с системой технического зрения, моделирование полета и проверка своих решений при помощи специально разработанного симулятора). Такой подход дает участникам возможность глубоко понять объект исследования и принцип его работы.

Уже несколько лет участники профиля «Беспилотные авиационные системы» поступают в вузы, в том числе и в МАИ, и успешно в них обучаются.

Первый отборочный этап

Задачи первого этапа. Информатика

Первая попытка. Задачи 8–11 класса

Задача I.1.1.1. Расчет скидки (20 баллов)

Темы: задачи для начинающих, простая математика.

Одна продуктовая сеть в рамках акции выдает скидочные купоны двух видов. По первому купону можно получить скидку в 8% от стоимости покупки, но не более 100 рублей. По второму купону можно получить скидку в 5% от стоимости покупки без других ограничений. Предъявлять можно только один купон, разделять покупку на части нельзя. Покупатель делает покупку на p рублей. У него есть оба купона. Напишите программу, которая вычислит максимальный размер скидки, которую покупатель сможет получить.

Формат входных данных

На вход подается одно целое число — размер покупки в рублях. Число не превосходит 10000.

Формат выходных данных

Вывести одно число — размер скидки в рублях. Ответ может оказаться не целым.

Методика проверки

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
810
Стандартный вывод
64.8

Пример №2

Стандартный ввод
1530
Стандартный вывод
100

Пример №3

Стандартный ввод
10000
Стандартный вывод
500

Решение

В этой задаче требуется рассмотреть два варианта. Обозначим сумму покупки за p .

При использовании купона со скидкой в 8% сумма скидки является минимумом из 100 и $0,08p$. При использовании купона со скидкой в 5% сумма скидки будет равна $0,05p$.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
1 p=int(input())
2 print(max(p*0.05,min(p*0.08,100)))
```

Задача I.1.1.2. Произведение многочленов (20 баллов)

Темы: реализация, простая математика.

Многочлены — это одни из самых распространенных математических объектов, которые используются практически во всех прикладных областях. Задан многочлен $a_n x^n + a_{n-1} x^{n-1} + \dots + a_2 x^2 + a_1 x + a_0$. От вас требуется написать программу, которая найдет произведение этого многочлена на $x + 1$. Многочлен задан своими коэффициентами $a_n, a_{n-1}, \dots, a_2, a_1, a_0$. Обратите внимание, что многочлен степени n состоит из $n + 1$ одночлена. Некоторые из одночленов могут отсутствовать. В этом случае соответствующий коэффициент считается равным нулю.

Например, многочлен $2x^3 + 3x^2 + 1$ будет задан набором коэффициентов 2 3 0 1. Результатом умножения будет многочлен четвертой степени с набором коэффициентов 2 5 3 1 1, что можно проверить, раскрыв скобки.

$$(2x^3 + 3x^2 + 1)(x + 1) = 2x^4 + 3x^3 + x + 2x^3 + 3x^2 + 1 = 2x^4 + 5x^3 + 3x^2 + x + 1$$

Формат входных данных

На вход программы в первой строке подается одно натуральное число n — степень многочлена. $1 \leq n \leq 100$. Далее во второй строке через пробел подается $n + 1$ целое число — коэффициенты многочлена $a_n, a_{n-1}, \dots, a_2, a_1, a_0$. Каждый из коэффициентов не превосходит 1000 по абсолютной величине. $a_n \neq 0$.

Формат выходных данных

Требуется вывести через пробел $n + 2$ коэффициента полученного многочлена.

Если вы программируете на Python, то убрать перенос строки в функции `print` можно при помощи именованного параметра `end`, например, `print(a, end=' ')`.

Методика проверки

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
3
2 3 0 1
Стандартный вывод
2 5 3 1 1

Решение

Найдем произведение многочлена $a_n x^n + a_{n-1} x^{n-1} + \dots + a_2 x^2 + a_1 x + a_0$ и $x + 1$.

$$\begin{aligned}
 & (a_n x^n + a_{n-1} x^{n-1} + \dots + a_2 x^2 + a_1 x + a_0)(x + 1) = \\
 & = (a_n x^{n+1} + a_{n-1} x^n + \dots + a_2 x^3 + a_1 x^2 + a_0 x) + \\
 & \quad + (a_n x^n + a_{n-1} x^{n-1} + \dots + a_2 x^2 + a_1 x + a_0) = \\
 & = a_n x^{n+1} + (a_{n-1} + a_n) x^n + \dots + (a_1 + a_2) x^2 + (a_0 + a_1) x + a_0
 \end{aligned}$$

Таким образом каждый коэффициент, кроме первого и последнего, является суммой двух соседних элементов исходного списка.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 input()
2 x=map(int,input().split())
3 prev=int(next(iter(x)))
4 print(prev,end=' ')

```

```
5 for t in x:
6     print(t+prev,end=' ')
7     prev=t;
8 print(prev)
```

Ниже представлено решение в функциональном стиле на языке Python 3.

```
1 x=list(map(int,input().split()))
2 print(*[a+b for (a,b) in zip([0]+x,x+[0])])
```

Задача I.1.1.3. Очередь (20 баллов)

Темы: структуры данных.

Студенческая группа сдает зачет преподавателю. Он расположил студентов по некоторому неизвестному порядку и сообщил, кто после кого сдает зачет. Теперь студенты хотят выяснить, в каком порядке им приходиться, чтобы не стоять всей группой за дверью.

Формат входных данных

На вход программы в первой строке подается одно натуральное число n — количество студентов в группе. $2 \leq n \leq 30$. Далее в $n - 1$ строке через пробел подается по два имени: имя студента, сдающего зачет, и имя того студента, который будет сдавать перед ним. Имена не содержат пробелов и состоят только из строчных и прописных символов латиницы. Гарантируется, что очередь задана корректно, имена студентов в группе не повторяются.

Формат выходных данных

Требуется вывести имена всех студентов группы в том порядке, в котором они сдают зачет. Каждое имя выводится в отдельной строке.

Методика проверки и пояснение к тесту

В приведенном примере в группе 5 студентов. Первой сдает зачет Лиза, поскольку только для нее не указано, кто приходит раньше. После Лизы приходит Иван, далее Мария, затем Петр и последним сдает Игорь.

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
5 Petr Mariya Ivan Liza Mariya Ivan Igor Petr
Стандартный вывод
Liza Ivan Mariya Petr Igor

Решение

В этой задаче требуется продемонстрировать умение работать со структурами данных.

Наиболее простым способом решения будет создание ассоциативного массива (словаря), в котором для каждого студента будет указано имя следующего в очереди.

Дополнительной сложностью является нахождение имени первого студента. Для этого надо найти имя, которое есть среди ключей, но которого нет среди значений словаря. Такая операция может быть записана как разность множества ключей и множества значений словаря. По условию такая разность будет содержать только один элемент, который и будет искомым именем.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  n=int(input())
2  nxt=dict()
3  for i in range(n-1):
4      val,key=input().split()
5      nxt[key]=val
6  cur=next(iter(nxt.keys()-nxt.values()))
7  print(cur)
8  for i in range(n-1):
9      cur=nxt[cur]
10     print(cur)

```

Задача I.1.1.4. Четырехугольник (20 баллов)

Темы: геометрия, неравенство треугольника, перебор, реализация.

Сергею на уроке геометрии задали следующее задание. Даны 5 чисел. Требуется нарисовать произвольный четырехугольник с одной диагональю так, чтобы длины

сторон и этой диагонали равнялись заданным числам. Сергею надо выбрать длину диагонали и каждую из сторон так, чтобы было возможно нарисовать требуемую фигуру. Если вариантов решения задачи несколько, можно выбрать любой. Нарисованная диагональ не должна лежать на одной из сторон. Возможно, что нарисовать требуемый четырехугольник не получится. В этом случае надо будет вывести ноль.

Формат входных данных

На вход через пробел подаются 5 натуральных чисел от 1 до 1000.

Формат выходных данных

Требуется вывести ответ в следующем порядке. В первой строке вывести число — длину диагонали четырехугольника. Во второй строке 2 числа — длины отрезков, лежащих с одной стороны от диагонали. В третьей строке еще 2 числа — длины отрезков, лежащих с другой стороны от диагонали. Если построить четырехугольник невозможно, то вывести 0.

В этой задаче можно выводить любой правильный ответ. В частности, можно переставить числа в одной строке или поменять местами вторую и третью строку.

Методика проверки и пояснение к тестам

В первом тесте в качестве диагонали можно взять отрезок длины 7. Тогда с одной стороны от диагонали будут стороны с длинами 3 и 5, а с другой — 9 и 3. Вторую и третью строку, а также числа в этих строках можно вывести в любом порядке. Также возможно нарисовать четырехугольник с диагональю 5 и длинами сторон 9, 7 и 3, 3. Кроме того, возможен вариант с диагональю 3 и длинами сторон 5, 3 и 7, 9. Любой из этих вариантов будет считаться верным.

Во втором тесте нарисовать четырехугольник невозможно. Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
3 9 5 3 7
Стандартный вывод
7
3 5
9 3

Пример №2

Стандартный ввод
3 9 5 1 7
Стандартный вывод
0

Решение

Для решения этой задачи требуется знать неравенство треугольника, которое говорит о том, что в невырожденном треугольнике сумма длин двух любых сторон больше, чем длина третьей. Четырехугольник с одной диагональю можно представить как два треугольника, смежных по одной стороне.

Решение должно заключаться в переборе нескольких вариантов. Этот перебор можно организовать разными способами. Один из способов заключается в использовании функций для генерации перестановок. В Python это функция `permutations` из модуля `itertools`. Можно получить все перестановки пяти заданных чисел и проверить их по неравенству треугольника.

Чтобы получить другой способ решения, можно заметить, что две самых длинных стороны в любом случае выгоднее использовать при построении одного треугольника. Тогда количество вариантов сокращается до трех.

Пример программы-решения

Ниже представлено решение на языке Python 3 с полным перебором.

```

1  from itertools import permutations
2  def triangle(a,b,c):
3      return a<b+c and b<a+c and c<a+b
4  x=map(int,input().split())
5  for t in list(permutations(x)):
6      if triangle(t[0],t[1],t[2]) and triangle(t[0],t[3],t[4]):
7          print(t[0])
8          print(t[1],t[2])
9          print(t[3],t[4])
10         break
11  else:
12         print(0)

```

Ниже представлено решение на языке Python 3 с сортировкой.

```

1  x=sorted(list(map(int,input().split())),reverse=True)
2  if x[0]<x[1]+x[2] and x[2]<x[3]+x[4]:
3      print(x[2])
4      print(x[0],x[1])
5      print(x[3],x[4])
6  elif x[0]<x[1]+x[3] and x[1]<x[2]+x[4]:
7      print(x[1])
8      print(x[0],x[3])
9      print(x[2],x[4])
10 elif x[0]<x[1]+x[4] and x[1]<x[2]+x[3]:
11     print(x[1])
12     print(x[0],x[4])
13     print(x[2],x[3])
14 else:
15     print(0)

```

Задача I.1.1.5. Рыцари (20 баллов)

Темы: структуры данных, реализация, алгоритмическая сложность.

В королевстве Логрес за круглым столом собираются рыцари. Каждый рыцарь гордится своими победами, поэтому делает на своих доспехах царاپины по количеству побежденных противников. Ранг рыцаря определяется по количеству царापин. Рыцари весьма горды и тот, чей ранг ниже, должен оказывать почтение тому, чей ранг выше. Если вдруг возникает ситуация, что подряд сидят рыцари с одинаковым рангом, то они устраивают между собой турнир, по итогам которого определяется один победитель. Он ставит на доспехи новые царاپины (если в турнире участвовало k рыцарей, то победитель поставит $k - 1$ новую царापину) и возвращается за стол на свое место. Остальные участники турнира уходят залечивать раны и уязвленное самолюбие. Если после этого вновь возникает аналогичная ситуация, то проводится новый турнир, и так далее.

За круглым столом собралось n рыцарей, и они предусмотрительно расселись так, чтобы ранги соседей были различными. Но опоздавший Галахад все испортил. Он сел на случайно выбранное место, и карусель турниров снова закрутилась.

Известны ранги всех n рыцарей, изначально сидевших за столом. Также известен ранг Галахада и место, на которое он сел. Напишите программу для определения количества рыцарей, которые останутся за столом после того, как все турниры завершатся.

Формат входных данных

В первой строке на вход подается число n — количество рыцарей без учета Галахада. $1 \leq n \leq 300000$. Во второй строке через пробел записаны n натуральных чисел r_1, \dots, r_n — ранги всех рыцарей. $r_i \leq 10^6$; $r_i \neq r_{i+1}$; $r_1 \neq r_n$. В третьей строке через пробел записаны два натуральных числа p и t . $1 \leq p \leq n$; $t \leq 10^6$. Число p задает место, на которое сел Галахад и означает, что он оказался между рыцарями с номерами p и $p + 1$. Если $p = n$, то Галахад находится между первым и последним рыцарем. Число t задает исходный ранг Галахада.

Формат выходных данных

Требуется вывести одно число — ответ к задаче.

Методика проверки и пояснение к тесту

После того, как Галахад сядет за стол после второго рыцаря, ранги рыцарей за столом будут (7 5 5 6 8 9 10 12 10 8). Два рыцаря с рангом 5 проведут турнир, останется один из них, ранг которого повысится до 6 (7 6 6 8 9 10 12 10 8). Из двух рыцарей с рангом 6 вновь останется 1 с рангом 7 (7 7 8 9 10 12 10 8). После очередного турнира получим (8 8 9 10 12 10 8). Помня, что стол круглый, видим 3 рыцарей с рангом 8, сидящих подряд. Из них останется один с рангом 10 (10 9 10 12 10). Будет проведен еще один турнир, после которого оставшиеся 4 рыцаря не будут иметь соседей с одинаковым рангом (11 9 10 12).

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
9 7 5 6 8 9 10 12 10 8 2 5
Стандартный вывод
4

Решение

В этой задаче требовалось придумать достаточно простой в реализации способ решения, который при этом удовлетворял бы требованию по времени работы программы. Для этого необходимо, чтобы элементы не удалялись из массива, так как эта операция достаточно затратная по времени. Дополнительной сложностью является то, что последовательность элементов должна быть циклической.

Заметим, что все изменения могут происходить вокруг позиции Галахада, поэтому будет удобно, если его ранг не будет храниться в последовательности, а номера рыцарей слева от него будут начинаться с нуля. Тогда справа от Галахада будет последний элемент последовательности. Такого состояния можно добиться при помощи циклического сдвига всей последовательности.

Далее будем использовать метод двух указателей. Номер рыцаря слева от Галахада будем хранить в переменной i , а номер правого рыцаря — в переменной j . Вместо удаления элементов будем просто смещать значения этих переменных.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  n=int(input())
2  lst=list(map(int,input().split()))
3  p,r=map(int,input().split())
4  lst=lst[p:]+lst[:p]
5  i=0
6  j=n-1
7  while i<j:
8      if lst[i]==lst[j]==r:
9          r+=2
10         i+=1
11         j-=1
12     elif lst[i]==r:
13         r+=1
14         i+=1
15     elif lst[j]==r:
16         r+=1
17         j-=1
18     else:
19         break
20 if i==j and lst[i]==r:
21     i+=1
22 print(j-i+2)

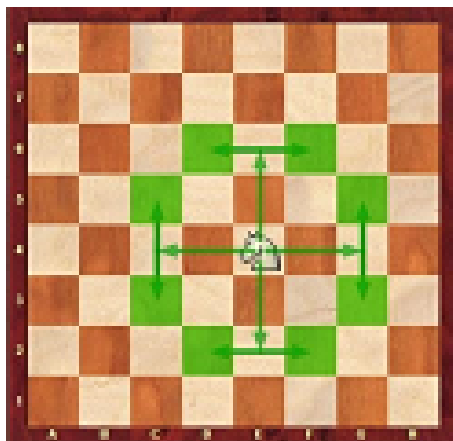
```

Вторая попытка. Задачи 8–11 класса

Задача I.1.2.1. Конь (20 баллов)

Темы: задачи для начинающих, перебор.

Шахматный конь стоит на доске размером 8×8 в i -той строке и j -том столбце. Напишите программу, которая определит, сколько ходов он может сделать.



Конь ходит, как показано на рисунке. Из центральной части доски он может сделать 8 ходов, но если конь находится ближе к краю доски, то количество ходов уменьшится, так как он не может выйти за ее границы.

Формат входных данных

На вход подается два натуральных числа в диапазоне от 1 до 8 — номер клетки, в которой находится конь, по горизонтали и вертикали. Каждое число записано в отдельной строке.

Формат выходных данных

Вывести одно число — количество возможных ходов коня.

Методика проверки

Программа проверяется на 40 тестах. Прохождение каждого теста оценивается в 0,5 балла. Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
5
3
Стандартный вывод
8

Решение

В предложенном решении этой задачи перебираются все клетки шахматной доски, и выполняется проверка того, что конь может сходить в эту клетку. Возможны и другие решения.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 a=int(input())
2 b=int(input())
3 ans=0
4 for i in range(1,9):
5     for j in range(1,9):
6         if a!=i and b!=j and abs(a-i)+abs(b-j)==3:
7             ans+=1
8 print(ans)

```

Задача I.1.2.2. Королевство чисел (20 баллов)

Темы: задачи для начинающих, простая математика.

Алиса и Боб стали королями в королевствах на множестве натуральных чисел. Подданными Алисы являются все натуральные числа, которые делятся на 3 без остатка, а все остальные числа стали подданными Боба. Алиса дружит с Бобом, и они хотят, чтобы их подданные тоже дружили между собой. Они разбили все числа на пары, причем i -тое по порядку число из королевства Алисы будет дружить с i -тым по порядку числом из королевства Боба. Вам задан набор из n чисел. Напишите программу для нахождения друга каждого из чисел.

Первые 10 чисел из королевства Алисы — это $\{3, 6, 9, 12, 15, 18, 21, 24, 27, 30, \dots\}$. Первые 10 чисел из королевства Боба — это $\{1, 2, 4, 5, 7, 8, 10, 11, 13, 14, \dots\}$. Таким образом, парами друзей являются $(3, 1)$ $(6, 2)$ $(9, 4)$ и так далее.

Формат входных данных

На вход в первой строке подается натуральное число n — количество чисел в наборе. $1 \leq n \leq 10^5$. Во второй строке через пробел подается n натуральных чисел a_1, a_2, \dots, a_n . Числа не превосходят 10^{18} . Обратите внимание, что для хранения таких чисел в программе на C++ вам потребуется тип `long long`. В программе на PascalABC такой тип называется `Int64`.

Формат выходных данных

Программа должна вывести через пробел n натуральных чисел b_1, b_2, \dots, b_n . Число b_i должно быть другом числа a_i .

Если вы программируете на Python, то заменить перенос строки на пробел в функции `print` можно при помощи именованного параметра `end`, например `print(a, end=' ')`.

Методика проверки

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется. В первых 5 тестах $n \leq 10$, $a_i \leq 1000$. В следующих пяти тестах $n \leq 10^5$, $a_i \leq 10^6$. В последних 10 тестах $a_i \leq 10^{18}$.

Примеры

Пример №1

Стандартный ввод
10
1 2 3 4 5 6 7 8 9 10
Стандартный вывод
3 6 1 9 12 2 15 18 4 21

Решение

Можно заметить, что соответствующие числа в различных множествах различаются примерно в два раза. Поэтому требуемые формулы можно получить, умножая или деля заданное число на 2. В зависимости от четности к результату надо будет прибавить некоторую константу. С использованием свойств целочисленной арифметики программу можно упростить.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 n=int(input())
2 for x in map(int,input().split()):
3     if x%3!=0:
4         print(3*(x-x//3),end=' ')
5     else:
6         print((x-1)//2,end=' ')

```

Задача I.1.2.3. Алфавит (20 баллов)

Темы: строки, символы, структуры данных.

Панграммой называется строка, в которой присутствуют все буквы алфавита. Однако при этом строка может содержать пробелы. У Алисы есть строка, состоящая из строчных и заглавных символов латиницы. Она хочет убедиться, что эта строка является панграммой. Алиса действует следующим образом. Для начала она удаляет из строки все пробелы, а также заменяет все заглавные буквы на строчные. Далее она вырезает из строки символы и составляет из них вторую строку по такому алгоритму. Она перебирает последовательно все буквы алфавита от «a» до «z» и пытается найти самое первое вхождение этой буквы в строке. Если вхождение нашлось, то Алиса вырезает из строки эту букву и добавляет ее справа ко второй строке и переходит к следующей букве.

Например, из строки «**A Bra Kada Bra**» будет получена строка «**abrakadabra**», далее из нее последовательно будут вырезаны самые первые вхождения букв **a**, **b**, **d**, **k**, **r**, после чего она превратится в строку **aaabra**.

Вы должны написать программу, которая определит содержимое обеих строк после преобразований Алисы.

Формат входных данных

На вход подается одна строка, содержащая не более 200 строчных и заглавных символов латиницы и пробелов. Гарантируется, что хотя бы один из символов латиницы входит в строку несколько раз.

Формат выходных данных

Программа должна вывести обе полученные строки без пробелов.

Методика проверки

Программа проверяется на 10 тестах. Прохождение каждого теста оценивается в 2 балла. Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
A Bra Kada Bra
Стандартный вывод
aaabra
abdkr

Решение

В этой задаче проверяется знание некоторых функций Python для работы со строками. В частности, используются: метод `replace()` для удаления всех пробелов, метод `lower()` для смены регистра, метод `join()` для объединения символов в строку. Для хранения уникальных символов в приведенном решении используются множества.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 unique=set()
2 other=''
3 for x in input().replace(' ','').lower():
4     if x in unique:
5         other+=x

```

```
6     else:
7         unique.add(x)
8     print(other)
9     print(''.join(sorted(list(unique))))
```

Задача I.1.2.4. Велосипедисты (20 баллов)

Темы: математика, уравнения, двоичный поиск.

Два велосипедиста выехали одновременно из пункта А по одной дороге с различными скоростями u и v метров в секунду. Через t секунд им вдогонку выехал электромобиль и через некоторое время обогнал одного, а затем и другого велосипедиста. При этом интервал между моментами обгона составил d секунд.

Вы должны написать программу, которая вычислит скорость движения электромобиля.

Формат входных данных

На вход через пробел подаются четыре натуральных числа: u , v , t , d . При этом $u \neq v$; $u, v \leq 50$; $t, d \leq 10000$. Гарантируется, что введенные данные будут таковы, что ответ не превысит 200.

Формат выходных данных

Программа должна вывести одно вещественное число — скорость электромобиля.

Методика проверки и пояснение к тесту

Ответ участника считается верным, если он отличается от ответа жюри не более чем на 10^{-8} .

Программа проверяется на 10 тестах. Прохождение каждого теста оценивается в 2 балла. При этом в первых 5 тестах ответ обязательно будет целым числом. Тест из условия задачи при проверке не используется.

Рассмотрим тест из примера. Утверждается, что для заданных параметров ответом является 12. Проверим это. Можно вычислить, что электромобиль, двигаясь со скоростью 12 м/с, обгонит более медленного велосипедиста на расстоянии 480 метров, а более быстрого на расстоянии в 1200 метров. Действительно, электромобиль преодолет 480 и 1200 метров за 40 и 100 секунд соответственно. Таким образом, интервал между моментами обгона действительно равен 60. Велосипедисты до моментов обгона будут двигаться на 20 секунд дольше, по 60 и 120 секунд соответственно. И, проверив пройденное расстояние $60 \cdot 8 = 480$ и $120 \cdot 10 = 1200$, убедимся, что ответ верен. **Обратите внимание, что это пояснение лишь показывает, как проверить правильность ответа, но не является алгоритмом решения.**

Примеры

Пример №1

Стандартный ввод
10 8 20 60
Стандартный вывод
12.0

Решение

Обозначим скорость электромобиля за x . Тогда до старта электромобиля велосипедист проехал tu метров. Электромобиль догнал велосипедиста через $\frac{tu}{x-u}$ секунд. Предполагая, что второй велосипедист двигался быстрее чем первый, получим уравнение:

$$\frac{tv}{x-v} - \frac{tu}{x-u} = d$$

Уравнение можно решить аналитически, а также тернарным или бинарным поиском, учитывая, что с ростом x интервал между моментами обгона будет сокращаться.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 u,v,t,d=map(int,input().split())
2 if u>v:
3     u,v=v,u
4 d/=t
5 b=(d+1)*v+(d-1)*u
6 print((b+(b*b-4*d*d*u*v)**0.5)/(2*d))

```

Ниже представлено решение бинарным поиском на языке Python 3.

```

1 u,v,t,d=map(int,input().split())
2 if u>v:
3     u,v=v,u
4 left=max(u,v)
5 right=200
6 for i in range(100):
7     mid=(left+right)/2
8     if t*v/(mid-v)-t*u/(mid-u)>d:
9         left=mid
10    else:
11        right=mid
12 print(right)

```

Задача I.1.2.5. Много единиц (20 баллов)

Темы: системы счисления, битовые операции, реализация.

Алиса учится работать с двоичными числами. Она уже поняла, что число в двоичной записи получается в несколько раз длиннее, чем в десятичной. А еще она

поняла, что нули писать дольше, чем единицы. И теперь ее любимые числа — это те, двоичная запись которых содержит как можно больше единиц. Алисе дали задание — выбрать одно произвольное число из заданного закрытого интервала $[a; b]$ и перевести его в двоичную запись. И теперь Алиса просит, чтобы вы написали программу, которая найдет в этом интервале число, двоичная запись которого содержит наибольшее количество единиц. Если таких чисел будет несколько, то Алиса будет рада любому из них.

Формат входных данных

На вход через пробел подаются два натуральных числа a и b . При этом $1 \leq a \leq b \leq 10^{18}$. Обратите внимание, что для хранения таких чисел в программе на C++ вам потребуется тип **long long**. В программе на PascalABC такой тип называется **Int64**.

Формат выходных данных

Программа должна вывести одно целое число из заданного диапазона, двоичная запись которого содержит наибольшее количество единиц. Само число следует выводить в десятичной записи.

Методика проверки и пояснение к тесту

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. При этом в первых пяти тестах $1 \leq a \leq b \leq 1000$. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
150 200
Стандартный вывод
191

Пример №2

Стандартный ввод
1 255
Стандартный вывод
255

Пример №3

Стандартный ввод
127 200
Стандартный вывод
127

Решение

Рассмотрим некоторое число в двоичной записи, которое содержит k единиц. Чтобы получить ближайшее сверху число в записи которого $k + 1$ единица, требуется заменить на единицу самый правый ноль. Таким образом решение задачи состоит в том, чтобы взять число из начала интервала и заменять самые правые нули до тех пор, пока результат не превысит правую границу интервала.

Реализовать такой алгоритм можно, представляя число в виде строки, а также при помощи арифметических или битовых операций.

Пример программы-решения

Ниже представлено поиском на языке Python 3 с использованием строк.

```

1 a,b = map(int,input().split())
2 a=list(bin(a)[2:])
3 b=list(bin(b)[2:])
4 while len(a)<len(b):
5     a=['0']+a
6 for i in range(len(a)-1,-1,-1):
7     a[i]='1'
8     if a>b:
9         a[i]='0'
10        break
11 print(int(''.join(a),2))

```

Ниже представлено поиском на языке Python 3 с использованием арифметических операций.

```

1 a,b = map(int,input().split())
2 p=1
3 while a+p<=b:
4     if (a//p)%2==0:
5         a+=p
6     p*=2
7 print(a)

```

Ниже представлено поиском на языке Python 3 с использованием битовых операций.

```

1 a,b = map(int,input().split())
2 while a | (a+1) <= b:
3     a = a | (a+1)
4 print(a)

```

Третья попытка. Задачи 8–11 класса

Задача I.1.3.1. Урожай (20 баллов)

Темы: задачи для начинающих.

Дядя Саша с сыном Колей копают картошку. Урожай выдался, как всегда, отменным, и они накопили n мешков. Дядя Саша пригнал грузовичок, в который может

поместиться не более a мешков картошки, а в Колин грузовичок поместится не более b мешков. Урожай они хотят поделить поровну. Если количество мешков не будет делиться на 2, то лишний мешок на правах старшего заберет дядя Саша. Вместе с тем, никто не сможет забрать мешков больше, чем поместится в его грузовик. И, конечно же, они не оставят ни одного мешка на поле.

Напишите программу, которая определит, сколько мешков увезет дядя Саша, а сколько Коля.

Формат входных данных

На вход подаются натуральные числа n , a и b по одному числу в строке. Числа не превосходят 1000. Гарантируется, что $n \leq a + b$.

Формат выходных данных

Программа должна вывести в одной строке через пробел два числа — количество мешков, которое увезут дядя Саша и Коля на своих грузовичках.

Методика проверки и пояснение к тестам

Программа проверяется на 10 тестах. Прохождение каждого теста оценивается в 2 балла. Тесты из условия задачи при проверке не используются.

В первом тесте 59 мешков будут разделены почти поровну — 30 мешков дяде Саше и 29 — Коле. Во втором тесте Коля не сможет увезти причитающиеся ему 29 мешков и отдаст лишнее дяде Саше.

Примеры

Пример №1

Стандартный ввод
59
35
40
Стандартный вывод
30 29

Пример №2

Стандартный ввод
59
41
25
Стандартный вывод
34 25

Решение

Данную задачу можно решать различными способами. В предлагаемом варианте решения мешки сначала делятся поровну, а далее, в случае необходимости, перемещаются из одного грузовика в другой.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  n=int(input())
2  a=int(input())
3  b=int(input())
4  x, y = (n+1)//2, n//2
5  if x>a:
6      y+=x-a
7      x=a
8  if y>b:
9      x+=y-b
10     y=b
11  print(x,y)

```

Задача I.1.3.2. Скайраннинг (20 баллов)

Темы: задачи для начинающих.

Скайраннингом называется бег в горной местности по неподготовленным трассам, которые обязательно проходят через одну или несколько вершин. Важной характеристикой маршрута в скайраннинге является набор высоты, который равен сумме перепада высот на всех участках подъема. Например, если на маршруте имеется три участка подъема, причем конечная точка каждого участка выше начальной на 200 метров, то набор высоты на маршруте будет равен 600 метров. Кроме того, весь маршрут может быть поделен на высотные зоны. В нашей задаче мы будем рассматривать две высотные зоны: ниже 2000 метров и выше 2000 метров. В этом случае все параметры, в том числе и набор высоты, рассчитываются для разных высотных зон.

Рассмотрим такой пример. Пусть профиль трассы содержит семь точек с высотами 1200, 2300, 2100, 2900, 3100, 1000, 1800 метров. На этой трассе есть четыре участка подъема: (1200; 2300), (2100; 2900), (2900; 3100), (1000;1800). На первом участке подъема 800 метров набирается в первой высотной зоне и 300 метров во второй. На втором и третьем участках набирается по 800 и 200 метров соответственно во второй высотной зоне. На четвертом участке набирается 800 метров в первой зоне. Таким образом в первой высотной зоне набирается 1600 метров, а во второй — 1300 метров.

Ваша задача — написать программу, которая по заданному профилю трассы найдет набор высоты для двух высотных зон: ниже 2000 и выше 2000 метров.

Формат входных данных

На вход в первой строке подается одно натуральное число n — количество точек в профиле высоты. $2 \leq n \leq 100$. Во второй строке через пробел записаны n целых чисел a_1, \dots, a_n , задающих высоту каждой точки. $-416 \leq a_i \leq 8848$.

Формат выходных данных

Программа должна вывести в одной строке через пробел два числа — набор высоты для первой и второй высотной зоны.

Методика проверки

Программа проверяется на 10 тестах. Прохождение каждого теста оценивается в 2 балла. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
7 1200 2300 2100 2900 3100 1000 1800
Стандартный вывод
1600 1300

Пример №2

Стандартный ввод
5 2000 2675 2675 1215 -416
Стандартный вывод
0 675

Решение

В этой задаче необходимо перебрать все участки трассы, на которых происходит подъем, и рассмотреть три случая: весь участок ниже 2000 метров, весь участок выше 2000 метров, участок проходит через высоту в 2000 метров.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  n = int(input())
2  h = list(map(int, input().split()))
3  a1, a2 = 0, 0
4  for i in range(1, n):
5      if h[i] > h[i-1]:
6          if h[i-1] >= 2000:
7              a2 += h[i] - h[i-1]
8          elif h[i] <= 2000:
9              a1 += h[i] - h[i-1]
10         else:
11             a1 += 2000 - h[i-1]
12             a2 += h[i] - 2000
13  print(a1, a2)

```

Задача I.1.3.3. Шоколадка (20 баллов)

Темы: перебор, сортировки, структуры данных.

