Домашна работа 8

Прасета в космоса

Въведение

Порки и Прасчо, интелектуално извисени свине аристократи, решили да си основат фирма за запълване на озонови дупки. Регистрирали предприятие с минимален капитал, наели бригада от зайци бездомници и се насочили към работната площадка. Готови за екшън!

Оказва се, че озоновият слой е като българския асфалт и рядко има само една дупчица. Обикновено са много, като в допълнение могат да се местят и без предупреждение. За щастие, традициите в запълването на космически аномалии, отдавна преминали рамките на научната фантастика.

Нашите приятели позиционирали телесата си в космическия апарат и след минути вече се носили в околоземна орбита. Пред зурлите им се изправила щетата, огромна дупка с размери чутовни, която очаквала моментално запушване. Позиционирали машината над целта, натиснали бутона за активиране на операцията, но “о, беда”. Оказало се, че апаратурата, поръчана в бързината, от “Али Експрес”, идвала с напълно флашнат програматор за запълване на дупки, който трябва да си програмираш сам.

Проклинайки късият си акъл, двамата предприемачи започнали да блъскат дебелите си глави в опит да открият решение на проблема. За щастие цялата бригада от зайци бездомници наскоро се записали на дистанционен курс по програмиране и поназнайвали туй онуй за оперативните функции на **озонобъркачките**. На лице е наръчник с инструкции и кислород за 168 часа. Помогнете на нашите космати приятели да напишат компилатор за управление на озонобъркачката и да се измъкнат невредими от дългата ръка на корпоративното безхаберие.

Спецификация

Разполагате с програмна среда, която емулира действието на **озонобъркачките**, както и работната площ ,върху която трябва да се развихрите.

- озонобъркачката разполага с **три роботски ръце**, които могат да извършват финни манипулации по озоновия слой. Роботските ръце имат специфичен идентификатор с латински букви съответно – **A**, **B** и **C**. В рамките на описанието ще разберете как се ползват тези идентификатори;
- озоновият слой е координатна система, която се идентифицира с положителни координати по X и Y;
- всяка една от дупките, които трябва да запълним има специфични координати;
- при стартиране на нашата програма (мисия), всяка една от роботските ръце, с които разполагаме, се намира на координати “X - 1” и “Y - 1”.

Забележка: Операцията по запълване на озоновите дупки, ще наричаме **мисия**.

За да можете да манипулирате роботските ръце е необходимо да програмирате поведението на стандартни команди, описани в наръчника на младия озонобъркач-програмист. Командите са описани в таблицата, като подробна информация и примери ще намерите в спецификацията по-долу.

команда	аргумент	описание
turnon	-	включва системата в оперативна готовност
load	име	активира роботска ръка
unload	име	деактивира роботска ръка
moveX	посока / число	премества роботската ръка по хоризонтала
moveY	посока / число	премества роботската ръка по вертикала
identify	-	идентифицира дупка
cut	число	изпълнява операция изрязване
fill	число	изпълнява операция запълване
finish	-	изпълнява операция запечатване
status	-	връща информация за статуса на мисията

Гориво и кислород

Озонобъркачката работи с **кислород** и **гориво**:

- всяко **движение на роботските ръце** консумира **кислород**;
- всяко **изпълнение на операция** консумира **гориво**.

Вашата мисия стартира със следните параметри:

- **168 часа** кислород
- **90 литра** гориво

Включване на озонобъркачката

Можете да стартирате изпълнението на програмата само и единствено ако подадете **команда за включване** на оперативния агрегат, описана в наръчника с команди по-горе. Ако машината не е включена, всеки опит за активиране на каквато и да е произволна друга команда трябва да върне съобщение за грешка: "**Оперативният агрегат не е включен**", след което да очаква нова команда.

Пример - озонобъркачката не е включена

> moveX
>
> 10
>
> Озонобъркачката не е включена - включете я преди да правите каквото и да е било

Пример - включваме озонобъркачката

> turnon
>
> Системата е включена

Включване и изключване на роботска ръка

За да може да работи една роботска ръка тя трябва да бъде включена. За тази цел използваме командата **load** и подаваме името на роботската ръка като аргумент. Ако се подаде грешен параметър (име на роботска ръка различно от трите налични), системата трябва да отговори със съобщение за грешка.

Важно: Всяко включване на роботска ръка, консумира *1 литър гориво*!

Пример - включване на роботска ръка A

> load
>
> A
>
> Ръка **A** е включена успешно

Пример - включване на несъществуваща роботска ръка D

> load
>
> D
>
> Не разполагате с подобно устройство

В даден момент системата може да използва само **една единствена роботска ръка**. Ако искаме да включим втора ръка, трябва изрично да изключим първата. Изключването става с командата **unload** , която не приема никакви параметри.

