Второй отборочный этап

Участникам финала 2021 года предстоит организовать производственную линию для строительства модульных сооружений — домов, состоящих из различных блоков. Командам предстоит поработать над несколькими этапами строительства зданий, например, разработать гриппер (захват), позволяющий захватывать модули любой формы, и запрограммировать производственную линию для сборки модульного дома. Данный подход к проектированию и производству вещей демонстрирует новые подходы четвертой промышленной революции, в концепции которой входит производство вещей по требованию и адаптированной под конкретного заказчика. Сам процесс при этом высоко автоматизирован, гибок и не требует участия человека. Сборка предварительно отрабатывается в симуляторе, что позволяет ускорить процесс разработки и снизить риск повреждения оборудования.

Задание рассчитано на команду из 3 человек, однако возможно и меньшее количество участников. Для успешного выполнения задачи необходимо правильно распределить роли, участники могут совмещать их и делить обязанности.

Роли и необходимые навыки

- Инженер-проектировщик: умение работать в САПР (предпочтительно Fusion 360, Inventor, Компас 3D), использование других программ возможно по согласованию с организаторами. Знание рабочего процесса проектирования деталей и сборок (чертежи и сборка должны быть определены, выставлены взаимосвязи, детали должны проектироваться с учетом технологии производства).
- Технолог: умение описывать процессы, документировать процессы производства и сборки. Работа в CAM программах для 3D печати (Cura и др.).
- Программист (PC): знание одного из популярных языков программирования (C++, python), умение работать с библиотекой openCV, умение находить координаты объектов по изображению с камер.
- Программист (микроконтроллер): Проектирование несложных программ для МК (рекомендовано Arduino, но можно использовать и другие).

Индивидуальная часть

Программирование

Задача II.1.1.1. (1 балл)

На одном известном форуме участник попросил помочь ему разобраться с ssh. В частности, он спросил пример UNIX-команды для подключения по ssh к другому серверу. Добродушные участники форума быстро ответили новичку, прислав команду:

ssh -p 10205 -q test_host@my_name.com -E user

Помогите юноше узнать, что является именем пользователя, чтобы участник смог заменить имя и войти под своим логином.

Ответом является одно слово — имя пользователя, указанное в примере команды.

Ответ: test_host.

Задача II.1.1.2. (1 балл)

На одном известном форуме участник попросил помочь ему разобраться с ssh. В частности, он спросил пример UNIX-команды для подключения по ssh к другому серверу. Неопытные участники форума быстро ответили новичку, прислав свои версии команд. Помогите юноше узнать, какая команда верная, чтобы участник смог войти под своим логином на свой локальный сервер.

Имя пользователя, под которым необходимо зайти: Vovan

IP адрес сервера: 192.168.1.224

Разрешенный порт: 192

1. ssh -p 192 -q -E Vovan host@192.168.1.224
2. ssh -p 293 -q Vovan@192.168.1.192 -E ip4
3. ssh -p 192 -q 192.168.1.224@Vovan
4. ssh -p 10205 -q Vovan@192.168.1.224 -E 192
5. ssh -p 192 -q Vovan@192.168.1.224 -E 168

Ответ: 5.

Задача II.1.1.3. (2 балла)

Неопытный участник команды хочет зайти на сервер и посмотреть log-файл работы устройства, но он не может этого сделать. Помогите сформировать UNIX-команду для того, чтобы участник смог зайти на сервер по следующим параметрам и выполнить задание.

Имя пользователя, под которым необходимо зайти: admin

IP адрес сервера: rasbiHost.com

Разрешенный порт: 25

Внимание! Введите строку для выполнения только операции присоединения к серверу по ssh по необходимому порту, не используйте лишние опции утилиты ssh.

Ответ: ssh -p 25 admin@rasbiHost.com; ssh admin@rasbiHost.com -p 25;
ssh -p25 admin@rasbiHost.com; ssh admin@rasbiHost.com -p25.

Задача II.1.1.4. (2 балла)

Вас попросили написать программу для манипулятора и запустить ее на заводе, который находится далеко от вас в другой стране. Завод работает автономно без

участия людей, поэтому запускать вашу программу некому. Единственный выход — передать информацию по какому-нибудь протоколу связи (ssh) и запустить. На данном этапе вам предстоит только отправить файл, запускать его не требуется. Для доступа к манипулятору вам предоставили следующие реквизиты:

Имя пользователя: admin_robot13

IP адрес сервера: autoFactory_234.us

Разрешенный порт: 23

Ваш файл называется: Bug_fix.py

Директория, в которой лежат скрипты манипулятора на удаленном сервере: `~/Programm\_files`

Ответ: scp -P23 Bug_fix.py admin_robot13@autoFactory_234.us: /Programm_files; scp -P 23 Bug_fix.py admin_robot13@autoFactory_234.us: /Programm_files.

G-code

Для решения данного типа заданий необходимо знать базовый набор команд для управления станков с ЧПУ. Данный набор команд можно посмотреть на https://marlinfw.org/meta/gcode/

Задача II.1.2.1. (1 балл)

Укажите верные команды G-код для линейного перемещения.

- G92
- G2
- G28
- G1
- G30
- G0

Ответ: G1; G0.

Задача II.1.2.2. (1 балл)

Сопоставьте команду с описанием ее действия.

1. G92	a. Установить координаты текущей позиции.
2. G28	b. Перемещение по дуге.
3. G90	c. Автопарковка осей станка.
4. G0	d. Использовать абсолютную систему координат.
5. G2	e. Использовать относительную систему координат.
6. G91	f. Линейное перемещение.

Ответ: 1 — a; 2 — c; 3 — d; 4 — f; 5 — b; 6 — e.

Задача II.1.2.3. (1 балл)

Выберите скрипт(-ы) для перемещения на 10 мм по X и на 5,3 мм по Y относительно произвольной начальной позиции головы.

1. G90 G92 X43 Y2 G1 X53 Y7.3
2. G91 G92 X32 Y0 G1 X42 Y5.3
3. G90 G92 X32 Y45 G1 X40 Y2.6 G91 G0 X2 Y 2.7

Ответ: 1; 3.

Задача II.1.2.4. (1 балл)

Выберите скрипт для перемещения на позицию (3.2, 10, 0.5) относительно произвольной начальной позиции головы (необходимо, чтобы после выполнения команд станок считал, что он находится в точке (3.2, 10, 0.5)):

1. G90 G92 X24 Y33.3 G1 X20 Y2.6 Z5 G91 G0 Y3.4 Z-4.5 G92 X 3.2
2. G91 G92 X24 Y33.3 G1 X20 Y2.6 Z5 G90 G0 Y3.4 Z-4.5 G92 X 3.2
3. G90 G28 G0 X3.2 Z0.5 G1 Y10 G92 X0

Ответ: 1.

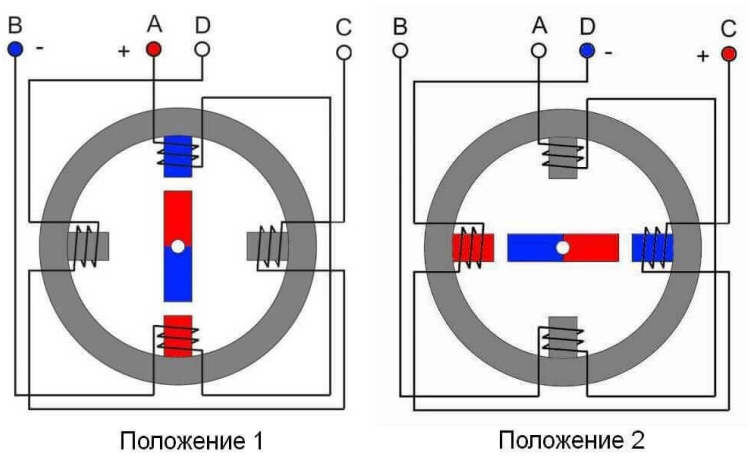
Arduino

Для решения данного типа задач необходимо изучить базовый набор процедур для инициализации пинов и их управлением, также потребуется владение языком Wiring, синтаксис которого был взят из языков C/C++. Ознакомиться с фреймворком Arduino можно по ссылке: https://www.arduino.cc/reference/en/

Задача II.1.3.1. Arduino (8 баллов)

У вас на финале неожиданно сгорел драйвер для управления шаговым двигателем, запасных драйверов нет. Вы решаете быстро сделать свой драйвер управления шагового двигателя на Arduino, но для этого вам необходимо написать программу.

Шаговый двигатель состоит из 2 обмоток.



Управляется такой двигатель с помощью последовательного изменения полярности обмоток статора.

Чтобы запустить мотор в полушаговом режиме, необходимо реализовать следующую последовательность сигналов, для того чтобы мотор сделал один шаг:

A	**B**	**C**	**D**
1	0	0	0
1	1	0	0
0	1	0	0
0	1	1	0
0	0	1	0
0	0	1	1
0	0	0	1
1	0	0	1

Т. к. частота работы микроконтроллера Arduino очень высокая (16 кГц), то для корректной работы мотора необходимо выдерживать микропаузы между переключением обмоток (2 мс).