У Аленки есть шоколадка прямоугольной формы, размером $n \times m$ долек. Аленка разламывает ее на две части по вертикали или по горизонтали и съедает одну из двух частей. Если Аленка чувствует, что не наелась, то она снова может разломить оставшийся кусок на две части и съесть одну из них, и так далее. Всю шоколадку Аленка есть не будет, а оставит про запас, как минимум, одну дольку.

Производителям стало известно о такой привычке девочки и они захотели узнать, какое количество долек может быть съедено при таком алгоритме поедания шоколада. Они хотят, чтобы вы написали программу, которая по известному размеру шоколадки найдет все возможные количества съеденных долек и выведет их в порядке возрастания.

Рассмотрим такой пример. Шоколадка имеет размер 3×3 . Тогда Аленка может разломить ее на две части 3×1 и 3×2 . Таким образом, она сможет съесть 3 или 6 долек и остановиться. Но также Аленка сможет съесть кусок из 6 долек, а от оставшегося отломить 1 или 2 дольки и съесть еще и их. Таким образом, она сможет съесть 7 или 8 долек. Наконец, она сможет съесть кусок 3×1 , а от оставшегося куска 3×2 отломить и съесть еще 2 дольки, тогда количество съеденных долек будет равно 5. Очевидно, что съесть 1, 2 или 4 дольки Аленка не сможет.

Формат входных данных

На вход в одной строке подается два натуральных числа n и m — размеры шоколадки. $1 \leq n, m \leq 100$; $n + m \geq 3$.

Формат выходных данных

Программа должна вывести в одной строке через пробел все числа, являющиеся ответами к задаче. Числа должны выводиться в порядке возрастания без повторов.

Если вы программируете на Python, то убрать перенос строки в функции `print` можно при помощи именованного параметра `end`, например, `print(a, end=' ')`.

Методика проверки

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
3 3
Стандартный вывод
3 5 6 7 8

Решение

Заметим, что у Аленки всегда остается один кусок шоколадки прямоугольной формы. Поэтому можно перебрать все прямоугольные области меньшего размера и найти разность между количеством долек во всей шоколадке и в прямоугольниках. Полученные значения могут повторяться, поэтому потребуется отбросить одинаковые числа. Для этого можно использовать множество или булевский массив.

Пример программы-решения

Ниже представлено решение на языке Python 3.

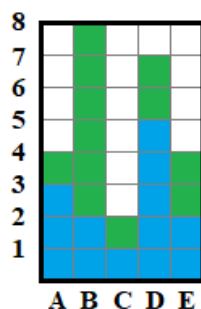
```

1 n, m = map(int, input().split())
2 k=[False]*(n*m)
3 for i in range(1,n+1):
4     for j in range(1,m+1):
5         k[n*m-i*j]=True
6 for i in range(1,n*m):
7     if k[i]:
8         print(i, end=' ')

```

Задача I.1.3.4. Стекочная гистограмма (20 баллов)

Темы: реализация, структуры данных.



По определению Википедии, стекочная гистограмма отображает зоны, представляющие свойства одной категории, друг поверх друга. Имеется n категорий с двумя свойствами. Требуется нарисовать стекочную гистограмму без подписей в консоли с использованием символов «.*#», как показано в примере.

Формат входных данных

В первой строке на вход подается одно натуральное число n — количество категорий в гистограмме. $2 \leq n \leq 50$. Во второй строке через пробел записано n натуральных чисел a_1, \dots, a_n — значения первого свойства для каждой категории. В третьей строке аналогично записаны натуральные числа b_1, \dots, b_n — значения второго свойства. $1 \leq a_i, b_i \leq 20$.

Формат выходных данных

Требуется вывести в консоль текстовое представление гистограммы. Текст должен содержать ровно $\max(a_i + b_i)$ строк, каждая строка должна содержать ровно n символов и завершаться переводом строки. Если рассматривать текст по столбцам, то i -тый столбец должен содержать снизу ровно a_i решеток, далее b_i звездочек, а еще выше — точки.

Методика проверки и пояснение к тесту

Программа проверяется на 5 тестах. Прохождение каждого теста оценивается в 4 балла. Тест из условия задачи при проверке не используется.

На картинке изображена гистограмма из примера. Синие квадратики соответствуют решеткам, зеленые — звездочкам, белые — точкам. Количество строк равно высоте самого высокого столбика.

Примеры

Пример №1

Стандартный ввод
5
3 2 1 5 2
1 6 1 2 2
Стандартный вывод
.*. . .
.*. *.
.*. *.
.*. #.
** . #*
#* . #*
####
#####

Решение

Для решения этой задачи можно создать двумерный символьный массив и заполнить его требуемыми символами. Строки надо будет выводить на экран в обратном порядке. Можно обойтись и без двумерного массива. Для вывода можно использовать два вложенных цикла и определять вид символа по номеру строки и столбца. Чтобы определить количество строк, потребуется найти максимум из сумм значений по категориям.

Пример программы-решения

Ниже представлено решение на языке Python 3 с двумерным массивом.

```
1 n=int(input())
2 a=list(map(int,input().split()))
```

```

3 b=list(map(int,input().split()))
4 h=max([a[i]+b[i] for i in range(n)])
5 s=[['.']*n for i in range(h)]
6 for i in range(n):
7     for j in range(a[i]):
8         s[j][i]='#'
9     for j in range(a[i],a[i]+b[i]):
10        s[j][i]='*'
11 for x in reversed(s):
12     print(''.join(x))

```

Ниже представлено решение на языке Python 3 без двумерного массива.

```

1 n=int(input())
2 a=list(map(int,input().split()))
3 b=list(map(int,input().split()))
4 h=max([a[i]+b[i] for i in range(n)])
5 for j in range(h):
6     for i in range(n):
7         if h-j <= a[i]:
8             print('#',end='')
9         elif h-j <= a[i]+b[i]:
10            print('*',end='')
11        else:
12            print('.',end='')
13    print()

```

Задача I.1.3.5. Префиксный код (20 баллов)

Темы: математика, строки, перебор, структуры данных.

Префиксное кодирование является очень распространенным способом сжатия информации для ее хранения и передачи. Каждый символ кодируется некоторым кодовым словом — строкой из нулей и единиц. Кодовые слова могут иметь различную длину, но при этом должно выполняться следующее свойство: ни одно кодовое слово не начинается с другого кодового слова. Например, множество кодовых слов $\{00, 011, 1000, 11\}$ подходит для префиксного кодирования, а $\{00, 101, 11, 1010\}$ — нет, так как 101 является началом 1010.

От вас требуется написать программу, которая найдет, какое максимальное количество кодовых слов заданной длины m можно добавить к некоторому уже построенному множеству слов для префиксного кодирования. Например $m = 4$, и уже построено множество $\{10, 001, 000, 0111, 11111, 11000, 111101\}$. Без нарушения условия префиксного кодирования можно добавить следующие слова длины 4: $\{0100, 0101, 0110, 1101, 1110\}$. Таким образом, ответ равен 5.

Формат входных данных

В первой строке на вход подается два натуральных числа n и m — количество уже известных кодовых слов и длина новых кодовых слов. $m \leq 60$. Во второй строке через пробел записано n кодовых слов. Все слова состоят из нулей и единиц и удовлетворяют условию префиксного кодирования. Суммарная длина всех слов не превосходит 10^6 .

Формат выходных данных

Требуется вывести одно целое число — ответ к задаче. Ответ может быть равным нулю.

Методика проверки

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. При этом в 3 тестах $m \leq 10$ и $n \leq 100$, еще в 3 тестах длина всех заданных слов не превосходит m , еще в 4 тестах длина всех заданных слов не меньше чем m . Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
7 4 10 001 000 0111 11111 11000 111101
Стандартный вывод
5

Решение

Сразу отметим, что существует 2^m различных кодовых слов длины m . Каждое кодовое слово, ранее добавленное в код, не позволит использовать некоторое количество возможных слов длины m . Так, если некоторое уже добавленное слово a имеет длину k , меньшую чем m , то любое слово, полученное приписыванием к a справа $m - k$ символов, будет недоступно. Существует 2^{m-k} способов дописать справа к a некоторую последовательность из нулей и единиц, поэтому такую величину надо будет вычесть из ответа.

Если кодовое слово имеет длину большую m , то его префикс длины m также нельзя использовать. Однако различные слова могут иметь одинаковые префиксы, поэтому среди них надо найти все различные, например, при помощи множеств.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 n,m=map(int,input().split())
2 ans=2**m
3 wset=set()
4 for x in input().split():
5     if len(x)>m:
6         wset.add(x[:m])
7     else:
8         ans-=2**(m-len(x))
9 print(ans-len(wset))

```

Четвертая попытка. Задачи 8–11 класса

Задача I.1.4.1. Кросс нации (20 баллов)

Темы: математика, задачи для начинающих.

Некоторые измеряют скорость в километрах, пройденных за час. А вот люди, занимающиеся бегом, измеряют темп бега в секундах на километр, то есть указывают количество минут и секунд, за которые они пробегают ровно 1 километр. Например, скорость в 12,5 км/ч соответствует темпу бега в 4 минуты 48 секунд.

Ваша задача — написать программу, которая определит темп бега по скорости.

Формат входных данных

На вход подается единственное число u — скорость в километрах за час. Скорость является вещественным положительным числом не более чем с двумя знаками после точки. $5 \leq u \leq 50$.

Формат выходных данных

Вывести через пробел два целых числа — время в минутах и секундах, за которое бегун пробегает 1 километр. Секунды должны быть округлены до ближайшего целого числа по стандартным правилам.

Методика проверки

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
12.5
Стандартный вывод
4 48

Пример №2

Стандартный ввод
15.03
Стандартный вывод
4 0

Пример №3

Стандартный ввод
15.04
Стандартный вывод
3 59

Решение

В одном часе 3600 секунд. Поэтому, если бегун пробегает за 3600 секунд a километров, то один километр он пробежит за $\frac{3600}{a}$ секунд. Это число надо округлить и привести к целому типу, далее выделить из него минуты и секунды, используя деление и остаток от деления на 60.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 a=float(input())
2 a=int(round(3600/a))
3 print(a//60,s%60)

```

Задача I.1.4.2. Говорящий конь (20 баллов)

Темы: задачи для начинающих, реализация.

Говорящий конь Юлий шел по дороге и встретил трех богатырей.

- *Здравствуй, богатырь номер 1,— сказал говорящий конь Юлий.*
- *Здравствуй, говорящий конь Юлий,— сказал богатырь номер 1.*
- *Здравствуй, богатырь номер 2,— сказал говорящий конь Юлий.*
- *Здравствуй, говорящий конь Юлий,— сказал богатырь номер 2.*
- *Здравствуй, богатырь номер 3,— сказал говорящий конь Юлий.*
- *Здравствуй, говорящий конь Юлий,— сказал богатырь номер 3.*
- *Здравствуй, лошадь богатыря номер 1,— сказал говорящий конь Юлий.*
- *Здравствуй, говорящий конь Юлий,— сказала лошадь богатыря номер 1.*
- *Здравствуй, лошадь богатыря номер 2,— сказал говорящий конь Юлий.*
- *Здравствуй, говорящий конь Юлий,— сказала лошадь богатыря номер 2.*
- *Здравствуй, лошадь богатыря номер 3,— сказал говорящий конь Юлий.*
- *Здравствуй, говорящий конь Юлий,— сказала лошадь богатыря номер 3.*
- *До свиданья, богатырь номер 1,— сказал говорящий конь Юлий.*
- *До свиданья, говорящий конь Юлий,— сказал богатырь номер 1.*
- *До свиданья, богатырь номер 2,— сказал говорящий конь Юлий.*
- *До свиданья, говорящий конь Юлий,— сказал богатырь номер 2.*
- *До свиданья, богатырь номер 3,— сказал говорящий конь Юлий.*

- До свиданья, говорящий конь Юлий, – сказал богатырь номер 3.
- До свиданья, лошадь богатыря номер 1, – сказал говорящий конь Юлий.
- До свиданья, говорящий конь Юлий, – сказала лошадь богатыря номер 1.
- До свиданья, лошадь богатыря номер 2, – сказал говорящий конь Юлий.
- До свиданья, говорящий конь Юлий, – сказала лошадь богатыря номер 2.
- До свиданья, лошадь богатыря номер 3, – сказал говорящий конь Юлий.
- До свиданья, говорящий конь Юлий, – сказала лошадь богатыря номер 3.

И говорящий конь Юлий пошел по дороге дальше.

Маленькому Ванечке очень нравится эта сказка и он любит слушать ее в разных вариантах. Ему особенно нравится вариант, в котором говорящий конь Юлий встречает на дороге 40 разбойников, но почему-то усталые взрослые не хотят ее рассказывать.

Напишите программу, которая по номеру предложения сказки о встрече говорящего коня Юлия с 40 разбойниками будет выводить это предложение. Поскольку родители считают, что Ванечка с самого юного возраста должен изучать иностранные языки, от вас требуется вывести ответ по-английски.

Формат входных данных

Одно натуральное число n — номер предложения сказки. $1 \leq n \leq 322$.

Формат выходных данных

Вывести требуемое предложение сказки в одной строке. Слова должны быть разделены ровно одним пробелом. Перед знаками препинания пробел не ставится. Перевод всех предложений и формат вывода смотрите в примерах.

Методика проверки

Программа проверяется на 50 тестах. Прохождение каждого теста оценивается в 0,4 балла. **Первые 10 контрольных тестов совпадают с тестами из условия задачи.**

Примеры

Пример №1

Стандартный ввод
1
Стандартный вывод
Talking horse Julius was walking along the road and met 40 robbers.

Пример №2

Стандартный ввод
2
Стандартный вывод
- Hello, robber number 1,- said talking horse Julius.

Пример №3

Стандартный ввод
3
Стандартный вывод
- Hello, talking horse Julius,- said robber number 1.

Пример №4

Стандартный ввод
82
Стандартный вывод
- Hello, horse of robber number 1,- said talking horse Julius.

Пример №5

Стандартный ввод
83
Стандартный вывод
- Hello, talking horse Julius,- said horse of robber number 1.

Пример №6

Стандартный ввод
162
Стандартный вывод
- Goodbye, robber number 1,- said talking horse Julius.

Пример №7

Стандартный ввод
163
Стандартный вывод
- Goodbye, talking horse Julius,- said robber number 1.

Пример №8

Стандартный ввод
242
Стандартный вывод
- Goodbye, horse of robber number 1,- said talking horse Julius.

Пример №9

Стандартный ввод
243
Стандартный вывод
- Goodbye, talking horse Julius,- said horse of robber number 1.

Пример №10

Стандартный ввод
322
Стандартный вывод
And talking horse Julius went on along the road.

Решение

Текст сказки состоит из 10 различных предложений. Каждое предложение встречается в определенном диапазоне номеров на четных или нечетных позициях. Исходя из этого, можно записать программу через одну инструкцию ветвления с 10 вариантами работы.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  n=int(input())
2  if n==1:
3      print('Talking horse Julius was walking along the road and met 40 robbers.')
4  elif n<82 and n%2==0:
5      print('- Hello, robber number '+str(n//2)+',- said talking horse Julius.')
6  elif n<82:
7      print('- Hello, talking horse Julius,- said robber number '+str(n//2)+'.')
8  elif n<162 and n%2==0:
9      print('- Hello, horse of robber number '+str((n-80)//2)+',- said talking horse
   ↪ Julius.')
10 elif n<162:
11     print('- Hello, talking horse Julius,- said horse of robber number
   ↪ '+str((n-80)//2)+'.')
12 elif n<242 and n%2==0:
13     print('- Goodbye, robber number '+str((n-160)//2)+',- said talking horse
   ↪ Julius.')
14 elif n<242:
15     print('- Goodbye, talking horse Julius,- said robber number
   ↪ '+str((n-160)//2)+'.')
16 elif n<322 and n%2==0:
17     print('- Goodbye, horse of robber number '+str((n-240)//2)+',- said talking
   ↪ horse Julius.')
18 elif n<322:
19     print('- Goodbye, talking horse Julius,- said horse of robber number
   ↪ '+str((n-240)//2)+'.')
20 else:
21     print('And talking horse Julius went on along the road.')
```

Задача I.1.4.3. Нумерация (20 баллов)

Темы: задачи для начинающих, реализация, структуры данных.

Власти города Байтленда решили построить новый микрорайон, который будет состоять из одной длинной улицы. Процесс постройки домов будет выглядеть следующим образом. Сначала будет построен самый первый дом, который получит номер 0. (Разумеется, в Байтленде все нумерации начинаются с нуля.) Далее все дома будут пристраиваться слева или справа от существующей застройки. Дома будут получать номера в порядке их ввода в эксплуатацию.

Рассмотрим пример. Пусть дом номер 1 был построен слева от дома номер 0. Тогда нумерация домов слева направо будет выглядеть как 1 0. Далее был построен дом номер 2 слева от существующих. Теперь дома на улице будут иметь номера 2 1 0. Далее был построен дом номер 3 справа от существующих. Теперь на улице будут стоять дома с номерами 2 1 0 3. Наконец, если дом номер 4 будет построен слева, а дома с номерами 5 и 6 справа, то последовательность номеров домов превратится в 4 2 1 0 3 5 6.

На улице построили n домов с номерами от 0 до $n - 1$. Для каждого дома с ненулевым номером нам стало известно с какой стороны от существующих он был построен. Теперь ваша задача — написать программу, которая перечислит номера домов в порядке движения по улице слева направо.

Формат входных данных

В первой строке на вход подается число n — количество построенных домов. $2 \leq n \leq 400000$. Во второй строке записана последовательность из $n - 1$ символов «L» или «R» (без кавычек). Символ «L» в i -той позиции означает, что дом с номером i был построен слева от предыдущих, а символ «R» — справа.

Формат выходных данных

Вывести через пробел в одной строке последовательность из n чисел — нумерацию домов после завершения строительства.

Методика проверки и описание тестов

Программа проверяется на 10 тестах. Прохождение каждого теста оценивается в 2 балла. Тест из условия задачи при проверке не используется.

В первых пяти тестах $n \leq 100$.

Примеры

Пример №1

Стандартный ввод
7 LLRLRR
Стандартный вывод
4 2 1 0 3 5 6

Решение

По условию задачи требуется составить последовательность чисел, добавляя их слева или справа к существующим. Для решения можно использовать структуру данных дек или изменить порядок выполнения команд добавления элементов. Требуемую последовательность можно получить, выполнив два прохода по строке, задающей команды. Во время первого прохода будем добавлять числа слева от нуля. Чтобы получить нужный порядок, по строке потребуется пройти справа налево. Далее необходимо вывести ноль и сделать второй проход для вывода чисел справа от нуля.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 n=int(input())
2 x=input()
3 ans=''
4 for i in range(n-1,0,-1):
5     if x[i-1]=='L':
6         print(i,end=' ')
7 print(0,end=' ')
8 for i in range(1,n):
9     if x[i-1]=='R':
10        print(i,end=' ')

```

Задача I.1.4.4. Космическая связь (20 баллов)

Темы: реализация.

В этой задаче от вас потребуется написать программу для составления расписания сеансов связи со спутником. Каждый сеанс заключается в обмене короткими сообщениями, поэтому мы считаем, что он происходит мгновенно. Поскольку ресурсы оборудования ограничены, следует стремиться к минимизации количества сеансов. Вместе с тем, интервал времени между сеансами не должен превышать d миллисекунд. Кроме того, существуют промежутки времени, в течении которых связь невозможна. При этом на границах промежутка мгновенный сеанс связи возможен. Расписание составляется на t миллисекунд. Первый сеанс должен обязательно состояться в момент 0, а последний — в момент t .

Рассмотрим пример. Пусть $t = 100$, $d = 20$ и задано 3 промежутка недоступности связи: (5;25), (27;40), (75;90). Тогда потребуется восемь сеансов связи, которые можно провести в моменты времени 0, 5, 25, 45, 65, 75, 90, 100, 5, 25, 45, 65, 75, 90, 100. Конкретное расписание может быть другим, но в любом случае количество сеансов не может быть меньше 8.

Ваша программа должна по имеющейся информации найти минимальное возможное количество сеансов связи.

Формат входных данных

В строке 1 через пробел записаны 3 натуральных числа n , d и t — количество интервалов недоступности связи, максимальный интервал между сеансами

и время, на которое составляется расписание. $n \leq 200000$, $d, t \leq 10^9$. Далее в n строках заданы по два целых неотрицательных числа a_i и b_i — начало и конец каждого интервала недоступности связи. $b_i - a_i \leq d$. Интервалы недоступности связи не пересекаются, каждый следующий интервал начинается строго после окончания предыдущего. $0 \leq a_1 < b_1 < a_2 < b_2 < \dots < a_n < b_n \leq t$.

Формат выходных данных

Вывести число — количество сеансов связи в графике.

Методика проверки

Программа проверяется на 25 тестах. Прохождение каждого теста оценивается в 0,8 балла. Тест из условия задачи при проверке не используется.

В первых 10 тестах $t \leq 1000$. В следующих 10 тестах $t \leq 10^6$.

Примеры

Пример №1

Стандартный ввод
3 20 100
5 25
27 40
75 90
Стандартный вывод
8

Решение

Для решения этой задачи можно использовать идею жадного алгоритма. Каждый следующий сеанс связи будем проводить как можно позже. Если время предыдущего сеанса было равно t , то следующий сеанс будем проводить в момент $t + d$. Однако, если $t + d$ попадет внутрь некоторого недоступного интервала $[a; b]$, то время сеанса потребуется сместить до левой границы интервала.

Для получения полных баллов за решение требуется составить алгоритм линейной сложности относительно количества недоступных для связи интервалов. Таким образом, на каждом шаге цикла будет рассматриваться один интервал, который будем называть текущим.

Решение этой задачи требует аккуратной реализации. Инвариантом предлагаемого решения является утверждение о том, что t никогда не попадет внутрь некоторого недоступного интервала, и $t + d$ будет больше, чем правая граница текущего интервала, что гарантирует правильность работы на следующих итерациях цикла.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
1 n,d,s=map(int,input().split())
2 m=1
3 t=0
4 for i in range(n):
5     a,b=map(int,input().split())
6     k=(a-t)//d
7     t+=d*k
8     m+=k
9     if t+d<b:
10        t=a
11        m+=1
12 print(m+(s-t+d-1)//d)
```

Задача I.1.4.5. Простая задача (20 баллов)

Темы: реализация.

В этой задаче не будет длинного условия и простого решения. Все будет наоборот. Требуется провести непрерывную линию произвольного вида из точки с координатами (x_1, y_1) в точку с координатами (x_2, y_2) так, чтобы минимизировать количество точек на этой линии, в которых хотя бы одна координата является целым числом. Ответом к задаче будет являться количество таких точек. Если начальная или конечная точка линии будет иметь хотя бы одну целочисленную координату, то ее тоже надо учитывать.

Формат входных данных

Каждый тест в этой задаче будет содержать n запросов. $1 \leq n \leq 100$. Натуральное число n будет записано в первой строке. Далее в n строках записаны запросы. Каждый запрос располагается в отдельной строке и состоит из четырех чисел x_1, y_1, x_2, y_2 , которые задают координаты двух точек. Точки не совпадают. Координаты могут быть целыми или вещественными числами не более чем с 2 знаками после точки. Координаты не превосходят 10^9 по абсолютной величине. Если координата является целым числом, то ее запись не содержит десятичной точки.

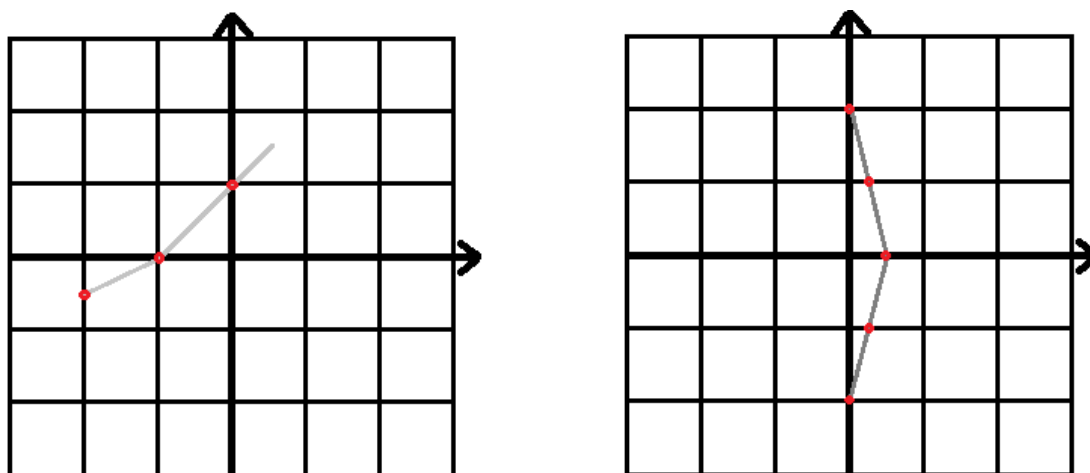
Формат выходных данных

Требуется вывести ответы на запросы по одному ответу в каждой строке.

Методика проверки и пояснение к тесту

Программа проверяется на 10 тестах. Прохождение каждого теста оценивается в 2 балла. Тест из условия задачи при проверке не используется.

Следующие рисунки поясняют ответ к тесту. Обратите внимание, что никакой фрагмент линии не может лежать на сетке, так как в этом случае количество точек с целочисленной координатой будет бесконечно большим.



Примеры

Пример №1

Стандартный ввод
2 -2 -0.5 0.5 1.5 0 -2 0 2
Стандартный вывод
3 5

Решение

Для решения задачи потребуется найти количество горизонтальных и вертикальных прямых целочисленной сетки, которые будут пересекаться нарисованной линией. Поскольку одна вертикальная и одна горизонтальная прямая могут быть пересечены в одной точке, из полученных значений надо будет взять максимальное. Далее надо будет проверить, являются ли координаты концов линии целыми числами, и, в случае необходимости, учесть эти точки в ответе.

Для поиска количества точек пересечения в предлагаемом решении используется функция *bw*. Сначала границы диапазона смещаются на некоторую небольшую величину, чтобы их координаты перестали быть целочисленными. Далее верхняя граница диапазона округляется вниз, а нижняя вверх, после чего вычисляется количество целых чисел в диапазоне.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 import math
2 def bw(a,b):
3     if a>b:
4         a,b=b,a

```

```
5     return max(0,math.floor(b-0.001)-math.ceil(a+0.001)+1)
6
7 n=int(input())
8 for i in range(n):
9     x1,y1,x2,y2=map(float,input().split());
10    print(max(bw(x1,x2),bw(y1,y2))+
11 int((x1==round(x1) or y1==round(y1))+
12 int((x2==round(x2) or y2==round(y2))))))
```

Задачи первого этапа. Физика

Первая попытка. Задачи 8–9 класса

Задача I.2.1.1. Электросамокат (20 баллов)

Темы: сила тока и тепло.

Емкость батареи электросамоката равна $q = 7500 \text{ мА} \cdot \text{ч}$, а ее внутреннее сопротивление составляет $r = 0,05 \text{ Ом}$. Какое тепло выделится на батарее, если ее зарядить полностью постоянным током за время $t = 6 \text{ ч}$?

Ответ дать с точностью до десятых долей кДж.

Решение

Тепло равно:

$$Q = I^2 r t = \left(\frac{q}{t}\right)^2 r t = \frac{r q^2}{t} = 1,7 \text{ кДж.}$$

Точность 0,1 кДж.

Ответ: $1,7 \pm 0,1$.

Задача I.2.1.2. Электросамокат (20 баллов)

Темы: мощность электромотора.

Продолжение задачи [I.2.1.1.](#)

Максимальный ток разрядки батареи электросамоката в десять раз превышает средний ток зарядки батареи. Какова максимальная мощность электромотора самоката, если постоянное напряжение батареи равно $U = 24 \text{ В}$?

Ответ дать с точностью до 1 Вт.

Решение

Максимальная мощность равна:

$$P = 10IU = 10U \frac{q}{t} = 300 \text{ Вт.}$$

Точность 1 Вт.

Ответ: 300 ± 1 .

Задача I.2.1.3. Электросамокат (20 баллов)

Темы: скорость и мощность.

Продолжение задач I.2.1.1 и I.2.1.2.

С какой максимальной скоростью может ехать человек на электросамокате по горизонтальной дороге, если сила сопротивления воздуха, действующая на человека, при этой скорости равна $F = 30$ Н. Силой трения качения пренебречь.

Ответ дать с точностью до 1 м/с.

Решение

Максимальная скорость равна:

$$V_m = \frac{P}{F} = 10 \text{ м/с.}$$

Точность 1 м/с.

Ответ: 10 ± 1 .

Задача I.2.1.4. Электросамокат (20 баллов)

Темы: закон Ома и сила сопротивления.

Продолжение задач I.2.1.1, I.2.1.2 и I.2.1.3.

Среднее сопротивление электрической цепи двигателя электросамоката равно $R = 8,0$ Ом. Какое расстояние проедет с постоянной скоростью электросамокат по горизонтальной дороге, израсходовав весь заряд батареи? Силу сопротивления воздуха считать прямо пропорциональной скорости.

Ответ дать с точностью до 1 км.

Решение

Средняя мощность двигателя равна механической мощности на скорости V :

$$\frac{U^2}{R} = F \frac{V}{V_m} V.$$

Время движения самоката равно:

$$t = q \frac{R}{U}.$$

Отсюда путь равен:

$$S = Vt = q \sqrt{\frac{RV_m}{F}} = 44 \text{ км.}$$

Точность 1 км.

Ответ: 44 ± 1 .

Задача I.2.1.5. Электросамокат (20 баллов)

Темы: сила тяжести.

Продолжение задач I.2.1.1, I.2.1.2, I.2.1.3 и I.2.1.4.

После полной перезарядки батареи человек массой 50 кг поднимается с постоянной скоростью на электросамокате массой 12 кг в гору высотой $h = 50$ м на $S = 1$ км пути. Среднее сопротивление электрической цепи двигателя электросамоката равно $R = 8,0$ Ом. Найти эту скорость электросамоката.

Ответ дать с точностью до 0,1 м/с.

Ускорение свободного падения $g = 10$ Н/кг.

Решение

Средняя мощность двигателя равна механической мощности на скорости V :

$$\frac{U^2}{R} = F \frac{V}{V_m} V + (m + M)g \frac{h}{S/V}.$$

Отсюда получаем квадратное уравнение для скорости V :

$$\frac{F}{V_m} V^2 + (m + M)g \frac{h}{S} V - \frac{U^2}{R} = 0.$$

Его решение: $V = 2,0$ м/с.

