

# Упражнение: Повторения с цикли – While-цикъл

Задачи за упражнение и домашно към курса ["Основи на програмирането" в СофтУни](#).

Тествайте решението си в judge системата: <https://judge.softuni.bg/Contests/Index/2408#0>

## 1. Старата Библиотека

Ани отива до родния си град след много дълъг период извън страната. Прибирайки се вкъщи тя вижда старата библиотека на баба си и си спомня за любимата си книга. Помогнете на Ани, като напишете функция в която тя въвежда търсената от нея книга(текст). Докато Ани не намери любимата си книга или не провери всички в библиотеката, програмата трябва да чете всеки път на нов ред името на всяка следваща книга (текст). Книгите в библиотеката са свършили щом получите текст "No More Books".

- Ако не открие книгата да се отпечата на два реда:
  - "The book you search is not here!"
  - "You checked {брой} books."
- Ако открие книгата си се отпечатва един ред:
  - "You checked {брой} books and found it."

## Примерен вход и изход

Вход	Изход	Обяснения
(["Troy", "Stronger", "Life Style", "Troy"])	You checked 2 books and found it.	Книгата която Ани търси, в случая е Troy, а библиотеката съдържа 3 книги. Първата е Stronger, втората е Life Style, третата книга е търсената – Troy и програмата приключва.
(["The Spot", "Hunger Games", "Harry Potter", "Toronto", "Spotify", "No More Books"])	The book you search is not here! You checked 4 books.	Книгата, която търси Ани е "The Spot". Библиотеката съдържа 4 книги. Първата е Hunger Games, втората Harry Potter, третата Toronto, а четвъртата Spotify. Понеже няма повече книги в библиотеката четенето на имена приключва. Ани не намери книгата, която търсеше.
(["Bourne", "True Story", "Forever", "More Space", "The Girl", "Spaceship", "Strongest", "Profit", "Tripple", "Stella", "The Matrix", "Bourne"])	You checked 10 books and found it.	

## Насоки

1. Извадете любимата книга от масива с данни, който приема функцията.

```
let favouriteBook = input[0];
```

2. Направете **още две помощни променливи в началото**, които да следят, **дали книгата е намерена или всички книги са проверени**. Едната променлива ще е **брояч** и трябва да е **число** и с **първоначална стойност едно**. С нея ще следим, **колко книги са проверени**. Другата променлива трябва да е с **началната стойност false**.

```
let index = 1;  
let bookIsFound = false;
```

3. Ако книгата, която получихте от аргумента **съвпада с любимата книга на Ани**, **презапишете стойността на променливата от булев тип, и прекратете цикъла**, в противен случай **увеличете брояча с едно**.

```
let nextBookName = input[index];  
while (nextBookName !== "No More Books") {  
    if (nextBookName === favouriteBook) {  
        bookIsFound = true;  
        break;  
    }  
    index++;  
    nextBookName = input[index];  
}
```

4. Според това, дали книгата е намерена, **принтирайте нужните съобщения**.

```
if (bookIsFound === false) {  
    console.log("The book you search is not here!");  
    console.log(`You checked ${index - 1} books.`);  
} else {  
    console.log(`You checked ${index - 1} books and found it.`);  
}
```

## 2. Подготовка за изпит

Напишете функция, в която Марин решава задачи от изпити **докато не получи съобщение "Enough"** от лектора си. При всяка решена задача той получава оценка. **Функцията трябва да приключи прочитането на данни при команда "Enough" или ако Марин получи определеният брой незадоволителни оценки**. **Незадоволителна е всяка оценка, която е по-малка или равна на 4**.

### Вход

- На първи ред - брой незадоволителни оценки - цяло число в интервала [1...5]
- След това многократно се четат по два реда:
  - Име на задача - текст (низ)

- Оценка - цяло число в интервала [2...6]

## Изход

- Ако Марин стигне до командата "Enough", отпечатайте на 3 реда:
  - "Average score: {средна оценка}"
  - "Number of problems: {броя на всички задачи}"
  - "Last problem: {името на последната задача}"
- Ако получи определения брой незадоволителни оценки:
  - "You need a break, {брой незадоволителни оценки} poor grades."

Средната оценка да бъде форматирана до втория знак след десетичната запетая.