Напишите программу для управления мотором с шагом 1,8 градуса на Arduino UNO для циклического вращения ротора мотора на 5 оборотов в одном направлении, а затем — на 5 оборотов в другом направлении вращения.

Считайте, что обмотки подключены следующим образом:

- A → D3
- B → D4
- C → D7
- D → D8

Внимание! Программа пишется на языке C++.

Решение

```
1  void setup(){
2      pinMode(D3, OUTPUT);
3      pinMode(D4, OUTPUT);
4      pinMode(D7, OUTPUT);
```

```
5       pinMode(D8, OUTPUT);
6   }
7
8   void loop(){
9   // put your loop code there
10      for (int i = 0; i <1000; i++){
11          digitalWrite(D3,1);
12          digitalWrite(D4,0);
13          digitalWrite(D7,0);
14          digitalWrite(D8,0);
15          delay(2);
16          digitalWrite(D3,1);
17          digitalWrite(D4,1);
18          digitalWrite(D7,0);
19          digitalWrite(D8,0);
20          delay(2);
21          digitalWrite(D3,0);
22          digitalWrite(D4,1);
23          digitalWrite(D7,0);
24          digitalWrite(D8,0);
25          delay(2);
26          digitalWrite(D3,0);
27          digitalWrite(D4,1);
28          digitalWrite(D7,1);
29          digitalWrite(D8,0);
30          delay(2);
31          digitalWrite(D3,0);
32          digitalWrite(D4,0);
33          digitalWrite(D7,1);
34          digitalWrite(D8,0);
35          delay(2);
36          digitalWrite(D3,0);
37          digitalWrite(D4,0);
38          digitalWrite(D7,1);
39          digitalWrite(D8,1);
40          delay(2);
41          digitalWrite(D3,0);
42          digitalWrite(D4,0);
43          digitalWrite(D7,0);
44          digitalWrite(D8,1);
45          delay(2);
46          digitalWrite(D3,1);
47          digitalWrite(D4,0);
48          digitalWrite(D7,0);
49          digitalWrite(D8,1);
50          delay(2);
51      }
52      for (int i = 0; i <1000; i++){
53          digitalWrite(D3,1);
54          digitalWrite(D4,0);
55          digitalWrite(D7,0);
56          digitalWrite(D8,1);
57          delay(2);
58          digitalWrite(D3,0);
59          digitalWrite(D4,0);
60          digitalWrite(D7,0);
61          digitalWrite(D8,1);
62          delay(2);
63          digitalWrite(D3,0);
64          digitalWrite(D4,0);
```

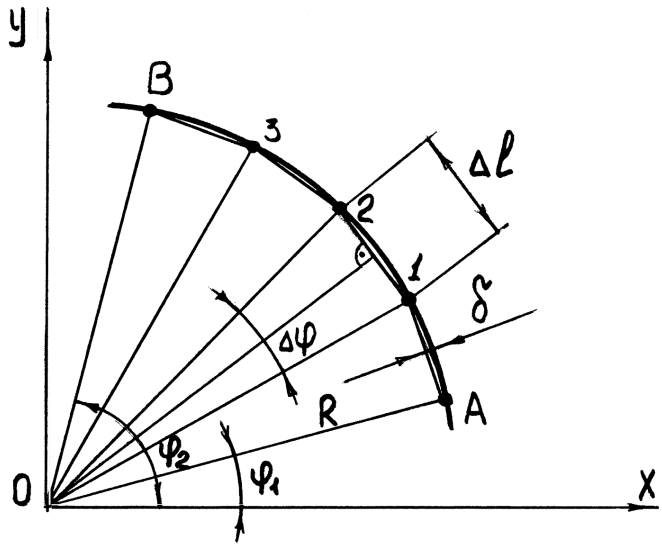
```
65          digitalWrite(D7,1);
66          digitalWrite(D8,1);
67          delay(2);
68          digitalWrite(D3,0);
69          digitalWrite(D4,0);
70          digitalWrite(D7,1);
71          digitalWrite(D8,0);
72          delay(2);
73          digitalWrite(D3,0);
74          digitalWrite(D4,1);
75          digitalWrite(D7,1);
76          digitalWrite(D8,0);
77          delay(2);
78          digitalWrite(D3,0);
79          digitalWrite(D4,1);
80          digitalWrite(D7,0);
81          digitalWrite(D8,0);
82          delay(2);
83          digitalWrite(D3,1);
84          digitalWrite(D4,1);
85          digitalWrite(D7,0);
86          digitalWrite(D8,0);
87          delay(2);
88          digitalWrite(D3,1);
89          digitalWrite(D4,0);
90          digitalWrite(D7,0);
91          digitalWrite(D8,0);
92          delay(2);
93      }
94  }
```

Расчет перемещения и генерация G-code

Для решения данного вида задач необходимы базовые знания линейной алгебры, в частности, — матрица вращения вектора.

Задача II.1.4.1. Напишите программу для интерполяции перемещения по дуге с заданной точностью (15 баллов)

В вашей системе ЧПУ не предусмотрено движение по дуге. Вам необходимо выполнить аппроксимацию движения по дуге и реализовать данный функционал путем создания списка команд линейного перемещения (G0, G1).



Формат входных данных

В первой строке через пробел задаются параметры $x0$, $y0$ (координаты центра окружности) $x1$, $y1$ (точка, откуда исходит дуга) a (градусная мера дуги окружности). $-10000 < x0, y0, x1, y1 < 10000$, $-360 < a < 360$.

Во второй строке задается Δl (см. рисунок).

Если последнее перемещение не равно Δl, то найти все точки удовлетворяющие Δl. Последняя точка должна быть $\leqslant \Delta l$.

Формат выходных данных

Список команд G1 с указанием позиции перемещения. Необходимо выводить по одной команде в каждой строке.

Примеры

Пример №1

Стандартный ввод
0 0 0 100 90 20

Стандартный вывод
G1 X0.000 Y100.000 G1 X-18.930 Y98.192 G1 X-37.718 Y92.614 G1 X-55.421 Y83.238 G1 X-71.044 Y70.376 G1 X-83.720 Y54.690 G1 X-92.869 Y37.086 G1 X-98.267 Y18.536 G1 X-100.000 Y-0.085

Решение

```
from math import atan, radians, acos, pi, sin, cos

def sign(klop):
    if klop < 0:
        return -1
    elif klop > 0:
        return 1
    return 0

x, y, x1, y1, a = map(float, input().split())
l = float(input())
x_absolute, y_absolute = x1-x, y1-y
r = (x_absolute**2 + y_absolute**2) ** 0.5
a = radians(a)
b = acos((r**2 + r**2 - l**2)/(2*r*r))
times = abs(int(a//b))
x_current, y_current = x_absolute, y_absolute
print("G1 ", "X", "%.3f" % (x1), " Y", "%.3f" % (y1), sep="")
for i in range(times):
    A = -2 * x_current
    B = -2 * y_current
    C = x_current ** 2 + y_current ** 2 + r ** 2 - l ** 2
    x0, y0 = -(A * C) / (A * A + B * B), -(B * C) / (A * A + B * B)
    d = r * r - C * C / (A * A + B * B)
    mult = (d / (A ** 2 + B ** 2)) ** 0.5
    ax, bx = 0, 0
    if a > 0:
        ax = x0 + B * mult
        ay = y0 - A * mult
    else:
        ay = y0 + A * mult
        ax = x0 - B * mult
    print("G1 ", "X", "%.3f" % (ax + x), " Y", "%.3f" % (ay + y), sep="")
    x_current, y_current = ax, ay
if int(a%b * 10000) > 0:
    l = (r**2 + r**2 - 2 * r * r * cos(a%b))**0.5
    A = -2 * x_current
    B = -2 * y_current
    C = x_current ** 2 + y_current ** 2 + r ** 2 - l ** 2
    x0, y0 = -(A * C) / (A * A + B * B), -(B * C) / (A * A + B * B)
    d = r * r - C * C / (A * A + B * B)
    mult = (d / (A ** 2 + B ** 2)) ** 0.5
    ax, bx = 0, 0
    if a > 0:
        ax = x0 + B * mult
        ay = y0 - A * mult
    else:
        ay = y0 + A * mult
        ax = x0 - B * mult
    print("G1 ", "X", "%.3f" % (ax + x), " Y", "%.3f" % (ay + y), sep="")
```

Задача II.1.4.2. Напишите программу для оптимизации перемещения (17 баллов)

Вам в пользование предоставили мостовой кран с ЧПУ, который должен перемещать грузы на складе, но во время выполнения задачи программа для управле-

ния перемещением крана дала сбой, и он начал делать много лишних перемещений, из-за чего его производительность заметно уменьшилась. Ваша задача — повысить продуктивность перемещений путем написания программы, которая должна будет анализировать существующий G-код и на основе данных о перемещении составить новый — такой, чтобы все перемещения грузов совершались линейно. Гарантируется, что грузы не могут быть поставлены друг на друга.

Формат входных данных

На вход подается число n (количество строк G-кода), далее в n строках передаются команды линейного перемещения. Гарантируется, что все значения перемещения задаются в абсолютной CO относительно точки (0, 0).