Точность 0,1 м/с.

Ответ: $2,0 \pm 0,1$.

Первая попытка. Задачи 10–11 класса**Задача I.2.2.1. Гепард на охоте (20 баллов)**

Темы: равноускоренное движение.

Гепард начинает бросок за жертвой из состояния покоя и бежит по кривой линии с постоянным тангенциальным ускорением a_0 промежуток времени $t_1 = 3,0$ с. Какой путь преодолет гепард, чтобы достичь к концу промежутка времени значения скорости $V_0 = 70$ км/ч?

Ответ дать с точностью до 1 м.

Решение

Путь равен:

$$S = \frac{V_0 t_1}{2} = 29 \text{ м.}$$

Точность 1 м.

Ответ: 29 ± 1 .

Задача I.2.2.2. Гепард на охоте (20 баллов)*Темы: ускорение.*Продолжение задачи [I.2.2.1](#).

Найти радиус окружности, по которой бежит гепард в момент времени 2,0 с. Тангенциальное ускорение равно центростремительному ускорению в этот момент.

Ответ дать с точностью до 1 м.

РешениеВ момент времени t скорость равна $V = a_0 t$.

Радиус окружности равен:

$$R = \frac{V^2}{a_0} = \frac{V_0 t^2}{t_1} = 26 \text{ м.}$$

Точность 1 м.

Ответ: 26 ± 1 .**Задача I.2.2.3. Гепард на охоте (20 баллов)***Темы: скорость.*Продолжение задач [I.2.2.1](#) и [I.2.2.2](#).

После времени $t_1 = 3,0$ с гепард начинает устывать и его тангенциальное ускорение уменьшается по линейному закону $a = a_0 - kt$, где коэффициент $k = a_0^2/V_0$ и время t отсчитывается с момента времени t_1 . Найти максимальную скорость гепарда.

Ответ дать с точностью до 1 км/ч.

Решение

В момент времени $t = \frac{a_0}{k} = t_1$ ускорение равно нулю и скорость достигает максимального значения равно:

$$V = V_0 + a_0 t - k \frac{t^2}{2} = \frac{3}{2} V_0 = 105 \text{ км/ч.}$$

Точность 1 км/ч.

Ответ: 105 ± 1 .**Задача I.2.2.4. Гепард на охоте (20 баллов)***Темы: путь.*Продолжение задач [I.2.2.1](#), [I.2.2.2](#) и [I.2.2.3](#).

Гепард промахивается и постепенно останавливается. Какой путь пробежит гепард с момента времени t_1 до остановки, если ускорение гепарда изменяется по линейному закону из задачи I.2.2.3 и после остановки становится равным нулю?

Ответ дать с точностью до 1 м.

Решение

В момент времени $t_2 = (1 + \sqrt{3}) t_1$ гепард останавливается. Тогда путь гепарда до остановки равен:

$$S = V_0 t_0 + a_0 \frac{t_2^2}{2} - k \frac{t_2^3}{6} = V_0 t_1 (1 + \sqrt{3}) \left[1 + \frac{(1 + \sqrt{3})}{2} - \frac{(1 + \sqrt{3})^2}{6} \right] = 179 \text{ м.}$$

Точность 1 м.

Ответ: 179 ± 1 .

Задача I.2.2.5. Гепард на охоте (20 баллов)

Темы: мощность.

Продолжение задач I.2.2.1, I.2.2.2, I.2.2.3 и I.2.2.4.

Найти максимальную удельную мощность, развиваемую гепардом за все время охоты.

Ответ дать с точностью до 1 Вт/кг.

Решение

Удельная мощность гепарда равна произведению горизонтальной силы на скорость отнесенному к массе гепарда:

$$\frac{P}{m} = a(t)V(t).$$

При движении с уменьшающимся ускорением удельная мощность изменяется по закону:

$$\frac{P}{m} = a(t)V(t) = (a_0 - kt) \left(V_0 + a_0 t - \frac{kt^2}{2} \right).$$

Причем максимальная мощность достигается в момент времени t_1 после начала движения гепарда ($t = 0$):

$$\frac{P_{max}}{m} = a_0 V_0 = \frac{V_0^2}{t_1} = 126 \text{ Вт/кг.}$$

Точность 1 Вт/кг.

Ответ: 126 ± 1 .

Вторая попытка. Задачи 8–9 класса

Задача I.2.3.1. Айсберг (10 баллов)

Темы: плотность.

Столообразный айсберг, плавающий в южной полярной области, ограничен двумя одинаковыми горизонтальными основаниями: надводным и подводным. Айсберг имеет ровные вертикальные боковые стенки и формой похож на кусок мороженого между двумя вафлями. Площадь каждого основания айсберга равна $S = 100 \text{ км}^2$, а его толщина составляет $H = 200 \text{ м}$. Айсберг состоит из пресного льда плотностью $\rho_1 = 920 \text{ кг/м}^3$, в толще которого находятся пузырьки сжатого воздуха, занимающие 10% объема айсберга. Плотность сжатого воздуха равна $\rho_2 = 1,5 \text{ кг/м}^3$. Найти массу айсберга m_0 с точностью до 1 миллиона тонн.

Ответ дать в миллионах тонн.

Решение

Масса айсберга равна:

$$m_0 = \rho_1 V_1 + \rho_2 V_2 = (0,9\rho_1 + 0,1\rho_2)SH = 16563 \text{ миллионов тонн.}$$

Точность 1 миллион тонн.

Ответ: 16563 ± 1 .

Задача I.2.3.2. Айсберг (15 баллов)

Темы: сила Архимеда.

Продолжение задачи [I.2.3.1](#).

Айсберг плавает в морской воде плотностью $\rho = 1025 \text{ кг/м}^3$. Найти высоту h его надводной части с точностью до десятой доли метра.

Решение

Сила тяжести равна силе Архимеда. Отсюда высота надводной части равна:

$$h = H - \frac{m_0}{\rho S} = 38,4 \text{ м.}$$

Точность 0,1 м.

Ответ: $38,4 \pm 0,1$.

Задача I.2.3.3. Айсберг (20 баллов)

Темы: сила тока.

Продолжение задач I.2.3.1 и I.2.3.2.

По всей площади надводного основания айсберга скопилась тонкая прослойка морской воды. Между надводным основанием айсберга и его горизонтальным сечением на уровне поверхности моря существует электрическое напряжение $U = 130$ В. Найти силу тока, протекающего в надводной части айсберга между основанием и сечением. Считать, что линии электрического тока вертикальны. Удельное сопротивление вещества айсберга равно $\rho_0 = 5 \cdot 10^6$ Ом · м.

Ответ дать с точностью до 1 А.

Решение

Сопротивление надводной части айсберга равно $R = \frac{\rho_0 h}{S}$, и силу тока находим из закона Ома:

$$I = \frac{US}{\rho_0 h} = 68 \text{ А.}$$

Точность 1 А.

Ответ: 68 ± 1 .

Задача I.2.3.4. Айсберг (30 баллов)

Темы: тепловой баланс.

Продолжение задач I.2.3.1 и I.2.3.2.

Подводная часть айсберга нагревается теплой морской водой и медленно тает. Рассмотрим этот процесс. Вокруг айсберга образовалась масса m_1 пресной воды при температуре $t_1 = 0, 0^\circ\text{C}$. В результате теплового контакта с теплой морской водой массы m_2 и температурой t установилось тепловое равновесие и температура пресной воды массой m_1 повысилась до $t_2 = 4, 0^\circ\text{C}$. Благодаря течению к пресной воде с массой m_1 поступает следующая порция теплой морской воды массы m_2 с температурой t и устанавливается новое тепловое равновесие с температурой $t_3 = 6, 0^\circ\text{C}$. Найти температуру t теплой морской воды с точностью до $0, 1^\circ\text{C}$. Считать, что морская вода с пресной водой не перемешиваются и они имеют разные теплоемкости.

Решение

Уравнения теплового баланса для первого контакта:

$$m_1 C_1 (t_2 - t_1) = m_2 C_2 (t - t_2)$$

и для второго контакта:

$$m_1 C_1 (t_3 - t_2) = m_2 C_2 (t - t_3).$$

Из уравнений находим температуру t теплой морской воды:

$$t = \frac{t_1 t_3 - t_2^2}{t_3 - 2t_2 + t_1} = 8,0^\circ\text{C}.$$

Точность $0,1^\circ\text{C}$.

Ответ: $8,0^\circ\text{C} \pm 0,1^\circ\text{C}$.

Задача I.2.3.5. Айсберг (25 баллов)

Темы: теплообмен.

Продолжение задачи I.2.3.1.

Процесс таяния айсберга приводит к тому, что его масса $m(\tau)$ уменьшается со временем τ по закону: $m(\tau) = m_0 + b\tau - a\tau^2$, где константы a и b связаны с начальной массой айсберга m_0 и характерным временем $T = 1$ год соотношениями: $b = aT$ и $a = \frac{m_0}{4T^2}$. Найти время жизни айсберга (полностью растает) с точностью до 0,1 года.

Решение

Когда айсберг полностью растает, то его масса обратится в ноль:

$$m_0 + b\tau - a\tau^2 = 0 \text{ и } \tau^2 - T\tau - 4T^2 = 0.$$

Тогда время жизни айсберга равно:

$$\tau_{\text{ж}} = 2,6T = 2,6 \text{ года.}$$

Точность 0,1 года.

Ответ: $2,6 \pm 0,1$.

Вторая попытка. Задачи 10–11 класса

Задача I.2.4.1. Гонки по вертикали (10 баллов)

Темы: динамика материальной точки.

В мотоаттракционе «Гонки по вертикальной стене» человек на мотоцикле движется по внутренней стороне вертикального деревянного цилиндра радиуса $R = 6$ м и высотой $H = 8$ м. Коэффициент трения скольжения резиновой шины по дереву равен $\mu = 0,6$. Ускорение свободного падения $g = 10$ Н/кг. В задачах I.2.4.1, I.2.4.2 и I.2.4.3 считать мотоцикл с человеком материальной точкой. Найти скорость движения мотоциклиста V в горизонтальной плоскости, если она в полтора раза превышает минимально возможную скорость.

Ответ дать с точностью до 1 м/с.

Решение

Из второго закона Ньютона минимально возможная скорость равна $V_0 = \sqrt{gR/\mu}$. Тогда скорость мотоциклиста $V = 1,5\sqrt{gR/\mu} = 15$ м/с.

Точность 1 м/с.

Ответ: 15 ± 1 .

Задача I.2.4.2. Гонки по вертикали (15 баллов)

Темы: КПД.

Найти КПД двигателя мотоцикла, если он движется со скоростью из задачи I.2.4.1. Мощность двигателя мотоцикла составляет 12 л. с. Расход бензина равен 6 л на 100 км пути. Удельная теплота сгорания бензина равна 44 МДж/кг, а плотность бензина составляет 730 кг/м³.

Ответ дать с точностью до 1%.

Решение

КПД равен отношению мощности двигателя к мощности сгорания бензина:

$$\eta = \frac{P}{Q_{\text{уд}}\rho V q_{\text{расход}}} = 31\%.$$

Точность 1%.

Ответ: 31 ± 1 .

Задача I.2.4.3. Гонки по вертикали (20 баллов)

Темы: путь.

Продолжение задачи I.2.4.1.

Найти путь мотоцикла от основания до вершины цилиндра по винтовой линии с шагом между витками (по вертикали) $h = 0,5$ м. Параметры цилиндра даны в задаче I.2.4.1.

Ответ дать с точностью до 1 м.

Решение

Путь равен длине одного витка умноженной на число витков.

$$S = \frac{H}{h} \sqrt{(2\pi R)^2 + h^2} = 603 \text{ м.}$$

Точность 1 м.

Ответ: 603 ± 1 .

Задача I.2.4.4. Гонки по вертикали (25 баллов)

Темы: момент силы.

Продолжение задачи I.2.4.1.

Заменяем человека с мотоциклом на однородный тонкий равнобедренный треугольник, скользящий углами основания по внутренней поверхности цилиндра, найти угол наклона α треугольника к горизонтальной плоскости. Высота треугольника, проведенная к его основанию равна $a = 1,7$ м, а основание треугольника шириной $b = 1,8$ м расположено горизонтально. Скорость углов основания треугольника равна скорости мотоцикла из задачи I.2.4.1. Ускорение свободного падения $g = 10$ Н/кг.

Ответ дать с точностью до десятой доли градуса.

Решение

Центр масс однородного тонкого треугольника находится на его высоте на расстоянии $a/3$ от основания и движется по окружности радиуса:

$$r = \sqrt{R^2 - \left(\frac{b}{2}\right)^2} - \frac{a}{3} \cos \alpha,$$

с угловой скоростью $\omega = V/R$. Запишем равенство моментов центробежной силы и силы тяжести относительно основания треугольника:

$$m\omega^2 r \frac{a}{3} \sin \alpha = mg \frac{a}{3} \cos \alpha.$$

Отсюда получаем уравнение для угла α :

$$\operatorname{tg} \alpha = \frac{gR^2}{V^2 \left[\sqrt{R^2 - \left(\frac{b}{2}\right)^2} - \frac{a}{3} \cos \alpha \right]}.$$

Так как второе слагаемое в знаменателе в десять раз меньше первого, то уравнение можно решить методом последовательных приближений.

Тогда угол равен $\alpha = 16,5^\circ$.

Точность $0,2^\circ$.

Ответ: $16,5^\circ \pm 0,2^\circ$.

Задача I.2.4.5. Гонки по вертикали (30 баллов)

Темы: центробежная сила.

Продолжение задач I.2.4.1 и I.2.4.4.

Найти силу давления одного угла основания треугольника на поверхность цилиндра. Масса треугольника равна массе мотоцикла с человеком $m = 250$ кг. Ускорение свободного падения $g = 10$ Н/кг.

Ответ дать с точностью до десятой доли кН.

Решение

Центробежная сила уравнивает удвоенную проекцию силы реакции вертикальной стенки. По третьему закону Ньютона сила давления равна силе реакции:

$$m\omega^2 r = 2N \frac{\sqrt{R^2 - \left(\frac{b}{2}\right)^2}}{R}.$$

Отсюда:

$$N = \frac{mV^2}{2R} \left(1 - \frac{a \cos \alpha}{3\sqrt{R^2 - \left(\frac{b}{2}\right)^2}} \right) = 4,3 \text{ кН.}$$

Точность 0,1 кН.

Ответ: $4,3 \pm 0,1$.

Третья попытка. Задачи 8–9 класса**Задача I.2.5.1. Корпус атомного реактора (10 баллов)**

Темы: равномерное движение.

Первый корпус современного атомного реактора ВВЭР-ТОИ совершил путешествие от прокатного стана до Курской АЭС, преодолев 1500 км по воде на барже и 300 км по суше на автоплатформе. Общее время путешествия составило 150 суток. 85% этого времени корпус не двигался во время подготовок к передвижениям. Средняя скорость передвижения по воде в два раза превышает среднюю скорость на суше. Найти среднюю скорость на суше с точностью до 0,1 км/ч.

Решение

Используя формулы равномерного движения, получим среднюю скорость на суше:

$$V = \frac{S_1 + S_2/2}{0,15t} = 1,9 \text{ км/ч.}$$

Точность 0,1 км/ч.

Ответ: $1,9 \pm 0,1$.

Задача I.2.5.2. Корпус атомного реактора (15 баллов)

Темы: плавание тела.

Продолжение задачи I.2.5.1.

Масса корпуса атомного реактора с внутренним оборудованием составляет $m = 606$ т. Баржа имеет прямоугольное днище: 50 м в длину и 18 м в ширину. Борты баржи вертикальны. На какую глубину увеличится осадка баржи при погрузке на нее корпуса реактора. Плотность речной воды $\rho = 1000 \text{ кг/м}^3$.

Ответ дать с точностью до сантиметра.

Решение

Дополнительная сила тяжести равна дополнительной силе Архимеда. Отсюда увеличение глубины равно:

$$\Delta h = \frac{m}{\rho ab} = 67 \text{ см.}$$

Точность 1 см.

Ответ: 67 ± 1 .

Задача I.2.5.3. Корпус атомного реактора (20 баллов)

Темы: сила давления.

Продолжение задач I.2.5.1 и I.2.5.2.

Для перевозки корпуса атомного реактора по суше использовалась автоплатформа на $N = 192$ колесах с массой $m_1 = 80$ т. Давление в шинах колес равно $p = 8 \cdot 10^5$ Па. Ускорение свободного падения $g = 10$ Н/кг. Найти площадь пятна контакта каждой шины с дорогой с точностью до 1 см². На автоплатформе находится корпус атомного реактора.

Решение

Из формулы для силы давления получаем площадь пятна контакта каждой шины:

$$S = \frac{(m + m_1)g}{Np} = 447 \text{ см}^2.$$

Точность 5 см².

Ответ: 447 ± 5 .

Задача I.2.5.4. Корпус атомного реактора (25 баллов)

Темы: центр тяжести.

Продолжение задач I.2.5.2 и I.2.5.3.

На каком расстоянии x от поверхности дороги находится центр тяжести автоплатформы с лежащим на ней корпусом атомного реактора? Считать корпус сплошным цилиндром диаметром $d = 6$ м и длиной 12 м, а автоплатформу считать однородной прямоугольной пластиной длиной 35 м, шириной $c = 6$ м и высотой $h = 2$ м. Цилиндр закреплен горизонтально вдоль платформы в ее середине. Массой крепежа пренебречь. Образующая цилиндра касается верхней плоскости пластины по ее оси симметрии.

Ответ дать с точностью до 0,1 м.

Решение

Используя правило рычага, получим расстояние x от поверхности дороги до центра тяжести автоплатформы с корпусом атомного реактора:

$$x = \frac{m_1 h/2 + m(h + d_2/2)}{m_1 + m} = 4,5 \text{ м.}$$

Точность 0,1 м.

Ответ: $4,5 \pm 0,1$.

Задача I.2.5.5. Корпус атомного реактора (30 баллов)

Темы: момент силы.

Продолжение задач I.2.5.2, I.2.5.3 и I.2.5.4.

На какую максимальную высоту над дорогой можно поднять левый нижний край пластины с закрепленным цилиндром, чтобы еще не произошло их боковое опрокидывание? В этом случае пластина стоит на дороге на своем правом нижнем крае. Расстояние от левого края до правого равно $c = 6$ м.

Ответ дать с точностью до 0,1 м.

Решение

Боковое опрокидывание начнется в тот момент, когда вертикальная прямая, опущенная из центра тяжести, пройдет через правый край пластины. Тогда максимальную высоту над дорогой левого края пластины найдем из подобия прямоугольных треугольников:

$$H = \frac{c^2}{2\sqrt{x^2 + (c/2)^2}} = 3,3 \text{ м.}$$

Точность 0,1 м.

Ответ: $3,3 \pm 0,1$.

Третья попытка. Задачи 10–11 класса**Задача I.2.6.1. Колесо автомобиля (10 баллов)**

Темы: газовые законы.

Колесо автомобиля состоит из колесного диска и бескамерной шины. Ширина шины $b = 205$ мм, внутренний диаметр $d_1 = 406$ мм, а внешний диаметр $d_2 = 632$ мм. Давление воздуха внутри шины равно $p = 0,25$ МПа, его молярная масса $\mu = 29$ г/моль. Найти массу воздуха внутри шины, считая, что ее поперечное сечение имеет форму прямоугольника ширины b . Температура воздуха $T = 20^\circ\text{C}$. Шину считать нерастяжимой. Универсальная газовая постоянная $R = 8,31$ Дж/(моль · К).

Ответ дать с точностью до 1 г.

Решение

Объем шины прямоугольного сечения равен:

$$\nu = \pi b \left((d_2/2)^2 - (d_1/2)^2 \right) = 0,038 \text{ м}^3.$$

Массу воздуха находим из уравнения состояния идеального газа:

$$m = \frac{\mu r \pi b \left((d_2/2)^2 - (d_1/2)^2 \right)}{RT} = 112 \text{ г.}$$

Точность 2 г.

Ответ: 112 ± 2 .

Задача I.2.6.2. Колесо автомобиля (15 баллов)

Темы: сила давления.

Продолжение задачи [I.2.6.1](#).

Автомобиль стоит на горизонтальной поверхности. Вертикальная нагрузка на одно колесо равна $F = 5000 \text{ Н}$. Найти длину a пятна контакта шины с плоскостью. Считать, что пятно имеет форму прямоугольника ширины b . Боковой деформацией шины и изменением ее объема пренебречь. Шина имеет форму цилиндрического слоя с диаметрами d_1 , d_2 и срезанным пятном контакта.

Ответ дать с точностью до 1 мм.

Решение

Приравниваем нагрузку силе давления, тогда длина пятна равна $a = \frac{F}{pb} = 98 \text{ мм}$.

Точность 1 мм.

Ответ: 98 ± 1 .

Задача I.2.6.3. Колесо автомобиля (20 баллов)

Темы: кинематика вращения.

Продолжение задачи [I.2.6.2](#).

Автомобиль поехал с постоянной скоростью $V_0 = 120 \text{ км/ч}$. Определить скорость верхней точки колеса, если оно катится без проскальзывания. Использовать данные о пятне контакта из задачи [I.2.6.2](#).

Ответ дать с точностью до 0,1 м/с.

Решение

Скорости точек, лежащих на вертикальной оси, проходящей через центр колеса, пропорциональны расстоянию до поверхности. Отсюда скорость верхней точки

равна:

$$V = V_0 \left(1 + \frac{1}{\sqrt{1 - \left(\frac{a}{d_2}\right)^2}} \right) = 67,1 \text{ м/с.}$$

Точность 0,1 м/с.

Ответ: $67,1 \pm 0,1$.

Задача I.2.6.4. Колесо автомобиля (25 баллов)

Темы: электростатика.

Продолжение задачи I.2.6.2.

На всю внешнюю поверхность шины диаметром d_2 (кроме пятна контакта) нанесли электрический заряд с постоянной поверхностной плотностью $\sigma = 5 \cdot 10^{-8}$ Кл/м². Найти напряженность электрического поля в центре колеса. Использовать данные о пятне контакта из задачи I.2.6.2. В расчетах пренебрегать длиной a и шириной b пятна контакта по сравнению с диаметром d_2 . Коэффициент пропорциональности $k = 9 \cdot 10^9$ Н · м²/Кл².

Ответ дать с точностью до 1 В/м.

Решение

Напряженность электрического поля в центре кольца такая же, как от части внешней поверхности шины, опирающейся на пятно контакта. Поскольку эта часть поверхности имеет малые размеры, используем формулу для напряженности поля точечного заряда:

$$E = \frac{k\sigma ab}{(d_2/2)^2} = 90 \text{ В/м.}$$

Точность 2 В/м.

Ответ: 90 ± 2 .

Задача I.2.6.5. Колесо автомобиля (30 баллов)

Темы: газовые законы.

Продолжение задачи I.2.6.1.

В шине произошел прокол и образовалось маленькое отверстие площадью $S = 0,012$ мм². Воздух выходит из шины очень медленно с постоянной скоростью $u = 10$ м/с. Температура воздуха в шине остается постоянной $T = 20^\circ\text{C}$ и объем шины с размерами из задачи I.2.6.1 не меняется. Найти время уменьшения давления p воздуха внутри шины в два раза.

Ответ дать с точностью до 1 часа.

Решение

За малое время Δt из отверстия выходит воздух массой $\frac{m}{\nu}Su\Delta t$. Тогда уменьшение массы воздуха m в шине объема ν описывается уравнением:

$$\frac{\Delta m}{\Delta t} = -\frac{m}{\nu}Su.$$

Решение этого уравнения находим по аналогии с законом радиоактивного распада. Тогда время уменьшения массы воздуха m и давления p воздуха внутри шины в два раза равно:

$$T_{1/2} = \frac{\nu \ln 2}{Su} = 61 \text{ час.}$$

Точность 1 час.

Ответ: 61 ± 1 .

Четвертая попытка. Задачи 8–9 класса**Задача I.2.7.1. Моноколесо (10 баллов)**

Темы: источник тока.

Емкость батареи моноколеса равна $Q = 1110 \text{ Вт} \cdot \text{ч}$. Какой постоянный ток будет протекать через батарею, если полное время зарядки батареи равно $t = 6 \text{ ч}$? Напряжение на клеммах батареи равно $U = 84 \text{ В}$. Внутренним сопротивлением батареи пренебречь.

Ответ дать с точностью до десятых долей А.

Решение

Ток равен:

$$I = \frac{Q}{Ut} = 2,2 \text{ А.}$$

Точность 0,1 А.

Ответ: $2,2 \pm 0,1$.

Задача I.2.7.2. Моноколесо (15 баллов)

Темы: мощность двигателя.

Продолжение задачи [I.2.7.1](#).

Максимальный ток разрядки батареи моноколеса в десять раз превышает постоянный ток зарядки батареи. Какова максимальная мощность электромотора самоката, если постоянное напряжение батареи равно $U = 84 \text{ В}$?

Ответ дать с точностью до 1 Вт.

Решение

Максимальная мощность равна:

$$P_m = 10IU = 1850 \text{ Вт.}$$

Точность 1 Вт.

Ответ: 1850 ± 1 .

Задача I.2.7.3. Моноколесо (20 баллов)

Темы: мощность силы.

Продолжение задач I.2.7.1 и I.2.7.2.

С какой максимальной скоростью может ехать человек на моноколесе по горизонтальной дороге, если сила сопротивления воздуха, действующая на человека и моноколесо, при этой скорости равна $F = 132 \text{ Н}$. Силой трения качения пренебречь.

Ответ дать с точностью до 1 км/ч.

Решение

Максимальная скорость равна:

$$V_m = P_m / F = 50 \text{ км/ч.}$$

Точность 2 км/ч.

Ответ: 50 ± 2 .

Задача I.2.7.4. Моноколесо (25 баллов)

Темы: равномерное движение.

Продолжение задач I.2.7.1, I.2.7.2 и I.2.7.3.

Человек на моноколесе по горизонтальной дороге проезжает расстояние $S = 70 \text{ км}$, израсходовав весь заряд батареи. Какое время затратит человек на это путешествие, если движение происходит с постоянной скоростью? Силу сопротивления воздуха считать прямо пропорциональной скорости.

Ответ дать с точностью до 1 мин.

Решение

Средняя мощность двигателя равна механической мощности на скорости V :

$$\frac{Q}{t} = F \frac{V}{V_m} V.$$

Весь путь равен $S = Vt$. Тогда время движения моноколеса равно:

$$t = \frac{FS^2}{V_m Q} = 193 \text{ мин.}$$

Точность 3 мин.

Ответ: 193 ± 3 .

Задача I.2.7.5. Моноколесо (30 баллов)

Темы: наклонная плоскость.

Продолжение задач I.2.7.1, I.2.7.2, I.2.7.3 и I.2.7.4.

После полной перезарядки батареи человек массой 100 кг поднимается с постоянной скоростью на моноколесе массой 25 кг в гору высотой $h = 50$ м на $l = 1$ км пути. Средняя мощность двигателя моноколеса равно $P = 500$ Вт. Найти постоянную скорость моноколеса.

Ответ дать с точностью до 0,1 м/с.

Ускорение свободного падения $g = 10$ Н/кг.

Решение

Средняя мощность двигателя равна механической мощности на скорости V :

$$P = F \frac{V}{V_m} V + (m + M)g \frac{h}{l/V}.$$

Отсюда получаем квадратное уравнение для V :

$$\frac{F}{V_m} V^2 + (m + M)g \frac{h}{l} V - P = 0.$$

Его решение: 4,7 м/с.

Точность 0,1 м/с.

Ответ: $4,7 \pm 0,1$.

Четвертая попытка. Задачи 10–11 класса

Задача I.2.8.1. Гонки на квадроцикле (10 баллов)

Темы: второй закон Ньютона.

В аттракционе «Гонки по вертикальной стене» гонщик на квадроцикле движется по внутренней стороне вертикального бетонного цилиндра радиуса $R = 7$ м и высотой $H = 9$ м. Коэффициент трения скольжения резиновой шины по бетону равен $\mu = 0,75$. Ускорение свободного падения $g = 10$ Н/кг. В задачах I.2.8.1, I.2.8.2 и I.2.8.3 считать квадроцикл с гонщиком материальной точкой. Найти постоянную скорость движения гонщика V в горизонтальной плоскости на высоте 4 м, если она в два раза превышает минимально возможную скорость.

Ответ дать с точностью до 1 м/с.

Решение

Из второго закона Ньютона минимально возможная скорость равна $V_0 = \sqrt{gR/\mu}$. Тогда скорость мотоциклиста:

$$V = 2\sqrt{gR/\mu} = 19 \text{ м/с.}$$

Точность 1 м/с.

Ответ: 19 ± 1 .

Задача I.2.8.2. Гонки на квадроцикле (15 баллов)

Темы: КПД.

Продолжение задачи I.2.8.1.

Найти КПД двигателя квадроцикла, если он движется со скоростью из задачи I.2.8.1. Мощность двигателя квадроцикла составляет 14 метрических лошадиных сил. Расход бензина равен 4,5 л на 100 км пути. Удельная теплота сгорания бензина равна 44 МДж/кг, а плотность бензина составляет 730 кг/м³.

Ответ дать с точностью до 1 %.

Решение

КПД равен отношению мощности двигателя к мощности сгорания бензина:

$$\eta = \frac{P}{Q_{\text{уд}}\rho V q_{\text{расход}}} = 37\%.$$

Точность 1 %.

Ответ: 37 ± 1 .

Задача I.2.8.3. Гонки на квадроцикле (20 баллов)

Темы: путь.

Продолжение задачи I.2.8.1.

Найти путь квадроцикла от основания до вершины цилиндра по винтовой линии с шагом между витками (по вертикали) $h = 0,75$ м. Параметры цилиндра даны в задаче I.2.8.1.

Ответ дать с точностью до 1 м.

Решение

Путь равен длине одного витка умноженной на число витков.

$$S = \frac{H}{h} \sqrt{(2\pi R)^2 + h^2} = 528 \text{ м.}$$

Точность 1 м.

Ответ: 528 ± 1 .

Задача I.2.8.4. Гонки на квадроцикле (25 баллов)

Темы: центр масс.

Продолжение задачи I.2.8.1.

Заменяем гонщика с квадроциклом на однородную прямоугольную пластину длиной $a = 1,8$ м, шириной $b = 1,0$ м и высотой $c = 1,6$ м. Пластина скользит всеми четырьмя углами основания ab по внутренней поверхности цилиндра на высоте 4 м, причем ребра a горизонтальны. Найти минимально возможную скорость углов V_m основания пластины для такого скольжения.

Ответ дать с точностью до 0,3 м/с.

Ускорение свободного падения $g = 10$ Н/кг.

Решение

Центр масс однородной прямоугольной пластины находится в ее геометрическом центре и движется по окружности радиуса:

$$r = \sqrt{R^2 - (a/2)^2} - (c/2)$$

с угловой скоростью $\omega = V_m/R$. На минимально возможной скорости V_m силы реакции в верхних углах основания обращаются в ноль. Тогда запишем равенство моментов центробежной силы и силы тяжести относительно горизонтальной оси, проходящей через нижние углы основания:

$$m\omega^2 r \frac{b}{2} = mg \frac{c}{2}.$$

Отсюда получаем минимально возможную скорость углов:

$$V_m = R \sqrt{\frac{gc}{rb}} = 11,3 \text{ м/с}.$$

Точность 0,3 м/с.

Ответ: $11,3 \pm 0,3$.

