# Упражнения: Масиви

Тествайте решението в Judge: <https://judge.softuni.org/Contests/4144/01-1-Arrays-Basics>.

## Ден от седмицата

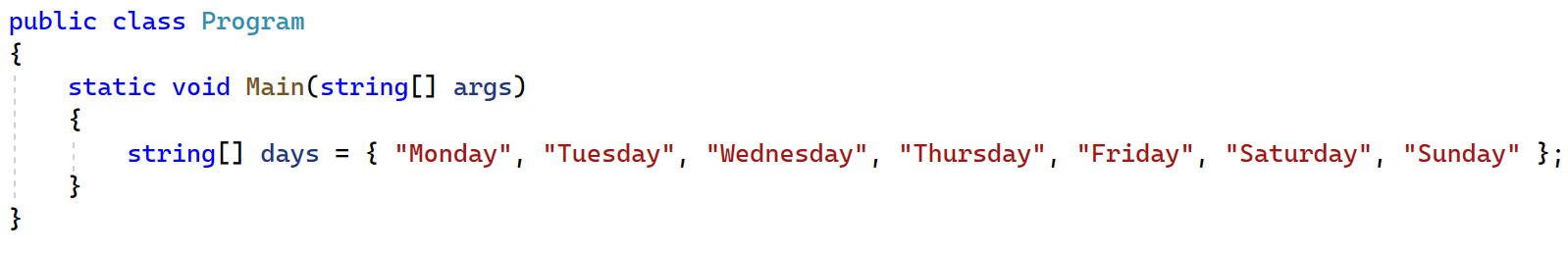
Въведете **ден от седмицата** [1…7] и отпечатайте **името** на деня (на английски) или **"Invalid day!"**

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 1 | Monday |
| 2 | Tuesday |
| 10 | Invalid day! |

### Насоки

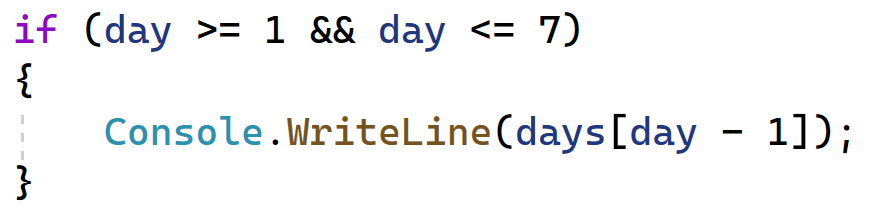
1. Първо създайте **масив** за **дните** от **седмицата**:



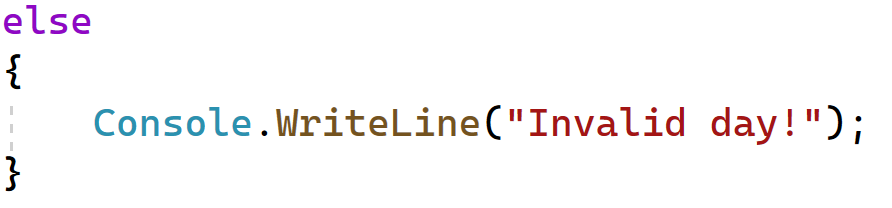
1. След това прочетете числото **day** от конзолата:



1. Създайте проверка дали **day** е **по-голямо или равно** на **1** и **по-малко или равно** на **7**. Ако това е така, отпечатайте съответния **ден от седмицата**:



1. В противен случай отпечатайте **"Invalid day!"**:



## Отпечатване на числа в обратен ред

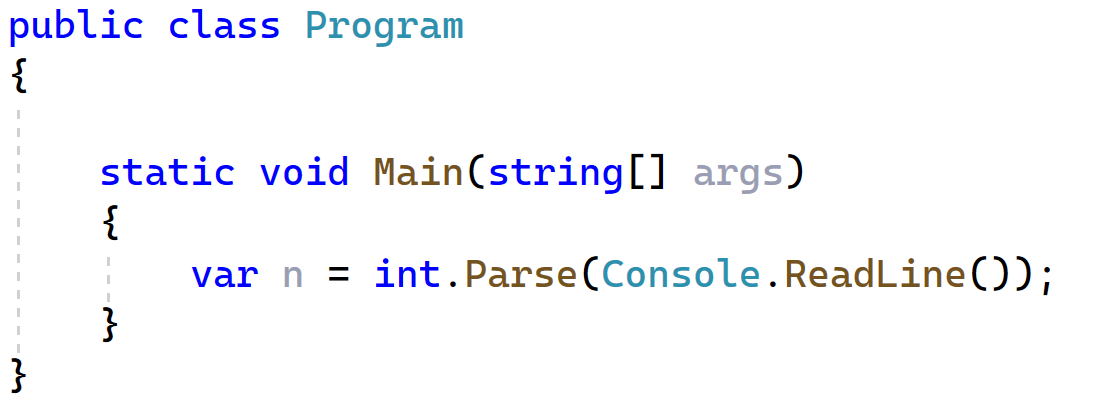
Прочетете **n** на брой числа и ги отпечатайте в **обратен ред**,изписани на един ред, разделени с **интервал**.

### Примери

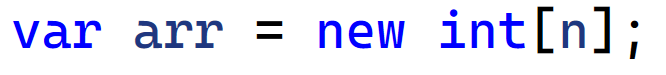
|  |  |  |  |
| --- | --- | --- | --- |
| **Вход** | **Изход** | **Вход** | **Изход** |
| 3  10  20  30 | 30 20 10 | 1  10 | 10 |

### Насоки

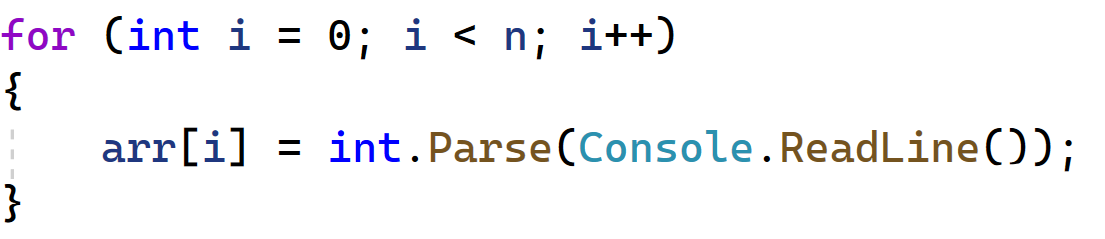
1. Прочетете **n** от конзолата:



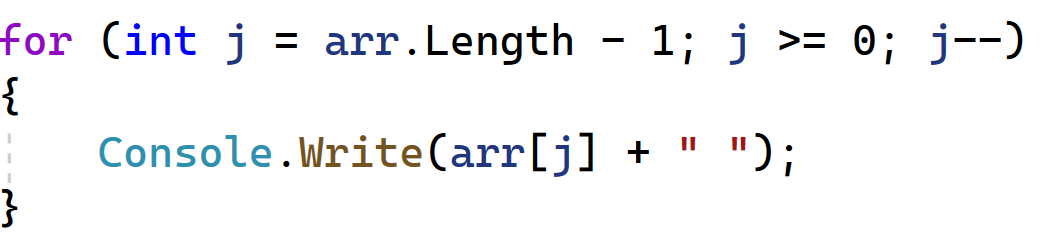
1. **Създавайте масив** от числа с размер n:



1. Прочете числата за масива чрез **for-цикъл**:



1. Отпечатайте масива в **обратен ред**:



## Закръглени числа

Прочетете **масив** от реални числа (разделени с **интервал**). Закръглете ги до **най-близкото число** и ги отпечатайте в следния формат:

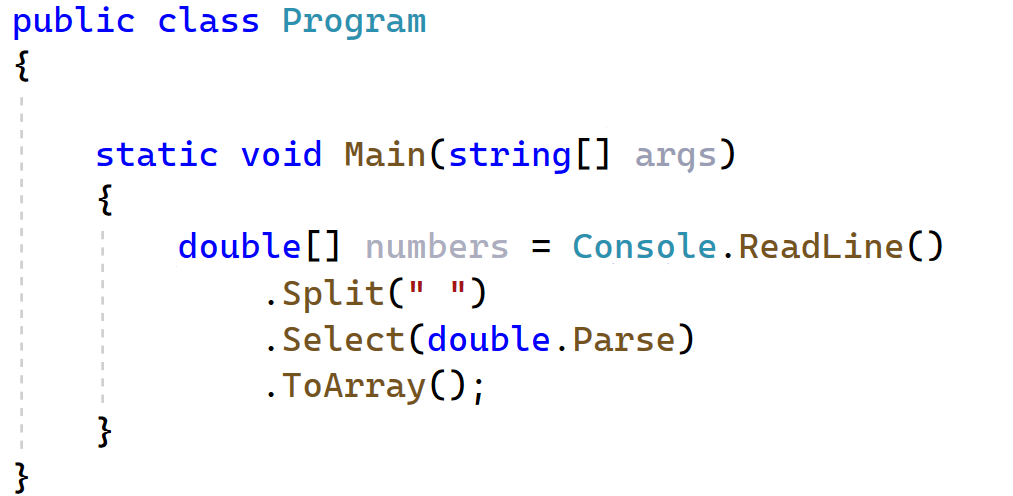
* **"{оригинално число} => {закръглено число}"**

### Примери

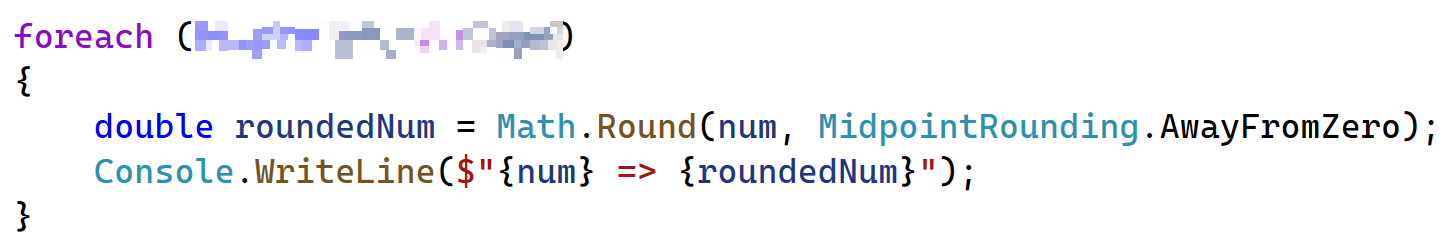
|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 0.9 1.5 2.4 2.5 3.14 | 0.9 => 1  1.5 => 2  2.4 => 2  2.5 => 3  3.14 => 3 |
| -5.01 -1.599 -2.5 -1.50 0 | -5.01 => -5  -1.599 => -2  -2.5 => -3  -1.50 => -2  0 => 0 |

### Насоки

1. **Прочетете** потребителския **вход** и го превърнете в **масив** **numbers**:



1. Създайте **foreach-цикъл**,с който да обходите всички числа от **масива** numbers.
2. На всяка итерация отпечатайте **оригиналното** и **закръгленото** число. Използвайте **Math.Round()** за закръгляне и настройката **MidpointRounding.AwayFromZero**, за да закръглите правилно всички стойности.



## Умножение на числа

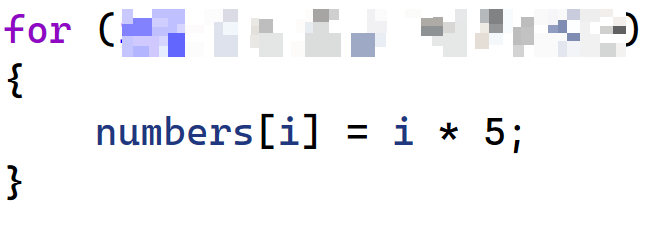
Напишете програма, която чете от конзолата цяло число **n** и създава **масив** от **n елемента**, като всеки елемент е **равен на индекса си**, **умножен** по **5**. Отпечатайте елементите на масива на конзолата, всеки на **нов ред**.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 4 | 0  5  10  15 |

### Насоки

1. Създайте променлива nи масив **numbers** с дължина n.
2. Създайте **for-цикъл.**
3. За всяка итерация на цикъла, присвоете в масива numbers на **индекс** i стойност i**\*5**:



1. Накрая отпечатайте **всяко число от масива** на нов ред.