Любое поднимание/опускание груза определяется с помощью перемещения в позицию -10 / 0 по Z (изначально груз не захвачен) и параметром E 10/5 (10 — груз отпущен, 5 — захвачен).

Формат выходных данных

Список команд G1 с указанием позиции перемещения. Необходимо выводить по одной команде в каждой строке.

Примеры

Пример №1

Стандартный ввод

```
16
G90
G28
G1 E10
G1 X4.47 Y62.03 F1521
G1 X175.74 Y368.93
G1 Z-10
G1 E5
G1 Z0
G1 X83.63 Y169.15
G1 Z-10
G1 E5
G1 Z0
G1 X4.47 Y34.02
G1 Z-10
G1 E10
G1 Z0
```

Стандартный вывод

```
G90
G28
G1 E10
G1 X175.74 Y368.93
G1 Z-10
G1 E5
G1 Z0
G1 X4.47 Y34.02
G1 Z-10
G1 E10
G1 Z0
```

Решение

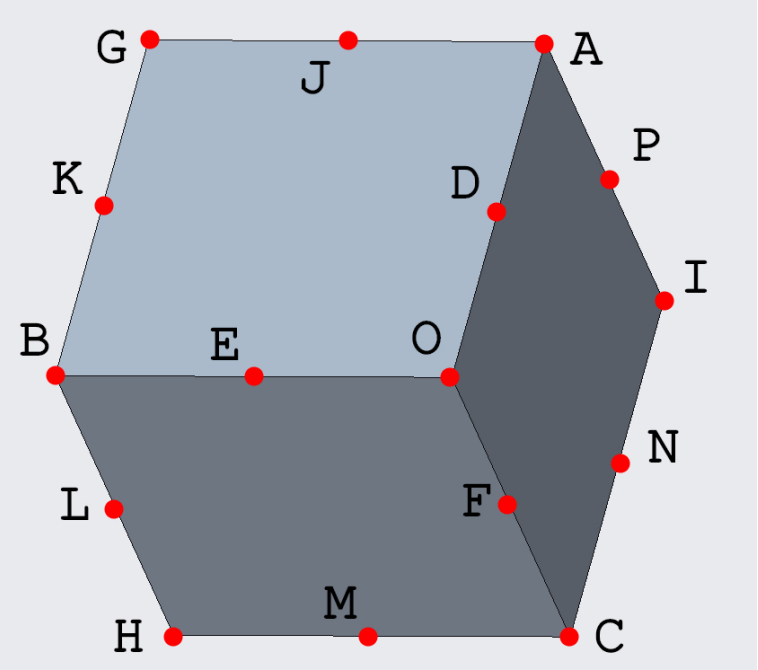
```
a=0
ay=0
ae=10
outlist=[]
sg=False
k=int(float(input()))
while k>0:
    cmd=input()
    k-=1
    if 'X' in cmd or 'Y' in cmd:
        sg=True
        cds=cmd.split()[1:]
        got_x='X' in cds[0]
        if got_x:
            x=float(cds[0][1:])
            del cds[0]
            ax=x
        if cds:
            got_y='Y' in cds[0]
            if got_y:
                y=float(cds[0][1:])
                ay=y
```

```
    elif 'Z' in cmd:
        z=int(cmd[4:])
        s=input()
        k-=1
        if 'E' in s:
            e=int(s[4:])
            if e!=ae:
                if sg:
                    sg=False
                    operation='G1 X{} Y{}'.format(ax, ay)
                    outlist.append(operation)
                outlist.append('G1 Z{}'.format(z))
                outlist.append('G1 E{}'.format(e))
                ae=e
                zz=int(input()[4:])
                k-=1
                outlist.append('G1 Z{}'.format(zz))
            else:
                zz=int(input()[4:])
                k-=1
    else:
        if sg:
            sg=False
            operation='G1 X{} Y{}'.format(ax, ay)
            outlist.append(operation)
        outlist.append(cmd)
for cmd in outlist:
    print(cmd)
```

Моделирование

Задача II.1.5.1. Базовое понимание 3D-геометрии (10 баллов)

Укажите форму сечения, проходящего через заданные точки:



Сопоставьте значения из двух списков:

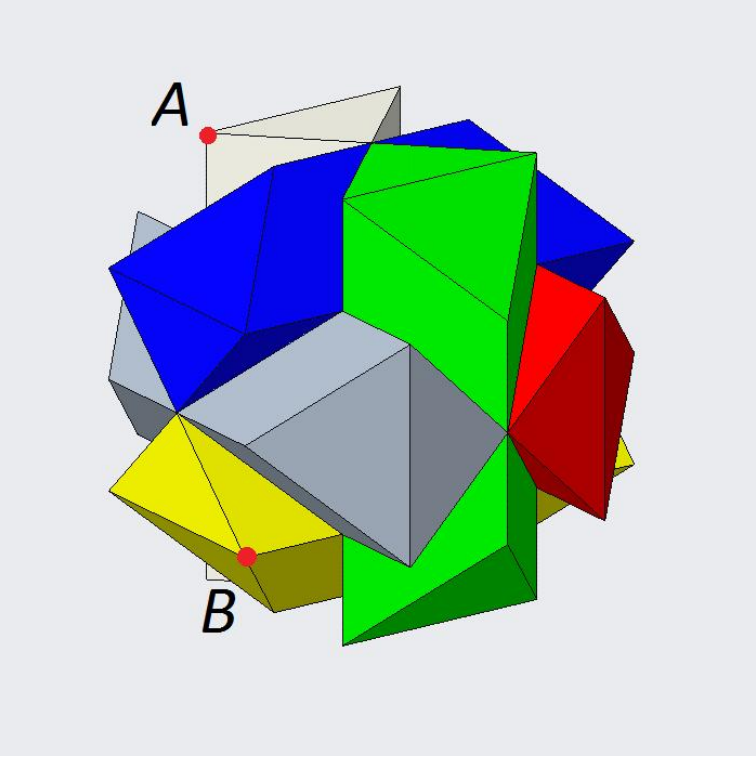
1. OJP	a. равносторонний треугольник
2. ABC	b. правильный шестиугольник
3. BCD	c. равнобедренный треугольник (боковая сторона меньше основания)
4. LFP	d. прямоугольник
5. KLM	e. квадрат
6. GOC	f. равнобедренный треугольник (боковая сторона больше основания)

Ответ: 1 — f; 2 — a; 3 — c; 4 — e; 5 — b; 6 — d.

Задача II.1.5.2. Импорт файлов в САПР (15 баллов)

Загрузите деталь STEP, откройте ее в САПР и произведите представленную на рисунке сборку, состоящую из 6 одинаковых деталей.

Ответом являются значение расстояния между точками A и B и объем получившейся конструкции.



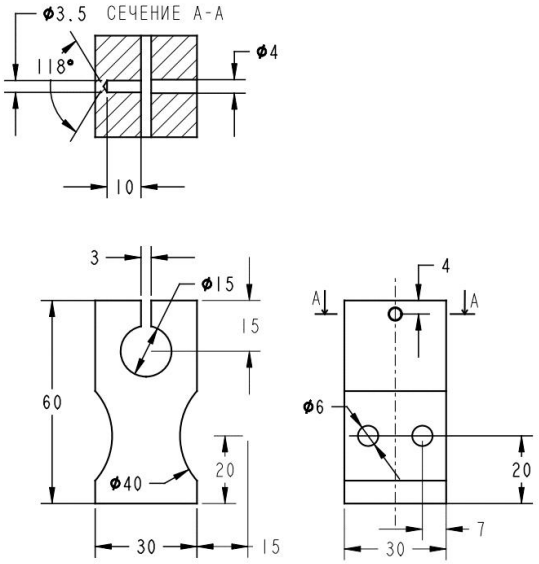
Ссылка для скачивания детали:

https://stepik.org/media/attachments/lesson/429726/Import.stp

Форма ответа: два числа, введеных через пробел, где первое число — расстояние (мм), второе — объем (мм^3). Ответы вводить с точностью до двух знаков после запятой БЕЗ округления.

Ответ: 93,54 ± 0,1; 543650,44 ± 10.

Задача II.1.5.3. Моделирование и анализ объектов в САПР (15 баллов)

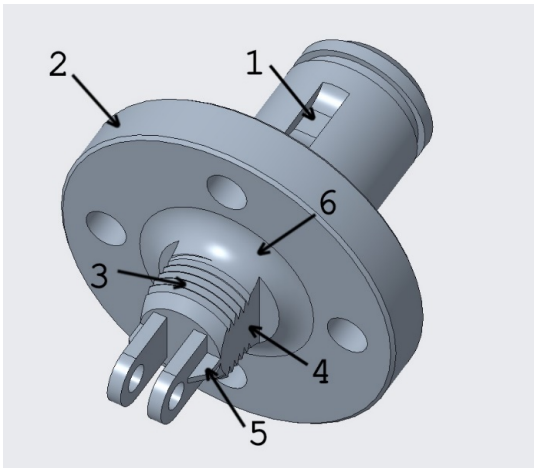


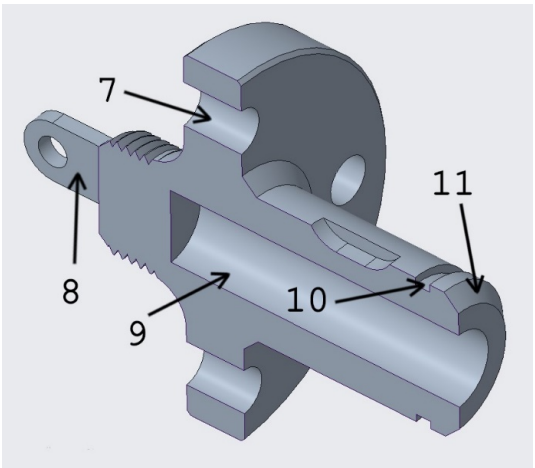
Постройте модель по данному чертежу, введите ее объем и массу при заданном материале (сталь, плотность 7,9 г/см^3).