## Примерен вход и изход

Вход	Изход	Обяснения
<pre>(["3", "Money", "6", "Story", "4", "Spring Time", "5", "Bus", "6", "Enough"])</pre>	<pre>Average score: 5.25 Number of problems: 4 Last problem: Bus</pre>	<p>Броя на позволени незадоволителни оценки е 3. Първата задача се казва Money, оценката на Марин е 6.</p> <p>Втората задача е Story, оценката на Марин е 4.</p> <p>Третата задача е Spring Time, оценката на Марин е 5.</p> <p>Четвъртата задача е Bus, оценката на Марин е 6.</p> <p>Следващата команда е Enough, програмата приключва.</p> <p>Средна оценка: <math>21 / 4 = 5.25</math></p> <p>Брой решени задачи: 4</p> <p>Последна задача: Bus</p>
Вход	Изход	Обяснения
<pre>(["2", "Income", "3", "Game Info", "6", "Best Player", "4"])</pre>	<pre>You need a break, 2 poor grades.</pre>	<p>Броят незадоволителни оценки е 2.</p> <p>Първата задача е Income, оценката на Марин е 3.</p> <p>Втората задача е Game Info, оценката на Марин е 6.</p> <p>Третата задача е Best Player, оценката на Марин е 4.</p> <p>Марин достигна допустимия брой незадоволителни оценки, време е за почивка.</p>

## Насоки

1. Направете четири помощни променливи в началото, които да следят броя добри оценки, броя незадоволителни оценки, сумата на всички оценки и коя е последната задача. Първата, втората и третата променливи са с първоначална стойност нула. Четвъртата е с първоначална стойност празен текст.
2. Създайте while цикъл, който продължава докато броя на незадоволителни оценки е по-малък от числото, което сте получили от аргумента. При всяко повторение на цикъла, вземете името на задачата и оценката за нея.
  - а. В случай, че получите команда Enough, намерете средната оценка на Марин и принтирайте нужните съобщения и прекратете цикъла.
3. При всяко повторение на цикъла, прибавете оценката на Марин към сбора на всичките му оценки и увеличете броя за оценките. Ако оценката е по-ниска или равна на 4 увеличете броя за незадоволителни оценки. Презапишете името на последната задача.

4. След цикъла ако броя **незадоволителни оценки** е достигнал **максималните незадоволителни оценки**, принтирайте нужното съобщение.

### 3. Почивка

Джеси е решила да събира пари за екскурзия и иска от вас да ѝ помогнете да разбере **дали ще успее да събере необходимата сума**. Тя спестява или харчи част от парите си всеки ден. Ако иска да похарчи повече от наличните си пари, то тя ще похарчи всичко, което има и ще ѝ останат 0 лева.

#### Вход

От масива се четат:

- Пари нужни за екскурзията - реално число в интервала [1.00.. 25000.00]
- Налични пари - реално число в интервала [0.00... 25000.00]

След това многократно се четат по два реда:

- Вид действие – текст с възможности "spend" и "save".
- Сумата, която ще спести/похарчи - реално число в интервала [0.01... 25000.00]

#### Изход

Функцията трябва да приключи при следните случаи:

- Ако **5 последователни дни** Джеси само харчи, на конзолата да се изпише:
  - "You can't save the money."
  - "{Общ брой изминали дни}"
- Ако Джеси събере парите за почивката на конзолата се изписва:
  - "You saved the money for {общ брой изминали дни} days."

#### Примерен вход и изход

Вход	Изход	Обяснения
(["2000", "1000", "spend", "1200", "save", "2000"])	You saved the money for 2 days.	Пари, нужни за екскурзията: <b>2000</b> Налични пари: <b>1000</b> <b>spend</b> - изваждаме от парите следващото число ( $1000 - 1200 = -200$ , което е по-малко от 0 => налични пари = 0) ~ последователни дни, в които харчи = 1 - общо дни : 1 <b>save</b> - добавяме към парите следващото число ( $0 + 2000 = 2000$ ) ~ последователни дни, в които харчи = 0 - общо дни : 2 Наличните пари ( <b>2000</b> ) >= Пари, нужни за екскурзията ( <b>2000</b> )
(["110", "60", "spend", "10", "spend", "10", "spend", "10", "spend", "10", "spend", "10"])	You can't save the money. 5	Пари, нужни за екскурзията: <b>110</b> Налични пари: <b>60</b> <b>spend</b> – изваждаме от парите следващото число ( $60 - 10 = 50$ ) ~ последователни дни, в които харчи = 1 - общо дни : 1 <b>spend</b> – изваждаме от парите следващото число ( $50 - 10 = 40$ ) ~ последователни дни, в които харчи = 2 - общо дни : 2 <b>spend</b> – изваждаме от парите следващото число ( $40 - 10 = 30$ ) ~ последователни дни, в които харчи = 3