Задача I.2.8.5. Гонки на квадроцикле (30 баллов)

Темы: сила давления.

Продолжение задач I.2.8.1 и I.2.8.4.

Найти силу давления одного угла основания ab на поверхность цилиндра. Масса пластины равна массе квадроцикла с гонщиком $m = 220$ кг. Углы пластины двигаются с минимально возможной скоростью V_m из задачи I.2.8.4. Ускорение свободного падения $g = 10$ Н/кг.

Ответ дать с точностью до десятой доли кН.

Решение

Центробежная сила уравнивает удвоенную проекцию силы реакции вертикальной стенки в нижнем угле основания. По третьему закону Ньютона сила давления равна силе реакции:

$$m\omega^2 r = 2N \frac{\sqrt{R^2 - (a/2)^2}}{R}.$$

Отсюда:

$$N = \frac{mV_m^2}{2R} \left(1 - \frac{c}{2\sqrt{R^2 - (a/2)^2}} \right) = 1,8 \text{ кН.}$$

Точность 0,1 кН.

Ответ: $1,8 \pm 0,1$.

Второй отборочный этап

Индивидуальная часть

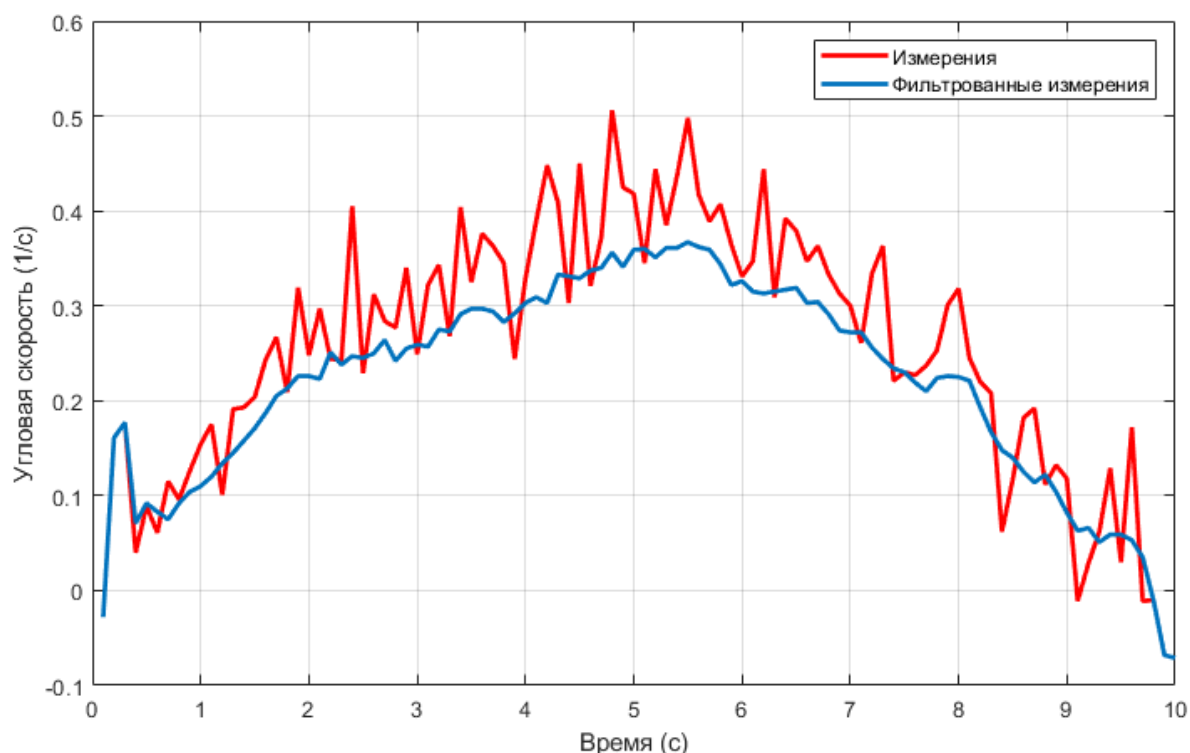
Задача П.1.1. Фильтрация данных (5 баллов)

Темы: фильтрация данных, программирование.

Алгоритмы фильтрации данных являются неотъемлемой частью бортовых навигационных систем БЛА.

Условие

При любых измерениях физических величин используются измерительные приборы, каждый из которых имеет некоторую погрешность. Для минимизации этой погрешности применяются различные методы фильтрации. Одним из самых простых является метод простого скользящего среднего (SMA, Simple Moving Average).



Входом такого фильтра является необработанный сигнал (в виде массива значений) и единственный параметр — размер окна (значение этого параметра всегда целое нечетное число). Выходом является массив значений фильтрованного сигнала, элементы которого вычисляются по формуле:

$$y_i = \frac{1}{N} \sum_{i-(N-1)/2}^{i+(N-1)/2} x_i,$$

где N — размер окна, x_i — элемент необработанного сигнала, y_i — элемент фильтрованного сигнала.

Таким образом вычисляется среднее арифметическое N элементов вокруг i -ого элемента исходного массива. На рисунке приведен пример расчета элементов массива фильтрованных данных с размером окна $N = 5$.

$$x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}$$

$$y_5 = \frac{x_3 + x_4 + x_5 + x_6 + x_7}{N}$$

$$y_3 = \frac{x_1 + x_2 + x_3 + x_4 + x_5}{N}$$

При таком подходе возникает проблема с несколькими элементами в начале и в конце исходного массива. Например, если $N = 5$, то для 0-го и 1-го элементов мы просто не сможем выбрать 5 элементов вокруг них для вычисления среднего арифметического. Существует несколько способов учета таких элементов. Первый, и самый простой, отбросить эти элементы (в этом случае длина массива фильтрованных данных будет на $N - 1$ меньше, чем длина исходного массива). Второй способ — оставить эти элементы без изменений и просто перенести их в массив с фильтрованными данными. Третий способ — фильтровать эти элементы с уменьшенным размером окна.

Напишите программу на языке C++ или Python, осуществляющую фильтрацию входного массива данных методом простого скользящего среднего. Крайние значения входного массива должны быть учтены без изменений.

Формат входных данных

На вход программы первой строчкой поступает размер окна — целое нечетное число, не превышающее размер массива входного сигнала. На второй строчке идет строка чисел входного сигнала, округленных до 3-го знака после запятой и разделенных пробелами. Количество значений входного сигнала может быть различным, но не больше 25.

Формат выходных данных

Необходимо вывести на стандартный вывод отфильтрованный массив данных. Все значения должны быть выведены в одну строку через пробел и округлены до трех знаков после запятой по правилам математического округления. Количество значений в выходной строке всегда должно совпадать с количеством значений входного сигнала.

Примеры

Пример №1

Стандартный ввод
5 0.039 0.086 0.277 0.278 0.17 0.719 0.606 0.572 0.678 0.876
Стандартный вывод
0.039 0.086 0.170 0.306 0.410 0.469 0.549 0.690 0.678 0.876

Список литературы

1. Функция mean модуля statistics в Python, <https://docs-python.ru/standard-library/modul-statistics-python/funktsija-mean-fmean-modulja-statistics/>
2. Фильтр скользящего среднего <http://enc.fxeuroclub.com/105/>

Решение

Для решения задачи необходимо:

1. Распределить числа из строки входных данных в целочисленный массив для удобства дальнейшей обработки.
2. Перенести «начало» массива по вышеописанным правилам.
3. Отфильтровать «середицу» массива.
4. Перенести «конец» массива аналогично «началу».

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 import statistics as st
2 win = float(input())
3 signal = [float(i) for i in input().split()]
4 half = int((win - 1) / 2)
5 Y = []
6 for i in range(half):
7     Y.append(signal[i])
8 for i in range(half, len(signal) - half):
9     Y.append(st.mean(signal[i - half : i + half + 1]))
10 for i in range(len(signal) - half, len(signal)):
11     Y.append(signal[i])
12 for i in Y:
13     print("{:2.3f}".format(i), end = ' ')

```

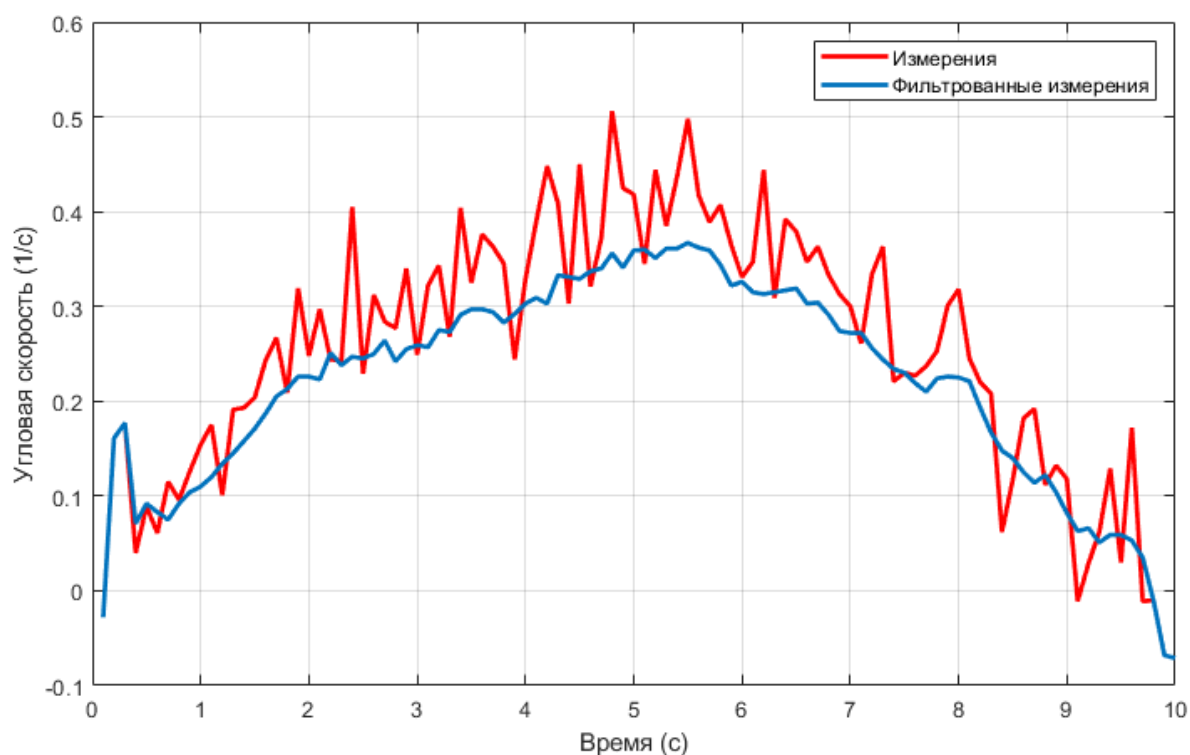
Задача П.1.2. Определение угла курса по измерениям гироскопа (10 баллов)

Темы: *фильтрация данных, программирование.*

Алгоритм расчета текущей ориентации по измерениям гироскопа необходим при построении систем автономной навигации.

Условие

В процессе полета БЛА необходимо непрерывно определять свое угловое положение относительно плоскости горизонта. От углов ориентации зависит не только направление полета, но и величина подъемной силы, угловая скорость разворота и множество других параметров движения БЛА. Одним из самых распространенных датчиков ориентации является гироскоп. Этот датчик измеряет угловую скорость, по которой можно рассчитать изменение углов ориентации БЛА. Однако, как и у большинства других измерительных систем, выходной сигнал гироскопа «зашумлен» — в нем присутствует случайная ошибка. В связи с этим перед расчетом углов ориентации, как правило, осуществляют фильтрацию измерений гироскопа.



Напишите программу на языке C++ или Python, определяющую угол разворота БЛА по полученным измерениям курсового гироскопа. Начальный угол курса $\psi_0 = 0$ град. Время между двумя последовательными измерениями $dt = 0,1$ с. Количество измерений во входном массиве может быть произвольным. Перед расчетом угла курса требуется осуществить фильтрацию измерений фильтром простого скользящего среднего (SMA) с размером окна $N = 7$ (крайние значения массива остаются без изменений).

Формат входных данных

Значения измерений угловой скорости разворота БЛА по углу курса (1/с), разделенные пробелами. Количество значений может быть произвольным.

X.XXX Y.YYY ... Z.ZZZ

Формат выходных данных

Значение угла разворота БЛА по курсу в градусах с точностью до второго знака после запятой.

RR.RR

Примеры

Пример №1

Стандартный ввод
-0.028 0.161 0.177 0.040 0.089 0.061 0.115 0.096 0.126 0.154 0.175 0.101
0.191 0.193 0.204 0.243 0.267 0.209 0.319 0.248 0.297 0.244 0.243 0.405
0.229 0.312 0.284 0.277 0.340 0.249 0.322 0.343 0.268 0.404 0.325 0.376
0.363 0.345 0.244 0.329 0.390 0.448 0.409 0.303 0.450 0.321 0.372 0.506
0.425 0.418 0.345 0.444 0.385 0.437 0.498 0.417 0.389 0.407 0.365 0.331
0.347 0.444 0.309 0.392 0.379 0.347 0.363 0.333 0.313 0.300 0.261 0.334
0.363 0.221 0.230 0.227 0.237 0.253 0.301 0.318 0.245 0.220 0.208 0.062
0.117 0.182 0.192 0.112 0.132 0.118 -0.011 0.029 0.062 0.129 0.030 0.172
-0.011 -0.010 -0.068 -0.071
Стандартный вывод
146.08

Список литературы

1. Фильтр скользящего среднего <http://enc.fxeuroclub.com/105/>
2. Датчик угловой скорости (гироскоп) https://ru.wikipedia.org/wiki/Датчик_угловой_скорости

Решение

Для решения задачи необходимо:

1. Написать функцию (или подключить библиотеку), реализующую фильтр скользящего среднего.
2. Отфильтровать входной массив данных, сохранив первые 3 и последние 3 значения без изменений.
3. Проинтегрировать отфильтрованный массив данных.
4. Перевести полученное значение угла из радианов в градусы.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 import statistics as st
2 def SMA2(signal, win):
3     half = int((win - 1) / 2)
4     Y = []
5     for i in range(half):
6         Y.append(signal[i])
7     for i in range(half, len(signal) - half):
8         Y.append(st.mean(signal[i - half : i + half + 1]))
9
10    for i in range(len(signal) - half, len(signal)):
11        Y.append(signal[i])
12    return Y
13
14 arr = [float(i) for i in input().split()]
15 filt = SMA2(arr, 7)
16 angle_filt = 0.0
17 dt = 0.1
18 pi = 3.1415926
19 for sig in filt:
20     angle_filt += sig * dt
21 print("{:2.2f}".format(angle_filt * 180.0 / pi))

```

Задача II.1.3. Разработка алгоритма нормализации изображения (5 баллов)

Темы: алгоритмы, программирование, обработка изображений, нормализация яркости.

Нормализация яркости изображений является одним из простейших алгоритмов обработки изображений.

Условие

В алгоритмах компьютерного зрения и в других алгоритмах обработки больших наборов однородных данных исходные данные, как правило, подвергаются предварительной нормализации. Нормализация — это преобразование данных к неким безразмерным единицам в рамках заданного диапазона, например, $[0..1]$ или $[-1..1]$.

Нормализацию значения из исходного диапазона $m \in [0..M]$ в диапазон $n \in [0..N]$ можно произвести по следующей формуле:

$$n = m \frac{1}{M} N,$$

где M — максимальное значение данных исходного диапазона; N — максимальное значение нормализованного диапазона данных.

Одним из самых простых форматов представления изображений является формат PGM (Portable Gray Map). В этом формате данных черно-белое изображение представляется набором чисел, каждое из которых лежит в диапазоне $[0..255]$ и характеризует яркость одного пикселя изображения. Значение 0 соответствует черному цвету, 255 — белому цвету. На практике измерительные устройства (например,

камеры) редко позволяют получить изображение с полным диапазоном значений, поэтому требуется проводить нормализацию полученного изображения. Напишите программу на языке C++ или Python, выполняющую нормализацию входных данных в диапазоне [0..255].

Формат входных данных

На вход программы поступает строка целых чисел, разделенных пробелами. В конце строки пробел не ставится. Количество чисел в строке может быть произвольным, но не более 30.

XX YY ... ZZ

Формат выходных данных

Необходимо вывести на стандартный вывод нормализованные в диапазоне от 0 до 255 значения исходной строки. Все числа должны быть округлены до целых значений по правилам математического округления. Необходимо вывести в одну строку. Количество значений в выходной строке всегда должно совпадать с количеством значений во входных данных.

XX YY ... ZZ

Примеры

Пример №1

Стандартный ввод
67 74 81 94 105 106 95 11
Стандартный вывод
161 178 195 226 253 255 229 26

Список литературы

1. Формат PGM, <https://www.online-convert.com/ru/file-format/pgm>

Решение

Для решения задачи необходимо:

1. Распределить числа из строки входных данных в целочисленный массив для удобства дальнейшей обработки.
2. Определить самое большое число в массиве входных данных.
3. Разделить все числа в массиве на максимальное число, умножить их на 255 и округлить до целых значений.
4. Вывести нормализованные числа на экран через пробел.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  # Read console input
2  arr = [float(i) for i in input().split()]
3  # Find max value
4  m = max(arr)
5  # For each element in the array
6  for num in arr:
7      # print the normalized value
8      print(round(num / m * 255), end = ' ')

```

Задача II.1.4. Обнаружение объектов системой технического зрения (10 баллов)

Темы: программирование, техническое зрение, анализ изображений.

Задача поиска объектов на фотоизображении позволяет участникам познакомиться с основами технического зрения.

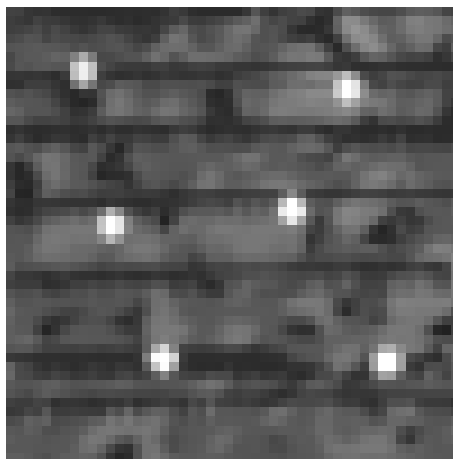
Условие

Одной из самых распространенных задач технического зрения для БЛА является задача поиска ориентиров. Ориентирами могут являться какие-то особенности местности, по которым БЛА может уточнить свое местоположение или объекты наблюдения (например, машины, люди, или даже коровы).



Напишите программу на языке C++ или Python, выполняющую нормализацию яркости исходного изображения и определяющую количество контрастных ориентиров на этом изображении.

Исходным изображением является черно-белое изображение в формате PGM. Размер изображения 50×50 пикселей. На изображении присутствуют несколько ориентиров в виде групп ярких пикселей. Яркость ориентиров неравномерная и может изменяться в небольшом диапазоне. Пример входного изображения показан на рисунке.



На вход программы поступает текст PGM файла, сохраненного в текстовом формате (содержимое PGM файла вводится в виде строки через консоль). Текст файла состоит из двух основных частей: заголовка и тела. Описание структуры PGM файла можно найти по ссылке: <http://netpbm.sourceforge.net/doc/pgm.html>

Все числа представляют собой целые значения, разделенные пробелами. Тело PGM файла содержит 2500 значений, описывающих яркость каждого из пикселей изображения размером 50×50 .

Формат входных данных

P2 RR RR M
X1 X2 X3 ... XN
Заголовок
Тело файла

Формат выходных данных

Необходимо вывести на стандартный вывод одно целое число — количество контрастных ориентиров на изображении.

Примеры

Пример №1

Стандартный ввод
P2 50 50 255 60 59 49 ... 79
Стандартный вывод
6

Решение

Для решения задачи необходимо:

1. Для удобства работы распределить числа из тела файла в целочисленный двумерный массив.

2. Нормализовать числа в массиве.
3. Выбрать порог, после которого пиксель будет считаться пикселем ориентира, например, 220.
4. Создать структуру данных для записи координат пикселей, превышающих порог, например, список.
5. Перебрать все элементы двумерного массива в поисках чисел, превышающих порог.
 - 5.1. Если такой элемент, превышающий порог, найден, то проверить находятся ли рядом с ним какие-либо пиксели, которые уже записаны в список для светлых пикселей, превышающих порог яркости.
 - 5.2. Если в предыдущем пункте нашелся хотя бы один ранее записанный пиксель, значит рассматривается пиксель уже найденного ориентира. В противном случае считаем, что нашли новый ориентир.
 - 5.3. Записываем пиксель в список найденных пикселей.
6. Выводим на экран количество найденных ориентиров.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  trash = input()
2  res = input()
3  trash = input()
4
5  # Определение разрешения
6  res += ' '
7  res_int = []
8  a = 0
9  b = 0
10 for i in range(len(res)):
11     if res[i] == ' ':
12         b = i
13         res_int.append(int(res[a:b]))
14         a = b + 1
15
16 # Распределение тела файла в двумерный массив
17 pgm_arr = []
18 for i in range(res_int[0]):
19     pgm_arr2 = []
20     for j in range(res_int[1]):
21         pgm_arr2.append(int(input()))
22     pgm_arr.append(pgm_arr2)
23
24 # Определение максимального элемента
25 max = pgm_arr[0][0]
26 for i in range(res_int[0]):
27     for j in range(res_int[1]):
28         if pgm_arr[i][j] > max:
29             max = pgm_arr[i][j]
30
31 # Нормализация
32 for i in range(res_int[0]):
33     for j in range(res_int[1]):
34         pgm_arr[i][j] = int(pgm_arr[i][j] * 255 / max)
35

```

```

36 # Поиск пикселей ориентиров
37 N = 0 # Количество найденных ориентиров
38 found_xy = [] # Массив с координатами найденных пикселей ориентиров
39 for i in range(res_int[0]):
40     for j in range(res_int[1]):
41         if pgm_arr[i][j] >= 220:
42
43             already_found = False # Проверка, относится ли этот пиксель к ранее
44             ↪ найденным ориентирам
45             for n in range(-2, 3):
46                 for m in range(-2, 3):
47                     if [i + n, j + m] in found_xy:
48                         already_found = True
49
50             found_xy.append([i, j])
51             if not already_found:
52                 N += 1
53
54 print(N)

```

Задача П.1.5. Методы статистики для оценки качества измерений (5 баллов)

Темы: математика, статистика.

Применение простейших методов статистики позволяет отбраковать дефектные измерения бортовых приборов БЛА.

Условие

Для получения информации об окружающем мире и своем положении в нем на борту БЛА используются различные измерительные системы. Каждая из этих систем имеет некоторую погрешность измерений. Для одних систем, например инерциальных датчиков, величина погрешности постоянна и указана в техническом паспорте изделия. Для других систем, например приемника GPS, величина погрешности все время меняется и зависит от большого количества внешних факторов. Как правило, такие системы обладают встроенными алгоритмами оценки текущей величины погрешности, которые позволяют судить о текущей точности измерений.

Расчет вероятности получения измерения $X_{\text{изм}}$ при истинном значении величины $X_{\text{ист}}$ и погрешности измерительной системы σ можно осуществить по формуле нормального закона распределения случайной величины:

$$f(x) = e^{-\frac{1}{2} \left(\frac{X_{\text{изм}} - X_{\text{ист}}}{\sigma} \right)^2};$$

$$0 \leq f(x) \leq 1.$$

Напишите программу на языке C++ или Python, для расчета вероятности $f(x)$. На вход программа должна получать измерение текущей широты $\varphi_{\text{изм}}$ местоположения БЛА, при известных точности измерения σ и истинном значении широты $\varphi_{\text{ист}} = 55,811008$ град. широты.

Формат входных данных

Значение полученного измерения широты места $\varphi_{\text{изм}}$ в градусах и точность этого измерения σ , заданные последовательно, через пробел.

DD:DDDDDD SS.SSSSS

Формат выходных данных

Значение вероятности $f(x)$ получения измерения $\varphi_{\text{изм}}$ в процентах (округление до целого числа).

FFF

Примеры

Пример №1

Стандартный ввод
55.811000 0.00001
Стандартный вывод
073

Список литературы

1. Географические координаты, <https://v-ipc.ru/guides/coord>
2. Нормальное распределение, <https://bigenc.ru/mathematics/text/2671173>

Решение

Для нахождения вероятности $f(x)$ необходимо подставить в предоставленную в условии формулу следующие значения: $\varphi_{\text{изм}}$ вместо $x_{\text{изм}}$; $\varphi_{\text{ист}} = 55,811008$ вместо $x_{\text{ист}}$; и значение σ .

$$f(x) = e^{-\frac{1}{2} \left(\frac{\varphi_{\text{изм}} - \varphi_{\text{ист}}}{\sigma} \right)^2}.$$

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 import math
2
3 # Read console input
4 arr = [float(i) for i in input().split()]
5 fizm = arr[0]
6 sigma = arr[1]
7 fist = 55.811008
8 fx = math.exp(-0.5 * ((fizm - fist) / sigma)**2)
9 print("{:03.0f}".format(fx*100))

```

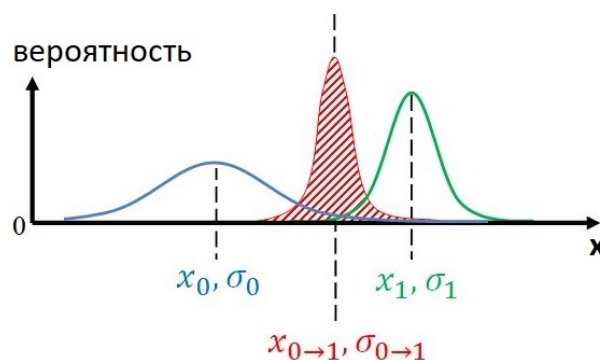
Задача П.1.6. Определение местоположения БЛА по измерениям GPS (10 баллов)

Темы: математика, статистика, качество измерений, программирование.

Применение простейших методов статистики позволяет отбраковать дефектные измерения бортовых приборов БЛА.

Условие

В процессе полета БЛА осуществляет непрерывный контроль своего местоположения по измерениям GPS и других бортовых систем. Однако любые измерительные системы обладают погрешностями и могут выдавать неверные данные. Поэтому при обработке измерений навигационная система БЛА должна учитывать не только последние сделанные измерения, но и измерения, сделанные на протяжении всего полета БЛА. Если измерения представлены в виде вероятностной функции, например, нормального закона распределения, то учет нового измерения осуществляется путем обновления текущего состояния БЛА на основании последних измеренных значений и их прогнозируемой точности. Так, например, если текущие координаты БЛА 55 град широты и 37 град долготы, а через несколько секунд он получает измерение 20 град широты и 1 град долготы — очевидно, произошел сбой системы и новым координатам верить нельзя. Но очевидно это только для человека, а для машины требуется описать математический аппарат, который позволит ей правильно учитывать измерения как в таких, экстремальных, случаях, так и в штатном режиме работы.



В случае, когда измерения представлены в виде нормального закона распределения (значения измерения x_1 и его точности σ_1), процедуру обновления текущего состояния БЛА (x_0, σ_0) можно описать соотношениями:

$$x_{0 \rightarrow 1} = \frac{x_0 \sigma_1 + x_1 \sigma_0}{\sigma_0 + \sigma_1};$$

$$\sigma_{0 \rightarrow 1} = \frac{1}{\frac{1}{\sigma_0} + \frac{1}{\sigma_1}}.$$

Напишите программу на языке C++ или Python, осуществляющую расчет местоположения БЛА (его географической широты φ и долготы λ в градусах) и точности полученного решения (параметр σ) по известным начальным значениям и четырем последовательным измерениям широты и долготы с заданной точностью каждого

из измерений. Начальное местоположение БЛА: $\varphi_0 = 56,096215$, $\lambda_0 = 35,876446$, $\sigma_0 = 0,1$.

Расчет конечного местоположения БЛА должен учитывать все четыре сделанных измерения GPS ($\varphi_{0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4}$, $\lambda_{0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4}$, $\sigma_{0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4}$).

Формат входных данных

Строка входных данных включает в себя три параметра для каждого из четырех измерений: широта (F), долгота (L), точность (S). Всего 12 значений, разделенных пробелами. Измерения записаны последовательно, от первого к четвертому. Значения широты и долготы задаются в градусах с десятичными долями.

```
FF.FFFFFFFF LL.LLLLLL S.SSSSSS FF.FFFFFFFF LL.LLLLLL S.SSSSSS
FF.FFFFFFFF LL.LLLLLL S.SSSSSS FF.FFFFFFFF LL.LLLLLL S.SSSSSS
```

Формат выходных данных

Значения широты (F), долготы (L) и точности (S) определения местоположения БЛА в градусах с точностью до 6-го знака после запятой. Все значения должны быть разделены пробелами.