Ако се опитаме да активираме ръка без да изключим предишната, трябва да изведем съобщение за грешка в конзолата.

Пример - опит за включване на нова ръка - без да сме изключили предишната

> load
>
> A
>
> Ръка **A** е включена успешно
>
> load B
>
> Не може да включите, нова ръка

Правилния подход е да изключим ръка A и след това да включим ръка B

Пример - опит за включване на нова ръка - след успешно изключване на вече включена такава

> load
>
> A
>
> Ръка **A** е включена успешно
>
> unload
>
> Ръка **A** е изключена успешно
>
> load B
>
> Ръка **B** е включена успешно

Преместване на роботска ръка

За преместване на роботската ръка можем да използваме командите:

- **moveX**
- **moveY**

Особености на командите:

- и двете команди приемат като аргумент посоката на движение и номера на работната площадка (*вижте примера по-долу*);
- аргументите за посока са: **left**, **right**, **up**, **down** *(те трябва да се прочетат от клавиатурата);*
- номера на работната площадка е число прочетено от клавиатурата.

Операцията по преместване на роботска ръка консумира кислород. Повече детайли по-долу!

Пример 1: Намираме се на координа “X 1” и искаме да преместим ръката на координат “X 10”. Трябва да преминем през стъпките:

- зареждаме/включваме роботска ръка A;
- преместваме я надясно.

Резултатът в конзолата би изглеждал по следния начин:

```
load

A

moveX

right

10
```

Важно: Обърнете внимание, че аргументите се подават винаги на нов ред!

Пример 2: Намираме се на координа “Y 1” и искаме да преместим ръката на координат “Y 10”. Трябва да преминем през стъпките:

- зареждаме/включваме роботска ръка A;
- преместваме я нагоре.

```
load
A
moveY
up
10
```

Важно: Преместването на роботската ръка консумира толкова кислород, колкото позиции преместим ръката. В нашия случай първоначално се намираме на позиция 1 и местим ръката на позиция 10, тоест консумираме 9 часа кислород.

Идентифициране на дупка

За да стартираме процедури свързани с оправяне на дупка, трябва първо да **идентифицираме размера на дупката**.

Използваме командата "**identify**". Командата генерира случайно число в интервала от 1 до 5, което е размера на текущо идентифицираната дупка. Имайте предвид, че това число ще стане важно впоследствие.

Резултатът от операцията е съобщението: "***Дупката е идентифицирана!***".

Извършване на операция по запълване на дупка

Имаме налице **3 операции** свързани с един ремонт:

1. **изрязване** на озонова дупка – използваме командата "cut";
2. **запълване** на озонова дупка – използваме командата "fill";
3. **запечатване** на запълнената дупка – използваме командата "finish".

Операциите могат да се изпълняват само и единствено в този ред и не могат да бъдат прескачани. Ако се опитаме да запълним идентифицирана дупка преди да сме я изрязали, трябва да получим съобщение за грешка.

Пример

```
fill
Грешка - некоректна последователност на използваната команда
```

Операциите могат да бъдат стартирани само и единствено след успешно идентифициране на дупка!

Важно: Ако една дупка е идентифицирана от една роботска ръка, няма нужда да бъде повторно идентифицирана в рамките на същите координати. Тоест, ако роботска ръка “A” идентифицира дупка на координати “X - 9” и “Y - 5”, то роботска ръка “B” няма нужда да идентифицира същата дупка на същите координати.

Когато въведете командата “**cut**”:

1. **Системата еднократно** ще генерира случайно число от 1 до 20.
2. Системата ще ви подкани да го налучкате числото.
3. Операцията се повтаря, докато не успеете да налучкате числото..
4. Всеки път, когато налучкате грешно, системата консумирате **1 час кислород**.

Пример:

```
cut

Генерирах число от 1 до 20 познай го - за да отрежа дупката 15

17

Неуспешен опит пробвай пак!

1

Неуспешен опит пробвай пак!

10

Операцията cut е изпълнена успешно
```

Когато въведете командата “**fill**”:

1. Системата ще генерира случайно число от 1 до 20 и ще го събере с генерираното число от предишната операция “**cut**”
2. Системата ще ви подкани да го налучкате числото.
3. Операцията се повтаря, докато не успеете да налучкате числото.
4. Всеки път, когато налучкате грешно, системата консумирате **1 литър гориво**.