Форма ответа: два числа введеных через пробел, где первое число — объем (мм^3), второе — масса детали (г). Ответы вводить с точностью до двух знаков после запятой **без** округления.

Ответ: 41172,72 ± 15; 325,26 ± 0,1.

Задача II.1.5.4. Сопоставте номера элементов с их названиями (10 баллов)





Сопоставьте значения из двух списков:

1	a. Ушко
2	b. Сквозное отверстие
3	c. Ребро
4	d. Проточка
5	e. Лыска
6	f. Внешняя резьба
7	g. Шпоночный паз
8	h. Глухое отверстие
9	i. Канавка
10	j. Фланец
11	k. Фаска
Отсутствует	l. Галтель

Ответ: 1- g; 2- j; 3- f; 4- e; 5- c; 6- l; 7- b; 8- a; 9- h; 10- d; 11- k; Отсутствует- i.

Командная часть

Программирование

Задача II.2.1.1. Что это такое? (30 баллов)

Перед тем как собирать дом с помощью робота, нужно понять, из чего вы его будете собирать, а так же где какая деталь должна находиться. На стройплощадке в хаотичном порядке разбросаны различные блоки, необходимые для возведения строения. Чтобы проиндексировать стройматериалы, манипулятор запускает режим сканирования, при котором каждый найденный объект он фиксирует на камеру. Помогите манипулятору определить тип детали, чтобы манипулятор смог успешно завершить сканирование.

Формат входных данных

На вход подается строка URL для скачивания снимка, который сделал манипулятор.

Формат выходных данных

В качестве ответа принимается одно слово — форма фигуры на фотографии. Заранее известно, что на стройплощадке могли находиться только следующие блоки (фигуры): Cube (куб), Arch (арка), Cylinder (цилиндр), Cone (конус), Pyramid (четырехугольная пирамида), Sphere (сфера), Tor (тор).

Примеры изображений фигур:

1. Arch: https://stepik.org/media/attachments/lesson/433326/Arch_cr.png
2. Cone: https://stepik.org/media/attachments/lesson/433326/Cone_cr.png
3. Cube: https://stepik.org/media/attachments/lesson/433326/Cube_cr.png
4. Cylinder:

https://stepik.org/media/attachments/lesson/433326/Cylinder_cr.png

5. Pyramid:
   https://stepik.org/media/attachments/lesson/433326/Pyramid_cr.png
6. Sphere:
   https://stepik.org/media/attachments/lesson/433326/Sphere_cr.png
7. Tor: https://stepik.org/media/attachments/lesson/433326/Tor_cr.png

Все фигуры неизменны. Меняется только ракурс съемки.

Список доступных библиотек:

https://stepik.org/lesson/59057/step/14?unit=36621

Примеры

Пример №1

Стандартный ввод
https://stepik.org/media/attachments/lesson/433326/Sphere_cr_2332.png
Стандартный вывод
Cube

Решение

Примерный алгоритм решения:

1. Перевести изображение в одноканальное.
2. Выбрать контур проекции фигуры на плоскость обзора.
3. Посчитать количество вершин контура.
4. При необходимости добавить условия существования фигур, чтобы отсечь лишние предположения.

```python
import urllib.request

import cv2
import numpy as np


def get_image_from_url(image_url: str, reading_mode=cv2.IMREAD_GRAYSCALE):
    """
    Download image from the given url, decode and return it (in grayscale by default).

    :param image_url: url path to the image
    :param reading_mode: Image reading mode. Default is cv2.IMREAD_GRAYSCALE
    :returns: cv2 image object
    """
    url_response = urllib.request.urlopen(image_url)
    return cv2.imdecode(
        np.array(bytearray(url_response.read()), dtype=np.uint8), reading_mode
    )


def get_cleaned_hough_lines(canny_edges):
```

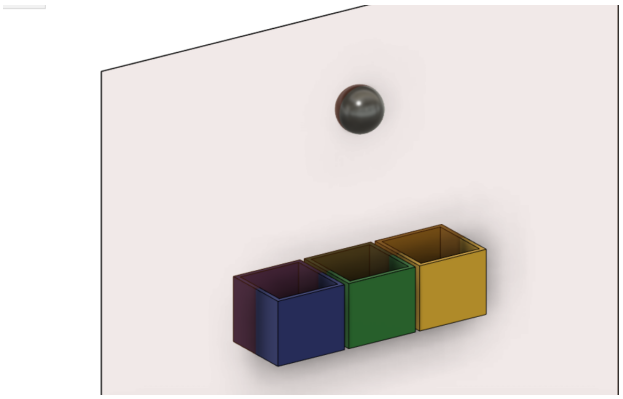
```
22      """
23      Cyclically finds lines on the image using cv2.HoughLines function. After finding
    ↪   the lines, paints the first one
24      with black and so on until there are no lines left. Thanks to this, all segments
    ↪   found on one straight line will be
25      counted as one line.
26
27      :param canny_edges: Image with edges
28      :returns: Found lines as list in the format that returns cv2.HoughLines function
29      """
30      working_edges = canny_edges.copy()
31      total_lines = []
32      while True:
33          lines = cv2.HoughLines(working_edges, 1, np.pi / 360, 100)
34
35          if lines is None:
36              break
37          else:
38              found_line = lines[0]
39              total_lines.append(found_line)
40
41              rho, theta = found_line[0]
42              a = np.cos(theta)
43              b = np.sin(theta)
44              x0 = a * rho
45              y0 = b * rho
46              x1 = int(x0 + 1000 * -b)
47              y1 = int(y0 + 1000 * a)
48              x2 = int(x0 - 1000 * -b)
49              y2 = int(y0 - 1000 * a)
50              cv2.line(working_edges, (x1, y1), (x2, y2), 0, 3)
51
52      return total_lines
53
54
55  def get_parallel_lines_count(hough_lines):
56      existing_angles = {}
57      for angle in [line[0][-1] for line in hough_lines]:
58          for existing_angle, count in existing_angles.items():
59              if abs(angle - existing_angle) < 0.05:
60                  existing_angles[existing_angle] += 1
61                  break
62          else:
63              existing_angles[round(angle, 5)] = 1
64      parallel_lines_count = 0
65      for angle_count in existing_angles.values():
66          if angle_count > 1:
67              parallel_lines_count += 1
68      return parallel_lines_count
69
70
71  url = input()
72  img = get_image_from_url(url, cv2.IMREAD_UNCHANGED)
73
74  # Convert the image to grayscale
75  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
76
77  # Blur the image
78  blur = cv2.bilateralFilter(gray, 12, 120, 140)
79
```

```
80  # Find edges in the image
81  edges = cv2.Canny(blur, 80, 180, apertureSize=3)
82
83  dilated = cv2.dilate(edges, None, iterations=1)
84
85  # Find contours
86  _, contours_, hierarchy = cv2.findContours(dilated, cv2.RETR_CCOMP,
    ↪  cv2.CHAIN_APPROX_SIMPLE)
87  contours = []
88
89  contour = None
90  biggest_area = 0
91  for cntr in contours_:
92      cntr_area = cv2.contourArea(cntr)
93      if cntr_area > 1000:
94          contours.append(cntr)
95          if cntr_area > biggest_area:
96              biggest_area = cntr_area
97              contour = cntr
98
99  # Approximate the contour
100 contour_approx = cv2.approxPolyDP(contour, 0.003 * cv2.arcLength(contour, True), True)
101
102 # Find circles in the image
103 # circles = cv2.HoughCircles(edges, cv2.HOUGH_GRADIENT, 1, 1000, param1=50, param2=30,
    ↪  minRadius=0, maxRadius=0)
104
105 lines = get_cleaned_hough_lines(edges)
106 lines_count = len(lines)
107 parallel_lines_count = get_parallel_lines_count(lines)
108
109 # Sphere or Tor, if there are no lines on the image
110 if lines_count == 0:
111     # Tor, if several contours are found
112     if len(contours) // 2 >= 2:
113         print('Tor')
114     # Sphere otherwise
115     else:
116         print('Sphere')
117 # Arch, if contour is not convex
118 elif not cv2.isContourConvex(contour_approx):
119     print('Arch')
120 # Cone or Cylinder, if there are exactly two lines
121 elif lines_count == 2:
122     # Cone, if these lines are not parallel
123     if parallel_lines_count == 0:
124         print('Cone')
125     # Cylinder otherwise
126     else:
127         print('Cylinder')
128 # Remained Cube or Pyramid
129 else:
130     if lines_count >= 8 or parallel_lines_count >= 2:
131         print('Cube')
132     else:
133         print('Pyramid')
```

Задача II.2.1.2. Куда пропала деталька? (20 баллов)

У вас есть 3 коробки, в которые бросают предметы различных форм (куб и шар, аналогичные предметам из предыдущего задания). Для съемки попадания поставили камеру, которая при фиксации движения начинает снимать объект. К сожалению, за время падения предмета камера успевает делать только 2 снимка. Определите в какую коробку упадет предмет. Считайте, что рывок предмета при броске настолько велик, что ускорение свободного падения не влияет на траекторию движения и объект движется прямолинейно. Учитывайте, что камера может снимать под различными ракурсами.