```
FF.FFFFFFFF LL.LLLLLL S.SSSSSS
```

Примеры

Пример №1

Стандартный ввод
56.097741 35.876253 0.030000 56.097262 35.877680 0.072000
56.097884 35.878860 0.120000 56.096238 35.879214 0.004000
Стандартный вывод
56.096485 35.878737 0.003169

Список литературы

1. Географические координаты, <https://v-ipc.ru/guides/coord>
2. Нормальное распределение, <https://bigenc.ru/mathematics/text/2671173>

Решение

Используя соотношения из условия, рассчитаем местоположение БЛА после получения первого измерения:

$$\varphi_{0 \rightarrow 1} = \frac{\varphi_0 \sigma_1 + \varphi_1 \sigma_0}{\sigma_0 + \sigma_1};$$

$$\lambda_{0 \rightarrow 1} = \frac{\lambda_0 \sigma_1 + \lambda_1 \sigma_0}{\sigma_0 + \sigma_1};$$

$$\sigma_{0 \rightarrow 1} = \frac{1}{\frac{1}{\sigma_0} + \frac{1}{\sigma_1}}.$$

Повторяем эту процедуру для второго измерения, используя полученные значения $\varphi_{0 \rightarrow 1}$, $\lambda_{0 \rightarrow 1}$, $\sigma_{0 \rightarrow 1}$ вместо φ_0 , λ_0 , σ_0 :

$$\varphi_{0 \rightarrow 1 \rightarrow 2} = \frac{\varphi_{0 \rightarrow 1} \sigma_2 + \varphi_2 \sigma_{0 \rightarrow 1}}{\sigma_{0 \rightarrow 1} + \sigma_2};$$

$$\lambda_{0 \rightarrow 1 \rightarrow 2} = \frac{\lambda_{0 \rightarrow 1} \sigma_2 + \lambda_2 \sigma_{0 \rightarrow 1}}{\sigma_{0 \rightarrow 1} + \sigma_2};$$

$$\sigma_{0 \rightarrow 1 \rightarrow 2} = \frac{1}{\frac{1}{\sigma_{0 \rightarrow 1}} + \frac{1}{\sigma_2}}.$$

Аналогичным образом определяем местоположение БЛА после получения третьего измерения:

$$\varphi_{0 \rightarrow 1 \rightarrow 2 \rightarrow 3} = \frac{\varphi_{0 \rightarrow 1 \rightarrow 2} \sigma_3 + \varphi_3 \sigma_{0 \rightarrow 1 \rightarrow 2}}{\sigma_{0 \rightarrow 1 \rightarrow 2} + \sigma_3};$$

$$\lambda_{0 \rightarrow 1 \rightarrow 2 \rightarrow 3} = \frac{\lambda_{0 \rightarrow 1 \rightarrow 2} \sigma_3 + \lambda_3 \sigma_{0 \rightarrow 1 \rightarrow 2}}{\sigma_{0 \rightarrow 1 \rightarrow 2} + \sigma_3};$$

$$\sigma_{0 \rightarrow 1 \rightarrow 2 \rightarrow 3} = \frac{1}{\frac{1}{\sigma_{0 \rightarrow 1 \rightarrow 2}} + \frac{1}{\sigma_3}}.$$

Итоговые координаты местоположения БЛА и точность их определения соответственно будут иметь вид:

$$\varphi_{0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4} = \frac{\varphi_{0 \rightarrow 1 \rightarrow 2 \rightarrow 3} \sigma_4 + \varphi_4 \sigma_{0 \rightarrow 1 \rightarrow 2 \rightarrow 3}}{\sigma_{0 \rightarrow 1 \rightarrow 2 \rightarrow 3} + \sigma_4};$$

$$\lambda_{0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4} = \frac{\lambda_{0 \rightarrow 1 \rightarrow 2 \rightarrow 3} \sigma_4 + \lambda_4 \sigma_{0 \rightarrow 1 \rightarrow 2 \rightarrow 3}}{\sigma_{0 \rightarrow 1 \rightarrow 2 \rightarrow 3} + \sigma_4};$$

$$\sigma_{0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4} = \frac{1}{\frac{1}{\sigma_{0 \rightarrow 1 \rightarrow 2 \rightarrow 3}} + \frac{1}{\sigma_4}}.$$

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1  # Read console input
2  arr = [float(i) for i in input().split()]
3  lat1 = arr[0]
4  lon1 = arr[1]
5  sig1 = arr[2]
6
7  lat2 = arr[3]
8  lon2 = arr[4]
9  sig2 = arr[5]
10
11 lat3 = arr[6]
12 lon3 = arr[7]
```

```

13 sig3 = arr[8]
14
15 lat4 = arr[9]
16 lon4 = arr[10]
17 sig4 = arr[11]
18
19 lat0 = 56.096215
20 lon0 = 35.876446
21 sig0 = 0.1
22
23 lat01 = (lat0*sig1+ lat1*sig0) / (sig0 + sig1)
24 lon01 = (lon0*sig1 + lon1*sig0) / (sig0 + sig1)
25 sig01 = 1 / ( 1 / (sig0) + 1 / (sig1) )
26 #print("0-1: {:.02.6f} {:.02.6f} {:.01.6f}".format(lat01, lon01, sig01))
27
28 lat012 = (lat01*sig2 + lat2*sig01) / (sig01 + sig2)
29 lon012 = (lon01*sig2 + lon2*sig01) / (sig01 + sig2)
30 sig012 = 1 / ( 1 / (sig01) + 1 / (sig2) )
31 #print("0-2: {:.02.6f} {:.02.6f} {:.01.6f}".format(lat012, lon012, sig012))
32
33 lat0123 = (lat012*sig3 + lat3*sig012) / (sig012 + sig3)
34 lon0123 = (lon012*sig3 + lon3*sig012) / (sig012 + sig3)
35 sig0123 = 1 / ( 1 / (sig012) + 1 / (sig3) )
36 #print("0-3: {:.02.6f} {:.02.6f} {:.01.6f}".format(lat0123, lon0123, sig0123))
37
38 lat01234 = (lat0123*sig4 + lat4*sig0123) / (sig0123 + sig4)
39 lon01234 = (lon0123*sig4 + lon4*sig0123) / (sig0123 + sig4)
40 sig01234 = 1 / ( 1 / (sig0123) + 1 / (sig4) )
41 print("{:.02.6f} {:.02.6f} {:.01.6f}".format(lat01234, lon01234, sig01234))

```

Задача II.1.7. Определение местоположения БЛА по измерениям GPS (10 баллов)

Темы: физика полета, уравнение, программирование.

Понимание основ физики полета БЛА самолетного типа необходимо для разработки системы автоматического управления для такого объекта.

Условие

В процессе полета БЛА самолетного типа не может совершать разворот на месте как, например, вертолет или квадрокоптер. БЛА самолетного типа совершает разворот по дуге окружности, радиус которой зависит от текущих угла крена и истинной скорости БЛА. Значение мгновенного радиуса разворота БЛА можно рассчитать по формуле:

$$R = \frac{V^2}{g \tan \gamma},$$

где V — истинная скорость БЛА; γ — угол крена; g — ускорение свободного падения.

Напишите программу на языке C++ или Python, выполняющую расчет мгновенного радиуса разворота БЛА по заданным значениям истинной скорости и угла крена. При решении задачи считать, что ускорение свободного падения $g = 9,81 \frac{M}{c^2}$.

Формат входных данных

Значение истинной скорости БЛА в м/с и значение его угла крена в градусах. Значения входных параметров разделены между собой пробелом.

V.VV G.GG

Формат выходных данных

Значение мгновенного радиуса разворота БЛА в метрах с точностью до первого знака после запятой.

RR.R

Примеры

Пример №1

Стандартный ввод
17.89 8.6
Стандартный вывод
215.7

Решение

Для нахождения мгновенного радиуса разворота БЛА необходимо подставить в предоставленную в условии формулу заданные значения истинной скорости, угла крена, а также постоянную величину ускорения свободного падения.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 import math
2 arr = [float(i) for i in input().split()]
3 V = arr[0]
4 gamma = arr[1]
5 g = 9.81
6 R = V ** 2 / (g * math.tan(gamma * math.pi / 180.0))
7 print("{:3.1f}".format(R))

```

Задача II.1.8. Определение угловой скорости азимутального разворота БЛА (10 баллов)

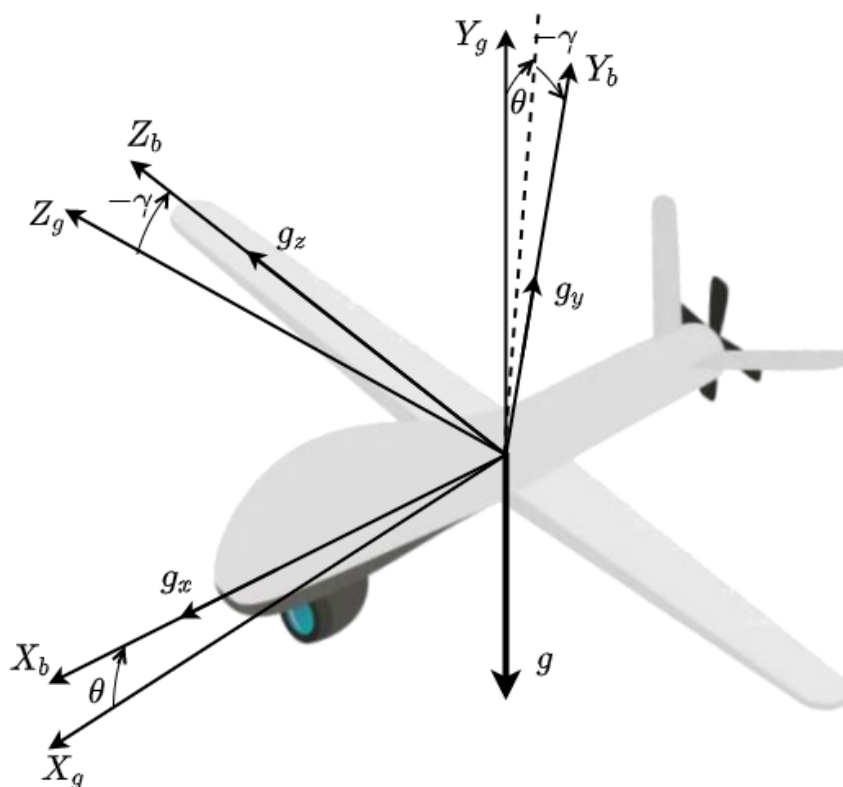
Темы: физика полета, уравнение, проекции, программирование.

Понимание основ физики полета БЛА самолетного типа необходимо для разработки системы автоматического управления для такого объекта.

Условие

Определение углов ориентации БЛА в процессе полета является нетривиальной задачей. Одним из источников информации об углах ориентации БЛА является датчик ускорений — акселерометр. Этот датчик позволяет измерять проекции полного вектора ускорения БЛА на его связанные оси: продольную, поперечную и вертикальную. Таким образом, зная расположение вектора полного ускорения БЛА, можно определить углы его ориентации.

Предположим, что БЛА самолетного типа совершает азимутальный разворот с постоянной скоростью. При этом известны проекции вектора ускорения свободного падения на оси связанной системы координат.



На рисунке обозначены:

- X_g, Y_g, Z_g — оси нормальной земной системы координат;
- X_b, Y_b, Z_b — оси связанной системы координат;
- g — вектор ускорения свободного падения;
- g_x, g_y, g_z — проекции ускорения свободного падения на оси X_b, Y_b, Z_b ;
- θ, γ — углы тангажа и крена соответственно.

Напишите программу на языке C++ или Python, определяющую угловую скорость азимутального разворота БЛА по заданным значениям истинной скорости полета и проекциям вектора ускорения свободного падения на его продольную, поперечную и вертикальные оси.

Формат входных данных

Значение истинной скорости в м/с и значения проекций вектора ускорения свободного падения (в м/с²) на оси X_b, Y_b, Z_b , связанной системы координат. Все значения входных параметров разделены между собой пробелами.

V.VV X.XX Y.YY Z.ZZ

Формат выходных данных

Значение угловой скорости азимутального разворота БЛА в град/с с точностью до первого знака после запятой.

RR.R

Примеры*Пример №1*

Стандартный ввод
22.6 -0.82 -9.65 1.71
Стандартный вывод
4.4

Список литературы

1. Связанная система координат https://oat.mai.ru/book/glava05/5_6/5_6.html

Решение

Первым шагом необходимо определить модуль ускорения свободного падения, действующего на БЛА. В рамках этой задачи он несколько отличается от стандартного значения $g = 9,81$ м/с²:

$$|g| = \sqrt{g_x^2 + g_y^2 + g_z^2}.$$

Теперь угол крена БЛА можно определить по формуле:

$$\gamma = \arcsin \frac{F_z}{g}.$$

Зная угол крена, можно посчитать радиус мгновенного разворота БЛА:

$$R = \frac{V^2}{g \tan \gamma}.$$

Угловая скорость азимутального разворота может быть найдена, используя соотношение для движения точки по окружности:

$$\dot{\Psi} = \frac{V}{R}.$$

Перевод угловой скорости из рад/с в град/с производится по формуле:

$$\dot{\Phi}_{\text{град/с}} = \dot{\Phi}_{\text{рад/с}} \frac{\pi}{180^\circ}.$$

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
1 import math
2 arr = [float(i) for i in input().split()]
3 V = arr[0]
4 Fx = arr[1]
5 Fy = arr[2]
6 Fz = arr[3]
7 F = math.sqrt(Fx**2 + Fy**2 + Fz**2)
8 gamma = math.asin(Fz/F)
9 R = V ** 2 / (F * math.tan(gamma))
10 dpsl = V / R
11 print("{:3.1f}".format(dpsl*180.0/math.pi))
```

Командная часть

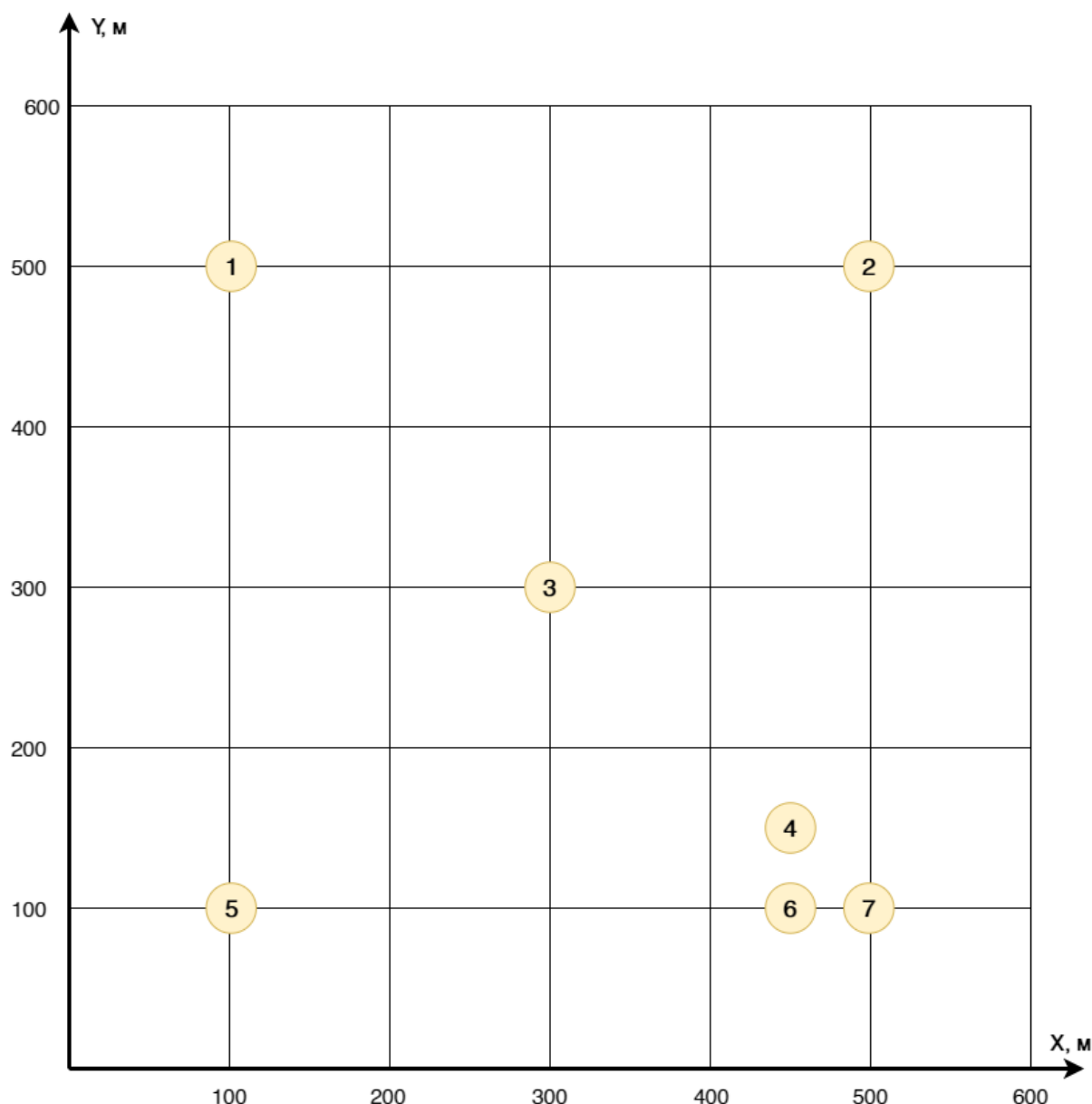
Задача П.2.1. Построение эффективной траектории полета БЛА (20 баллов)

Темы: комбинаторика, физика полета, формирование траектории.

Для решения практически любых задач, связанных с БЛА, требуется построение эффективных траекторий полета. Критерий эффективности часто учитывает не только физику полета самого БЛА, но и всевозможные сторонние ограничения.

Условие

Одной из задач, которые решает система автоматического управления (САУ) в процессе полета БЛА, является построение эффективной траектории полета. Эффективность траектории может оцениваться по разным критериям: затраченное время, пройденный путь, количество разворотов, энергопотребление и т. д. Однако одним из самых распространенных критериев эффективности является минимизация времени полета.



Рассмотрим ситуацию, когда БЛА совершает полет в горизонтальной плоскости. Задачей БЛА является достижение каждой из 7-ми обозначенных на рисунке путевых точек за наименьшее время.

Напишите программу на языке Python, реализующую достижение БЛА всех путевых точек за время, не превышающее 130 секунд. Для этого необходимо сформировать управляющие команды для БЛА, каждая из которых состоит из значений путевой скорости, угла крена и времени выполнения команды.

Моделирование полета БЛА на платформе Stepik осуществляется двумя скрытыми функциями:

```
set_pos(x0, y0, yaw0)
set_cmd(Vel, Roll, Time)
```

Функция `set_pos()` принимает в качестве аргументов начальные значения координат (x , y , в метрах) и угла курса (yaw , в градусах). Эту функцию можно вызвать лишь один раз, в самом начале программы. В противном случае программа выдаст автоматическое предупреждение. Пример использования функции:

```
set_pos(100, 0, 0) # Задать начальное положение БЛА
```

Функция `set_cmd()` принимает в качестве аргументов значения путевой скорости (`Vel`, в м/с), угла крена (`Roll`, в градусах) и времени выполнения команды (`Time`, в секундах). После получения этой команды осуществляется моделирование полета БЛА с заданными параметрами в течении заданного времени. В процессе выполнения команды путевая скорость и угол крена считаются постоянными. Изменение путевой скорости и угла крена БЛА при отправке новой команды происходит мгновенно, переходные процессы не учитываются. Пример использования функции:

```
set_cmd(20, 0, 30) # Лететь по прямой
```

Условия моделирования:

- Скорость полета БЛА ограничена диапазоном [36...72] км/ч.
- Максимальный угол крена БЛА ограничен диапазоном [-10...10] градусов.
- Время выполнения команды в функции `set_cmd()` задается в целых секундах.
- БЛА может совершать развороты вправо (по часовой стрелке) и влево (против часовой стрелки), а также лететь по прямой (при нулевом угле крена).
- Требуемая точность достижения каждой из путевых точек — 10 метров (БЛА должен попасть в область радиусом 10 метров от заданных координат путевой точки).
- В рамках задачи БЛА считается материальной точкой. Выход БЛА за пределы зоны обозначенной на рисунке разрешен.
- Начальное положение БЛА может быть произвольным и задается функцией `set_pos()`. Допускается задание начального положения БЛА в одной из путевых точек.

Координаты путевых точек:

1 — 100, 500

2 — 500, 500

3 — 300, 300

4 — 450, 150

5 — 100, 100

6 — 450, 100

7 — 500, 100

Порядок достижения путевых точек не имеет значения.

У этой задачи есть множество вариантов решения. Программа автоматической проверки решений оценивает количество достигнутых путевых точек и суммарное время полета БЛА.

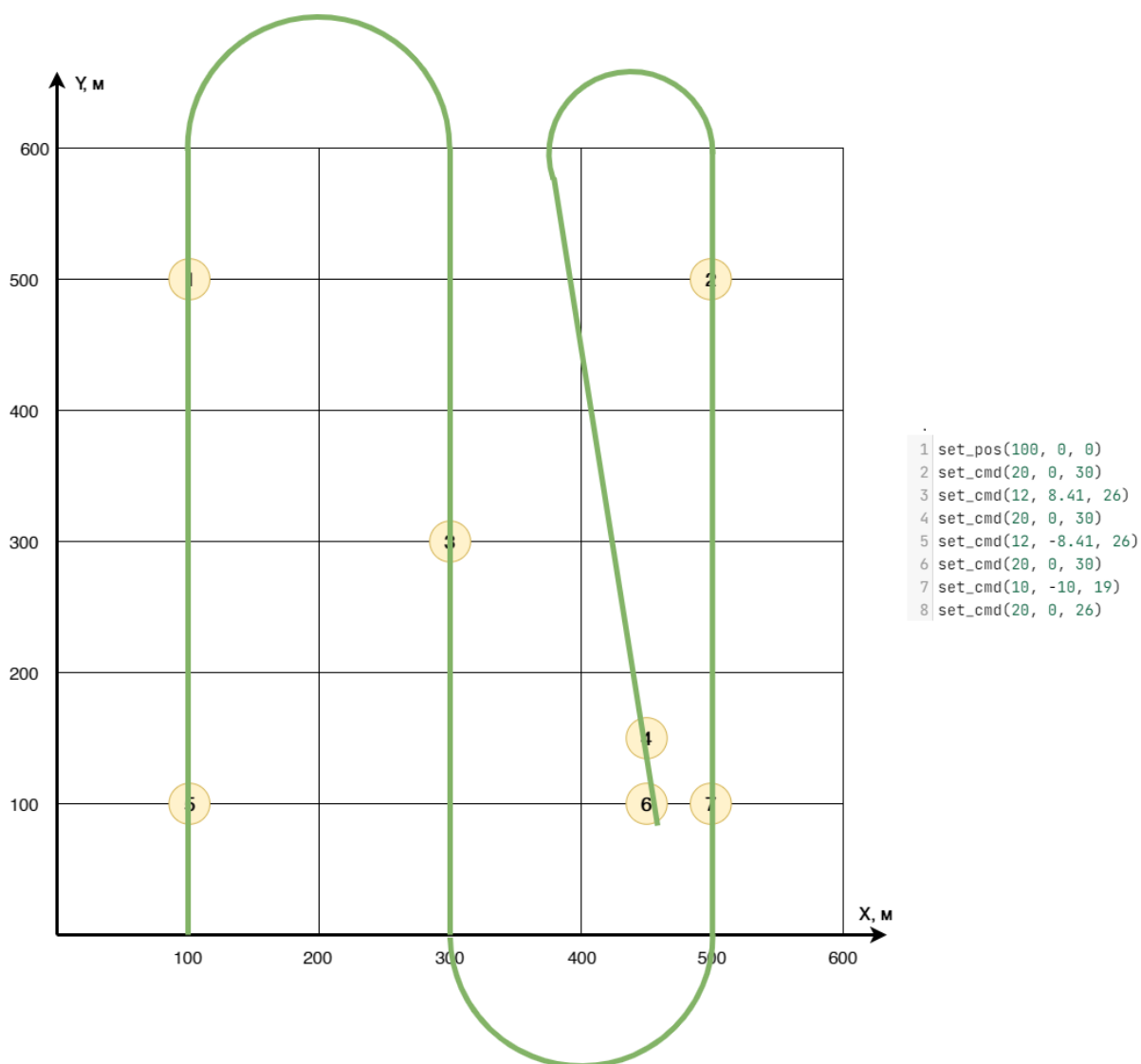
Формат входных данных

Все параметры моделирования указаны в условии задачи. Входных данных нет.

Формат выходных данных

Выходные данные формируются программой автоматически. Оценивается только количество достигнутых путевых точек (должны быть достигнуты все 7 точек) и затраченное время (не должно превышать 130 секунд). Весь остальной вывод в консоль игнорируется, можно выводить отладочную печать.

Пример выполнения программы



Пример результата программы

Test output:

```

Статус точек (1=достигнута): [1, 1, 1, 1, 1, 1, 1]
Порядок достижения точек: 5137246
Точек достигнуто: 7 время затрачено: 187

```

Решение

У задачи есть несколько вариантов решения. Один из вариантов выглядит так:

1. Выставка начального положения БЛА в первой путевой точки с координатами 100, 500 и углом курса 135 градусов. Таким образом, БЛА сразу достигает путевой точки 1.
2. Полет по прямой с максимальной путевой скоростью 20 м/с в течение 35 секунд. При этом осуществляется достижение путевых точек 3, 4 и 7.
3. Разворот БЛА на 235 градусов влево с минимальной путевой скоростью 10 м/с — разворот в течение 22 секунд с заданным углом крена -10 градусов и последующий разворот в течение 1 секунд с углом крена $-7,1$ градусов.
4. Полет по прямой с максимальной путевой скоростью 20 м/с в течение 26 секунд. При этом осуществляется достижение путевых точек 6 и 5.
5. Разворот БЛА на 135 градусов вправо с минимальной путевой скоростью 10 м/с — разворот в течение 14 секунд с заданным углом крена 10 градусов и последующий разворот в течение 1 секунд с углом крена 7,2 градусов.
6. Полет по прямой с максимальной путевой скоростью 20 м/с в течение 26 секунд. При этом осуществляется достижение путевой точки 2.

Время полета БЛА при использовании этого алгоритма составляет 124 секунды.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```

1 set_pos(100, 500, 135)
2 set_cmd(20, 0, 35)
3 set_cmd(10, -10, 22)
4 set_cmd(10, -7.1, 1)
5 set_cmd(20, 0, 26)
6 set_cmd(10, 10, 14)
7 set_cmd(10, 7.2, 1)
8 set_cmd(20, 0, 26)

```

Задача II.2.2. Определение координат объекта системой технического зрения (20 баллов)

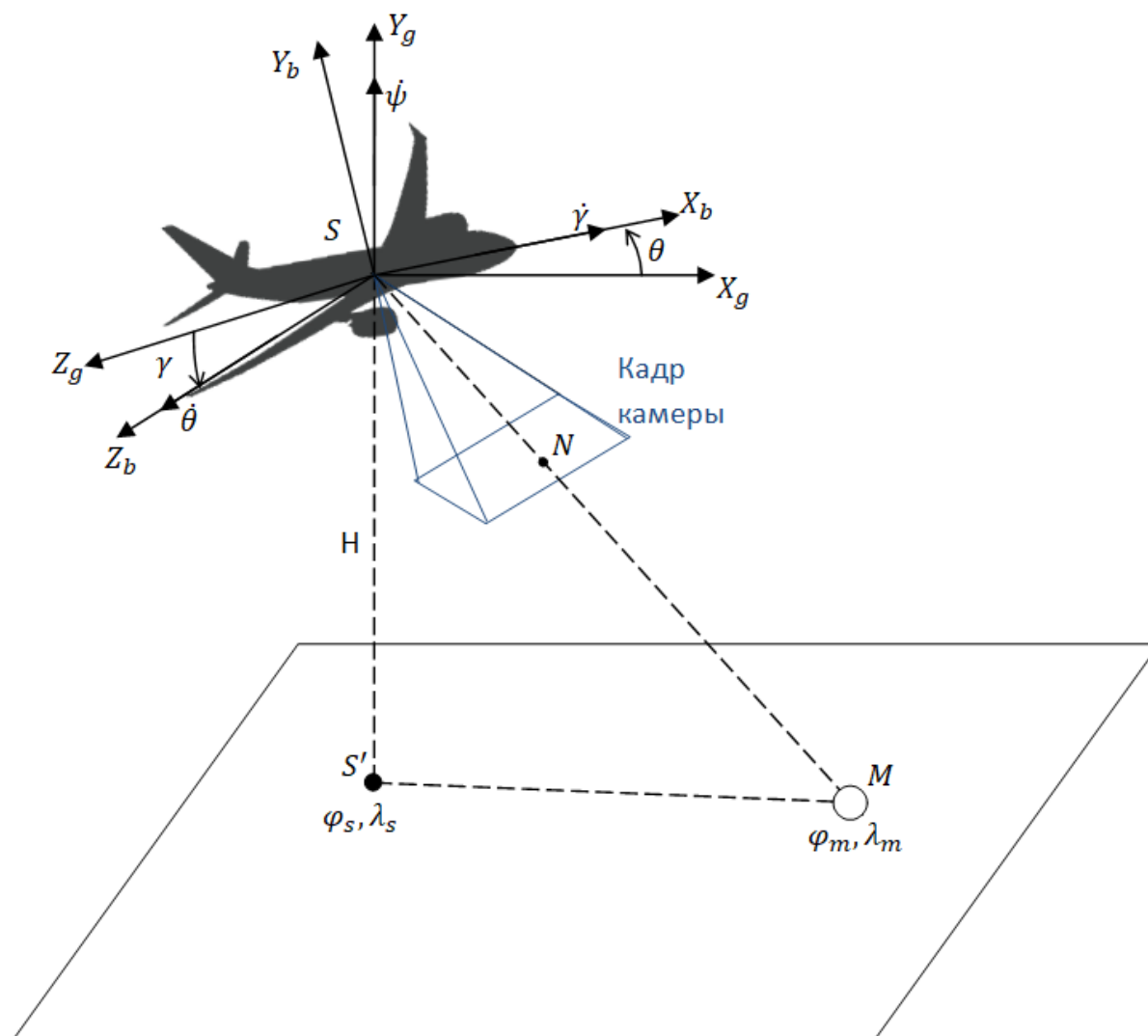
Темы: физика полета, проекции, оптическое искажение, техническое зрение.

Система технического зрения являются одним из основных источников информации об окружающем пространстве при решении задач автоматизированного поиска объектов. Навыки нахождения проекций и работы с несколькими системами координат пригодятся участникам в финале при разработке аналогичной системы для реального БЛА.

Условие

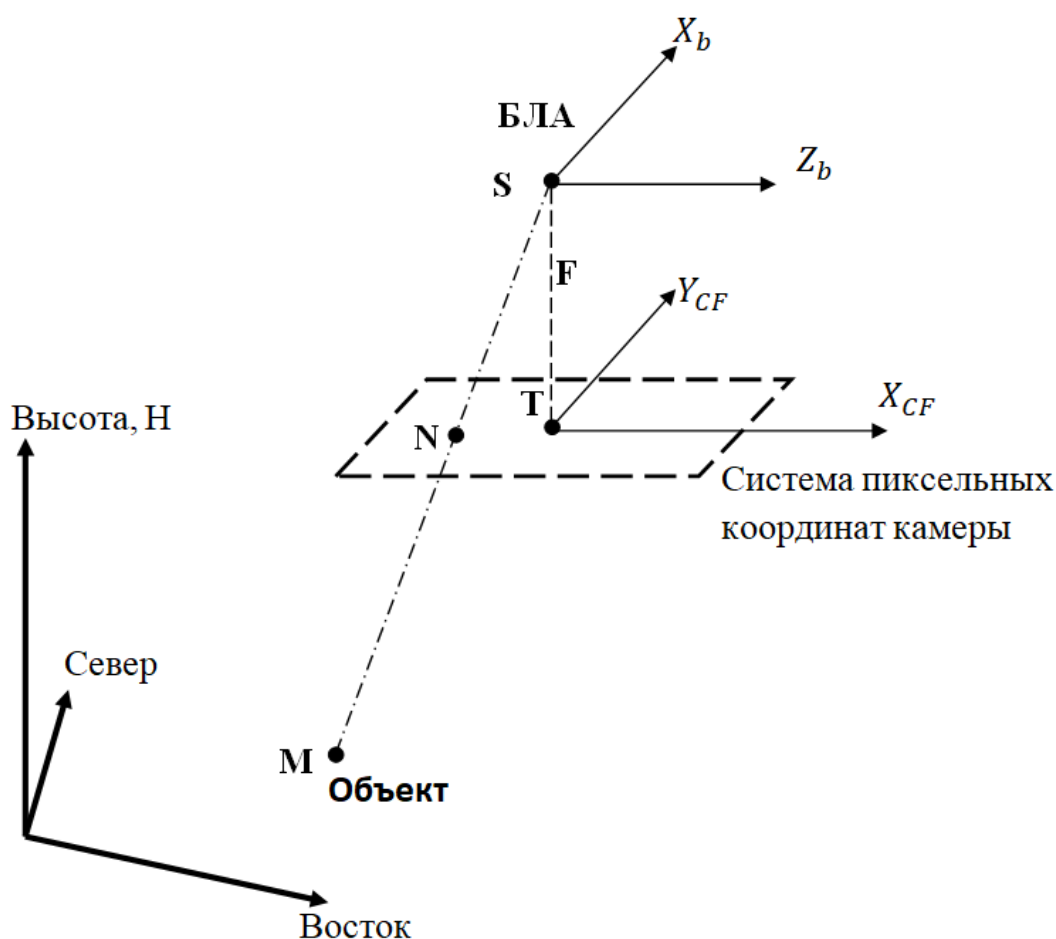
При разработке БЛА, оснащенного системой технического зрения, часто возникает задача определения координат объекта (или ориентира) в кадре камеры. При

этом линейное смещение объекта в кадре камеры будет зависеть не только от фактического расстояния от БЛА до объекта, но и от множества других параметров: высоты полета БЛА, углов его ориентации, параметров камеры.



В процессе полета БЛА обнаруживает объект M . Напишите программу на языке C++ или Python, которая определяет географические координаты объекта (точки M) φ_m, λ_m при заданных углах ориентации ЛА (курс ψ , тангаж θ , крен γ), текущем местоположении БЛА $\varphi_s = 56,097977$, $\gamma_s = 35,877670^\circ$, высоте полета БЛА H , и пиксельных координатах точки N в кадре бортовой камеры.

Считается, что камера направлена вертикально вниз при нулевых углах тангажа и крена БЛА (ось Y_{cf} кадра камеры совпадает с направлением оси X_b БЛА, а направление оси X_{cf} кадра совпадает с направлением оси Z_b БЛА) и поворачиваются вместе с БЛА при ненулевых значениях углов курса, тангажа и крена.

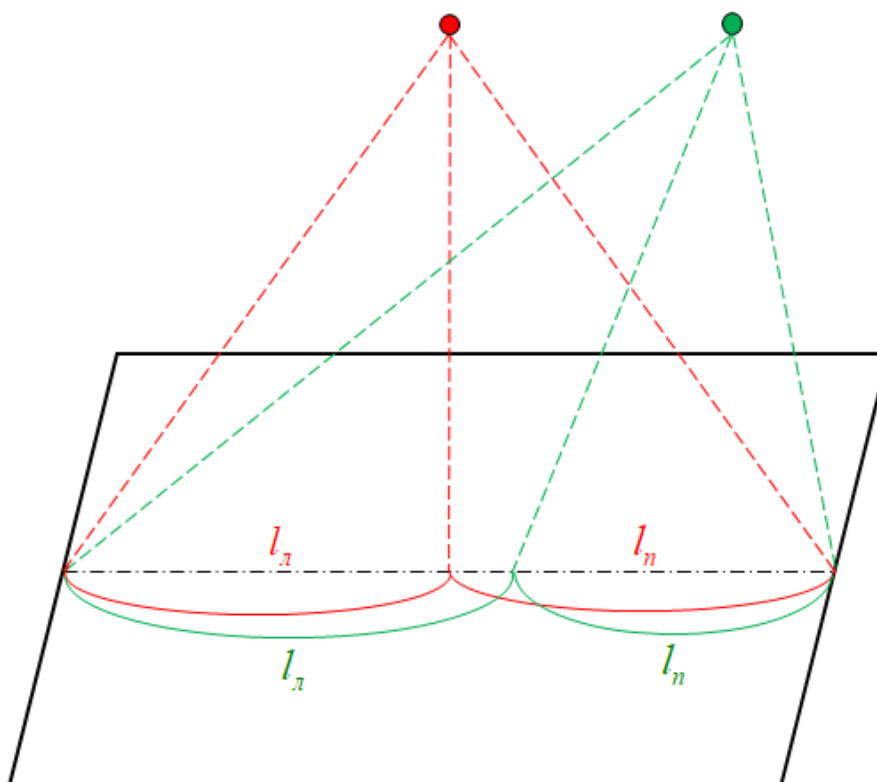


Характеристики камеры:

- разрешение $RESX \times RESY = 2000 \times 1600$;
- фокусное расстояние камеры $F = 4,8$ мм (расстояние от точки S до центра изображения T в системе пиксельных координат камеры);
- размер одного пикселя на изображении, полученном с камеры, $sp = 7,2$ мкм. Пиксели изображения считать квадратными (размер пикселя вдоль осей X_{cf} и Y_{cf} одинаков).

При пересчете долготы и широты в метрические координаты использовать коэффициенты пересчета, найденные для точки S. При нахождении этих коэффициентов считать Землю шаром с радиусом 6371 км. При расчетах считать, что подстилающая поверхность ровная, и ее высота равна 0 м.

При решении задачи необходимо также учитывать оптическое искажение изображения при наличии углов тангажа и крена БЛА.



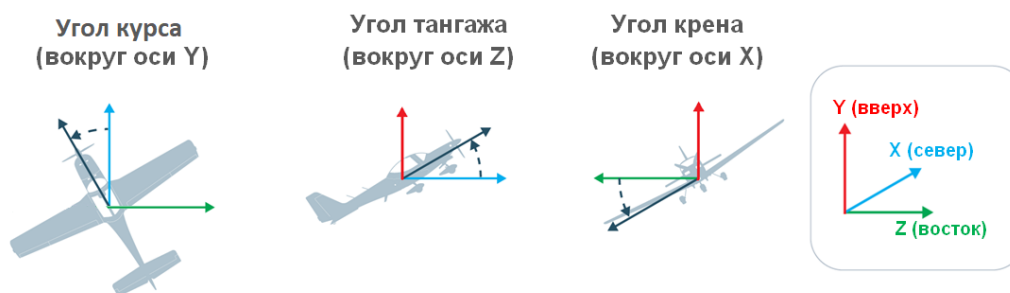
При наличии углов крена и тангажа БЛА возникает оптическое искажение изображения, полученного бортовой камерой. Это явление возникает, когда плоскость камеры не параллельна плоскости подстилающей поверхности.

Если углы тангажа и крена равны нулю — искажения не возникает и $l_n = l_d$. Этот случай обозначен на рисунке красными линиями.

Если же углы тангажа и крена не равны нулю — длина l_d на поверхности земли будет больше длины l_n . При этом разрешение снимка камеры остается постоянным. Это означает, что пикселям, лежащим в левой части кадра (в данном примере), будет соответствовать большее расстояние в метрах на поверхности, чем пикселям в правой части кадра, так как они находятся дальше от БЛА. Этот случай обозначен на рисунке зелеными линиями.

Величина оптического искажения в рамках решения данной задачи может достигать 5 метров.

Обратите внимание на расположение осей связанной с БЛА системы координат и на направление разворотов по углам курса, тангажа и крена!



Формат входных данных

Координаты объекта в системе пиксельных координат камеры (точка N) X и Y в градусах широты и долготы соответственно; высота полета БЛА H в метрах; углы курса Y , тангажа P и крена R в градусах. Все значения разделены пробелами.

XXXX YYYYY HH YY.Y PP.P RR.R

Формат выходных данных

Значения φ_m , λ_m географических координат объекта (точки M) в градусах широты и долготы с точностью до 6-го знака после запятой. Значения должны быть разделены пробелом.

FF.FFFFFFF LL.LLLLLL

Примеры

Пример №1

Стандартный ввод
400 250 80 30.5 3.1 4.2
Стандартный вывод
56.098427 35.877966

Список литературы

1. Географические координаты <https://v-ipc.ru/guides/coord>.
2. Углы Эйлера https://ru.wikipedia.org/wiki/Углы_Эйлера.

Критерии оценивания

Это самая сложная задача профиля, поэтому в ней используется гибкий критерий оценки. Количество баллов, начисляемое за задачу, зависит от точности решения задачи участниками. Сводная таблица оценивания задачи в зависимости от величины ошибки решения.

Значение ошибки	Баллы
0,0	20
< 1,0	10
< 2,0	9
< 3,0	8
< 5,0	7
< 7,5	6
< 10,0	5
< 20,0	4
< 30,0	3
< 40,0	2
< 50,0	1
≥ 50	0

Решение

Для решения задачи необходимо:

1. Определить смещение центра кадра камеры относительно текущего местоположения БЛА. Для этого вектор визирной линии камеры (направленный вертикально вниз) нужно повернуть на углы ориентации БЛА при помощи матрицы направляющих косинусов.
2. Определить смещение координат объекта относительно центра кадра в метрах.
3. Определить величину искажения изображения, вызванного наличием углов крена и тангажа.
4. Сложить полученные в пунктах 1–3 величины для каждой из осей.
5. Перевести полученное суммарное смещение объекта относительно БЛА в градусы широты и долготы.
6. Прибавить рассчитанное смещение к исходным координатам БЛА.

Пример программы-решения

Ниже представлено решение на языке C++.

