# *Tipificación de Datos en C#.*

## Numéricos Enteros:

| Descripción | Nombre C# | Nombre .Net | Precisión | Valor Mínimo | Valor Máximo | Espacio de Almacenamiento |
|---|---|---|---|---|---|---|
| Entero Octeto | sbyte | System.SByte | 100% | -128 | 127 | 1 Byte (8 bits) |
| Entero Octeto Sin Signo | byte | System.Byte | 100% | 0 | 255 | 1 Byte (8 bits) |
| Entero Corto | short | System.Int16 | 100% | -32.768 | 32.767 | 2 Bytes (16 bits) |
| Entero Corto Sin Signo | ushort | System.UInt16 | 100% | 0 | 65.535 | 2 Bytes (16 bits) |
| Entero (por defecto) | int | System.Int32 | 100% | -2.147.483.648 | 2.147.483.647 | 4 Bytes (32 bits) |
| Entero Sin Signo | uint | System.UInt32 | 100% | 0 | 4.294.967.295 | 4 Bytes (32 bits) |
| Entero Largo | long | System.Int64 | 100% | -9.223.372.036.854.775.808 | 9.223.372.036.854.775.807 | 8 Bytes (64 bits) |
| Entero Largo Sin Signo | ulong | System.UInt64 | 100% | 0 | 18.446.744.073.709.551.615 | 8 Bytes (64 bits) |

**Características:**

- Su selección se determina por el **rango** del número entero a almacenar o representar, buscando minimizar el espacio de almacenamiento.
- Todos presentan precisión del 100%. Es decir, no existirá error o diferencia entre el valor ingresado y el efectivamente almacenado/representado.
- Se codifican en código binario natural y, los números negativos, en complemento a dos.
- Las variables o constantes numéricas enteras se inicializan por defecto y automáticamente en 0.
- **Interpretación de constantes:**
  - Por defecto, cualquier constante numérica entera, se asume **int**. Por ejemplo: la constante **28** será interpretada como **int**.
  - Las constantes numéricas enteras con el identificador **U**, se asumirán **uint**. Por ejemplo: la constante **28U** será interpretada como **uint**.
  - Las constantes numéricas enteras con el identificador **L**, se asumirán **long**. Por ejemplo: la constante **28L** será interpretada como **long**.
  - Las constantes numéricas enteras con el identificador **UL**, se asumirán **ulong**. Por ejemplo: la constante **28UL** será interpretada como **ulong**.

## Numéricos Reales:

| Descripción | Nombre C# | Nombre .Net | Precisión | Rango de Valores Admitidos | Espacio de Almacenamiento |
|---|---|---|---|---|---|
| Punto Flotante de Precisión Simple | float | System.Single | **La más baja.** En el orden del **7°** dígito decimal de relevancia. | Aprox. desde $-3{,}4\times10^{+38}$ hasta $-1{,}4\times10^{-45}$, el 0 y desde $1{,}4\times10^{-45}$ hasta $3{,}4\times10^{+38}$. | 4 Bytes (32 bits) |
| Punto Flotante de Precisión Doble | double | System.Double | **Intermedia.** En el orden del **15°** o **16°** dígito decimal de relevancia. | Aprox. desde $-1{,}8\times10^{+308}$ hasta $-4{,}9\times10^{-324}$, el 0 y desde $4{,}9\times10^{-324}$ hasta $1{,}8\times10^{+308}$. | 8 Bytes (64 bits) |
| Punto Flotante de Precisión Extendida | decimal | System.Decimal | **La más alta.** En el orden del **28°** o **29°** dígito decimal de relevancia. | Aprox. desde $-7{,}9\times10^{+28}$ hasta $-7{,}9\times10^{+28}$ (menor rango y altísima precisión). Uso en cálculos de financieros y de precisión científica. | 16 Bytes (128 bits) |

**Características:**

- Su selección se determina por la **precisión** deseada en el tratamiento del número real a almacenar o representar, buscando minimizar el espacio de almacenamiento.
- Todos presentan precisión **menor o igual al 100%**. Es decir, que **puede existir error o diferencia** entre el valor ingresado y el efectivamente almacenado/representado.
- Se codifican en código binario, conforme el estándar IEEE 754.
- Las variables o constantes numéricas reales se inicializan por defecto y automáticamente en 0.0.
- **Interpretación de constantes:**
  - Cualquier constante numérica real, por defecto o con el identificador **D**, se asume **double**. Por ejemplo: las constantes **28.0** y **28.0D** serán interpretadas como **double**.
  - Las constantes numéricas reales con el identificador **F**, se asumirán **float**. Por ejemplo: la constante **28.0F** será interpretada como **float**.
  - Las constantes numéricas reales con el identificador **M**, se asumirán **decimal**. Por ejemplo: la constante **28.0M** será interpretada como **decimal**.