Грарантируется, что линейное перемещение объекта совершается только в одной плоскости (XY):



Формат входных данных

На вход подается 2 строки URL для скачивания снимков, которые расположены в порядке фиксации камерой.

Формат выходных данных

Выведите цвет коробки, куда предположительно упадет предмет: Green (зеленый), Yellow (желтый), Blue (синий).

Список доступных библиотек:

https://stepik.org/lesson/59057/step/14?unit=36621

Примеры

Пример №1

Стандартный ввод
https://stepik.org/media/attachments/lesson/433326/OpenCV_task_2_2_cr.png https://stepik.org/media/attachments/lesson/433326/OpenCV_task_2_3_cr.png

Стандартный вывод
Blue

Решение

Примерный алгоритм решения:

1. Перевести изображение в одноканальное.
2. Определить замкнутые области с помощью выделения границ.
3. Отсечь лишние области по количеству вершин.
4. Определить положение объектов по цвету из многоканального изображения.
5. Проделать пункты 1–4 со вторым изображением.
6. Провести прямую между начальным и конечным положением движущегося объекта.
7. Определить, центр какой коробки находится ближе всего к проведенной прямой.

```
import numpy as np
import urllib.request
import cv2
def url_to_image(url):
    with urllib.request.urlopen(url) as url:
        image = np.asarray(bytearray(url.read()), dtype="uint8")
        image = cv2.imdecode(image, cv2.IMREAD_COLOR)
        return image

def image_resize(image, width = None, height = None, inter = cv2.INTER_AREA):
        dim = None
        (h, w) = image.shape[:2]
        if width is None and height is None:
                return image
        if width is None:
                r = height / float(h)
                dim = (int(w * r), height)
        else:
                r = width / float(w)
                dim = (width, int(h * r))
        resized = cv2.resize(image, dim, interpolation = inter)
        return resized

def distance(p1, p2, p3):
    return np.linalg.norm(np.cross(p2-p1, p1-p3))/np.linalg.norm(p2-p1)


y_min = np.array((18, 69, 2), np.uint8)
y_max = np.array((40, 208, 255), np.uint8)

g_min = np.array((46, 49, 0), np.uint8)
g_max = np.array((65, 172, 255), np.uint8)

b_min = np.array((81, 77, 0), np.uint8)
b_max = np.array((200, 222, 255), np.uint8)

gr_min = np.array((11, 5, 49), np.uint8)
gr_max = np.array((51, 100, 255), np.uint8)


s1 = input()
s2 = input()
#2 3 48 49
```

```
44  # s1 = "https://stepik.org/media/attachments/lesson/433326/OpenCV_task_2_4_cr.png"
45  # s2 = "https://stepik.org/media/attachments/lesson/433326/OpenCV_task_2_5_cr.png"
46  # s1 = "https://stepik.org/media/attachments/lesson/433326/OpenCV_task_2_0_cr.png"
47  # s2 = "https://stepik.org/media/attachments/lesson/433326/OpenCV_task_2_1_cr.png"
48
49  img1 = url_to_image(s1)
50  img2 = url_to_image(s2)
51
52  img1 = image_resize(img1, width=300)
53  img2 = image_resize(img2, width=300)
54
55  hsv1 = cv2.cvtColor(img1, cv2.COLOR_BGR2HSV)
56  hsv2 = cv2.cvtColor(img2, cv2.COLOR_BGR2HSV)
57
58  gr1_thresh = cv2.inRange(hsv1, gr_min, gr_max)
59  gr2_thresh = cv2.inRange(hsv2, gr_min, gr_max)
60
61  y_thresh = cv2.inRange(hsv2, y_min, y_max)
62  g_thresh = cv2.inRange(hsv2, g_min, g_max)
63  b_thresh = cv2.inRange(hsv2, b_min, b_max)
64
65  try:
66      gr_cnts1, _ = cv2.findContours(gr1_thresh.copy(), cv2.RETR_EXTERNAL,
        ↪  cv2.CHAIN_APPROX_NONE)
67      gr_cnts2, _ = cv2.findContours(gr2_thresh.copy(), cv2.RETR_EXTERNAL,
        ↪  cv2.CHAIN_APPROX_NONE)
68
69      y_cnts, _ = cv2.findContours(y_thresh.copy(), cv2.RETR_EXTERNAL,
        ↪  cv2.CHAIN_APPROX_NONE)
70      g_cnts, _ = cv2.findContours(g_thresh.copy(), cv2.RETR_EXTERNAL,
        ↪  cv2.CHAIN_APPROX_NONE)
71      b_cnts, _ = cv2.findContours(b_thresh.copy(), cv2.RETR_EXTERNAL,
        ↪  cv2.CHAIN_APPROX_NONE)
72  except:
73      _, gr_cnts1, _ = cv2.findContours(gr1_thresh.copy(), cv2.RETR_EXTERNAL,
        ↪  cv2.CHAIN_APPROX_NONE)
74      _, gr_cnts2, _ = cv2.findContours(gr2_thresh.copy(), cv2.RETR_EXTERNAL,
        ↪  cv2.CHAIN_APPROX_NONE)
75
76      _, y_cnts, _ = cv2.findContours(y_thresh.copy(), cv2.RETR_EXTERNAL,
        ↪  cv2.CHAIN_APPROX_NONE)
77      _, g_cnts, _ = cv2.findContours(g_thresh.copy(), cv2.RETR_EXTERNAL,
        ↪  cv2.CHAIN_APPROX_NONE)
78      _, b_cnts, _ = cv2.findContours(b_thresh.copy(), cv2.RETR_EXTERNAL,
        ↪  cv2.CHAIN_APPROX_NONE)
79
80  x1, y1 = 0, 0
81  x2, y2 = 0, 0
82  x_y, y_y = 0, 0
83  x_g, y_g = 0, 0
84  x_b, y_b = 0, 0
85
86  if len(gr_cnts1) != 0:
87      c = max(gr_cnts1, key = cv2.contourArea)
88      x, y, w, h = cv2.boundingRect(c)
89      x1 = (x + x+w)/2
90      # x1 = x+w
91      y1 = (y + y+h)/2
92      y1 = y+h
93      # M = cv2.moments(c)
```

```
94      # if M["m00"] != 0:
95      #     x1 = int(M["m10"] / M["m00"])
96      #     y1 = int(M["m01"] / M["m00"])
97      # else:
98      #     x1, y1 = 0, 0
99      cv2.circle(img1, (int(x1), int(y1)), 2, (0, 0, 0), 2)
100     cv2.drawContours(img1, c, -1, (0, 0, 0), 3)
101
102 if len(gr_cnts2) != 0:
103     c = max(gr_cnts2, key = cv2.contourArea)
104     x, y, w, h = cv2.boundingRect(c)
105     x2 = (x + x+w)/2
106     y2 = (y + y+h)/2
107     y2 = y+h
108     # M = cv2.moments(c)
109     # if M["m00"] != 0:
110     #     x2 = int(M["m10"] / M["m00"])
111     #     y2 = int(M["m01"] / M["m00"])
112     # else:
113     #     x2, y2 = 0, 0
114     cv2.circle(img2, (int(x2), int(y2)), 2, (0, 0, 0), 2)
115     cv2.drawContours(img2, c, -1, (0, 0, 0), 3)
116
117 if len(y_cnts) != 0:
118     c = max(y_cnts, key = cv2.contourArea)
119     x, y, w, h = cv2.boundingRect(c)
120     x_y = (x + x+w)/2
121     y_y = (y + y+h)/2
122     # M = cv2.moments(c)
123     # if M["m00"] != 0:
124     #     x_y = int(M["m10"] / M["m00"])
125     #     y_y = int(M["m01"] / M["m00"])
126     # else:
127     #     x_y, y_y = 0, 0
128     cv2.circle(img2, (int(x_y), int(y_y)), 2, (0, 100, 100), 2)
129     cv2.drawContours(img2, c, -1, (0, 100, 100), 3)
130
131 if len(g_cnts) != 0:
132     c = max(g_cnts, key = cv2.contourArea)
133     x, y, w, h = cv2.boundingRect(c)
134     x_g = (x + x+w)/2
135     y_g = (y + y+h)/2
136     # M = cv2.moments(c)
137     # if M["m00"] != 0:
138     #     x_g = int(M["m10"] / M["m00"])
139     #     y_g = int(M["m01"] / M["m00"])
140     # else:
141     #     x_g, y_g = 0, 0
142     cv2.circle(img2, (int(x_g), int(y_g)), 2, (0, 255, 0), 2)
143     cv2.drawContours(img2, c, -1, (0, 255, 0), 3)
144
145 if len(b_cnts) != 0:
146     c = max(b_cnts, key = cv2.contourArea)
147     x, y, w, h = cv2.boundingRect(c)
148     x_b = (x + x+w)/2
149     y_b = (y + y+h)/2
150     # M = cv2.moments(c)
151     # if M["m00"] != 0:
152     #     x_b = int(M["m10"] / M["m00"])
153     #     y_b = int(M["m01"] / M["m00"])
```

```
154     # else:
155     #     x_b, y_b = 0, 0
156     cv2.circle(img2, (int(x_b), int(y_b)), 2, (255, 0, 0), 2)
157     cv2.drawContours(img2, c, -1, (255, 0, 0), 3)
158 
159 # print((x1, y1), (x2, y2), (x_y, y_y), (x_b, y_b), (x_g, y_g))
160 
161 d = {
162     'Yellow': distance(np.array((x1, y1)), np.array((x2, y2)), np.array((x_y, y_y))),
163     'Green': distance(np.array((x1, y1)), np.array((x2, y2)), np.array((x_g, y_g))),
164     'Blue': distance(np.array((x1, y1)), np.array((x2, y2)), np.array((x_b, y_b)))
165 }
166 
167 print(min(d, key=d.get))
168 
169 # cv2.line(img2, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255, 0), thickness=3,
    ↪  lineType=8)
170 # cv2.imshow('img1', img1)
171 # cv2.imshow('img2', img2)
172 # cv2.waitKey(0)
```