```

1  #include <iostream>
2  #include <cmath>
3  #include <iomanip>
4
5  using namespace std;
6
7  const double uavLat = 56.097977; // degrees
8  const double uavLon = 35.877670; // degrees
9  const double F = 4.8 * 1e-3;     // meters
10 const double sp = 7.2 * 1e-6;    // meters
11 const int width = 2000;          // pixels
12 const int height = 1600;         // pixels
13 const double R = 6371 * 1e+3;    // meters
14
15 void rotate(double &x, double &y, double angle)
16 {
17     double x1 = x,
18           y1 = y;
19
20     angle = angle / 180 * M_PI;
21
22     x = x1 * cos(angle) - y1 * sin(angle);
23     y = y1 * cos(angle) + x1 * sin(angle);
24 }
25
26 int main()
27 {
28     double X, Y, H;
29     double yaw, pitch, roll;
30     cin >> X >> Y >> H >> yaw >> pitch >> roll;
31
32     double x, y, z;
33     x = Y * sp;
34     y = -F;
35     z = X * sp;
36

```

```
37 rotate(y, z, roll);
38 rotate(x, y, pitch);
39 rotate(z, x, yaw);
40
41 x *= -H / y;
42 z *= -H / y;
43 y *= -H / y;
44
45 const double meters2degZ = 360 / (2 * M_PI * R * cos(uavLat / 180 * M_PI));
46 const double meters2degX = 360 / (2 * M_PI * R);
47
48 cout << fixed << setprecision(6);
49 cout << uavLat + x * meters2degX << ' ' << uavLon + z * meters2degZ << "\n";
50 //cout << x << ' ' << z << " ответ в метрах\n";
51 //cout << (56.097826 - uavLat) / meters2degX << ' ' << (35.877520 - uavLon) /
    ↪ meters2degZ << " верный ответ в метрах\n";
52 }
```

Заключительный этап

Индивидуальный предметный тур

Информатика. 8–11 класс

Задача III.1.1.1. Разработка новой игры (10 баллов)

Две конкурирующие фирмы-разработчики компьютерных игр решили объединиться и создать абсолютно новый игровой мир. Производственный процесс у первой фирмы подразумевает, что она разрабатывает в единицу времени a игровых уровней, а вторая фирма за этот же период может разработать b уровней. Известно, что $a \neq b$. Так как некоторые замыслы требуют использования уникального оборудования, предоставляемого третьей фирмой, то два уровня не могут разрабатываться одновременно.

В итоге в созданной игре оказалось n уровней и на ее производство было затрачено m единиц времени. Необходимо определить сколько из этих уровней разработала первая фирма, а сколько — вторая.

Формат входных данных

В первой строке содержатся через пробел числа a и b , $a \neq b$, $1 \leq a, b \leq 10^5$, во второй тоже через пробел содержатся числа n и m . $1 \leq n, m \leq 10^9$.

Формат выходных данных

Вывести два числа через пробел — количество уровней в игре, которые были разработаны первой и второй фирмой соответственно.

Примеры

Пример №1

Стандартный ввод
5 9 88 12
Стандартный вывод
25 63

Решение

Задачу можно решить при помощи системы из двух линейных уравнений с двумя неизвестными, а можно и при помощи следующих рассуждений.

Так как $a \neq b$, то найдем минимум из этих чисел (пусть для определенности это число a). Тогда за t единиц времени будет обеими фирмами разработано не менее $t \cdot a$ уровней. Если из общего числа уровней вычтем эту величину, то оставшиеся $n - t \cdot a$ уровней разработаны фирмой, которая производит больше (в данном случае это вторая фирма), и эта разность делится на разность $b - a$, за счет которой вторая фирма выпускает больше уровней. Поделим $n - t \cdot a$ на $b - a$ (более точно $n - t \cdot \min(a, b)$ на $\max(a, b) - \min(a, b)$) и получим количество уровней, которые разработала более быстро работающая фирма. Вычтем это из n и получим ответ для более медленно работающей фирмы.

Задача III.1.1.2. Полет по зацикленной программе (15 баллов)

Программируемый квадрокоптер в момент времени 0 находится над точкой с координатами $(0, 0)$. В него заложена программа, которая состоит из n команд вида N, E, S, W .

Команда N увеличивает вторую координату на 1, команда E увеличивает первую координату на 1, команда S уменьшает вторую координату на 1, команда W уменьшает первую координату на 1. За один такт квадрокоптер выполняет ровно одну команду. Программа зациклена, то есть после выполнения n -ой команды, квадрокоптер начинает выполнять всю программу заново.

Заряда аккумулятора у квадрокоптера хватит на выполнение t перемещений, после их выполнения квадрокоптер опустится и будет ожидать подзарядки. Требуется по загруженной программе и числу t определить точку, в которой окажется квадрокоптер.

Формат входных данных

В первой строке содержится два числа n и t . $1 \leq n \leq 10^5$, $1 \leq t \leq 10^{18}$.

Во второй строке содержится n команд из множества N, E, S, W — описание программы, загруженной в квадрокоптер.

Формат выходных данных

Вывести два числа через пробел — координаты квадрокоптера после выполнения им зацикленной программы из t команд.

Примеры

Пример №1

Стандартный ввод
11 20 NEESWNNENWEN
Стандартный вывод
3 5

Решение

Так как величина m очень большая, то просто промоделировать полет не удастся. Для ее решения нужно узнать, на какую величину изменится положение квадрокоптера по обеим координатам после однократного выполнения программы (пусть эти изменения равны Dx и Dy). Далее вычислить при помощи простого целочисленного деления, сколько раз квадрокоптер успеет выполнить программу целиком (m/n). Затем подсчитать, в какой точке окажется квадрокоптер после окончания последней целиком выполненной программы ($m/n \cdot Dx, m/n \cdot Dy$). После этого определить, сколько еще команд останется для выполнения в остатке, и выполнить их количество, начав из указанной точки.

Задача III.1.1.3. Синхронизация (20 баллов)

Группа из n спутников расположена на координатной прямой в точках с координатами $1, 2, \dots, n$. Каждый спутник характеризуется мощностью генерируемого им сигнала p_i . Если спутник в точке x_i имеет мощность сигнала p_i , то все спутники, находящиеся на расстоянии не большем чем p_i , могут получить сигнал с этого спутника. Сам спутник свой сигнал не получает.

Для синхронизации работы, все спутники одновременно послали свои сигналы всем остальным спутникам, до которых их сигнал доходит. Требуется для каждого спутника определить, от скольких других спутников он получил сигнал.

Формат входных данных

В первой строке находится число n — количество спутников, $2 \leq n \leq 10^5$. Во второй строке через пробел перечислены n целых чисел — мощности соответствующих спутников. В i -ой позиции второй строки содержится мощность спутника, находящегося в точке i на координатной прямой. Все мощности — целые числа в пределах от 1 до 10^9 .

Формат выходных данных

Вывести n чисел через пробел, характеризующие количество сигналов, полученных спутниками в процессе синхронизации. В i -ой позиции должно быть указано количество сигналов, полученных спутником, находящимся в точке i на координатной прямой.

Примеры

Пример №1

Стандартный ввод
10 5 3 1 1 7 4 6 2 1 5
Стандартный вывод
3 5 6 6 6 5 4 5 5 5

Решение

Для каждого спутника определим границы отрезка, в котором все спутники получат сигнал от рассматриваемого (при этом следует учесть границы всего множества спутников). Пусть для i -го спутника эти границы равны L_i и R_i . Теперь произведем для каждого такого отрезка прибавление к каждой его точке по 1. Но напрямую это сделать не получится в связи с тем, что спутников может быть до 10^5 . Тогда легко понять, что в явном виде можно сделать до 10^{10} прибавлений, что не уложится в ограничения по времени на многих тестах. Делаем более аккуратно: заведем массив балансов, заполним его нулями. Далее для каждого начала L_i прибавим в эту точку единицу, а для каждого окончания R_i вычтем из точки $R_i + 1$ единицу. Теперь на соответствующем отрезке для каждой его точки можно считать, что произошла прибавка единицы, и после его окончания она исчезла. Далее перебираем значения баланса, прибавляем его очередное значение к текущей переменной (которая изначально была равна 0) и выводим текущий ответ.

Задача III.1.1.4. Два беспилотника (25 баллов)

Из двух точек трехмерного пространства одновременно стартовали два беспилотных летательных устройства. Каждое движется неограниченно и прямолинейно с постоянной скоростью. Требуется определить минимальное расстояние между ними, которое будет в какой-либо момент времени.

Формат входных данных

В первой строке содержатся координаты через пробел (x_1, y_1, z_1) стартовой точки первого устройства.

Во второй строке содержатся координаты через пробел (vx_1, vy_1, vz_1) вектора, задающего направление и скорость полета первого устройства.

В третьей строке содержатся координаты через пробел (x_2, y_2, z_2) стартовой точки второго устройства.

В четвертой строке содержатся координаты через пробел (vx_2, vy_2, vz_2) вектора, задающего направление и скорость полета второго устройства.

Все координаты целые в пределах от -10^5 до 10^5 включительно. Координаты любого направляющего вектора одновременно не равны 0.

Формат выходных данных

Вывести одно вещественное число — минимальное расстояние, которое будет между беспилотниками в какой-либо момент времени. Число должно быть выведено с точностью ровно 7 знаков после десятичной точки.

Примеры*Пример №1*

Стандартный ввод
0 0 0 3 3 3 10 10 0 0 0 2
Стандартный вывод
3.2444284

Пример №2

Стандартный ввод
0 3 -100000 0 0 2 0 0 100000 0 0 1
Стандартный вывод
3.0000000

Пример №3

Стандартный ввод
0 0 -100000 0 0 1 0 0 100000 0 0 1
Стандартный вывод
200000.0000000

Решение

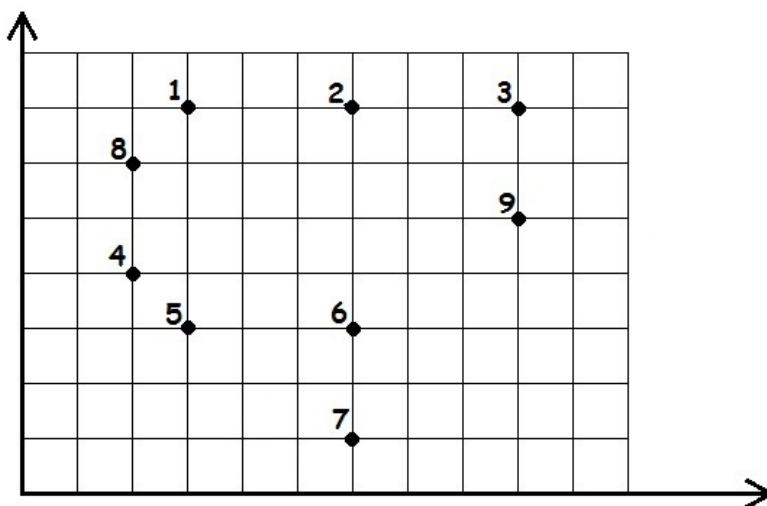
За счет прямолинейности движения и постоянной скорости можно показать, что беспилотники либо в любой момент времени находятся на одном и том же расстоянии, либо существует ровно один момент, когда расстояние минимально, до этого оно убывало, а после этого возрастает. Такие функции называются унимодальными и имеют ровно один глобальный экстремум (возможно на всей области определения). Отрезки возрастания или убывания могут тоже вырождаться в точку. В любом из этих случаев для поиска экстремума (в нашем случае минимума) можно использовать тернарный или троичный поиск на отрезке. Суть его в следующем: разобьем отрезок $[A, B]$, на котором находится минимум функции, на три равные части двумя точками $t_1 = A + (B - A)/3$ и $t_2 = B - (B - A)/3$ (вычисления вещественные). Удалим из отрезка ровно одну из двух крайних его третей путем сравнения значения целевой функции в точках t_1 и t_2 . Легко показать, что выбрать для удаления нужно (при поиске минимума) тот крайний отрезок, значение на границе которого в соответствующей точке больше. Если значения равны, то удалить можно любой крайний

отрезок. После удаления точка минимума останется внутри оставшихся двух третей. После этого повторим рассуждения для нового меньшего отрезка. При таком подходе длина следующего отрезка в 1,5 раза меньше длины предыдущего, то есть экспоненциально уменьшается.

В нашем случае целевой функцией является расстояние между двумя точками в трехмерном пространстве, которое вычисляется по обычной формуле: корень квадратный из суммы квадратов разностей координат. Работаем на числовой прямой, значения которой соответствуют моментам времени t . В качестве стартового значения A возьмем $t = 0$. Так как самое большое содержательное значение времени сближения может быть в случае, когда разница скоростей минимальна (равна 1 в нашем целочисленном случае), а направление совпадает, то не более чем через $2 \cdot 10^5$ сближение до минимального расстояния произойдет. Таким образом, R можно взять, например, 10^6 . Далее, например, 100 раз делаем одну и ту же операцию: делим текущий отрезок точками t_1 и t_2 , вычисляем координаты беспилотников в эти два момента времени, вычисляем расстояния между ними в эти два момента времени и отбрасываем тот конец отрезка, в соответствующей которому точке расстояние больше. Ответ будет равен расстоянию между беспилотниками в любой точке оставшегося отрезка.

Задача III.1.1.5. Надежность соединения (30 баллов)

Дано n точек на плоскости. Между любыми двумя точками можно передать сообщение как напрямую, так и используя цепочку соединений между несколькими промежуточными точками. Пусть требуется передать сообщение по цепочке A_1, A_2, \dots, A_k . В этом случае считаем, что сообщение передается последовательно между каждыми двумя соседними точками в этой цепочке. Тогда надежностью этой цепочки будем считать наибольшее расстояние между двумя соседними точками в этой цепочке. Чем это расстояние меньше, тем надежнее цепочка.



Рассмотрим, например, набор точек, приведенных на рисунке. Допустим, необходимо построить самое надежное соединение между точками 2 и 6. Если организовать его напрямую, то его надежность будет равна 4. Если построить цепочку $2 - 3 - 9 - 6$, то надежность этого соединения будет равна расстоянию между точками 9 и 6, то есть $\sqrt{13}$. Но лучше всего построить цепочку $2 - 1 - 8 - 4 - 5 - 6$. Тогда надежность

этой цепочки будет равна 3. Далее для любой пары точек будем рассматривать самую надежную цепочку между этими точками.

Можно убедиться, что в данном случае между любыми двумя точками можно организовать цепочку с надежностью не хуже, чем 3. То есть можно утверждать, что соединение между точками 2 и 6 входит в число самых ненадежных.

Для определения надежности всего множества нужно найти количество пар точек, соединение между которыми самое ненадежное.

Формат входных данных

В первой строке содержится число n — количество точек на плоскости, $2 \leq n \leq 1000$. В следующих n строках содержатся координаты x_i, y_i каждой точки, разделенные пробелом. Все координаты в пределах от -10^5 до 10^5 .

Формат выходных данных

Вывести одно число — количество пар точек, соединение между которыми самое ненадежное для этого множества точек.

Примеры

Пример №1

Стандартный ввод
9
3 7
6 7
9 7
2 4
3 3
6 3
6 1
2 6
9 5
Стандартный вывод
28

Решение

Для решения задачи нужно найти минимально возможное d , такое, что для любой пары точек самая надежная цепочка между ними имеет надежность не больше, чем d . Для наглядности поиска этого d можно привести такую наглядную динамику: для каждой точки, как для центра окружности, будем рисовать окружности одинакового непрерывно возрастающего радиуса R . Если другая точка попадает внутрь окружности для текущего центра, то эти две точки соединяются. Нас интересует такое минимальное R , при котором все множество наших точек будет соединено полностью, некоторые точки будут соединены при этом, возможно, посредством промежуточных. Такое R и будет являться нашим искомым значением d .

Для его нахождения нужно найти максимальное ребро в минимальном каркасе для полного графа, вершинами которого являются наши точки, а весами ребер — расстояния между ними. Для текущих условий и ограничений лучше всего воспользоваться алгоритмом Прима поиска минимального каркаса.

Найдя такое d , разобьем граф на несколько компонент связности таких, что между любыми двумя вершинами компоненты связности есть цепочка с надежностью строго меньше d (то есть внутри них любая пара имеет соединение лучшее, чем самое ненадежное во всем множестве). Если мы возьмем теперь любую пару точек из двух разных компонент, то эта пара будет иметь самое ненадежное соединение. Осталось для каждой компоненты подсчитать число ее элементов и далее сложить попарные произведения этих количеств.

Физика. 8–9 классы

Задача III.1.2.1. Трудная встреча (10 баллов)

Два тела, движущиеся вдоль одной прямой, в момент начала наблюдения находятся в одной точке. Пользуясь графиком зависимости проекции скорости этих тел от времени (рисунок III.1.1), определите, встретятся ли они вновь? Если да, то в какой момент времени. Ответ обоснуйте.

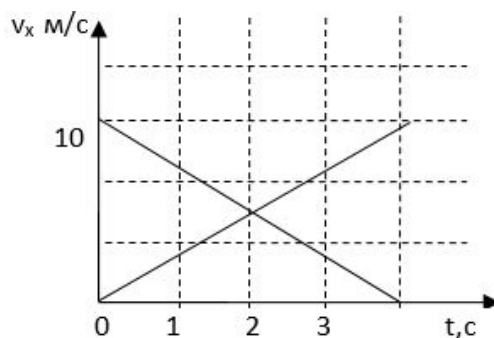


Рис. III.1.1

Решение

Перемещение численно равно площади фигуры, ограниченной графиками. Следовательно, встреча произойдет, когда площади станут равными и это произойдет через 4 с.

Ответ: 4 с.

Критерии оценивания

Задачи должны быть оформлены согласно ФГОС, а их решение демонстрировать знание законов и способы их применения в конкретной ситуации.

1. Правильно применены знания о геометрическом смысле перемещения, как площади под графиком, зависимости скорости от времени или записаны выражения для вычисления перемещения (координаты) при равноускоренном движении — 5 баллов.
2. Правильно проведены вычисления — 5 баллов.

Если решения нет, но приведены разумные рассуждения в направлении решения — 5 баллов.

Задача III.1.2.2. Исследование Венеры (25 баллов)



В декабре 1984 года с Земли к Венере были запущены автоматические станции «Вега-1» и «Вега-2», название которых отражали названия объектов исследования Венеры и кометы Галлеи. Спускаемый аппарат станций включал посадочный аппарат и аэростатный зонд. Аэростатный зонд диаметром 3,4 м состоял из тефлоновой оболочки и троса общей массой 14,6 кг. К шару была прикреплена гондола небольших размеров с размещенной на ней аппаратурой массой 6,9 кг. Ученых интересовала плотность атмосферы Венеры, где скорость ветра достигает 500 км/ч. Найдите ее, считая, что зонд находится на постоянной высоте. Объем шара рассчитывается по формуле: $V_{\text{ш}} = \frac{4}{3}\pi R^3$. Ответ предоставьте в СИ, округлив до сотых.

Решение

$$F_T = F_A.$$

$$F_T = (m_{\text{аэростата}} + m_{\text{гондолы}})g.$$

$$F_A = \rho_{\text{В}} V_{\text{аэростата}} g,$$

где $\rho_{\text{В}}$ — плотность атмосферы Венеры.

$$(m_{\text{аэростата}} + m_{\text{гондолы}})g = \rho_{\text{В}} V_{\text{аэростата}} g,$$

отсюда:

$$\rho_{\text{В}} = \frac{m_{\text{аэростата}} + m_{\text{гондолы}}}{V_{\text{аэростата}}}.$$

$$\rho_{\text{В}} = 1,0437 \text{ кг/м}^3.$$

Плотность атмосферы на этой высоте сравнима с земной.

Ответ: $\rho_{\text{В}} = 1,04 \text{ кг/м}^3$.

Критерии оценивания

Задачи должны быть оформлены согласно ФГОС, а их решение демонстрировать знание законов и способы их применения в конкретной ситуации.

- Правильно записано выражение для закона Архимеда — 5 баллов.
- Правильно записано выражение для условия плавания тел — 5 баллов.
- Правильно записано выражение для расчета силы тяжести — 5 баллов.
- Правильно записано выражение для расчета плотности — 10 баллов.

Если решения нет, но приведены разумные рассуждения в направлении решения — 10 баллов.

Задача III.1.2.3. Колыбель Ньютона (25 баллов)

Механическая система, в основе которой лежит соударение одинаковых по массе шариков, подвешенных на нити, часто называют «колыбель Ньютона» или маятник Ньютона на рисунке III.1.2 приведен пример маятника Ньютона. При отсутствии силы трения подобная система могла бы работать вечно. Сколько шариков отскочат вправо, если отклонить и отпустить два шарика слева. Ответ обоснуйте.

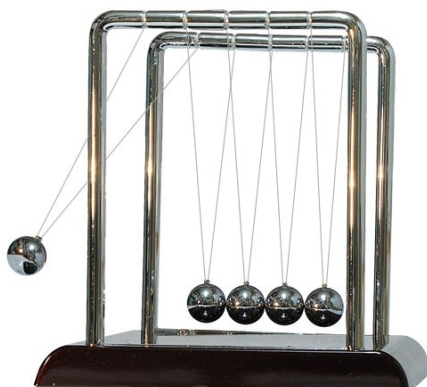


Рис. III.1.2. Маятник Ньютона

Решение

Из закона сохранения импульса:

$$N_1 m v_1 = N_2 m v_2,$$

следовательно:

$$\frac{v_1}{v_2} = \frac{N_2}{N_1}. \quad (\text{III.1.1})$$

Из закона сохранения механической энергии:

$$N_1 \frac{m v_1^2}{2} = N_2 \frac{m v_2^2}{2},$$

следовательно:

$$\frac{v_1^2}{v_2^2} = \frac{N_2}{N_1}. \quad (\text{III.1.2})$$

Из (III.1.1) и (III.1.2) получаем:

$$N_1 = N_2.$$

Ответ: 2.

Критерии оценивания

Задачи должны быть оформлены согласно ФГОС, а их решение демонстрировать знание законов и способы их применения в конкретной ситуации.

- Правильно записан закон сохранения импульса — 10 баллов.
- Правильно записан закон сохранения энергии — 10 баллов.
- Сделаны правильные выводы о количестве шаров — 5 баллов.

Если решения нет, но приведены разумные рассуждения в направлении решения — 10 баллов.

Задача III.1.2.4. Холодильник (25 баллов)

Холодильник не вырабатывает «холод». Охлаждение происходит благодаря отбору тепла и его отдаче окружающему пространству. В охлаждающих установках жидкость попадает в испаритель, поглощает тепло и переходит в парообразное состояние, при этом он отводит теплоту из внутреннего пространства камеры. Какое количество хладагента должно испариться чтобы перевести 500 мл воды, находящейся при температуре 20 °С в лед при температуре (−5 °С). Удельная теплоемкость воды 4200 $\frac{\text{Дж}}{\text{кг} \cdot \text{°С}}$, удельная теплоемкость льда 2100 $\frac{\text{Дж}}{\text{кг} \cdot \text{°С}}$, удельная теплоемкость хладагента 2000 $\frac{\text{Дж}}{\text{кг} \cdot \text{°С}}$, удельная теплота плавления льда 330000 Дж/кг, удельная теплота парообразования хладагента $2 \cdot 10^6$ Дж/кг.

