**Сортиране чрез броене(Counting Sort)**

1. **Какво представлява сортирането чрез броене(Counting sort) ?**

* Както името му подсказва, сортирането чрез броене работи, като брои броя на срещанията на всеки елемент поотделно в листа. Помощен/frequency масив пази тези срещания и картографира стойностите на отделните елементи с индексите на масива. Накрая алгоритъма прави итерация над другия/помощния масив, докато сортира първоначалния масив.

1. **Как работи този алгоритъм ?**

* За да разберем как работи алгоритъма, нека вземем един несортиран масив:

****

* **Graphical user interface, application

  Description automatically generated with medium confidenceСтъпка 1**: В този случай, най-големия елемент в масива е 9. Като друг масив е инициализиран с размер [max(numbers[ ] + 1)]. След това масива е обходен, за да пази броят на срещанията на всеки отделен елемент в този масив.

* **Стъпка 2**: В тази стъпка, масива(the frequency array) е обходен, за да се сдобием с информацията за всеки елемент поотделно и неговите срещания, които по-късно се използват, за да се сортира масива.

Diagram

Description automatically generated

1. **Как да имплементираме сортиране чрез броене в C#**

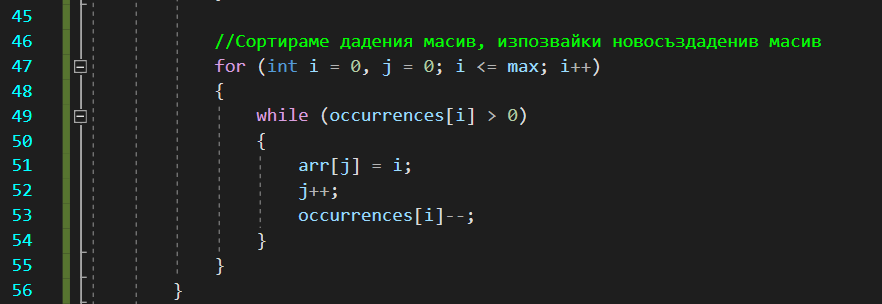
* 1. Нека създадем метод CountingSort. И да намерим най-големия елемент в масива.

Text

Description automatically generated

* Text

  Description automatically generated2. Създаваме нов масив, който да пази броя на срещанията на всеки елемент.
* 3. Сортираме масива.

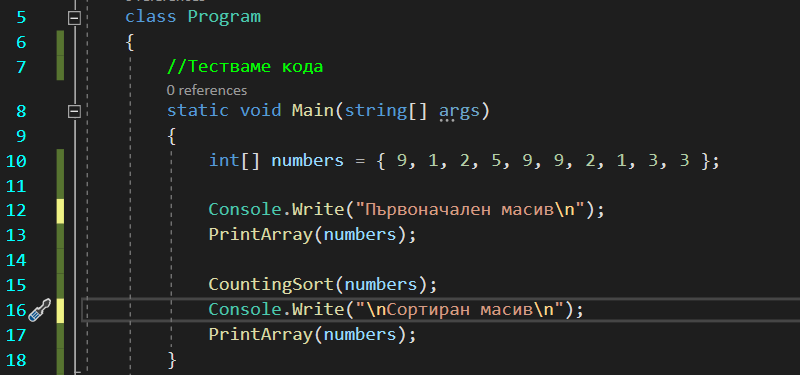


* + - 4. Създаваме си нов метод, който да изпечата масива.

Text

Description automatically generated

* + - 5. Тестваме програмата.



* + - Text

      Description automatically generated6. Като резултат получаваме:

1. **Памет и времева сложност на алгоритъма**
   * + Алгоритъма за сортиране чрез броене изисква допълнителен масив с размер k (max element + 1). Затова паметта е: **O(k)**.
     + Най-добър случай – появява се, когато елементите на масива имат диапазон **k**, който е равен на **1**. В този случай алгоритъма приема линейно време, като времевата сложност става: **O(1 + n) или O(n)**.
     + Среден случай – среща се, когато избираме произволни стойности, напр. от 1 до n. В този случай приемаме, че имаме масив с размер **n** и стойността на най-големия елемент е **k,** алгоритъма има**: O(n+k), като среден случай времева сложност.**
     + **Най-лошия случай -става, когато диапазона k на елементите е значително по-голям от останалите елементи.**

1. **Предимства на сортирането чрез броене**

* Алгоритъмът е **много ефективен** за сортиране на малък диапазон от числа. **Стабилен** е, което означава, че елементите с еднакви стойности ще запазят първоначалния си ред след сортиране.

1. **Недостатъци на сортирането чрез броене**

* Алгоритъмът **не е подходящ** за голям диапазон от стойности, защото ефективността на алгоритъма намалява с нарастването на входните данни.
* Сортирането чрез броене **не е** **in-place** сортиращ алгоритъм, защото изисква допълнително място.
* И накрая алгоритъма може да е по-бавен от другите алгоритми за сортиране, които използват **„разделяй и владей“** подходи като: **сортиране чрез сливане(merge sort) и бързо сортиране(quicksort)** за големи масиви.

1. **Обобщение(Summary)**
   * Сортиране чрез броене (counting sort) е много ефективен алгоритъм за сортиране (онагледяване).
   * Сортира малки числа чрез броене на техните срещания.
   * Не е базиран на сравнение.
   * Най-добър, среден и най-лош случай: **O (n + k).**
   * **k** е диапазонът на сортираните числа.
   * Памет**: O (n + k)**.
   * Място: **O(k)**
   * Стабилен: Да
   * Метод: Броене