## Еднаквост

Напишете програма, която чете от конзолата **два масива** от цели числа и проверява дали са **еднакви**. Ако са еднакви, отпечатайте **"Yes"**. В противен случай отпечатайте **"No"**.

### Насоки

1. Създайте два масива с име **nums1** и **nums2**.
2. Създайте **булев флаг** **areEqual** с начална стойност **true**.
3. Създайте **for-цикъл**, чрез който програмата ще сравнява всеки елемент на **nums1** със съответния елемент на **nums2**.
4. Ако има дори един елемент, който не съвпада, трябва да променим стойността на **areEqual** на **false** и да **излезем от цикъла**, за да оптимизираме изпълнението на програмата
5. Ако всички елементи на **nums1** съвпадат със съответните елементи на **nums2**, трябва да изпишем **"Yes"**. В противен случай изписваме **"No"**.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 1 2 3  1 2 3 | Yes |
| 1 2 3  1 3 2 | No |

## Цифри от индекс

Напишете програма, която чете от конзолата цяло число **n** и създава **масив** от **n** елемента, като всеки елемент е равен на **сумата от цифрите на индекса** си. Отпечатайте елементите на масива на конзолата, разделени с **нов ред**.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 5 | 0  1  2  3  4 |
| 11 | 0  1  2  3  4  5  6  7  8  9  1 |

### Насоки

1. Прочетете от конзолата числото n.
2. Създайте масив numbers с размер n.
3. Направете **for-цикъл** от **0** до n-1 включително.
4. Създайте копие на индекса **i** в променлива indexCopy. Ще използваме **копието**, за да намерим **сумата на цифрите**, а в **i** искаме да запазим **оригиналния**, непроменен **индекс**.
5. Изчислете **сумата от цифрите** в променливата indexCopy. Можете да намерите **последната цифра** чрез **модулно деление с 10**, а след това да **разделите** копието на индекса на **10**, за да **премахнете последната цифра** и да можете да вземете **предпоследната**.
6. **Продължете** да извършвате тези **две операции**, докато **indexCopy** стане равно на **0**. Тогава сте сумирали всички цифри.
7. Запазете резултата за сумата в numbers[i] .
8. Накрая отпечатайте всички елементи на масива numbers.

## Обърнат масив от низове

Прочетете **масив от низове**, разделени с **интервал**, **обърнете** го и **отпечатайте** елементите:

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| a b c d e | e d c b a |
| -1 hi ho w | w ho hi -1 |

## Подмножества

Напишете програма, която чете от конзолата **два масива** от цели числа и проверява дали вторият е **подмножество** на **първия**, т.е. **всички елементи** на **втория** масив се съдържат в **първия**. Ако е подмножество, отпечатайте на конзолата "**Yes**". Ако не, отпечатайте "**No**".

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 1 2 3 4 5 6 7 8 9 10  1 2 3 | Yes |
| 1 2 3 4 5 6 7 8 9 10  -1 1 2 3 | No |

## Влак

Ще ви бъде даден **броят на вагоните** на един влак – **n**. На следващите **n** реда ще бъде посочен **броят на хората** във **всеки вагон**. След като прочетете всички входни данни, **отпечатайте броя на хората** във всеки вагон, а на **следващия ред,** **общия брой на хората** във влака.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 3  13  24  8 | 13 24 8  45 |
| 6  3  52  71  13  65  4 | 3 52 71 13 65 4  208 |
| 1  100 | 100  100 |