# Работа с масиви

Тествайте задачите от тази тема в judge: <https://judge.softuni.bg/Contests/2636>

## Статистика на масив

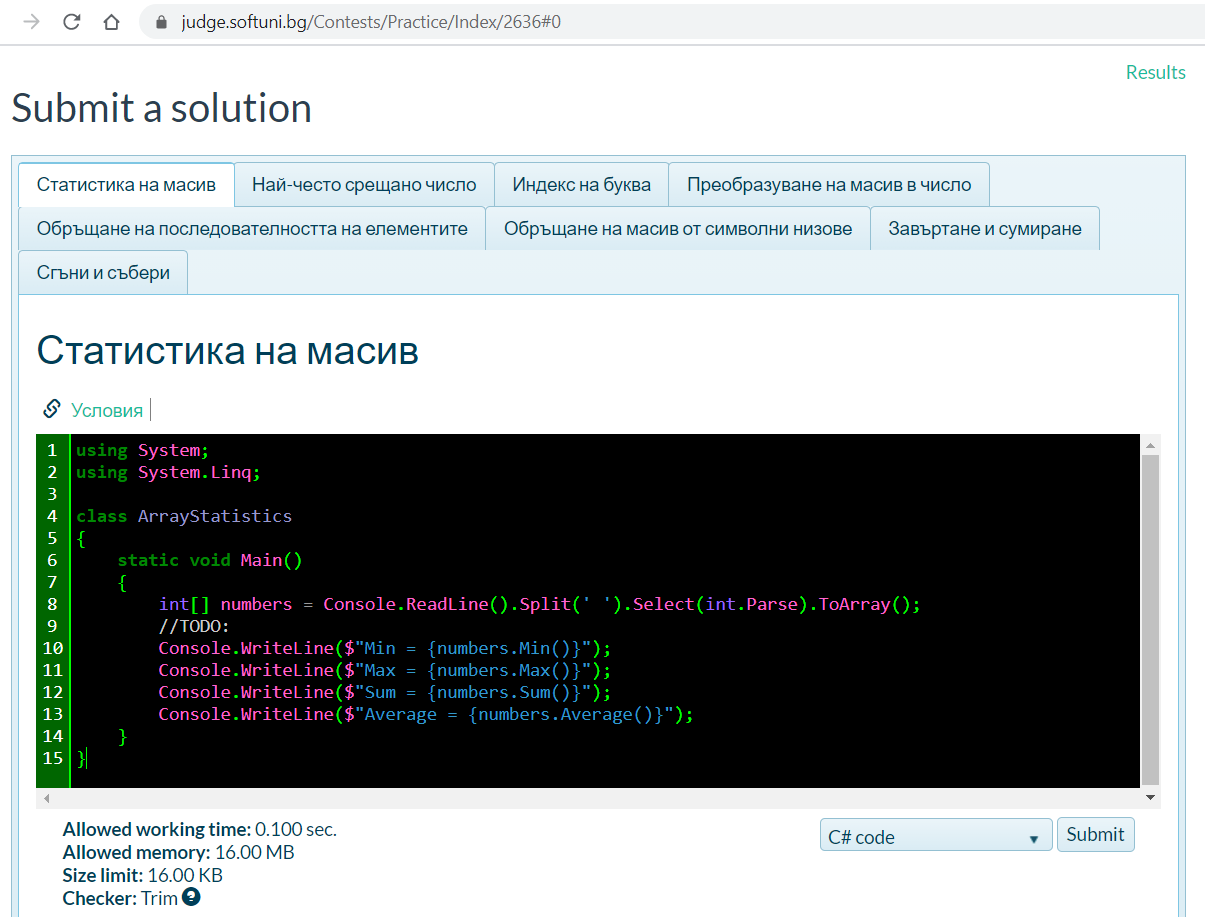
Напишете програма, която получава масив от цели числа (разделени с интервал) и извежда най-малкия елемент, най-големия елемент, сумата на елементите и средната им стойност.