Задача II.2.1.3. Написать библиотеку для управления шаговым мотором с помощью драйвера ШД A4988 (30 баллов)

Теперь ваша задача состоит в том, чтобы написать библиотеку на основе фреймворка Arduino по управлению шаговым мотором с помощью драйвера ШД A4988.

В библиотеке должно быть реализовано следующее:

1. Два режима работы: линейное перемещение и управление вращением.
2. Управление углом поворота вала шагового двигателя.
3. Поддержка всех возможных дроблений шага.
4. Управление линейным перемещением в относительной и абсолютной системе координат.
5. Установка текущей позиции мотора.
6. Управление ускорением движения мотора.

Ваша задача состоит в следующем:

- Написать библиотеку.
- Оформить код.
- Написать README.md файл, где необходимо изложить документацию по использованию вашей библиотеки.
- Написать и оформить примеры использования.
- Грамотно вести историю создания в Git.

Примечания:

1. Код должен быть выполнен в одном стиле.
2. Переменные должны писаться так же в одном стиле, но имена классов и структур должны отличаться от именования переменных.
3. Документация должна содержать подробное описание по поводу того, что необходимо передать в функцию, какой размерности должна быть эта величина

(отличным решением будет применение утилиты doxygen).
4. Примеры использования должны находиться в отдельной директории.
5. Имена файлов с примерами должны переносить смысл примера.
6. Сами примеры должны быть хорошо прокомментированы.
7. Если разработка ведется несколькими участниками, то тогда каждый должен работать в отдельной ветке (желательно использование issue/pull request)
8. Каждый коммит — маленькое логически завершенное действие

Решение

Пример решения: https://github.com/GuestKP/StepperNTI

Моделирование

Следующее задание будет состоять из нескольких частей:

1. Моделирование.
2. Оптимизация.
3. Сборка.
4. Программирование.

Каждую из частей необходимо загружать на своем шаге.

Внимание! На каждом шаге задание будет проверяться только по критерием для данной части задания. Если решение любой части задания было отправлено не в свой шаг, оно проверяться не будет.

Проверки будут совершаться преимущественно по воскресеньям.

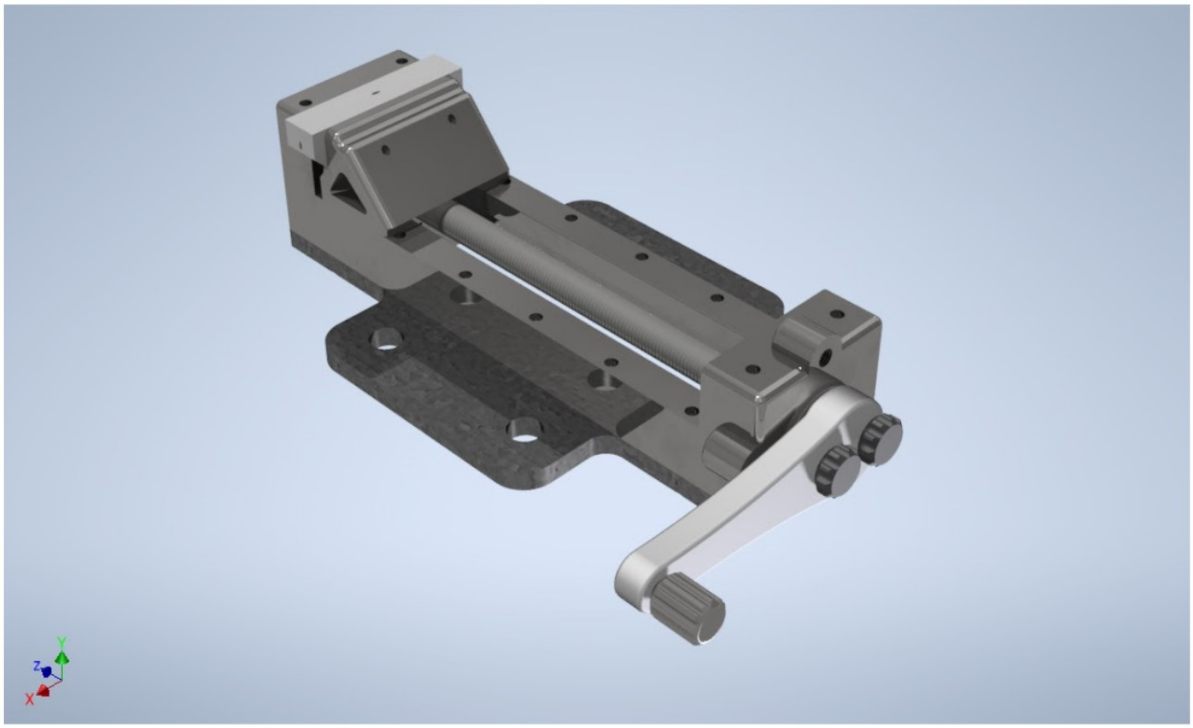
(Проверьте, чтобы у организаторов был свободный доступ к документам! Документы с закрытым доступом просматриваться и оцениваться не будут!)

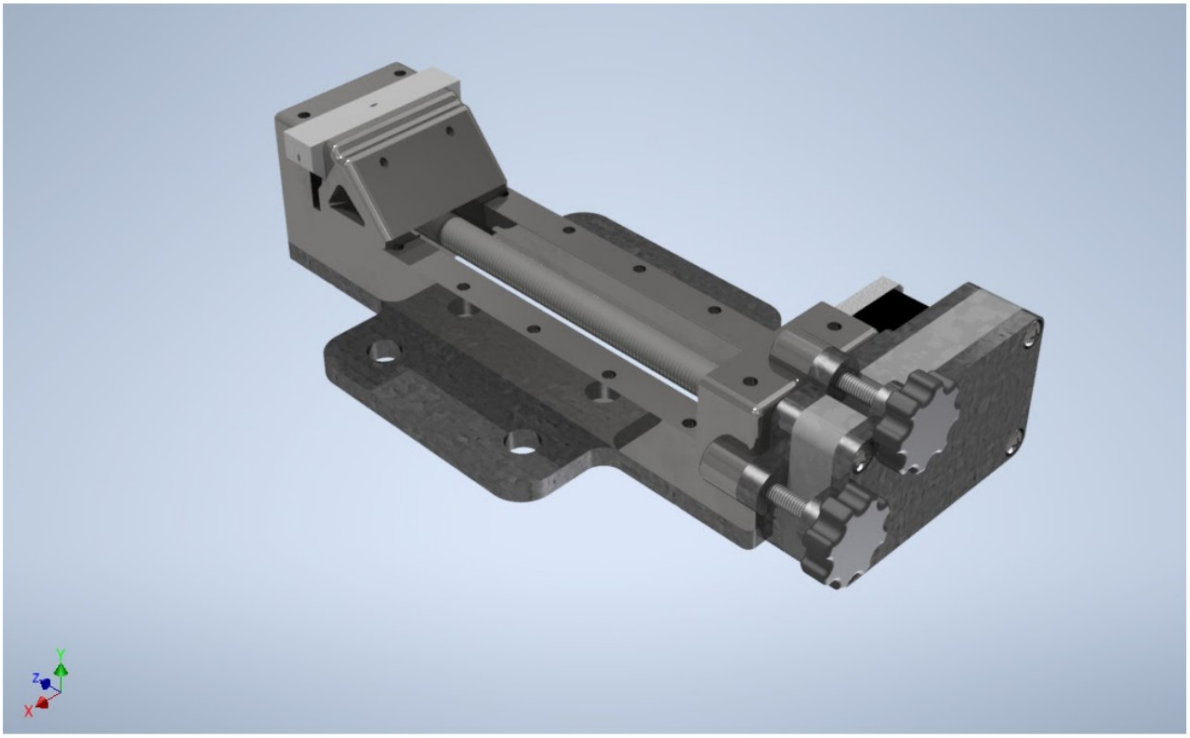
Задача II.2.2.1. Часть 1 (20 баллов)