Решение

$$\begin{aligned} Q_{\text{отд}} &= Q_{\text{пол}}. \\ Q_{\text{отд}} &= Lm_x. \\ Q_{\text{пол}} &= |c_{\text{в}}m_{\text{в}}\Delta t_1| + |\lambda m_{\text{л}}| + |c_{\text{л}}m_{\text{л}}\Delta t_2|. \\ m_x &= \frac{|c_{\text{в}}m_{\text{в}}\Delta t_1| + |\lambda m_{\text{л}}| + |c_{\text{л}}m_{\text{л}}\Delta t_2|}{L} = \\ &= \frac{4200 \cdot 0,5 \cdot 20 + 330000 \cdot 0,5 + 2100 \cdot 0,5 \cdot 5}{2000000} = 0,106 \text{ кг}. \end{aligned}$$

Ответ: 0,106 кг.

Критерии оценивания

Задачи должны быть оформлены согласно ФГОС, а их решение демонстрировать знание законов и способы их применения в конкретной ситуации.

- Правильно записано уравнение теплового баланса — 5 баллов.
- Правильно записаны выражения для отданного количества теплоты — 5 баллов.
- Правильно записаны выражения для полученного количества теплоты — 10 баллов.
- Верно произведены расчеты — 5 баллов.

Если решения нет, но приведены разумные рассуждения в направлении решения — 10 баллов.

Задача III.1.2.5. Теплоемкость сплава (15 баллов)

При изготовлении разъемов ответственных узлов самолетов используется латунь. Латунь обладает большой устойчивостью к коррозии и является сплавом меди, цинка и олова. Оцените удельную теплоемкость сплава, в котором 60% меди и 36% цинка, 4% олова. Удельная теплоемкость меди $380 \frac{\text{Дж}}{\text{кг} \cdot ^\circ\text{C}}$, цинка $400 \frac{\text{Дж}}{\text{кг} \cdot ^\circ\text{C}}$ и олова $230 \frac{\text{Дж}}{\text{кг} \cdot ^\circ\text{C}}$.

Решение

$$Q = cm\Delta t.$$

$$Q_{\text{м}} = c_{\text{м}}m_{\text{м}}\Delta t = 0,6c_{\text{м}}m\Delta t.$$

$$Q_{\text{ц}} = c_{\text{ц}}m_{\text{ц}}\Delta t = 0,36c_{\text{ц}}m\Delta t.$$

$$cm\Delta t = 0,6c_{\text{м}}m\Delta t + 0,36c_{\text{ц}}m\Delta t + 0,04c_{\text{о}}m\Delta t.$$

$$c = 0,6c_{\text{м}} + 0,36c_{\text{ц}} + 0,04c_{\text{о}}.$$

$$c = 0,6 \cdot 380 + 0,36 \cdot 400 + 0,04 \cdot 230 = 381,2 \frac{\text{Дж}}{\text{кг} \cdot ^\circ\text{C}}.$$

Ответ: $381,2 \frac{\text{Дж}}{\text{кг} \cdot ^\circ\text{C}}$.

Критерии оценивания

Задачи должны быть оформлены согласно ФГОС, а их решение демонстрировать знание законов и способы их применения в конкретной ситуации.

- Правильно записано выражение для расчета количества теплоты — 5 баллов.
- Правильно записано выражение для расчета результирующего количества теплоты — 5 баллов.
- Верно проведены вычисления — 5 баллов.

Если решения нет, но приведены разумные рассуждения в направлении решения — 5 баллов.

Физика. 10–11 классы

Задача III.1.3.1. Необычный пропеллер (10 баллов)

На самолетостроительном заводе придумали модель необычного пропеллера: он несимметричный. Пропеллер равномерно вращается вокруг закрепленной оси, проходящей через точку А перпендикулярно плоскости рисунка III.1.3. Длина двух лопастей 50 см, угловая скорость вращения 4 рад/с, линейная скорость малой лопасти 0,5 м/с. Чему равна линейная скорость конца большой лопасти? Ответ предоставьте в СИ.

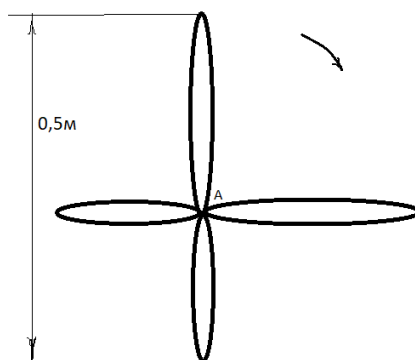


Рис. III.1.3

Решение

$$\omega_1 = \omega_2.$$

$$v_1 = \omega R_1.$$

$$R_1 = 0,125 \text{ м.}$$

$$R_2 = 0,5 - 0,125 = 0,375 \text{ м.}$$

$$v_2 = \omega R_2.$$

$$v_2 = 1,5 \text{ м/с.}$$

Ответ: $v_2 = 1,5 \text{ м/с}$.

Критерии оценивания

Задачи должны быть оформлены согласно ФГОС, а их решение демонстрировать знание законов и способы их применения в конкретной ситуации.

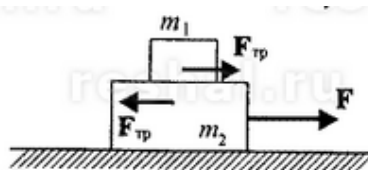
- Правильно записано выражение для расчета скорости — 5 баллов.
- Получено значение скорости — 5 баллов.

Если решения нет, но приведены разумные рассуждения в направлении решения — 5 баллов.

Задача III.1.3.2. Зимние забавы (25 баллов)

Во время игры ребята тянут санки по льду с силой, пропорциональной времени действия $F = 2t$. На горизонтальной поверхности санок находится мальчик, коэффициент трения мальчика о сиденье санок 0,15. Сколько времени мальчик сможет усидеть на санках, чтобы не слететь с них (мальчик не держится за санки)? С каким ускорением будут двигаться мальчик и санки в этот критический момент времени? Масса санок 3 кг, масса мальчика 30 кг, ускорение свободного падения принять равным 10 м/с^2 .

Решение



$$\sum \vec{F} = m_1 \vec{a}_1.$$

$$F_{\text{тр}} = m_1 a_1.$$

$$\sum \vec{F} = m_2 \vec{a}_2.$$

$$F - F_{\text{тр}} = m_2 a_2.$$

$$F_{\text{тр}} = \mu m_1 g.$$

$$a_2 = \frac{2t - \mu m_1 g}{m_2}.$$

$$a_1 = \mu g = \text{const}, \quad a_1 = 0,15 \cdot 10 = 1,5 \text{ м/с}^2.$$

$$a_2 \geq a_1 \Rightarrow \frac{2t - \mu m_1 g}{m_2} \geq \mu g.$$

Время, через которое мальчик слетит с санок, найдем из равенства:

$$\frac{2t - \mu m_1 g}{m_2} = \mu g.$$

$$t = \frac{(m_1 + m_2)\mu g}{2} = \frac{(30 + 3) \cdot 0,15 \cdot 10}{2} = 24,75 \text{ с.}$$

$$a_2 = \frac{2t - \mu m_1 g}{m_2} = \frac{2 \cdot 24,75 - 0,15 \cdot 30 \cdot 10}{3} = 1,5 \text{ м/с}^2.$$

Ответ: $a_1 = 1,5 \text{ м/с}^2$; $t = 24,75 \text{ с}$; $a_2 = 1,5 \text{ м/с}^2$.

Критерии оценивания

Задачи должны быть оформлены согласно ФГОС, а их решение демонстрировать знание законов и способы их применения в конкретной ситуации.

- Правильно записано выражение второго закона Ньютона и его применение в данной задаче — 5 баллов.
- Правильно записано выражение для расчета ускорения первого тела — 5 баллов.
- Правильно записано выражение для расчета ускорения второго тела — 5 баллов.
- Правильно записано выражение для расчета времени — 5 баллов.
- Правильно проведены вычисления — 10 баллов.

Если решения нет, но приведены разумные рассуждения в направлении решения — 10 баллов.

Задача III.1.3.3. Снеговик (25 баллов)

Сколько дней будет в марте таять один ком снеговика массой 20 кг, если средняя интенсивность излучения составляет 40% от значения солнечной постоянной 1367 Вт/м²? График среднего значения суточных температур представлен на рисунке III.1.4. Количество часов дневного времени суток на рисунке III.1.5. Удельная теплота плавления льда 330 кДж/кг. Плотность комка снега 400 кг/м³. Объем шара $V = \frac{4}{3}\pi R^3$; площадь сферы $S = 4\pi R^2$.

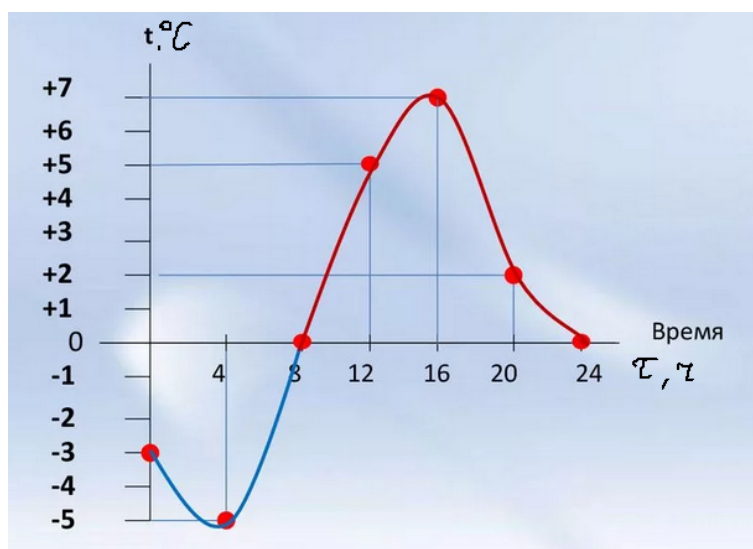


Рис. III.1.4

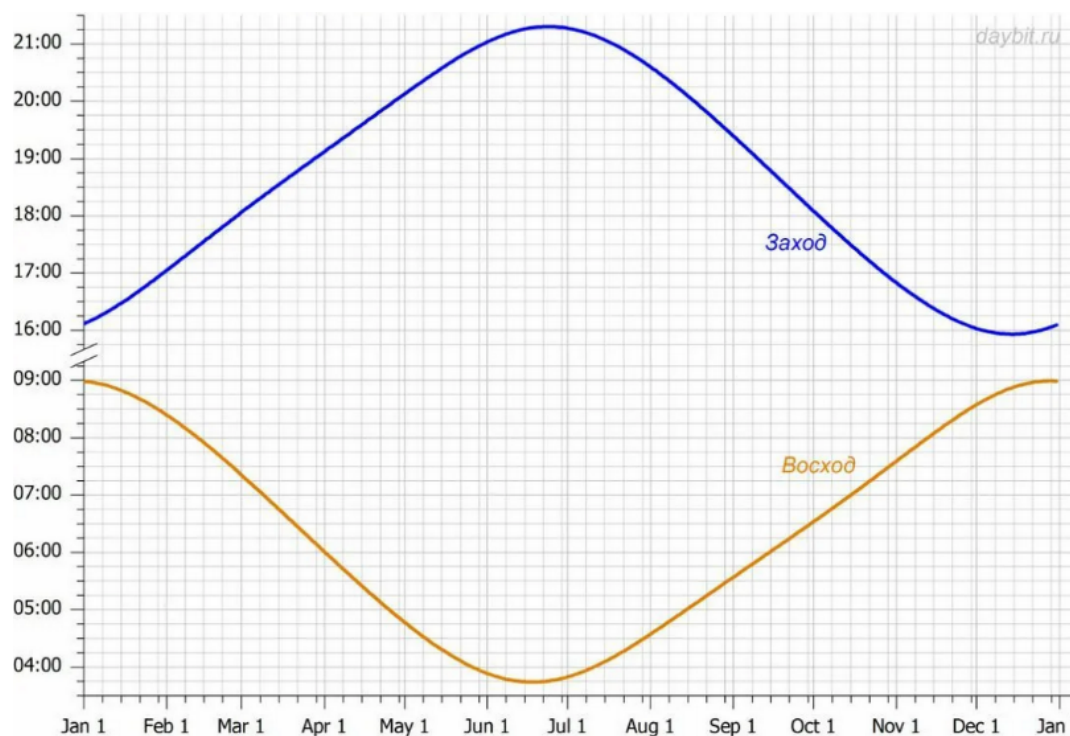


Рис. III.1.5

Решение

$$m = 20 \text{ кг}, \quad \lambda = 330 \frac{\text{кДж}}{\text{кг}}.$$

$$Q = \lambda m.$$

$$Q = 330 \cdot 10^3 \cdot 20 = 6600000 \text{ Дж}.$$

$$V = \frac{m}{\rho}.$$

$$V = \frac{20}{400} = 0,05 \text{ м}^3.$$

$$R = \sqrt[3]{\frac{3V}{4\pi}}.$$

$$R = \sqrt[3]{\frac{3 \cdot 0,05}{4 \cdot 3,14}} \approx 0,2289 \text{ м}.$$

$$S = 4\pi R^2.$$

$$S = 4 \cdot 3,14 \cdot 0,2289^2 \approx 0,658 \text{ м}^2.$$

$$P_c = IS.$$

$$P_c = 0,4 \cdot 1367 \cdot 0,658 = 359,84 \text{ Вт}.$$

$$\tau = \frac{Q}{P_c}.$$

$$\tau = \frac{6600000}{359,84} = 18341,4 \text{ с} \approx 5,09 \text{ ч} \approx 1 \text{ день}.$$

Ответ: 1 день.

Критерии оценивания

Задачи должны быть оформлены согласно ФГОС, а их решение демонстрировать знание законов и способы их применения в конкретной ситуации.

- Правильно записано выражение для расчета количества теплоты при плавлении — 5 баллов.
- Правильно записано выражение для расчета мощности излучения — 5 баллов.
- Правильно записано выражение для расчета поверхности излучения — 5 баллов.
- Правильно записано выражение для расчета плотности — 5 баллов.
- Правильно проведены вычисления и сделан вывод на основании графиков — 5 баллов.

Если решения нет, но приведены разумные рассуждения в направлении решения — 10 баллов.

Задача III.1.3.4. Сноубордист (25 баллов)



Сноубордист проезжает под гору с уклоном 4° длиной 100 м, а затем движется в гору с уклоном 8° . Коэффициент трения 0,03. Сможет ли спортсмен подняться на высоту 3 м? Ответ обоснуйте.

Решение

По закону сохранения энергии:

$$E_{n1} - A = E_{n2}, \quad A = F_{\text{тр}1}s_1 + F_{\text{тр}2}s_2.$$

$$mgh_1 - F_{\text{тр}1}s_1 - F_{\text{тр}2}s_2 = mgh_2.$$

$$h_1 = s_1 \cdot \sin \alpha.$$

$$s_2 = \frac{h_2}{\sin \beta}.$$

$$F_{\text{тр}1} = \mu mg \cos \alpha.$$

$$F_{\text{тр2}} = \mu mg \cos \beta.$$

$$mgs_1 \cdot \sin \alpha - \mu mgs_1 \cdot \cos \alpha - \mu mg \frac{h_2}{\sin \beta} \cdot \cos \beta = mgh_2.$$

$$h_2 = \frac{s_1 \cdot \sin \alpha - \mu s_1 \cdot \cos \alpha}{1 + \mu \operatorname{ctg} \beta}.$$

$$h_2 = \frac{100 \cdot \sin 4^\circ - 0,03 \cdot 100 \cdot \cos 4^\circ}{1 + 0,03 \operatorname{ctg} 8^\circ} \approx 3,28 \text{ м.}$$

Ответ: сможет, так как максимальная возможная высота подъема 3,28 м.

Критерии оценивания

Задачи должны быть оформлены согласно ФГОС, а их решение демонстрировать знание законов и способы их применения в конкретной ситуации.

- Правильно записан закон сохранения энергии — 5 баллов.
- Правильно записано выражение для расчета потенциальной энергии — 5 баллов.
- Правильно записано выражение для расчета работы — 5 баллов.
- Правильно записано выражение для расчета высоты подъема — 5 баллов.
- Правильно проведены вычисления — 5 баллов.

Если решения нет, но приведены разумные рассуждения в направлении решения — 10 баллов.

Задача III.1.3.5. Неудачный эксперимент (15 баллов)

При испытании двигателя 0,5 моль атомарного газа находился при 0 °С. В ходе эксперимента была получена диаграмма, представленная на рисунке III.1.6. Оцените с точностью до десятых работу газа за цикл и КПД этого двигателя. Универсальная газовая постоянная равна $8,31 \frac{\text{Дж}}{\text{моль} \cdot \text{К}}$.

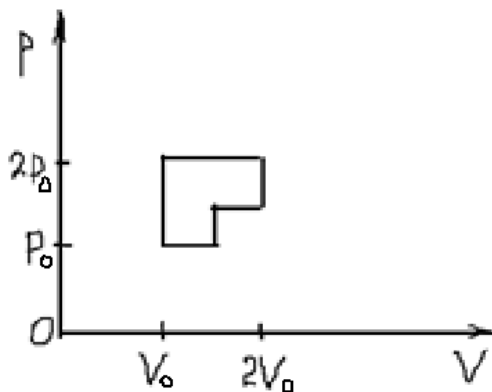


Рис. III.1.6

Решение

Работа газа в цикле:

$$A = \frac{3}{4}p_0V_0.$$

$$p_0V_0 = \nu RT.$$

$$A = \frac{3}{4} \cdot 0,5 \cdot 8,31 \cdot 273 \approx 850,7 \text{ Дж.}$$

$$\eta = \frac{A}{Q_{\text{н}}} \cdot 100\%.$$

$$Q_1 = \Delta U \frac{3}{2} \nu R \Delta T_1.$$

$$\frac{p_0}{T_0} = \frac{p_1}{T_1}, T_1 = 2T_0 = 546 \text{ К.}$$

$$\frac{V_1}{T_1} = \frac{V_2}{T_2}, T_2 = 2T_1 = 1092 \text{ К.}$$

$$Q_2 = \frac{5}{2} \nu R \Delta T_2.$$

$$Q_{\text{н}} = Q_1 + Q_2.$$

$$\eta = \frac{\frac{3}{4} \nu RT_0}{\frac{3}{2} \nu R \Delta T_1 + \frac{5}{2} \nu R \Delta T_2} = \frac{\frac{3}{4} T_0}{\frac{3}{2} \Delta T_1 + \frac{5}{2} \Delta T_2}.$$

$$\eta = \frac{\frac{3}{4} T_0}{\frac{3}{2} \Delta T_1 + \frac{5}{2} \Delta T_2} = \frac{\frac{3}{4} 273}{\frac{3}{2} 273 + \frac{5}{2} 546} \approx 0,115 \Rightarrow 11,5\%.$$

Ответ: $A = 850,7 \text{ Дж}$, $\eta = 11,5\%$ — малое значение КПД.

Критерии оценивания

Задачи должны быть оформлены согласно ФГОС, а их решение демонстрировать знание законов и способы их применения в конкретной ситуации.

- Правильно получено выражение для расчета работы газа — 5 баллов.
- Правильно получено выражение для расчета КПД — 5 баллов.
- Правильно сделаны вычисления — 5 баллов.

Если решения нет, но приведены разумные рассуждения в направлении решения — 5 баллов.

Командный практический тур

Задача участников — построить отказоустойчивую систему автоматического управления БПЛА самолетного типа при решении задачи поиска объектов. Для решения поставленной задачи используются: микроконтроллер и симулятор полета БПЛА самолетного типа (UAVIANT).

Требования к команде

Количество участников в команде: 3–4 человека.

Рекомендуемые роли участников в команде:

- **Капитан**, лидер команды. Осуществляет общее руководство работой команды, распределяет задачи и контролирует выполнение проекта.
- **Математик/физик**. Формирует алгоритмы управления полетом БЛА, составляет алгоритмы программ для обработки сигналов с датчиков, разрабатывает алгоритмы для обработки изображений.
- **Программист** (Python, C/C++ (среда Arduino IDE)). Реализует разработанные алгоритмы в виде программного обеспечения для САУ БЛА, разрабатывает ПО для реализации задач по техническому зрению.

Оборудование и программное обеспечение

Требования к рабочему месту участника:

1. ПК.
2. Web-камера.
3. Наушники и микрофон (для общения внутри команды).
4. Стабильное интернет-соединение (скорость не ниже 10 Мбит/сек).

Для успешного участия необходимо установить следующее ПО:

1. Программа Chrome Remote Desktop (<https://remotedesktop.google.com/>) для удаленной работы (проверка решения на симуляторе) на ПК организатора.
2. Программа TeamViewer (<https://disk.yandex.ru/d/7WH813uyzy5Faw>) для резервного варианта работы на ПК организатора.
3. Программа AnyDesk (<https://disk.yandex.ru/d/HkNuHuwVEJ2H0g>) для резервного варианта работы на ПК организатора.
4. При невозможности работы через программу Chrome Remote Desktop и перечисленные резервные программы на ПК организатора работа будет осуществляться на ПК участника, а видеотрансляция с ПК организатора (проверка решения задачи на симуляторе) будет осуществляться через сервис Webinar.
5. Среда программирования — Arduino IDE (<https://disk.yandex.ru/d/0XTEyZBkaD-4mA>).
6. Среда программирования на языке Python (<https://www.python.org/>).

7. Программа DISCORD (резервный вариант — конференции будут проходить через платформу Webinar).
8. Программа Telegram Desktop (резервный вариант).

Задачи

Задача III.2.3.1. (19 баллов)

Стабилизация бортовой фотоаппаратуры БЛА в процессе полета является необходимым условием работы системы технического зрения. Для решения этой задачи обычно используются управляемые стабилизирующие подвесы, способные не только стабилизировать углы поворота камеры относительно заданного положения по трем осям, но и компенсировать вибрации и прочие возмущения, возникающие в процессе полета БЛА.

Условие

Разработайте программу стабилизации прототипа подвеса камеры БЛА в плоскости горизонта. Прототип подвеса имеет одну ось вращения (по углу крена). Вращение площадки с камерой осуществляется посредством формирования соответствующих команд на сервопривод, установленный на этой оси. На площадке установлен инерциальный датчик, позволяющий измерять текущий угол поворота подвеса камеры. Управление подвесом реализуется на контроллере Arduino.

Формат входных данных

Шаблон программы на Arduino; измерения датчика углового положения подвеса.

Формат выходных данных

Программа для Arduino реализующая стабилизацию прототипа подвеса камеры БЛА в плоскости горизонта.

Решение

Поскольку измерения инерциального датчика имеют шумовую составляющую, целесообразно перед их использованием в законе управления провести фильтрацию полученных измерений. Один из самых простых и распространенных алгоритмов фильтрации — упрощенный фильтр скользящего среднего (Simple Moving Average, SMA).

Преимуществом данного фильтра является то, что его очень просто настроить, так как он содержит всего один параметр — размер окна n . Значения n всегда нечетные. Выходным значением фильтра будет среднее арифметическое всех значений, лежащих в пределах половины окна от заданного значения:

$$y_i = \frac{1}{n} \sum_{j=i-k}^{i+k} x_j, \quad k = \frac{n-1}{2}.$$

При разработке эталонного решения задачи использовался размер окна $n = 3$. Слишком большой размер окна приведет к запаздыванию управления. Управление подвесом камеры можно реализовать с помощью закона управления, в котором присутствует ПИД-регулятор. Для реализации регулятора могут понадобиться не все три составляющие (пропорциональная (П), интегральная (И) и дифференциальная (Д)), а только часть из них. Решение о виде регулятора и значениях его коэффициентов принимается по результатам тестирования в симуляторе так, чтобы параметры переходного процесса удовлетворяли поставленным требованиям.

Код программы для стабилизации прототипа подвеса камеры БЛА в плоскости горизонта.

```

/##### Участникам олимпиады - задача 1 #####/ float last_angle =
↪ 0.0;
float last1 = 0.0;
float last2 = 0.0;
float UARTControlSystemClass::gimbal_control(float uav_roll, float camera_roll)
{
    float gimbal_control = 0.0;
    // TODO: Реализуйте фильтрацию измерений текущего положения камеры camera_roll
    // И разработайте автоматический регулятор для выставления камеры в плоскость
    ↪ горизонта
    float k = 26.5;
    float value = (last2 + last1 + camera_roll) / 3;
    gimbal_control = (k * (uav_roll - value) - 7.0 * (value - last_angle)); last_angle
    ↪ = value;
    last2 = last1;
    last1 = camera_roll;
    return gimbal_control;
}
/#####*/
// Функция реализующая отправку на симулятор команд для управления подвесом камеры
void UARTControlSystemClass::SendSuspPWMPacket()
{
    float gam_planer = getPcktTele().getBody().Gam; // Угол крена БЛА float gam_camera
    ↪ = getPcktIce().getBody().Gamma; // Измерение текущего угла крена камеры
    ↪ (включает случайный шум измерений)
    float gim_ctrl = gimbal_control(gam_planer, gam_camera);

    // Код для отправки команды на симулятор. Пожалуйста не изменяйте его.
    getPcktSuspControl().getBodyMutable().PWM_Gamma =
    Saturation(gim_ctrl, -255, 255);
    getPcktSuspControl().updateChecksum();
    _formatter->SendBuf(getPcktSuspControl().getData(),
    getPcktSuspControl().dataSizeBytes);
}

```

Критерии оценивания

Оценивается качество переходного процесса стабилизации подвеса.

- Максимальная оценка за решение задачи — 15 баллов.
- Требуемое время переходного процесса не более 1,5 с. За каждые 0,1 с превышения этого времени снимается 1 балл.
- Требуемая точность стабилизации (установившееся значение переходного процесса на 5-й секунде) составляет 0,3 градуса. За каждую 0,1 градуса ошибки сверх эталонного значения снимается 1 балл.

- Правильный ответ на вопрос по теории прибавляет 2 балла (всего 2 вопроса).

Задача III.2.3.2. (19 баллов)

Одной из основных задач БЛА, оснащенных системой технического зрения, является задача обнаружения объектов. Решение этой задачи требуется буквально во всех областях применения БЛА: контроль зон чрезвычайных ситуаций или ситуации на дорогах, сельское хозяйство, мониторинг объектов инфраструктуры, поиск людей и объектов в слабозаселенных районах.

Условие

Разработайте программу на языке Python, осуществляющую определение координат местоположения (в метрах) объекта (его центра) на полученном снимке с бортовой камеры БЛА. В рамках этой задачи параметры бортовой камеры БЛА неизвестны, но известны его навигационные параметры и габаритные размеры искомого объекта.

Формат входных данных

Изображение с бортовой камеры, габаритные размеры искомого объекта, навигационные параметры БЛА (координаты и ориентация) на момент осуществления съемки.

Формат выходных данных

Программа реализующая расчет координат местоположения (в метрах) центра искомого объекта с использованием алгоритмов технического зрения и библиотеки OpenCV.

Решение

Пользуясь инструментами библиотеки OpenCV, произвести обработку изображения с целью выделения на исходном изображении искомого объекта и описать объект в виде матрицы значений. Затем с помощью встроенной функции найти собственные числа полученной матрицы и составить по ним вектора. С помощью векторов определить ориентацию объекта. Определить горизонтальную и вертикальную проекции объекта в метрах и пикселях. Найти размер одного пикселя изображения в метрах.

Найти местоположение искомого объекта в связанной с БЛА системе координат.

Для получения окончательного решения нужно перевести полученное решение в географическую систему координат. Это можно сделать с помощью матрицы направляющих косинусов, которая имеет следующий вид.

$$M_b^g = \begin{bmatrix} \cos\theta\cos\psi & -\sin\psi\sin\gamma - \cos\gamma\cos\psi\sin\theta & \cos\psi\sin\theta\sin\gamma - \sin\psi\cos\gamma \\ \sin\theta & \cos\gamma\cos\theta & -\cos\theta\sin\gamma \\ \cos\theta\sin\psi & \cos\psi\sin\gamma - \cos\gamma\sin\theta\sin\psi & \cos\gamma\cos\psi + \sin\theta\sin\gamma\sin\psi \end{bmatrix}.$$

Сам перевод координат можно записать следующим образом:

$$\begin{bmatrix} X_g \\ Y_g \\ Z_g \end{bmatrix} = M_b^g \times \begin{bmatrix} X_b \\ Y_b \\ Z_b \end{bmatrix}.$$

Матрица для реализации обратного разворота при необходимости может быть получена путем транспонирования:

$$M_b^g = (M_b^g)^T.$$

Код программы для расчета координат местоположения центра объекта.