### Примери

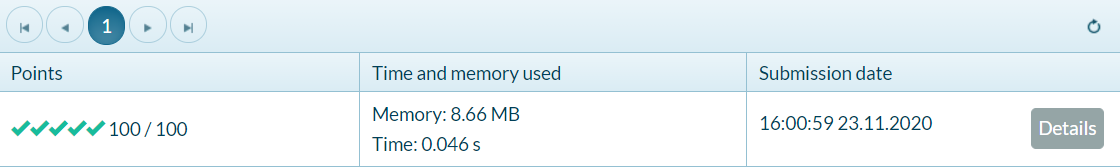
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Вход** | **Изход** |  | **Вход** | **Изход** |
| 2 3 4 5 6 1 | Min = 1  Max = 6  Sum = 21  Average = 3.5 | -1 200 124123 -400 -124214 | Min = -124214  Max = 124123  Sum = -292  Average = -58.4 |

**Проверете** решението си в **judge системата**.

Отворете страницата в judge за този урок: <https://judge.softuni.bg/Contests/2636>. Изберете задачата “**Статистика на масив**”. Копирайте и поставете в тъмното поле **сорс кода**. Натиснете бутона за изпращане **[Submit]**:



Трябва да получите **100 точки** (напълно вярна задача):



## Обръщане на последователността на елементите на масив

Напишете програма, която въвежда масив от цели числа, **Обръща го** иизвежда елементите. Входните данни са **числото** n (брой на елементите) + n цели числа, всяко на отделен ред. Изведете резултата на един ред, за разделител да се ползва интервал

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| **3**  10  20  30 | 30 20 10 |
| **4**  -1  20  99  5 | 5 99 20 -1 |

### Упътване

* Първо, въведете числото n.
* Създайте масив от n цели числа.
* Въведете с цикъл for числата.
* Вместо да обръщате масива, можете просто да изведете елементите му като го обходите от последния до първия

## Обръщане на масив от символни низове

Напишете програма, която да прочете масив от символни низове, обръща масива и печата на неговите елементи. Входът се състои от поредица от низове, разделени с интервал. Отпечатва резултата на един ред с разделител интервал.

### Примери

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

### Упътване

* Въведете масив от символни низове
* Разменете първият елемент (с индекс 0) с последния елемент (с индекс n--1)
* Продължете с тези размени с останалите елементи докато стигнете средата на масива



* Друг, по-кратък подход е да се ползва готовия extension метод .Reverse() от “System.Linq”.

## Индекс на буква

Напишете програма, която позволява да въведете дума с малки букви (lowercase) от конзолата и извежда "индекса" на всяка буква от "масива" с буквите от английската азбука (тоест на колко позиции спрямо 'a' e).

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| abcz | a -> 0  b -> 1  c -> 2  z -> 25 |
| easter | e -> 4  a -> 0  s -> 18  t -> 19  e -> 4  r -> 17 |

### Упътване

* Един низ е всъщност масив от символи. Думата, която сте въвели - също. Така че можете да обходите всички букви по същия начин, както обхождате масив от символи.
* Поредният номер на всяка буква в азбуката можете да получите, като от нея извадите 'a'. Това всъщност изважда от ASCII кода на съответната буква ASCII кода на буквата 'a'.

## Преобразуване на масив в число

Напишете програма, която въвежда масив от цели числа и г преобразува чрез сумиране на съседни двойки елементи, докато се получи едно цяло число. Например, ако имаме 3 елемента {2,10,3}, то събираме първите два и вторите два елемента и получаваме {2+10, 10+3} = {12, 13}, после събираме всички съседни елементи и получаваме obtain {12+13} = {25}.

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Коментари** |
| 2 10 3 | 25 | 2 10 3 🡪 2+10 10+3 🡪 12 13 🡪 12 + 13 🡪 25 |
| 5 0 4 1 2 | 35 | 5 0 4 1 2 🡪 5+0 0+4 4+1 1+2 🡪 5 4 5 3 🡪 5+4 4+5 5+3 🡪 9 9 8 🡪 9+9 9+8 🡪 18 17 🡪 18+17 🡪 35 |
| 1 | 1 | 1 is already condensed to number |

### Упътване

Докато имаме повече от един елемент в масива nums[], повтаряй следното:

* Създай нов масив condensed[] с размер nums.Length-1.
* Събирай числата от nums[] в condensed[]:
  + condensed[i] = nums[i] + nums[i+1]
* nums[] = condensed[]

Процесът е илюстриран по-долу:





1. **Най-често срещано число**

Напишете програма, която намира най-често срещаното число в дадена последователност.

