## Сортиране на масив

## Зад. 1 Сортиране чрез пряка селекция

Сортирайте един масив (или списък) от елементи с помощта на алгоритъма за пряка селекция (Selection Sort).

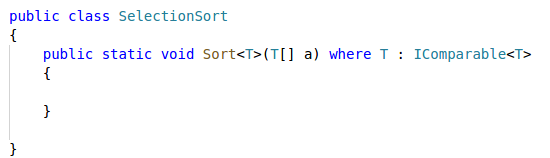
### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 5 4 3 2 1 | 1 2 3 4 5 |

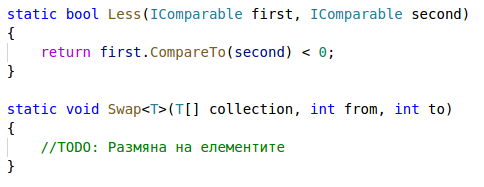
### Подсказки

Първо разгледайте описанието на алгоритъма от [Уикипедия](https://bg.wikipedia.org/wiki/%D0%91%D1%8A%D1%80%D0%B7%D0%BE_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%B0%D0%BD%D0%B5) и визуализацията му във [Visualgo.net](https://visualgo.net/en/sorting?slide=7). Или пък се облегнете на стола и му се насладете под формата на [цигански народен танц](https://www.youtube.com/watch?v=Ns4TPTC8whw), представен от [AlgoRythmics](https://www.youtube.com/channel/UCIqiLefbVHsOAXDAxQJH7Xw). ☺

После създайте клас SelectionSort с един-единствен метод Sort



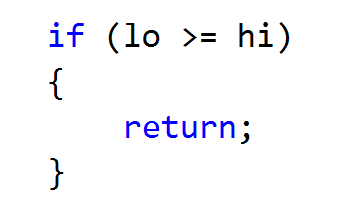
Създайте и два помощни метода - за сравняване на два елемента и за размяна на местата им:



В реализацията на метода Sort() опишете механизъма на самото сортиране. Алгоритъмът накратко е:

* Намираме **най-малката стойност** в масива
* **Разменяме я** с първата несортирана стойност ( в началото това е първия елемент, после е втория и т.н.)
* **Това се повтаря** върху останалата част от масива (без сортираните стойности). Тоест:
  + търсим най-малката стойност от втория елемент до края и я поставяме на втора позиция
  + търсим най-малката стойност от третия елемент до края и я поставяме на трета позиция
  + и т.н., докато стигнем да сравняваме предпоследния и последния елемент

Алгоритъмът може да бъде реализиран с вложени цикли или с рекурсия:

* В първия случай външния цикъл ни служи да укажем от кой елемент да започнем да търсим минимална стойност и позицията, на която ще я поставим, а вътрешния цикъл е за намирането на минималната стойност чрез обхождане до края на масива.
* Във втория случай ще трябва да създадем допълнителен частен рекурсивен метод **Sort()**   
  В него директно реализираме алгоритъма:
  + Ако е останал **само един елемент** в подмасива, то той **е вече сортиран** (което е и дъното на рекурсията).
  + Ако не е, намираме **най-малката стойност** в масива (от позиция **lo** до **hi)**
  + **Разменяме я** с елемента на позиция **lo**
  + Извикваме рекурсивно метода за останалата част от масива (след позиция **lo** до края)

Не забравяйте от публичния метод Sort() да извикате рекурсивния, като му подадете целия масив. ;-)

## Код на задачата:

Програмен файл:

static void Main(string[] args)

{

// Масив

var numbers = new int[] { 9, -9, 8, -8, 7, -7, 6, -6, 5, -5, 4, -4, 3, -3, 2, -2, 1, -1, 0 };

// Печат

Console.WriteLine(string.Join(" ",numbers));

//string.Join - връща стринг и обединява елементите на определен масив или членовете на колекция, като използва посочения разделител между всеки елемент или член.

// Сортиране по метода на пряката селекция = O(N ^ 2)

Sort\_Selection.Selection(numbers);

// Печат

Console.WriteLine(string.Join(" ", numbers));

//Добавете метод в класа, с който да се отпечатат от най-голямото към най-малкото и го извикайте в програмата

Клас:

class Sort\_Selection

{//IComparable - метод за сравнение, т.е where T : IComparable - списък от обекти, достъпни по индекс, които ще бъдат сравнявани

public static void Selection<T>(T[] elements) where T : IComparable

{

for (int i = 0; i < elements.Length; i++)

{

int min = i;

for (int j = i + 1; j < elements.Length; j++)

{

if (IsLess(elements[j], elements[min]))

{

min = j;

}

}

Swap(elements, min, i);

}

}

public static void Swap<T>(T[] array, int first, int second)

{

T temp = array[first];

array[first] = array[second];

array[second] = temp;

}

//метод, който връща булева стойност, проверява дали втория елемент е по малък от първия

public static bool IsLess(IComparable first, IComparable second)

{

return first.CompareTo(second) < 0; // връща true, ако първия елемент е по-малък от втория

}

//CompareTo - метод, който сравнява действителната стойност и връща 0, ако и двете числа са равни,

//-1, ако извикването на обект (first) е по-малко от предаването на обект (second)

//и 1, ако извикването на обект е по-голямо от предаването на обект.

}

}

## Зад. 2 Кросово бягане

Миналата седмица във вашия град бе проведено масово кросово бягане. Тъй като участниците са били наистина много, те са били разпределени да се състезават в два различни дни, по един и същ маршрут. На финиша е имало съдии, които са записали времето на пристигането на всеки състезател в секунди. На вас се пада честта да обедините тези два списъка в един и да оформите общата класация. Напишете програма, която да ви улесни в изпълнението на задачата.

### Вход

* Входните данни трябва да се прочетат от конзолата.
* На първия ред ще има цели числа, подредени в ненамаляващ ред и отделени едно от друго с интервал. Това са времената на състезателите, участвали в кроса през първия ден.
* На втория ред ще са времената на състезателите, участвали в кроса през втория ден. Те са в същия формат, както числата от първия ред, но не е сигурно, че са същия брой.
* Входните данни винаги ще са валидни и в описания формат. Не е необходимо да бъдат изрично проверявани.

### Изход

* Изходните данни (сортираният масив с класацията на всички състезатели, участвали в кроса, подредени в ненамаляващ ред) трябва да бъдат отпечатани на конзолата.

### Пример

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Коментар** |
| **956 989 1037 1095**  948 992 1025 1062 1160 | 948 **956** **989** 992 1025 **1037** 1062 **1095** 1160 | За яснота участниците от първия ден са **удебелени** |

**Забележка!!! Задачата ще решите, като използвате кода на класа от предходната задача**!