Масивът е поредица от еднотипни елементи с общо име и номерация. Елементите се разполагат последователно в паметта, а индексацията започва от 0. Последния елемент има индекс равен на дължината на масива – 1. Достъпа до елементите се осъществява с името на масива и индекса на елемента, записан в квадратни скоби, ***Пример***: numbers[3]

Масивът е статична(с фиксиран размер) структура - към него не може да се добават или махат елементи. Броя на елементите (дължината на масива) може да се провери със свойството Length, ***Пример***: numbers.Length

Основни действия с масив

```
//Деклариране и инициализация на масив от 10 цели числа
int[] numbers = new int[10];

//Въвеждане на стойности на елементите
for (int i = 0; i < numbers.Length; i++)
{
   Console.Write("Въведи число: ");
   numbers[i] = int.Parse(Console.ReadLine());
}

//Отпечатване на елементите на отделни редове
for (int i = 0; i < numbers.Length; i++)
{
   Console.WriteLine(numbers[i]);
}

//Отпечатване на елементите на един ред разделени със ,
Console.WriteLine(string.Join(", ", numbers));
```

Декларирането на масива може да стане в началото на програмата, а инициализацията на по-късен етап. Инициализацията става с ключовата дума **new** и в този момент в паметта за заделя необходимото място за указания брой елементи и те се запълват с неутрална за конкретния тип данни стойност. За числовите типове като **int** и **double** това е **нула**, а ако масива е от **стрингове** – **празен низ** (“”).

Пример за дефиниране и деклариране на различни места в програмата:

```
string[] names;
…
names = new string[10];
```

Основни задачи при работа с масиви – търсене и сортиране

Обхождането на масив представлява преминаване през всеки един от елементите с цел достъп до стойността му, като това се прави най-често с цикъл for минавайки по всички индекси. ***Пример***:

```
for (int i = 0; i < numbers.Length; i++)
{
   действие с текущия елемент numbers[i]
}
```

Примери за задачи свързани с обхождане:

- отпечатване на елементите, които отговарят на някакво условие
- модифициране на елементите, които отговарят на някакво условие
- намиране броя на елементите, които отговарят на дадено условие
- намиране на най-малко/най-голямо число
- сумиране/произведение/средно аритметично на елементите

Примери за задачи свързани със сортиране:

- подреждане по азбучен/обратен на азбучния ред
- подреждане в нарастващ/намаляващ ред

Известни алгоритми за сортиране:

- метод на „мехурчето“ (buble sort)
- сортиране чрез селекция/размяна (selection sort)
- сортиране чрез вмъкване (insertion sort)
- бързо сортиране (quick sort)
- сортиране чрез сливане (merge sort)
- heap sort
- counting sort
- bucket sort
- tree sort

Допълнителна информация от Wikipedia

Можете да видите интересна визуализация на повечето от изброените медоди на адрес: https://visualgo.net/en/sorting

Въвеждане на елементите на масив от 1 ред (без цикъл)

Стандартно въвеждането(прочитането) на елементите на масив става с помощта на цикъл, като при всяка итерация потребителя въвежда по една стойност, която се прочита и записва като стойност на съответния елемент от масива.

В съвременните езици има допълнително разработени методи, които ни дават възможност да работим с колекции от елементи. В C# те се намират в библиотеката System.Linq

Методите дефинирани в System.Linq ни дават възможност за търсене, броене, прилагане на действие към всеки елемент от колекция, намиране на минимален/максимален елемент, средно аритметично, филтриране по критерии, сортиране и много други.

Две от функциите(методите), които ни трябват от тази библиотека за да може да прочетем масив от 1 ред са **Select()** и **ToArray().**

Пример: Прочитане на цели числа разделени с интервал и записването им в масив с име numbers. В случая не се определя размера на масива с new, а в момента на изпълнението на функцията ToArray().

```
using System.Linq;

    …
int[] numbers = Console.ReadLine()  // Прочитане на 1 текстов ред   "2 3 4"
      .Split()                      // Разделяне на отделни низове "2" "3" "4"
      .Select(int.Parse)            // Конвертиране на всеки елемент в число  2 3 4
      .ToArray();                   // записване на елементите в масив [2,3,4]
```

Тук имаме навързани няколко метода, като всеки следващ взима резултата от предишния и го обработва. Възможно е да възникне грешка при Split() или Select() ако входните данни не са адекватни и тогава масива няма да бъде създаден, а програмата ще хвърли грешка Exception.

Независимо от начина на въвеждане на елементите на масива, винаги е добре да се предвиждат, прихващат и обработват възможните грешки.

Пример за филтриране – ако numbers е горния масив с Where() може да вземем само елементите които отговарят на някакво условие (в случая да са четни). В резултат се получава колекция от елементи, които отново с ToArray() записваме в нов масив с име result.

```
int[] result = numbers
          .Where(x => x % 2 == 0)
          .ToArray();
```

Изразът x => x % 2 == 0 представлява така наречената **ламбда** функция (анонимна функция). В случая тя приема елемент x като входен параметър и връща като резултат true или false в зависимост от това дали x е четно или нечетно.