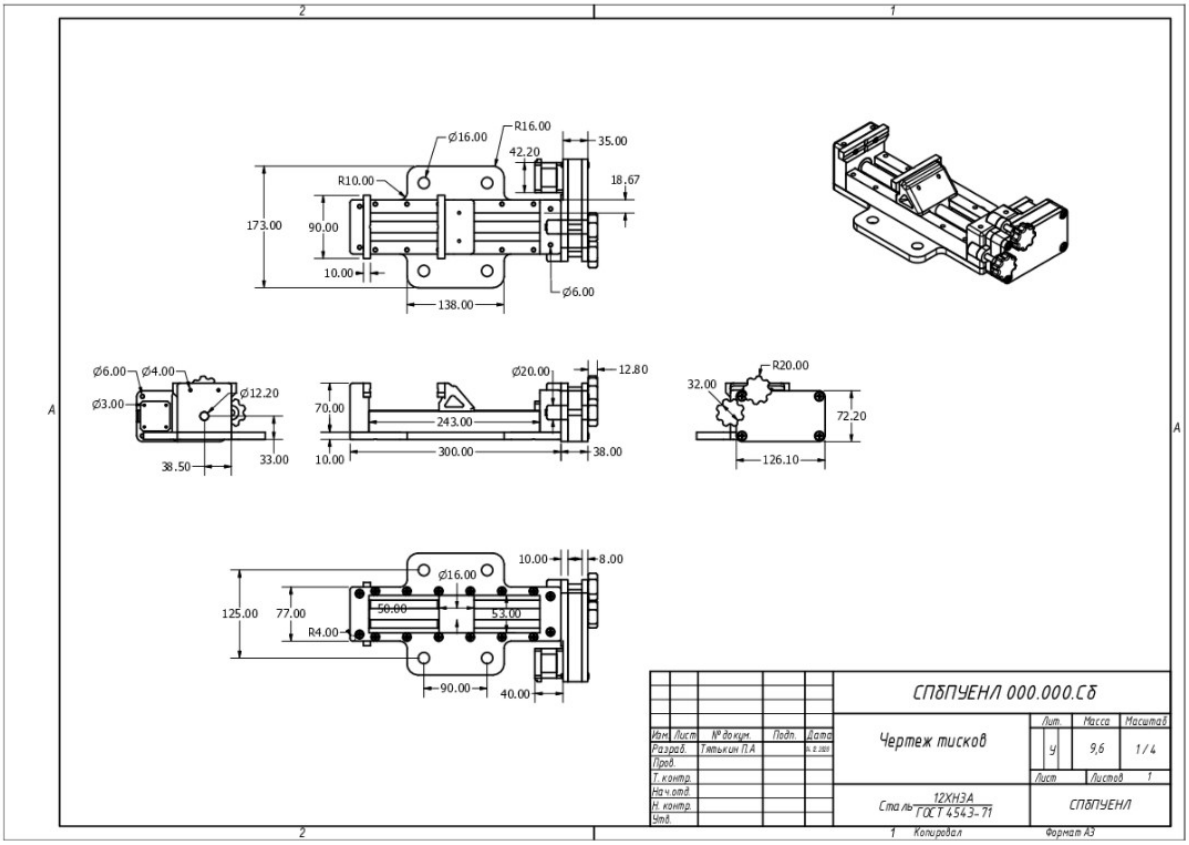
Вам предстоит разработать и создать 3D-модель тисков, удовлетворяющую техническому заданию. При разработке учитывайте удобство, функционал, простоту изготовления. Также не забывайте о том, что в дальнейшем созданную вами модель нужно будет оптимизировать и изготовить. Старайтесь не допускать технических решений, которые усложнят это. Показателем ваших результатов будут скриншоты или рендеры модели, демонстрирующие основные элементы конструкции и ее в целом, а так же STEP-файлы сборки тисков и сборок приводных частей и файл чертежа готового изделия в формате PDF с указанием габаритных размеров в сложенном положении и с установленным электрическим приводом.

ТЗ (часть 1): Тиски должны иметь габариты (Д×Ш×В) не более 400×200×100 мм, крепежные отверстия диаметром 16 мм, расположенные в углах прямоугольника со сторонами 125 мм по ширине и 90 мм по длине тисков. Также тиски должны обладать механизмом быстрой замены приводного механизма с электронного (шагового или редукторного двигателя) на ручной (ручку).

Решение





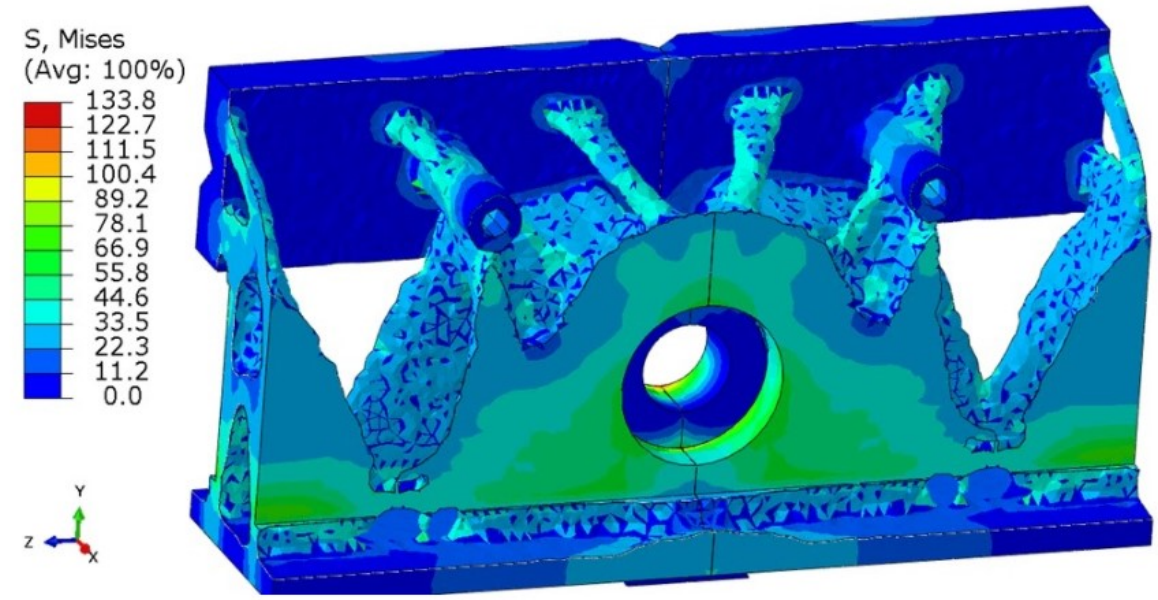


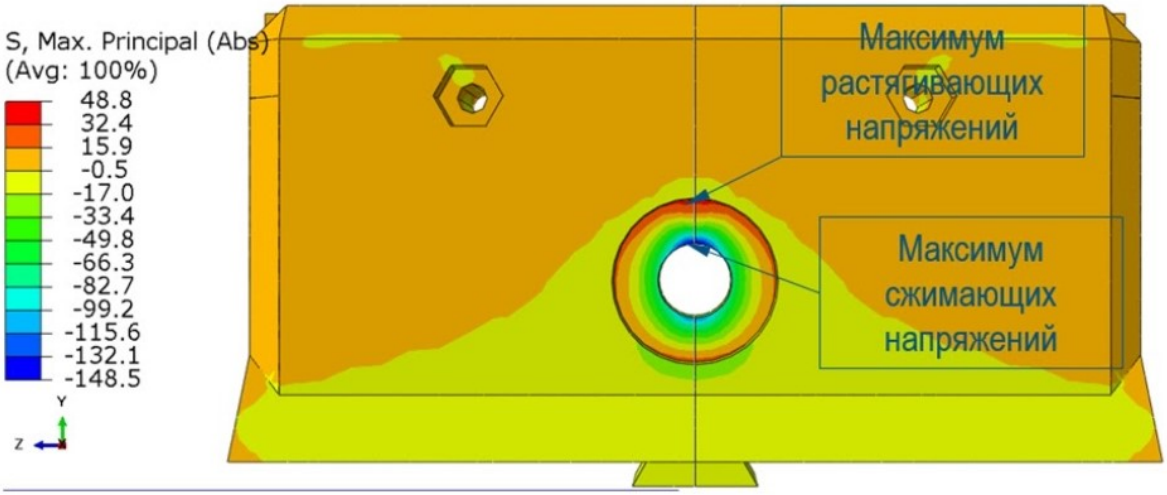
Задача II.2.2.2. Часть 2 (20 баллов)

