

Programación II

C

Clase 6 - Parte II

Maximiliano Neiner

Modificada por Federico Dávila

Temas a Tratar

- ❖ Arrays
- ❖ Manejo de Cadenas

Temas a Tratar

Arrays

- Descripción.
- Unidimensionales.
- Multidimensionales.

Manejo de Cadenas

Arrays (1/2)

- ❖ Un Array puede ser unidimensional, multidimensional o anidado (jagged).
- ❖ El valor por defecto de Array de elementos numéricos (**value types**) se establece en **cero**, mientras que los objetos (**reference types**) se establece en **null**.
- ❖ Un Array anidado es un Array de Arrays, por lo tanto, al ser tipos por referencia se inicializan en **null**.

Arrays (2/2)

- ❖ Los Arrays en C# son base-cero: un Array con ‘n’ elementos está indexado de 0 a n-1.
- ❖ Los elementos de un Array pueden ser de cualquier tipo, incluso de tipo Array (*Clase*).
- ❖ Los Arrays son **reference type**, heredan de la clase abstracta System.Array.
- ❖ Implementan la *interfaz IEnumerable* por lo tanto se pueden iterar usando foreach.

Temas a Tratar

Arrays

- Descripción.
- Unidimensionales.
 - Sintaxis.
 - La Clase System.Array.
- Multidimensionales.

Manejo de Cadenas

Temas a Tratar

Arrays

- Descripción.
- Unidimensionales.
 - Sintaxis.
 - La Clase System.Array.
- Multidimensionales.

Manejo de Cadenas

Sintaxis - Declaración

[acceso] **Tipo [] nombre_Array;**

- En la declaración de un Array en C# los corchetes se colocan detrás del tipo y no detrás de la variable.
- Esta pequeña diferencia sintáctica se debe a una importante diferencia semántica:
 - Aquí los Arrays son objetos derivados de la clase System.Array.
 - Por lo tanto, cuando se declara un Array en C# este aún no se habrá creado, es decir, no se habrá reservado aún memoria para él.
 - En consecuencia, los Arrays de C# son todos dinámicos, y antes de poder usarlos habrá que instanciarlos, como si fuera cualquier otro objeto.

Sintaxis - Instanciación

```
nombre_Array = new Tipo[CANT_ELEMENTOS];
```

- Los Arrays se pueden declarar e instanciar en una misma línea.

```
[acceso] Tipo [] nombre_Array = new Tipo[CANT_ELEMENTOS];
```

Temas a Tratar

Arrays

- Descripción.
- Unidimensionales.
 - Sintaxis.
 - La Clase System.Array.
- Multidimensionales.

Manejo de Cadenas

Temas a Tratar

Arrays

- Descripción.
- Unidimensionales.
 - Sintaxis.
 - La Clase System.Array.
 - Métodos y Propiedades.
- Multidimensionales.

Manejo de Cadenas

Métodos (1/2)

CopyTo:

- Copia todos los elementos del Array unidimensional actual al Array unidimensional especificado, desde el índice indicado.

Resize:

- Redimensiona un Array.

GetLength:

- Obtiene el número de elementos para una dimensión determinada del Array.

Métodos (2/2)

GetValue/SetValue:

- Obtiene o establece el valor de uno o varios elementos del Array.

Initialize:

- Inicializa todos los elementos del Array, llamando al constructor por defecto.

Propiedades

Length: Obtiene la cantidad total de elementos de todas las dimensiones del Array.

Rank: Obtiene el número de dimensiones del Array.

Temas a Tratar

Arrays

- Descripción.
- Unidimensionales.
- Multidimensionales.

Manejo de Cadenas

Temas a Tratar

Arrays

- Descripción.
- Unidimensionales.
- Multidimensionales.
 - Sintaxis.

Manejo de Cadenas

Sintaxis - Declaración

```
[acceso] Tipo [ , ] nombre_Array;
```

Sintaxis - Instanciación

```
nombre_Array = new Tipo[FILAS, COLUMNAS];
```

Temas a Tratar

❖ Arrays

❖ Manejo de Cadenas

- La clase **System.String**.
 - Métodos estáticos.
 - Métodos de instancia.
 - Propiedades
 - **StringBuilder**

Métodos Estáticos

Compare:

- Compara entre dos cadenas y devuelve tres valores posibles:
 - Cero, si $\text{cad1} = \text{cad2}$.
 - Positivo, si $\text{cad1} > \text{cad2}$
 - Negativo, si $\text{cad1} < \text{cad2}$

Concat:

- Concatena una o más cadenas.

Copy:

- Copia una cadena dentro de otra.

Temas a Tratar

❖ Arrays

❖ Manejo de Cadenas

- La clase **System.String**.
 - Métodos estáticos.
 - Métodos de instancia.
 - Propiedades
 - **StringBuilder**

Métodos de Instancia (1/3)

- ❖ **CompareTo:** Ídem método estático Compare().
- ❖ **Contains:** Indica si una cadena está o no contenida en el objeto actual.
- ❖ **CopyTo:** Ídem método estático Copy().
- ❖ **EndsWith/StartsWith:** Determina si el final/principio de la cadena contenida por el objeto coincide o no con la cadena pasada como argumento.
- ❖ **IndexOf/LastIndexOf:** Devuelve el índice de la primera/última ocurrencia de la cadena especificada.

Métodos de Instancia (2/3)

- ❖ **Insert:** Inserta una cadena a partir de una posición determinada.
- ❖ **Remove:** Borra todos los caracteres a partir del índice especificado.
- ❖ **Replace:** Reemplaza todas las ocurrencias de una cadena especificada en la instancia por otra cadena.
- ❖ **Substring:** Retorna una sub-cadena, a partir de un índice especificado y con una determinada longitud.

Métodos de Instancia (3/3)

- ❖ **ToLower/ToUpper:** Devuelve una copia de la cadena convertida a minúscula/mayúscula.
- ❖ **TrimEnd/TrimStart:** Remueve todas las ocurrencias de un conjunto de caracteres especificado en un Array, desde el final/principio de la instancia.
- ❖ **Trim:** Remueve todos los espacios en blanco del principio y del final de la instancia.

Temas a Tratar

❖ Arrays

❖ Manejo de Cadenas

- La clase **System.String**.
 - Métodos estáticos.
 - Métodos de instancia.
 - **Propiedades**
 - **StringBuilder**

Propiedades

- Chars: Obtiene el carácter situado en una posición de carácter especificada en el objeto System.String actual.
- Length: Retorna la cantidad de caracteres que posee la instancia.

Temas a Tratar

❖ Arrays

❖ Manejo de Cadenas

- La clase **System.String**.
 - Métodos estáticos.
 - Métodos de instancia.
 - Propiedades
 - **StringBuilder**

StringBuilder

- ❖ Representa una cadena de caracteres mutable.
- ❖ Esta clase no puede heredarse.
- ❖ Provee métodos y propiedades para operar con Strings.
- ❖ Mediante la propiedad Capacity podemos obtener o establecer el número máximo de caracteres que puede contener la memoria asignada para una instancia en particular. Una vez superada esa capacidad, el espacio utilizado en memoria se asignará dinámicamente (proceso más costoso).

StringBuilder - Uso

```
StringBuilder sb = new StringBuilder();
```

```
sb.Append("Algo de texto.");
```

```
sb.AppendLine("Algo de texto con salto de línea.");
```

```
sb.AppendFormat("{0} Resultado: {1} - {2}", "Gol", 1, 0);
```

```
sb.ToString();
```