"spend", "10"])		<p>- общо дни : 3</p> <p>spend – изваждаме от парите следващото число (<math>30 - 10 = 20</math>)</p> <p>~ последователни дни, в които харчи = 4</p> <p>- общо дни : 4</p> <p>spend – изваждаме от парите следващото число (<math>20 - 10 = 10</math>)</p> <p>~ последователни дни, в които харчи = 5</p> <p>- общо дни : 5</p> <p>5 последователни дни харчи =&gt; налични пари: 10</p> <p>Наличните пари (10) &lt; Пари, нужни за екскурзията (110)</p>
(["250", "150", "spend", "50", "spend", "50", "save", "100", "save", "100"])	You saved the money for 4 days.	<p>Пари, нужни за екскурзията: 250</p> <p>Налични пари: 150</p> <p>spend - изваждаме от парите следващото число (<math>150 - 50 = 100</math>)</p> <p>~ последователни дни, в които харчи = 1</p> <p>- общо дни : 1</p> <p>spend - изваждаме от парите следващото число (<math>100 - 50 = 50</math>)</p> <p>~ последователни дни, в които харчи = 2</p> <p>- общо дни : 2</p> <p>save - добавяме към парите следващото число (<math>50 + 100 = 150</math>)</p> <p>~ последователни дни, в които харчи = 0</p> <p>- общо дни : 3</p> <p>save - добавяме към парите следващото число (<math>150 + 100 = 250</math>)</p> <p>~ последователни дни, в които харчи = 0</p> <p>- общо дни : 4</p> <p>Наличните пари (250) &gt;= Пари, нужни за екскурзията (250)</p>

## 4. Стъпки

Габи иска да започне здравословен начин на живот и си е поставила за цел да върви **10 000 стъпки всеки ден**. Някои дни обаче е много уморена от работа и ще иска да се прибере преди да постигне целта си. Напишете функция, която **чете от масив по колко стъпки изминава** тя всеки път като излиза през деня и **когато постигне целта си да се изписва "Goal reached! Good job!"** и колко стъпки повече е извървяла "{разликата между стъпките} steps over the goal!"

Ако иска да се **прибере преди това**, тя ще въведе командата "Going home" и **ще въведе стъпките, които е извървяла докато се прибира**. След което, ако не е успяла да постигне целта си, на конзолата трябва да се изпише: "{разликата между стъпките} more steps to reach goal."

### Примерен вход и изход

Вход	Изход	Вход	Изход
(["1000", "1500", "2000", "6500"])	Goal reached! Good job! 1000 steps over the goal!	(["1500", "300", "2500", "3000", "Going home", "200"])	2500 more steps to reach goal.
Вход	Изход	Вход	Изход

(["1500", "3000", "250", "1548", "2000", "Going home", "2000"])	Goal reached! Good job! 298 steps over the goal!	(["125", "250", "4000", "30", "2678", "4682"])	Goal reached! Good job! 1765 steps over the goal!
---	---	---	--

## 5. Монети

Производителите на вендинг машини искали да направят машините си да връщат възможно **най-малко монети ресто**. Напишете функция, която приема **сума - рестото**, което трябва да се върне и изчислява **с колко най-малко монети може да стане това**. Монетите може да са от 2 лева, 1 лев, 50 стотинки, 20 стотинки, 10 стотинки, 5 стотинки, 2 стотинки или 1 стотинка

### Примерен вход и изход

Вход	Изход	Обяснения
(["1.23"])	4	Рестото ни е 1 лев и 23 стотинки. Машината ни го връща с 4 монети: монета от 1 лев, монета от 20 стотинки, монета от 2 стотинки и монета от 1 стотинка.
(["2"])	1	Рестото ни е 2 лева. Машината ни го връща с 1 монета от 2 лева.
(["0.56"])	3	Рестото ни е 56 стотинки. Машината ни го връща с 3 монети: монета от 50 стотинки, монета от 5 стотинки и монета от 1 стотинка.
(["2.73"])	5	Рестото ни е 2 лева и 73 стотинки. Машината ни го връща с 5 монети: монета от 2 лева, монета от 50 стотинки, монета от 20 стотинки, монета от 2 стотинки и монета от 1 стотинка.