Пример (допускаме че случайно генерираното число от операцията cut е 16):

```
fill

Генерирах число от 1 до 20 и го събрах с 16 познай го, за да запълня дупката
15

 17

Неуспешен опит пробвай пак!

19

Операцията fill е изпълнена успешно!
```

Когато въведете командата “**finish**”, системата извежда съобщението:

```
Дупка на координати X и Y е запълнена успешно като X и Y в този контекст, са координатите на дупката
```

Статус на мисията

Във всеки един момент можем да проверим ресурсите, с които разполагаме с помощта на командата “**status**”. Тя не получава аргументи, но връща данни за:

- наличното гориво
- наличният кислород
- количеството запълнени дупки

Пример

```
status

наличното гориво - 50 литра

наличен кислород - 24 часа

запълнени дупки - 1
```

Стартиране на мисията

При стартиране на програмата е необходимо да въведем нашата цел – количеството дупки, които ще се опитаме да запълним. След като въведем количество, ще получим координатите на първата дупка.

- Координата X на всяка една от дупките е случайно генерирано число в интервала 1 до 50;
- Координата Y на всяка една от дупките е случайно генерирано число в интервала 1 до 50.

Пример

```
Колко дупки искате да запълните

2

Дупка 1 - се намира на X - 5 и Y - 7
```

След този момент можем да започнем да въвеждаме командите, които сме програмирали и да решаваме проблема.

След **успешно запълване на дупка**, **ще получим координатите на следващата**. Нека да видим същата програма, но за следващата дупка!

Пример

```
Колко дупки искате да запълните

2

Дупка 1 - се намира на X - 5 и Y - 7

{ команди за запълване на дупката }

Дупка 2 - се намира на X - 15 и Y - 27
```

Приключване на мисията

Програмата приключва изпълнението си при настъпване на **едно** от трите условия:

1. свършило е горивото;
2. количеството кислород е равно на или е паднало под 24 часа;
3. запълнили сме количеството дупки, които сме си поставили за цел.

Ако програмата приключи при настъпване на **условие 1** – изведете съобщение:

- Няма гориво в резервоара, моля презареди!

Ако програмата приключи при настъпване на **условие 2** – изведете съобщение:

- Достигнахме критичен кислороден минимум!

Ако програмата приключи при **условие 3** – изведете съобщение:

- Мисията е изпълнена успешно!

Особености на интерпретатора

При реализацията на нашата програма, имайте предвид следните особености:

- **Особеност 1:** Аргументите на командите винаги се подават на нов ред след командата;
- **Особеност 2:** Освен ако изрично не е казано, няма нужда да проверявате за валидността на аргументите.

Примерен вход/изход в конзолата

Нека разгледаме примерен вход и изход, който вашата програма може да прочете и изведе за потребителите на системата.

```
Колко дупки искате да запълните?

1

Дупка 1 - се намира на X - 5 и Y - 7

turnon

load

A

moveX

right

5

moveY

up

7

identify

Дупката е идентифицирана!

cut

Генерирах число от 1 до 20 познай го - за да отрежа дупката!

16

Операцията cut е изпълнена успешно!

fill

Генерирах число от 1 до 20 и го събрах с 16 познай го, за да запълня дупката!

25

Операцията fill е изпълнена успешно!

finish

Дупка на координати 5 и 7 е запълнена успешно!

Мисията е изпълнена успешно!
```

Критерии за оценяване

Домашната работа ще се оценява в точкова система. Максималното количество точки, които можете да получите от решението на поставените ви условия е ***100 точки***.

Функционалност	***Точки***
Включване на озонобъркачката	10
Включване и изключване на роботска ръка	10
Преместване на роботска ръка	10
Идентифициране на дупка	10
Извършване на операция “cut”	10
Извършване на операция “fill”	10
Извършване на операция “finish”	10
Статус на мисията	10
Стартиране на мисията	10
Приключване на мисията	10
Общо (точки)	***100***

Повече информация за предаването на вашите решения можете да намерите в секция “***Начини и срокове за предаване***”.

Начини и срокове за предаване

Разработката на домашната работа се прикачва към съответното задание като прикачена хипервръзка (*линк*) към GitLab проекта, който сте създали, и след това се предава. За целта, трябва да го предадете като натиснете бутон “***Предай***”.

Предаването на заданието трябва да се случи не по-късно от определения краен срок, упоменат в заданието на класната стая (*Google Classroom*).

Стандартният краен срок за предаване е до вторник 20:30 ч. на следващата седмица.

Задължително е да си направите нов проект в GitLab със заглавие **pu-fmi-java-intro-w8**

Начини за комуникация с преподавателския колектив

При въпроси от ваша страна, използвайте частните коментари към заданието в класната стая (*ако въпросите са относно вашата имплементация и проблеми с нея*) и публичните коментари (*ако имате въпроси относно самото задание*).