```
import cv2 as cv
from math import atan2, cos, sin, sqrt, pi, atan
import numpy as np

def drawAxis(img, p_, q_, color, scale):
    p = list(p_)
    q = list(q_)

    ## [visualization1]
    angle = atan2(p[1] - q[1], p[0] - q[0]) # angle in radians
    hypotenuse = sqrt((p[1] - q[1]) * (p[1] - q[1]) + (p[0] - q[0]) * (p[0] - q[0]))

    # Here we lengthen the arrow by a factor of scale
    q[0] = p[0] - scale * hypotenuse * cos(angle)
    q[1] = p[1] - scale * hypotenuse * sin(angle)
    cv.line(img, (int(p[0]), int(p[1])), (int(q[0]), int(q[1])), color, 3, cv.LINE_AA)

    # create the arrow hooks
    p[0] = q[0] + 9 * cos(angle + pi / 4)
    p[1] = q[1] + 9 * sin(angle + pi / 4)
    cv.line(img, (int(p[0]), int(p[1])), (int(q[0]), int(q[1])), color, 3, cv.LINE_AA)

    p[0] = q[0] + 9 * cos(angle - pi / 4)
    p[1] = q[1] + 9 * sin(angle - pi / 4)
    cv.line(img, (int(p[0]), int(p[1])), (int(q[0]), int(q[1])), color, 3, cv.LINE_AA)
    ## [visualization1]

def getOrientation(pts, img):
    ## [pca]
    # Construct a buffer used by the pca analysis
    sz = len(pts)
    data_pts = np.empty((sz, 2), dtype=np.float64)
    for i in range(data_pts.shape[0]):
        data_pts[i,0] = pts[i,0,0]
        data_pts[i,1] = pts[i,0,1]

    # Perform PCA analysis
    mean = np.empty((0))
    mean, eigenvectors, eigenvalues = cv.PCACompute2(data_pts, mean)

    # Store the center of the object
    cntr = (int(mean[0,0]), int(mean[0,1]))
    ## [pca]

    ## [visualization]
```

```

# Draw the principal components
cv.circle(img, cntr, 3, (255, 0, 255), 2)
p1 = (cntr[0] + 0.02 * eigenvectors[0,0] * eigenvalues[0,0], cntr[1] + 0.02 *
    ↪ eigenvectors[0,1] * eigenvalues[0,0])
p2 = (cntr[0] - 0.02 * eigenvectors[1,0] * eigenvalues[1,0], cntr[1] - 0.02 *
    ↪ eigenvectors[1,1] * eigenvalues[1,0])
drawAxis(img, cntr, p1, (255, 255, 0), 1)
drawAxis(img, cntr, p2, (0, 0, 255), 5)

angle = atan2(eigenvectors[0,1], eigenvectors[0,0]) # orientation in radians
## [visualization]

# Label with the rotation angle
label = " Rotation Angle: " + str(-int(np.rad2deg(angle)) - 90) + " degrees"
textbox = cv.rectangle(img, (cntr[0], cntr[1]-25), (cntr[0] + 250, cntr[1] + 10),
    ↪ (255,255,255), -1)
cv.putText(img, label, (cntr[0], cntr[1]), cv.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1,
    ↪ cv.LINE_AA)

return (angle, cntr)

# Load the image
#color definition
red_lower = np.array([0, 200, 160], dtype="uint8")
red_upper = np.array([20, 255, 255], dtype="uint8")
red_lower_v = np.array([235, 120, 150], dtype="uint8")
red_upper_v = np.array([255, 255, 255], dtype="uint8")

# Main cycle
img = cv.imread('source.png')

# Normalize image brightness
cv.normalize(img, img, 0, 255, cv.NORM_MINMAX)
# Blur the image to reduce noise
blur = cv.medianBlur(img, 5)
# Convert BGR to HSV
hsv = cv.cvtColor(img, cv.COLOR_BGR2HSV)
# Find color masks
red3_mask = cv.inRange(hsv, red_lower, red_upper)
red2_mask = cv.inRange(hsv, red_lower_v, red_upper_v)
#white_mask = cv2.inRange(hsv, white_lower, white_upper)
full_mask = red2_mask + red3_mask
#full_mask = white_mask
#cv2.imwrite("deb.png", full_mask)
# Find counters
contours, hierarchy = cv.findContours(full_mask, cv.RETR_LIST, cv.CHAIN_APPROX_SIMPLE)
# img = full_mask

#for i, c in enumerate(contours):
comb = contours[0].copy()
for i in range(1, len(contours)):
    comb = np.concatenate((comb, contours[i]))
#print(comb)
c = comb
i = 0

# Calculate the area of each contour
area = cv.contourArea(c)

# Ignore contours that are too small or too large

```

```

# if area < 220 or 100000 < area:
# continue

# Draw each contour only for visualisation purposes
cv.drawContours(img, contours, i, (0, 200, 0), 2)

# Find the orientation of each shape
alpha, cntr = getOrientation(c, img)
alpha = -alpha - pi/2.0
lx = 5.8
ly = 3.4
x0 = 527.40
z0 = 467.70
xuav = 520.80
zuav = 473.72
h = 41
psi = 29.99 * pi / 180

lxx = sqrt(lx**2 + 1) + sqrt(0.5**2 + 0.5**2)
#print(0.5*1.4)
#alpha = alpha + atan2(1, lx)
cx = []
cy = []
for el in c:
    cx.append(el[0][0])
    cy.append(el[0][1])
#print(cx)
dpx = max(cx) - min(cx)
dpy = max(cy) - min(cy) # vertical
print("Object rotation angle: {:.2f}".format(alpha * 180 / pi))
dx = abs(lxx * cos(alpha)) # vertical
dz = abs(lxx * sin(alpha))
print("pixel {}, meters {}".format(dpx, dz))
print("pixel {}, meters {}".format(dpy, dx))
height, width, channels = img.shape
pixel_height = dx / dpy
pixel_width = dz / dpx
print("Pixel height: {}, width: {}".format(pixel_height, pixel_width))
center_dx = cntr[0] - width / 2 # right
center_dy = height / 2 - cntr[1] # up
print("Center shift hor: {}, ver: {}".format(center_dx, center_dy))
vect = [center_dy * pixel_height, center_dx * pixel_width]
print("Displacement in body frame X: {:.2f}, Z: {:.2f}".format(vect[0], vect[1]))
v2 = [1, 0]
#psi = 45 * pi / 180
v2[0] = vect[0] * cos(psi) - vect[1] * sin(psi)
v2[1] = +vect[0] * sin(psi) + vect[1] * cos(psi)
print("Actual displacement X: {:.2f}, Z: {:.2f}".format(v2[0], v2[1]))
print("Solution X: {:.2f}, Z: {:.2f}".format(xuav+v2[0], zuav+v2[1]))
print("Solutuion error: {:.2f} (X: {:.2f}, Z: {:.2f})".format(sqrt((xuav+v2[0] -
↵ x0)**2 + (zuav+v2[1] - z0)**2), xuav+v2[0] - x0, zuav+v2[1] - z0))

cv.imshow('Output Image', img)
cv.imwrite("sol_2.png", img)
cv.waitKey(0)
cv.destroyAllWindows()

```

Критерии оценивания

Оценивается точность определения координат местоположения искомого объекта.

- Максимальный балл за решение задачи — 15 баллов.
- Требуемая точность решения задачи — 0,2 метра. Если величина ошибки превышает это значение, за каждые дополнительные 0,5 метра ошибки снимается 1 балл.
- Правильный ответ на вопрос по теории прибавляет 2 балла (всего 2 вопроса).

Задача III.2.3.3. (28 баллов)

Одной из базовых функций, реализуемых БЛА, является автономный полет по заданной траектории. Для осуществления такого полета разрабатываются специальные системы автоматического управления, способные регулировать все параметры навигации БЛА не только в зависимости от его положения относительно заданной траектории, но и с учетом других факторов, например, текущих погодных условий.

Условие

Разработайте программу, реализующую автоматическое управление углом крена и высотой полета БЛА, для осуществления его движения по заданной траектории (полет по заданным точкам). При этом заданная траектория проходит через зону повышенной влажности, в пределах которой возможно обледенение БЛА, влияющее на параметры его полета. БЛА оснащен сигнализатором обледенения, позволяющим измерить текущий уровень этого параметра.

Формат входных данных

Координаты путевых точек, система автоматического управления БЛА (кроме регуляторов в каналах крена и высоты), навигационные параметры БЛА в каждый момент времени (координаты и углы ориентации), измерения сигнализатора обледенения.

Формат выходных данных

Программа для Arduino, реализующая автономный полет БЛА по заданным путевым точкам.

Решение

Управление по курсу осуществляется через изменение угла крена БЛА. Другими словами, входом системы управления является рассогласование $\Psi - \Psi^{\text{зад}}$, где Ψ — текущий угол курса, $\Psi^{\text{зад}}$ — требуемый угол курса, а выходной величиной — заданный угол крена $\gamma^{\text{зад}}$. Подходящим контроллером является пропорциональный регулятор (ПД-регулятор):

$$u = K_{\Psi} \varepsilon,$$

где ε — сигнал ошибки, u — управляющий сигнал, K — коэффициент усиления. Оба угла и зад изменяются в пределах $[-180^\circ, +180^\circ]$. Для обеспечения кратчайшего разворота сигнал ошибки ε рассчитывается как:

$$\varepsilon = \text{atan2}(\sin(\Psi - \Psi^{\text{зад}}), \cos(\Psi - \Psi^{\text{зад}})).$$

Чтобы заданный угол крена не превышал по модулю 30 градусов, управляющий сигнал регулятора ограничивается следующим образом:

$$\gamma^{\text{зад}} = \{u, |u| \leq 30^\circ\}.$$

Коэффициент K подбирается при таком малом сигнале рассогласования, что $|\gamma^{\text{зад}}| < 30^\circ$. Увеличение коэффициента ускоряет переходный процесс, однако, при больших значениях K регулятор курса может работать быстрее, чем система стабилизации крена, что приведет к перерегулированию и колебательности. Удовлетворительное качество процесса регулирования достигается при $K = 5 \dots 8$.

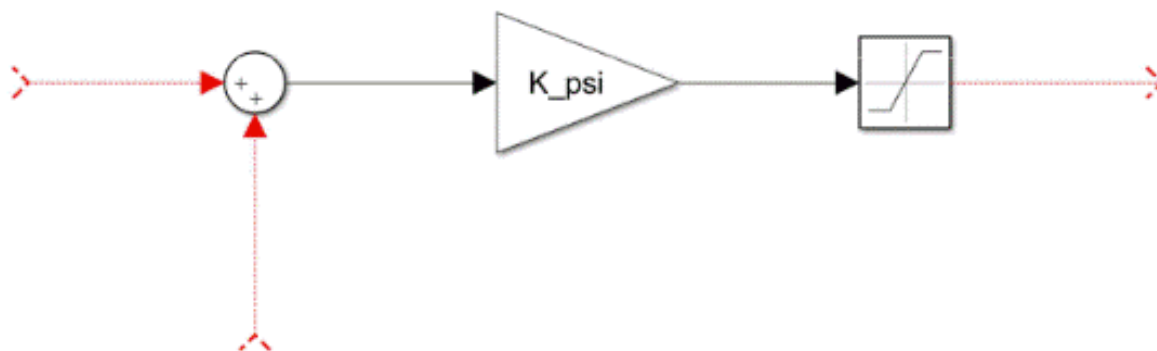


Рис. III.2.1. Структурная схема системы управления углом крена

Управление по высоте осуществляется через изменение угла тангажа БПЛА. Другими словами, входом системы управления является рассогласование $\alpha - \alpha^{\text{зад}}$, где α — текущий угол атаки, $\alpha^{\text{зад}}$ — требуемый угол атаки, а выходной величиной — заданный угол тангажа $\vartheta^{\text{зад}}$.

Подходящим контроллером является пропорциональный регулятор (ПД регулятор):

$$\Delta H = H^{\text{зад}} - H^{\text{тек}}.$$

$H^{\text{тек}}$ получаем из пакета телеметрии `_tele.H`.

V_y получаем из пакета телеметрии `_tele.dH`.

$$k_{h1} = K_h \cdot \Delta H,$$

$$\text{pitch_direct} = k_{h1} \cdot K_{vy1} - v_y \cdot K_{vy2}.$$

Чтобы заданный угол тангажа не превышал по модулю 30 градусов, управляющий сигнал регулятора ограничивается следующим образом:

$$\vartheta^{\text{зад}} = \{\text{pitch_direct}, |\text{pitch_direct}| \leq 30^\circ\},$$

$$V_{y\text{max}} = \{k_{h1}, |k_{h1}| \leq 7,5\text{м/с}\}.$$

Коэффициент K_h подбирается при таком малом сигнале рассогласования, что $|\vartheta^{\text{зад}}| < 30^\circ$. Увеличение коэффициента ускоряет переходный процесс, однако, при больших значениях K_h регулятор высоты может работать быстрее, чем система стабилизации, что приведет к перерегулированию и колебательности. Для удержания высоты и ограничения вертикальной скорости используется дифференциальное звено регулятора — коэффициенты K_{vy1} и K_{vy2} .

Удовлетворительное качество процесса регулирования достигается при $K_h = 0,5 \dots 0,6$, $K_{vy1} = 7 \dots 8$, $K_{vy2} = 9 \dots 10$.

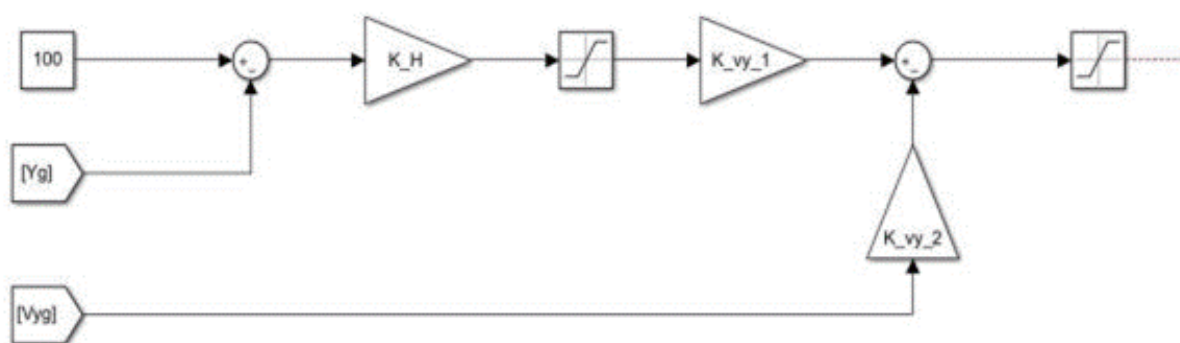


Рис. III.2.2. Структурная схема управления углом тангажа

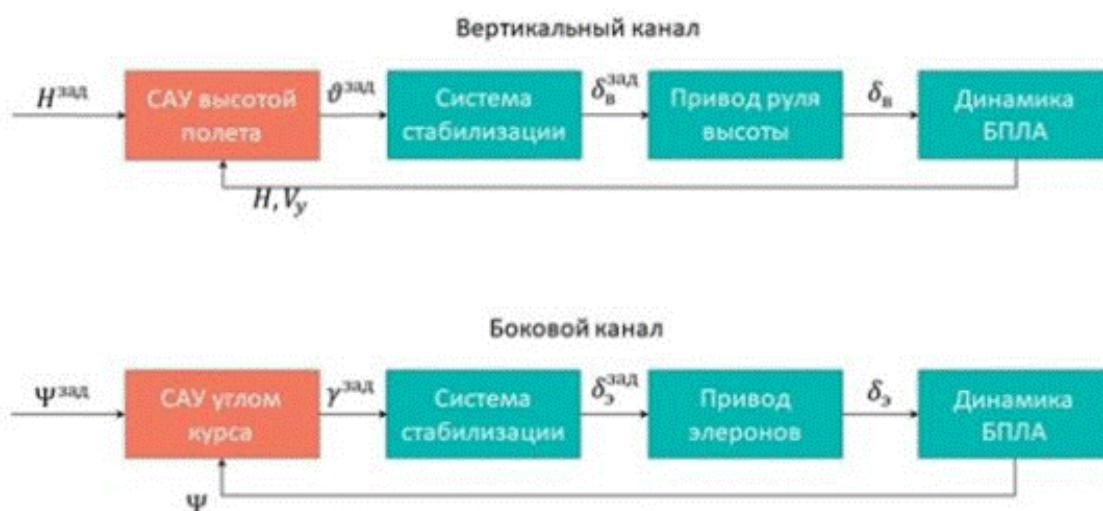


Рис. III.2.3. Обобщенная возможная система автоматизации управления

Траектория полета определяется порядком облета точек, который следует выбирать так, чтобы время полета БЛА не превышало заданное значение. Окончательный порядок облета точек определяется по результатам пролетов БЛА в симуляторе.

Поскольку в процессе полета БЛА может попасть в зону повышенной влажности, что повлечет за собой обледенение и, следовательно, ограничение управляемости БЛА, нужно внести поправки в законы управления. Для этого следует сформировать зависимость коэффициентов управления от измерений датчика обледенения. Окончательные значения соответствующих коэффициентов определяются по результатам пролетов БЛА в симуляторе.

Код программы для реализации полета по путевым точкам.

```

#include "UARTControlSystem.h"
#include <avr/eeprom.h>
#define sign(x) ({ decltype(x) _x = (x); _x < 0 ? -1 : _x ? 1 : 0; })
/***** Участникам олимпиады - задача 3 *****/ // Регулятор в
↳ канале управления высотой полета БЛА
// Функция должна возвращать требуемое значение угла тангажа (в град.) БЛА для
↳ достижения желаемой высоты
float UARTControlSystemClass::HeightReg(const Skywalker2015PacketTelemetry& _tele,
↳ const float &Hz) // _tele - структура принятого пакета телеметрии, Hz - заданная
↳ высота
{
    float Yg = _tele.H; // Получаем из принятого пакета телеметрии текущую высоту БЛА,
↳ м.
    float Vy = _tele.Vy1; // Получаем из принятого пакета телеметрии текущую
↳ вертикальную скорость БЛА, м/с.
    // Переменная _IceVortexFloat содержит показание датчика обморожения (изменяется
↳ от 0 до 1, где 0 - отсутствие обморожения)
    // TODO: Разработайте автоматический регулятор канала высоты! // Для достижения
↳ заданной высоты можно использовать PI регулятор // Для автоматического
↳ управления углом тангажа можно использовать PD регулятор
    float _Pitch_direct = 0.0;
    float K_H = 0.56;
    float K_vy_1 = 7.7591;
    float K_vy_2 = 9.8941;
    float kh1;
    K_H = K_H + _IceVortexFloat;
    K_vy_2 = K_vy_2 + _IceVortexFloat*15.7;
    K_vy_1 = K_vy_1 + _IceVortexFloat*7.9;
    kh1 = Saturation(K_H * (Hz - Yg), -7.5, 7.5); // Интегральное звено + Ограничитель
↳
    _Pitch_direct = kh1 * K_vy_1 - Vy * K_vy_2; // Дифференциальное звено +
↳ Ограничитель
    _Pitch_direct = Saturation(_Pitch_direct, -20, 20); // Ограничение максимального
↳ угла тангажа в 20 град.
    return _Pitch_direct; // Возвращаем заданный тангаж (град)
}
// Функция реализующая расчет требуемого угла курса для достижения путевой точки
float UARTControlSystemClass::ToPointXY(const Skywalker2015PacketTelemetry& _tele,
↳ const float& a_Xg, const float& a_Yg)
{
    // Разбор пакета телеметрии БЛА. Здесь вы сможете найти все необходимые для
↳ управления текущие параметры БЛА
    float _Xt = _tele.L;
    float _Yt = _tele.Z;
    float H = _tele.H; // Высота, м
    float Vx1 = _tele.Vx1; // Скорость Vx1, м/с
    float Vy1 = _tele.Vy1; // Скорость Vy1, м/с
    float Vz1 = _tele.Vz1; // Скорость Vz1, м/с
    float wx = _tele.wx; // Угловая скорость в связанной СК, 1/с float wy = _tele.wy;
↳ // Угловая скорость в связанной СК, 1/с float wz = _tele.wz; // Угловая
↳ скорость в связанной СК, 1/с float Theta = _tele.Tan; // Угол наклона
↳ траектории град.
    float Psi = _tele.Psi; // Угол курса в связанной СК, град. float Gamma =
↳ _tele.Gam; // Угол крена в связанной СК, град. float V = _tele.V; // Скорость
↳ БЛА, м/с
    // Переменная _IceVortexFloat содержит показание датчика обморожения (изменяется
↳ от 0 до 1, где 0 - отсутствие обморожения)
    // TODO: Рассчитайте требуемый угол курса!
    // Для определения угла курса регулятор не требуется, только геометрия float
↳ Psi_cmd = 0.0;

```



```

else
{
    x_og = _PointsArray[max_index - 1].L;
    z_og = _PointsArray[max_index - 1].Z;
}
float relx = _tele.L - x_og;
float rely = _tele.Z - z_og;
float delx = Xg_cmd - x_og;
float dely = Zg_cmd - z_og;
float U = (relx * delx + rely * dely) / (delx * delx + dely * dely); float cte =
↪ (rely * delx - relx * dely) / (delx * delx + dely * dely);
if (Size < 7.5)
{
    _Point_Index++;
}
// Выход за границы массива
if (_Point_Index >= max_index)
{
    _Point_Index = 0;
}
return _Point_Index;
}

inline float UARTControlSystemClass::AngDefines(const float& angl) {
    float z1 = angl * M_PI / 180;
    float z2 = atan2(sin(z1), cos(z1)) * 180 / M_PI;
    return z2;
}
// Функция ограничивающая минимальное и максимальное значения величины
inline float UARTControlSystemClass::Saturation(const float& c, const float& a, const
↪ float& b)
{
    if (c < a)
        return a;
    else if (c > b)
        return b;
    else
        return c;
}

```

Критерии оценивания

Оценивается количество пройденных БЛА путевых точек.

- Максимальный балл за решение задачи — 24 баллов.
- За каждую пройденную БЛА путевую точку начисляется 2 балла.
- Время полета БЛА не должно превышать 6 минут.
- Правильный ответ на вопрос по теории прибавляет 2 балла (всего 2 вопроса).

Задача III.2.3.4. (34 баллов)

Одним из основных факторов, определяющих автономность БЛА, является обеспечение надежности его функционирования. В условиях возможного отказа измерительных или управляющих систем БЛА должен быть способен продолжить выполнение миссии или принять взвешенное решение об ее отмене и вернуться на базу.

Условие

Сформируйте траекторию автоматического полета БЛА. В пределах карты симулятора находится искомый объект. Напишите программу, реализующую анализ изображений с бортовой камеры БЛА в процессе его полета по заданной траектории, с целью обнаружения искомого объекта и определения его координат. При этом в процессе полета БЛА могут происходить отказы измерительных систем: СНС и ИНС.

Формат входных данных

САУ БЛА, непрерывный поток изображений с бортовой камеры, параметры навигации БЛА в текущий момент времени, статус наличия отказов измерительных систем, габаритные размеры искомого объекта.

Формат выходных данных

Программа, реализующая расчет координат местоположения (в метрах) центра искомого объекта, с использованием алгоритмов технического зрения и библиотеки OpenCV; траектория полета БЛА.

Решение

Траектория полета задается с помощью путевых точек. Сначала это делается примерно, для того чтобы обнаружить приблизительное местоположение искомого объекта. Когда искомый объект обнаружен, траектория полета может корректироваться так, чтобы БЛА летел уже целенаправленно в сторону искомого объекта. Это нужно для уменьшения времени полета, если есть такая необходимость.

При попадании в зоны, где возникают отказы инерциальной навигационной системы и/или спутниковой навигационной системы, нужно компенсировать отсутствие соответствующих параметров. Один из способов решения этой задачи — интегрирование линейных и угловых скоростей, информация о которых не блокируется при попадании в соответствующие зоны, где происходят отказы указанных выше систем БЛА.

Для определения координат искомого объекта используется алгоритм из задачи [III.2.3.2](#).

Код программы.

```
# -*- coding: utf-8 -*-
import socket
from threading import Thread
import threading
from threading import Lock
import time
from math import atan2, cos, sin, sqrt, pi, atan
import cv2
import numpy as np
import uav_state
#from collections import deque
```

```

#queue = deque([], maxlen = 20)
# Мультипоточный класс для получения последнего кадра с бортовой камеры
class Camera:
    last_frame = None
    last_ready = None
    lock = Lock()

    def __init__(self, rtsp_link):
        capture = cv2.VideoCapture(rtsp_link)
        thread = threading.Thread(target=self.rtsp_cam_buffer, args=(capture,),
        ↪ name="rtsp_read_thread")
        thread.daemon = True
        thread.start()

    def rtsp_cam_buffer(self, capture):
        while True:
            with self.lock:
                self.last_ready, self.last_frame = capture.read()

    def getFrame(self):
        if (self.last_ready is not None) and (self.last_frame is not None):
            return self.last_frame.copy()
        else:
            return None

# Последние полученные данные телеметрии
last_telemetry = uav_state.uav_state()
running = True

# UDP сервер для параллельного получения данных телеметрии
def udp_server_thread():
    host = 'localhost'
    port = 7778
    addr = (host, port)
    server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server.bind(addr)

    while running:
        d = server.recvfrom(1024)
        received = d[0]
        addr = d[1]
        received = received.decode("utf-8")
        #print(received)
        received = received.replace("\t\r\n", "")
        arr = [float(i) for i in received.split('\t')]
        #print(arr)
        if len(arr) == 15:
            last_telemetry.update(arr)
            #queue.append(last_telemetry)
            #print(time.time())
    server.close()

# УЧАСТНИКАМ НТО - ЗАДАЧА 4 -----
def getOrientation(pts, img):
    ## [pca]
    # Construct a buffer used by the pca analysis
    sz = len(pts)
    data_pts = np.empty((sz, 2), dtype=np.float64)

```

```

for i in range(data_pts.shape[0]):
    data_pts[i,0] = pts[i,0,0]
    data_pts[i,1] = pts[i,0,1]

# Perform PCA analysis
mean = np.empty((0))
mean, eigenvectors, eigenvalues = cv2.PCACompute2(data_pts, mean)

# Store the center of the object
cntr = (int(mean[0,0]), int(mean[0,1]))
angle = atan2(eigenvectors[0,1], eigenvectors[0,0]) # orientation in radians

return (angle, cntr)

# Поток обработки данных с камеры
def camera_processing_thread():
    #color definition
    red_lower = np.array([0, 200, 160], dtype="uint8")
    red_upper = np.array([20, 255, 255], dtype="uint8")
    red_lower_v = np.array([235, 120, 150], dtype="uint8")
    red_upper_v = np.array([255, 255, 255], dtype="uint8")
    x0 = 345.1
    z0 = 682.0
    lx = 5.8
    ly = 3.4
    lxx = sqrt(lx**2 + 1) + sqrt(0.5**2 + 0.5**2)
    frame_count = 0
    vtime_p = 0.0
    vtime_o = 0.0
    vx = 0.0
    vz = 0.0
    vpsi = 0.0
    vh = 0.0

    # Создание объекта Camera
    #capture = Camera('rtsp://localhost:555/live')
    vcap = cv2.VideoCapture(0)
    vcap.set(cv2.CAP_PROP_FRAME_WIDTH, 1600);
    vcap.set(cv2.CAP_PROP_FRAME_HEIGHT, 900);

    print("Connected to livestream, ready to search for the object...")
    # Основной цикл программы
    while running:
        x = 0
        z = 0
        psi = 0
        h = 0
        lete = last_telemetry
        #if (len(queue) != 0):
        #     lete = queue.popleft()
        if (lete.ins_state == 0 and lete.gps_state == 0):
            # everything works fine
            x = lete.x
            z = lete.z
            psi = lete.yaw * pi / 180.0
            h = lete.h
            vtime_p = time.time()
            vtime_o = time.time()
            vx = x
            vz = z

```

```

    vpsi = psi
    vh = h
else:
    if (lete.ins_state == 0 and lete.gps_state == 1):
        tt = time.time()
        x = vx + lete.velocity * cos(lete.yaw * pi / 180.0) * (tt -
            ↪ vtime_p) / 1000
        z = vz + lete.velocity * sin(lete.yaw * pi / 180.0) * (tt -
            ↪ vtime_p) / 1000
        h = vh
        psi = lete.yaw * pi / 180.0
        vpsi = psi
    elif (lete.ins_state == 1 and lete.gps_state == 0):
        psi = atan2(lete.x - vx, lete.z - vz)
        x = lete.x
        z = lete.z
        h = lete.h
        vx = x
        vz = z
        vh = h
    else:
        tt = time.time()
        x = vx + lete.velocity * cos(vpsi) * (tt - vtime_p) / 1000
        z = vz + lete.velocity * sin(vpsi) * (tt - vtime_p) / 1000

# Получение последнего снимка с бортовой камеры
#print(time.time())
    # Normalize image brightness
    cv2.normalize(img, img, 0, 255, cv2.NORM_MINMAX)
    # Blur the image to reduce noise
    blur = cv2.medianBlur(img, 5)
    # Convert BGR to HSV
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    # Find color masks
    red3_mask = cv2.inRange(hsv, red_lower, red_upper)
    red2_mask = cv2.inRange(hsv, red_lower_v, red_upper_v)
    full_mask = red2_mask + red3_mask
    # Find counters
    contours, hierarchy = cv2.findContours(full_mask, cv2.RETR_LIST,
        ↪ cv2.CHAIN_APPROX_SIMPLE)

    if len(contours) != 0:
        comb = contours[0].copy()
        for i in range(1, len(contours)):
            comb = np.concatenate((comb, contours[i]))
        c = comb

    if len(comb) > 130:
        frame_count += 1
        if frame_count == 3:
            i = 0
            # Calculate the area of each contour
            area = cv2.contourArea(c)
            alpha, cntr = getOrientation(c, img)
            alpha = -alpha - pi/2.0

```

```

cx = []
cy = []
for el in c:
    cx.append(el[0][0])
    cy.append(el[0][1])
#print(cx)
dpx = max(cx) - min(cx)
dpy = max(cy) - min(cy) # vertical
dx = abs(lxx * cos(alpha)) # vertical
dz = abs(lxx * sin(alpha))
height, width, channels = img.shape
pixel_height = dx / dpy
pixel_width = dz / dpx
center_dx = cntr[0] - width / 2 # right
center_dy = height / 2 - cntr[1] # up
vect = [center_dy * pixel_height, center_dx * pixel_width]
v2 = [1, 0]
#psi = 45 * pi / 180
v2[0] = vect[0] * cos(psi) + vect[1] * sin(psi)
v2[1] = -vect[0] * sin(psi) + vect[1] * cos(psi)

print("Data used x: {} z: {} psi: {}".format(x, z, psi))
print("Solution X: {:.2f}, Z: {:.2f}".format(x+v2[0],
→ z+v2[1]))
print("Solutuion error: {:.2f} (X: {:.2f}, Z:
→ {:.2f})".format(sqrt((x+v2[0] - x0)**2 + (z+v2[1] -
→ z0)**2), x+v2[0] - x0, z+v2[1] - z0))
# Draw the principal components
cv2.circle(img, cntr, 3, (0, 200, 0), 2)
cv2.imwrite("sol_4.png", img)
break
else:
    continue
#print("Solutuion error: {:.2f} (X: {:.2f}, Z:
→ {:.2f})".format(sqrt((xuav+v2[0] - x0)**2 + (zuav+v2[1] -
→ z0)**2), xuav+v2[0] - x0, zuav+v2[1] - z0))
else:
    a=0
    #print("comb too small")
else:
    a=0
    #print("No contours")
else:
    a=0
    #print("no ret")
    #print("")
    #print("")
    time.sleep(0.05)

# УЧАСТНИКАМ НТО - ЗАДАЧА 4 -----

# Основная функция для управления потоками
if __name__ == '__main__':
    cv2.setNumThreads(3)
    capture_thr = Thread(target=udp_server_thread, args=[])
    capture_thr.daemon = True
    capture_thr.start()

```

```
process_thr = Thread(target=camera_processing_thread, args=[])
process_thr.daemon = True
process_thr.start()

try:
    while running:
        time.sleep(1)
except KeyboardInterrupt:
    print('\ninterrupted!')
```

Критерии оценивания

Оценивается точность определения координат местоположения искомого объекта.

- Максимальный балл за решение задачи — 30 баллов.
- 25 баллов начисляется за точность решения. Требуемая точность решения задачи — 3 метра. Если величина ошибки превышает это значение, за каждые 4 дополнительные метра ошибки снимается 1 балл.
- Дополнительно 5 баллов начисляется за предоставление снимка с результатами обработки системы технического зрения (картинки с обозначенными координатами центра объекта).
- Максимальное время полета БЛА — 10 минут.
- Правильный ответ на вопрос по теории прибавляет 2 балла (всего 2 вопроса).

Критерии определения победителей и призеров

Первый отборочный этап

В первом отборочном этапе участники решали задачи по двум предметам: физика и информатика, в каждом предмете максимально можно было набрать 100 баллов. Для того, чтобы пройти во второй этап участники должны были набрать в сумме по обоим предметам не менее 70 баллов, независимо от уровня.

Второй отборочный этап

Количество баллов, набранных при решении всех задач второго отборочного этапа, суммируется.

Победители второго отборочного этапа должны были набрать не менее 90 баллов, независимо от уровня.

Заключительный этап

Индивидуальный предметный тур

- Физика — максимально возможный балл за все задачи — 100 баллов;
- Информатика — максимально возможный балл за все задачи — 100 баллов.

Командный практический тур

Команды, прошедшие в заключительный этап, получали за командный практический тур от 0 до 100 баллов: команда, набравшая максимальное число очков, становилась командой-победителем.

В заключительном этапе олимпиады баллы участника складываются из двух частей, каждая из которых имеет собственный вес: баллы за индивидуальное решение задач по предметам (физика, информатика) с весом 0,15 каждый предмет и баллы за командное решение практических задач в области беспилотных авиационных систем с весом 0,7.

Итоговый балл определяется по формуле:

$$S = 0,15 \cdot (S1 + S2) + 0,7 \cdot S3, \text{ где}$$

$S1$ — балл первой части заключительного этапа по физике в столбальной системе ($S1_{\text{макс}} = 100$);

S_2 — балл первой части заключительного этапа по информатике в стобалльной системе ($S_{2\text{макс}} = 100$);

S_3 — итоговый балл командного тура в стобалльной системе ($S_{3\text{макс}} = 100$).

Итого максимально возможный балл по условиям общего рейтинга:

$$0,15 \cdot (100 + 100) + 0,7 \cdot 100 = 100 \text{ баллов.}$$

Критерий определения победителей и призеров (независимо от класса)

Критерий определения победителей и призеров (независимо от класса): был сделан общий рейтинг, где 8-9 классы участвовали на общих основаниях с 10–11 классами. С начала рейтинга были выбраны 2 победителя и 7 призеров (первые 25% участников рейтинга становятся победителями или призерами — первые 8% участников рейтинга становятся победителями, оставшиеся 17% — призерами).

Категория	Количество баллов
Победители	78,66 и выше
Призеры	от 67,15 до 78,60