## 6. Торта

Поканени сте на 30-ти рожден ден, на който рожденикът черпи с огромна торта. Той обаче не знае **колко парчета могат да си вземат гостите от нея**. Вашата задача е да напишете функция, която изчислява **броя на парчетата**, които гостите са взели, преди тя да свърши. Ще получите **размерите на тортата** (широчина и дължина – **цели числа** в интервала [1...1000]) и след това на всеки ред, до получаване на командата **"STOP"** или **докато не свърши тортата**, броят на парчетата, които гостите вземат от нея.

**Бележка:** Едно парче торта е с размер 1x1 см.

Да се **отпечата** на конзолата **един** от следните редове:

- "{брой парчета} pieces are left." - ако стигнете до **STOP** и не са свършили парчетата торта
- "No more cake left! You need {брой недостигащи парчета} pieces more."

### Примерен вход и изход

Вход	Изход	Обяснения
(["10", "10", "20", "20", "20"])	No more cake left! You need 1 pieces more.	Тортата е с дължина <b>10</b> и широчина <b>10</b> => броят на парчетата = <b>10 * 10 = 100</b> 1-во вземане -> <b>100 - 20 = 80</b> 2-ро вземане -> <b>80 - 20 = 60</b>

<pre>"20", "20", "21"])</pre>		3-то вземане -> $60 - 20 = 40$ 4-то вземане -> $40 - 20 = 20$ 5-то вземане -> $20 - 21 = -1 < 0$ => не остава повече торта, 1 парче не достига
<pre>[("10", "2", "2", "4", "6", "STOP"])</pre>	8 pieces are left.	Тортата е с дължина 10 и ширина 2 => броят на парчетата = $10 * 2 = 20$ 1-во вземане -> $20 - 2 = 18$ 2-ро вземане -> $18 - 4 = 14$ 3-то вземане -> $14 - 6 = 8$ 4-то вземане -> команда STOP => останали парчета: 8

## 7. Преместване

На осемнадесетия си рожден ден на Хосе взел решение, че ще се изнесе да живее на квартира. Опаковал багажа си в **кашони** и намерил подходяща обява за апартамент под наем. Той започва да пренася своя багаж **на части**, защото не може да пренесе целия наведнъж. Има ограничено **свободно пространство** в новото си жилище, където може да разположи вещите, така че мястото да бъде подходящо за живеене.

Напишете **програма**, която изчислява **свободния обем** от жилището на Хосе, който остава след като пренесе багажа си.

Бележка: Един кашон е с точни размери: 1m. x 1m. x 1m.

### Вход

Потребителят въвежда следните данни на отделни редове:

1. Широчина на свободното пространство - цяло число в интервала [1...1000]
2. Дължина на свободното пространство - цяло число в интервала [1...1000]
3. Височина на свободното пространство - цяло число в интервала [1...1000]
4. На следващите редове (до получаване на команда "Done") - брой кашони, които се пренасят в квартирата - цели числа в интервала [1...10000];

Функцията трябва да приключи прочитането на данни при команда "Done" или ако свободното място свърши.

### Изход

Да се **отпечата** на конзолата **един** от следните редове:

- Ако стигнете до командата "Done" и има още свободно място:  
**"{брой свободни куб. метри} Cubic meters left."**
- Ако свободното място свърши преди да е дошла команда "Done":  
**"No more free space! You need {брой недостигащи куб. метри} Cubic meters more."**

### Примерен вход и изход

Вход	Изход	Обяснение
<pre>(["10", "10", "2",</pre>	No more free space! You need 2 Cubic meters more.	$10 * 10 * 2 = 200$ кубични метра налични $20 + 20 + 20 + 20 + 122 = 202$ кубични метра $200 - 202 = 2$ недостигащи кубични метра



"20", "20", "20", "20", "122"])		
(["10", "1", "2", "4", "6", "Done"])	10 Cubic meters left.	$10 * 1 * 2 = 20$ кубични метра налични $4 + 6 = 10$ кубични метра $20 - 10 = 10$ кубични метра