# Caracteres:

| Descripción | Nombre C# | Nombre .Net | Empleo | Valores Admitidos | Espacio de Almacenamiento |
|---|---|---|---|---|---|
| Caracter | char | System.Char | **Almacenamiento y representación de caracteres.** | Entero corto sin signo (16 bits) representando el código de asignado a al carácter el estándar UNICODE. | 2 Bytes (16 bits) |

**Características:**

- Concebidos para el almacenamiento y representación de caracteres individuales.
- Opera en base al estándar ("código") de caracteres UNICODE, que norma 65536 caracteres de diversa taxonomía (numéricos, alfabéticos, simbólicos, de control, etc.), cuyos primeros 256 caracteres son coincidentes con lo definido en el estándar ASCII ampliado (8 bits). Por ello, por ejemplo, el entero asignado en el estándar UNICODE del carácter A es el mismo que el asignado por el estándar ASCII: 65.
- Se codifican en código binario natural
- Las variables o constantes caracter se inicializan por defecto y automáticamente en el caracter fin de cadena (**\0**) cuya codificación en el estándar corresponde al entero corto sin signo 0.
- **Interpretación de constantes:**
  - Los caracteres constantes se explicitan expresándolos entre apóstrofes. Ejemplo: **'A'** (UNICODE 65).

# Cadenas de Caracteres:

| Descripción | Nombre C# | Nombre .Net | Empleo | Valores Admitidos | Espacio de Almacenamiento |
|---|---|---|---|---|---|
| Cadena de Caracteres | string | System.String | **Almacenamiento y representación de cadenas de caracteres.** | Cadenas de caracteres de hasta aproximadamente 2.000.000.000 caracteres UNICODE de extensión. | 20 Bytes (160 bits) más dos Bytes (16 bits) por caracter. |

**Características:**

- Concebidos para el almacenamiento y representación de colecciones secuenciales (ordenadas) de sólo lectura de caracteres UNICODE.
- Esta clase se constituye como una **colección dinámica**. Es decir adapta su tamaño a las necesidades, aumentándolo o reduciéndolo en tiempo de ejecución.
- En **.Net**, las cadenas de caracteres **no** requieren la finalización en el caracter fin de cadena. Por lo cual, puede contener caracteres fin de cadena en la posición y cantidad que se desee.
- Las variables o constantes cadenas de caracter **no** se inicializan por defecto en una instancia de String. Su valor inicial será entonces **null**.
- **Interpretación de constantes:**
  - Las cadenas de caracteres constantes se explicitan expresándolos entre comillas (constante o literal de cadena **regular**), como por ejemplo **"casa", o** bien entre comillas pero precedido por el identificador @ (constante o literal de cadena **textual**), como por ejemplo **@"casa"**. La diferencia entre ambos literales afectará el tratamiento de los caracteres de barra invertida (**\n**, **\"**, **\f**, **\0**, etc.) incluidos dentro de la cadena.

# Lógicos:

| Descripción | Nombre C# | Nombre .Net | Empleo | Valores Admitidos | Espacio de Almacenamiento |
|---|---|---|---|---|---|
| Lógico o booleanos | bool | System.Boolean | **Almacenamiento y representación de los datos lógicos verdadero (true) o falso (false).** | **true** o **false** | 1 Byte (8 bits). |

**Características:**

- Concebidos para el almacenamiento y representación de valores lógicos verdadero o falso. Adicionalmente, si se declara como **bool?** admitirá también el valor **null**.
- Las variables o constantes lógicas declaradas como **bool** se inicializan por defecto y automáticamente en el valor **false**. Las declaradas como **bool?**, en **null**.
- Estos valores se emplean junto a los operadores lógicos comparación igualdad (**==**), comparación diferencia (**!=**) e inversión o negación (**!**).
- En C# ya no es posible la operatoria lógica con datos distintos de los lógicos. Por ejemplo, siendo x un valor numérico, el condicional simple **`if(x)`** ya no es sintácticamente válido, debiéndose expresar como **`if(x!=0).`**
- **Interpretación de constantes:**
  - Las valores lógicos constantes admitidos son sólo **true** y **false** y, como tal, expresarse como cadenas de caracteres ("true" o "false").