

Introducción a C# .NET

Módulo 3

Arreglos

Introducción

Hasta ahora se han aprendido un montón de características de las variables, entre ellas, que siempre había que saber con antelación el número de variables que el programa iba a necesitar.

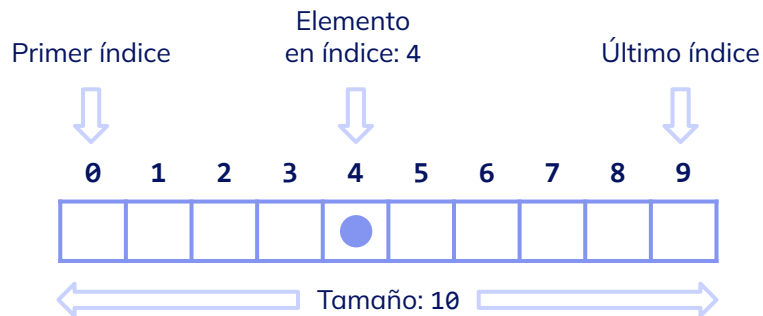
Sin embargo, habrá situaciones en las que no sea posible determinar este número hasta que el programa no se esté ejecutando. Pues bien, para solucionar estas dificultades contamos con las **estructuras de arreglos, *arrays*** en inglés.



Arreglos de una dimensión

Un **array** es una variable que es capaz de almacenar varios valores en forma simultánea. Cada uno de estos valores se identifica mediante un número que se denomina “índice”.

Para acceder al primer elemento del **array** se usa el índice cero, para el segundo el índice uno, para el tercero el índice dos, y así sucesivamente.



Al arreglo se lo declara de la siguiente forma:

```
tipo[] variable;
```

Es muy parecido a cómo se declara una variable normal, sólo que hay que poner corchetes detrás del tipo de dato.

En C# los arrays son objetos derivados de la clase `System.Array`. Por lo tanto, y esto es muy importante, cuando declaramos un array en C# este aún no se habrá creado en la memoria de la computadora, en consecuencia, antes de poder usarlos habrá que instanciarlos, como si fuera cualquier otro objeto.

Veamos un breve ejemplo:

```
string[] nombres; // Declaración del array  
nombres = new string[3]; // Instanciación del array
```

En efecto, el array **nombres** será utilizable únicamente a partir de su instanciación, y podrá tener un máximo de tres elementos, accedidos desde los índices 0, 1 y 2.

Nótese que índice del array comienza de CERO (0) y NO de uno (1), por lo cual el índice del último elemento es siempre el `TotalDeElementos` menos uno ($3-1=2$)

Operaciones de recorrida de arreglos

Un array es un conjunto de objetos, ordenado y de tamaño fijo.

Para acceder a cualquier elemento de este conjunto se aplica el operador postfijo `[]` sobre la tabla para indicar entre corchetes la posición que ocupa el objeto al que se desea acceder dentro del conjunto. Es decir, este operador se usa así:

```
[<posiciónElemento>]
```

Un ejemplo de su uso, en el que se asigna al elemento que ocupa la posición 3 en una tabla de nombre `arrayPrueba`, el valor del elemento que ocupa la posición 18 de dicha tabla es el siguiente:

```
arrayPrueba[3] = arrayPrueba[18];
```

Recorrido del array

Podemos recorrer el array de la siguiente forma:

```
for(i=0; i < losValores.Length; i++)  
{  
    Console.WriteLine(losValores[i]);  
}
```

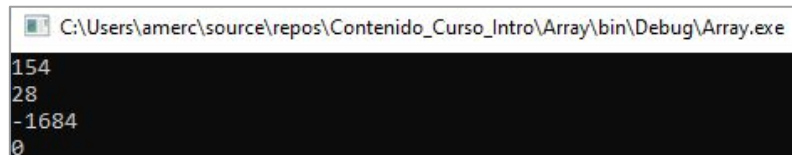
En la próxima slide veremos el código completo del programa, que crea el arreglo, lo instancia, lo llena con números y lo recorre.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Array
{
    class Arreglo
    {
        static void Main(string[] args)
        {
            int[] losValores = new int[4];
            losValores[0] = 154;
            losValores[1] = 28;
            losValores[2] = -1684;
            losValores[3] = 0;
```

```
...
        for (int i = 0; i < losValores.Length; i++)
        {
            Console.WriteLine(losValores[i]);
        }
        Console.ReadKey();
    }
}
```



```
C:\Users\amerc\source\repos\Contenido_Curso_Intro\Array\bin\Debug\Array.exe
154
28
-1684
0
```


Si quisiéramos recorrer el array de atrás hacia adelante lo haríamos de la siguiente manera:

```
for (i=losValores.Length-1; i>=0; i--)  
{  
    Console.WriteLine(losValores[i]);  
}
```

No es necesario utilizar todos los elementos de un arreglo, por lo que, al trabajar con ellos, se puede utilizar una variable entera adicional que nos indique el número de datos que realmente estamos utilizando.

El tamaño del arreglo nos dice cuanta memoria se ha reservado para almacenar datos del mismo tipo de dato, no cuántos datos del mismo tipo tenemos realmente en el arreglo.

Ejemplo

Suma de todos los elementos de un vector:

```
int suma=0;  
int n=losValores.Length;  
  
for (int j=0; j < n; j++)  
{  
    suma= suma + losValores[j];  
}  
Console.WriteLine(suma);
```

Búsqueda del número máximo y mínimo de un arreglo

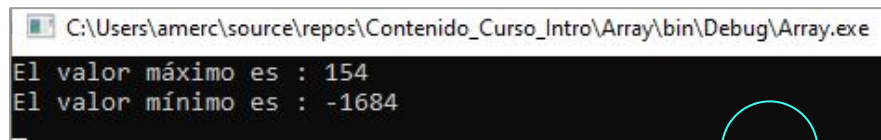
```
int maximo = losValores[0], minimo = losValores[0];

// Recorremos desde la posición 1, ya que el valor de la posición 0 se asignó a maximo y a minimo
for (int a = 1; a < losValores.Length; a++)
{
    if (losValores[a] > maximo)
    {
        maximo = losValores[a];
    }
    if (losValores[a] < minimo)
    {
        minimo = losValores[a];
    }
}

Console.WriteLine("El valor máximo es : " + maximo);
Console.WriteLine("El valor mínimo es : " + minimo);
```



Y el resultado de su ejecución:



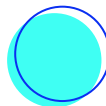
```
C:\Users\amerc\source\repos\Contenido_Curso_Intro\Array\bin\Debug\Array.exe
El valor máximo es : 154
El valor mínimo es : -1684
```

Contar cuántas veces aparece el número 10

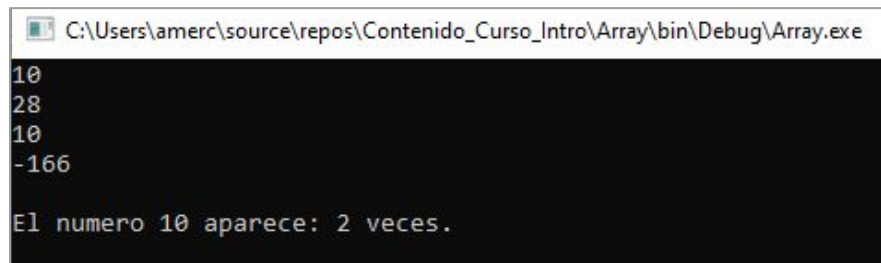
```
int contador = 0;

for (int a = 0; a < osValores.Length; a++)
{
    if (losValores[a] == 10)
    {
        contador++;
    }
}

Console.WriteLine("El numero 10 aparece: " + contador + " veces.");
```



Y el resultado de su ejecución:



```
C:\Users\amerc\source\repos\Contenido_Curso_Intro\Array\bin\Debug\Array.exe
10
28
10
-166

El numero 10 aparece: 2 veces.
```



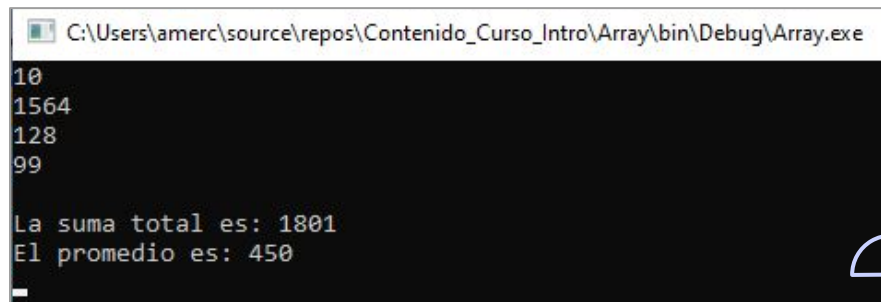
Totalizar el contenido de un arreglo numérico y calcular el promedio

```
int suma = 0;

for (int a = 0; a < losValores.Length; a++)
{
    suma += losValores[a];
}

Console.WriteLine("La suma total es: " + suma);
Console.WriteLine("El promedio es: " + suma / losValores.Length);
```

Y el resultado de su ejecución:



```
C:\Users\amerc\source\repos\Contenido_Curso_Intro\Array\bin\Debug\Array.exe
10
1564
128
99
La suma total es: 1801
El promedio es: 450
_
```

**¡Sigamos
trabajando!**