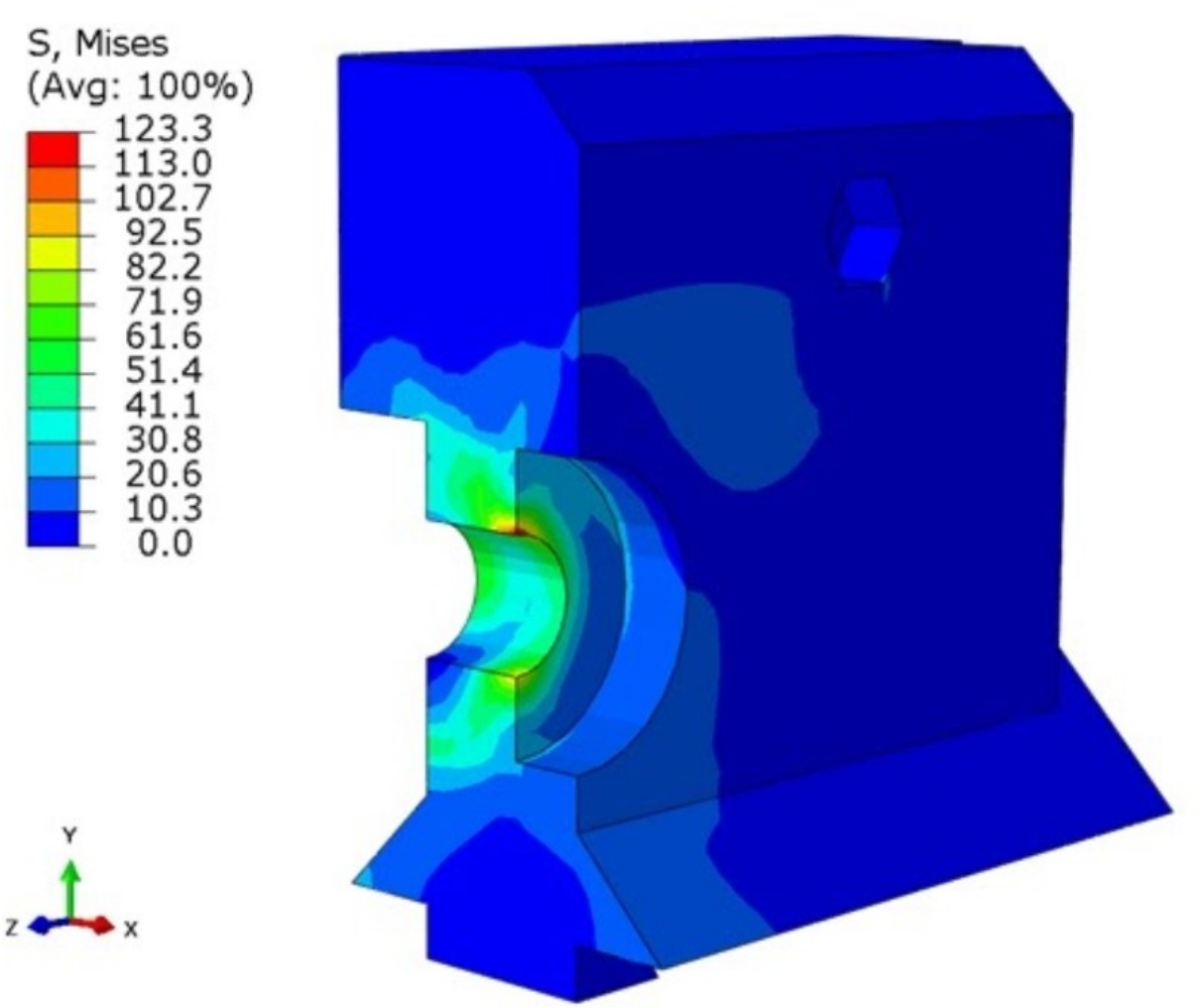
Далее ваша задача состоит в следующем: оптимизируйте с помощью топологической оптимизации или создайте с помощью генеративного дизайна подвижное крепление губок тисков. Произведите анализ нагрузок полученной детали, симулирующий сжатие тисков. Анализ должен показать самые слабые (нагруженные) части полученной или ранее созданной детали. Допускается выполнение задания «по частям»: выполнение только оптимизации или только анализа созданной в первой части детали. В таком случае невыполненная часть задания будет оценена нулем баллов. В качестве ответа на задание предоставьте скриншоты, демонстрирующие полученную деталь и выполненный анализ нагрузок, а также STEP-файл полученной детали.

ТЗ (часть 2): Тиски должны позволять производить пиление зажатой в них стальной пластины/профиля/уголка толщиной 3 мм, при этом не деформируясь и не ослабляя удержание заготовки. Также тиски должны иметь возможность зажимать круглые изделия как горизонтально, так и вертикально.

Решение


S, Mises
(Avg: 100%)
133.8
122.7
111.5
100.4
89.2
78.1
66.9
55.8
44.6
33.5
22.3
11.2
0.0
Y
Z
X

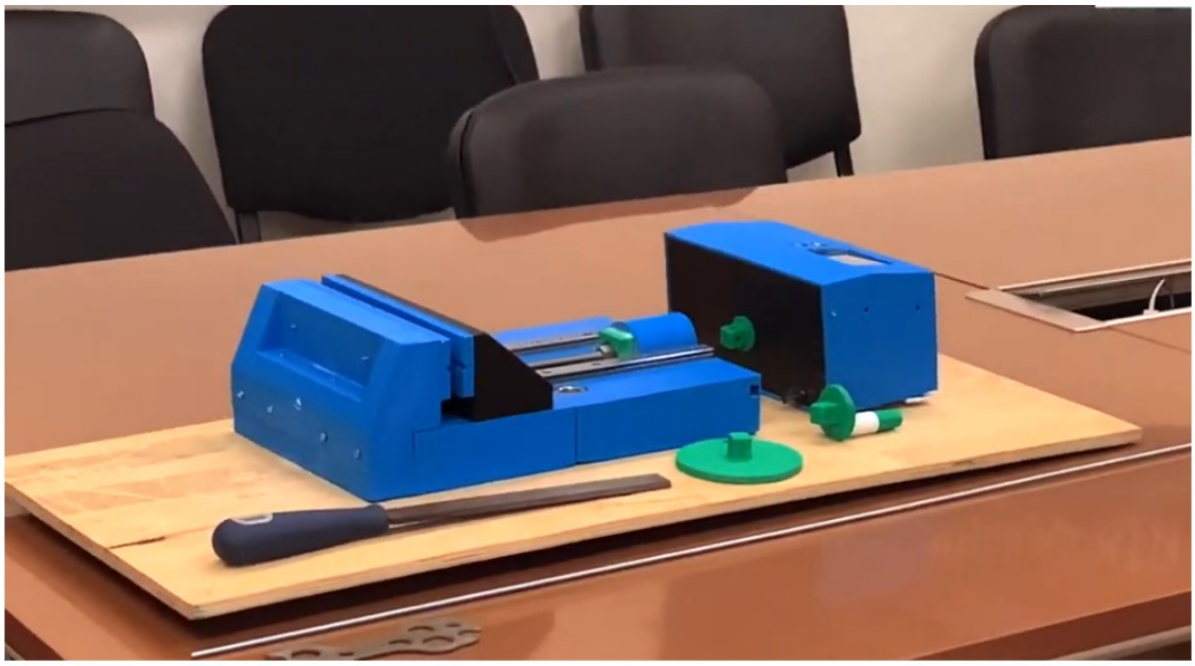

S, Max. Principal (Abs)
(Avg: 100%)
48.8
32.4
15.9
-0.5
-17.0
-33.4
-49.8
-66.3
-82.7
-99.2
-115.6
-132.1
-148.5
Y
Z
Максимум растягивающих напряжений
Максимум сжимающих напряжений



Задача II.2.2.3. Часть 3 (30 баллов)

Используя станки с ЧПУ и ручной инструмент, произведите детали и соберите тиски. При выполнении этого этапа постарайтесь средства, которые используются на реальном производстве. Для демонстрации результата загрузите в папку облачного хранилища видео с названием вашей команды, демонстрирующее полученный прототип и его работу (по второй части ТЗ),, ссылку на папку отправьте в качестве ответа.

Решение



Задача II.2.2.4. Часть 4 (30 баллов)

Вы уже пробовали поуправлять шаговым мотором. Получилось? Теперь ваша задача стоит в том, чтобы написать библиотеку на основе фреймворка Arduino по управлению шаговым мотором с помощью драйвера ШД A4988.

В библиотеке должно быть реализовано следующее:

1. Два режима работы: линейное перемещение и управление вращением.
2. Управление углом поворота вала шагового двигателя.
3. Поддержка всех возможных дроблений шага.
4. Управление линейным перемещением в относительной и абсолютной системе координат.
5. Установка текущей позиции мотора.
6. Управление ускорением движения мотора.

Ваша задача состоит в следующем:

- Написать библиотеку.
- Оформить код.
- Написать README.md файл, где необходимо изложить документацию по использованию вашей библиотеки.
- Написать и оформить примеры использования.
- Грамотно вести историю создания в Git.

В качестве ответа принимается ссылка на GitLab/GitHub или иной Git-репозиторий вашего проекта.

Внимание! Проверьте настройки приватности. Настройте все так, чтобы у организаторов был доступ к просмотру.