* Числата ще са в интервала [0…65535].
* В случай, че има няколко най-често срещани числа, изведете най-лявото от тях.

**Примери**

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Коментари** |
| **4** 1 1 **4** 2 3 **4 4** 1 2 **4** 9 3 | 4 | Числото **4** е най-често срещаното (среща се 5 пъти) |
| **2 2 2 2** 1 **2 2 2** | 2 | Числото **2** е най-често срещаното (среща се 7 пъти) |
| **7 7 7** 0 2 2 2 0 10 10 10 | 7 | Числата **2**, **7** и **10** имат максимална честота (всяко се среща 3 пъти). Най-лявото е **7**. |

### Упътване

Най-лесно и бързо тази задача се решава така:

1. Правите си целочислен масив **counts** от 65536 елемента - той ще съдържа колко пъти се е повтаряло всяко число.
2. Обхождате първия масив и увеличавате с единица бройката на това число (тоест на елемента с този индекс) в **counts**:  
    var number = nums[i];  
    counts[number]++;
3. Намирате максималната стойност в **counts** - това е колко пъти максимално се е повтаряло това число. Запомняте и индекса, на който се намира - това е числото, което се е повтаряло най-много.

Има и друг начин, който хаби по-малко памет:

1. Създавате си още два масива - **numbers** (за числата, които се повтарят) и **counts** (за това колко пъти се повтарят). Помислете в най-лошия случай от колко елемента трябва да са.
2. Трябва ви и още една променлива **repCount** за броя различни повтарящи се числа. Колко е тя отначало?
3. Обхождате първия масив с различни повтарящите се числа и за всяко число проверявате има ли го в numbers.
   1. ако да - увеличавате бройката на съответния елемент в counts
   2. ако не - значи в numbers[repCount] записвате новото число, а в counts[repCount] - 1, защото сте видели първото срещане на това ново число. После увеличавате repCount с 1, за да отпразнувате случая, че имате още едно число, на което ще броите повторенията. ☺
4. Накрая в counts проверявате кое е било най-повтаряното число, а съответният елемент в numbers ще ви отговори на въпроса кое е това число.

## Завъртане и сумиране

“Завъртане на масив на дясно” означава да преместим неговия последен елемент на първо място: {1, 2, 3} 🡪 {3, 1, 2}.

Напишете програма, която въвежда масив от **n** цели числа (разделени с интервал на един ред) и цяло число **k**, завърта **k** пъти надясно и сумира получените масиви след всяко завъртане както е показано по-долу:

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Коментари** |
| 3 2 4 -1  2 | 3 2 5 6 | rotated1[] = -1 3 2 4  rotated2[] = 4 -1 3 2  sum[] = 3 2 5 6 |
| 1 2 3  1 | 3 1 2 | rotated1[] = 3 1 2  sum[] = 3 1 2 |
| 1 2 3 4 5  3 | 12 10 8 6 9 | rotated1[] = 5 1 2 3 4  rotated2[] = 4 5 1 2 3  rotated3[] = 3 4 5 1 2  sum[] = 12 10 8 6 9 |

### Упътване

* След r завъртания, елементът на позиция i отива на позиция (i + r) % n.
* Масивът sum[] може да бъде изчислен с два вложени цикъла : for r = 1 … k; for i = 0 … n-1.

## Сгъни и събери

Въведете масив от **4\*k** цели числа, сгънете го както е указано по-долу и изведете сумата на горния и долния ред (всеки, съдържащ 2 \* k цели числа):



### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Коментари** |
| 5 **2 3** 6 | 7 9 | 5 6 +  2 3 =  7 9 |
| 1 2 **3 4 5 6** 7 8 | 5 5 13 13 | 2 1 8 7 +  3 4 5 6 =  5 5 13 13 |
| 4 3 -1 **2 5 0 1 9 8**  6 7 -2 | 1 8 4 -1 16 14 | -1 3 4 -2 7 6 +  2 5 0 1 9 8 =  1 8 4 -1 16 14 |

### Упътване

* Създайте първия ред след сгъването: първите **k** числа **обърнати**, последвани от последните **k** числа, също обърнати.
* Създайте втория ред след сгъването, като вземете средните **2\*к** числа
* **Сумирайте** първи и втори ред