

# **Programación II**

## **C#**

### **Clase 1**

**Maximiliano Neiner**

**Modificado por: Federico Dávila**

# Temas a Tratar

- ✖ Tipos de datos
- ✖ Operadores
- ✖ Estructuras de Control
- ✖ Entrada y Salida de Datos

# Temas a Tratar

## Tipos de datos

- CTS (Common Type System)
- Valores predeterminados
- Conversiones básicas en C#

## Operadores

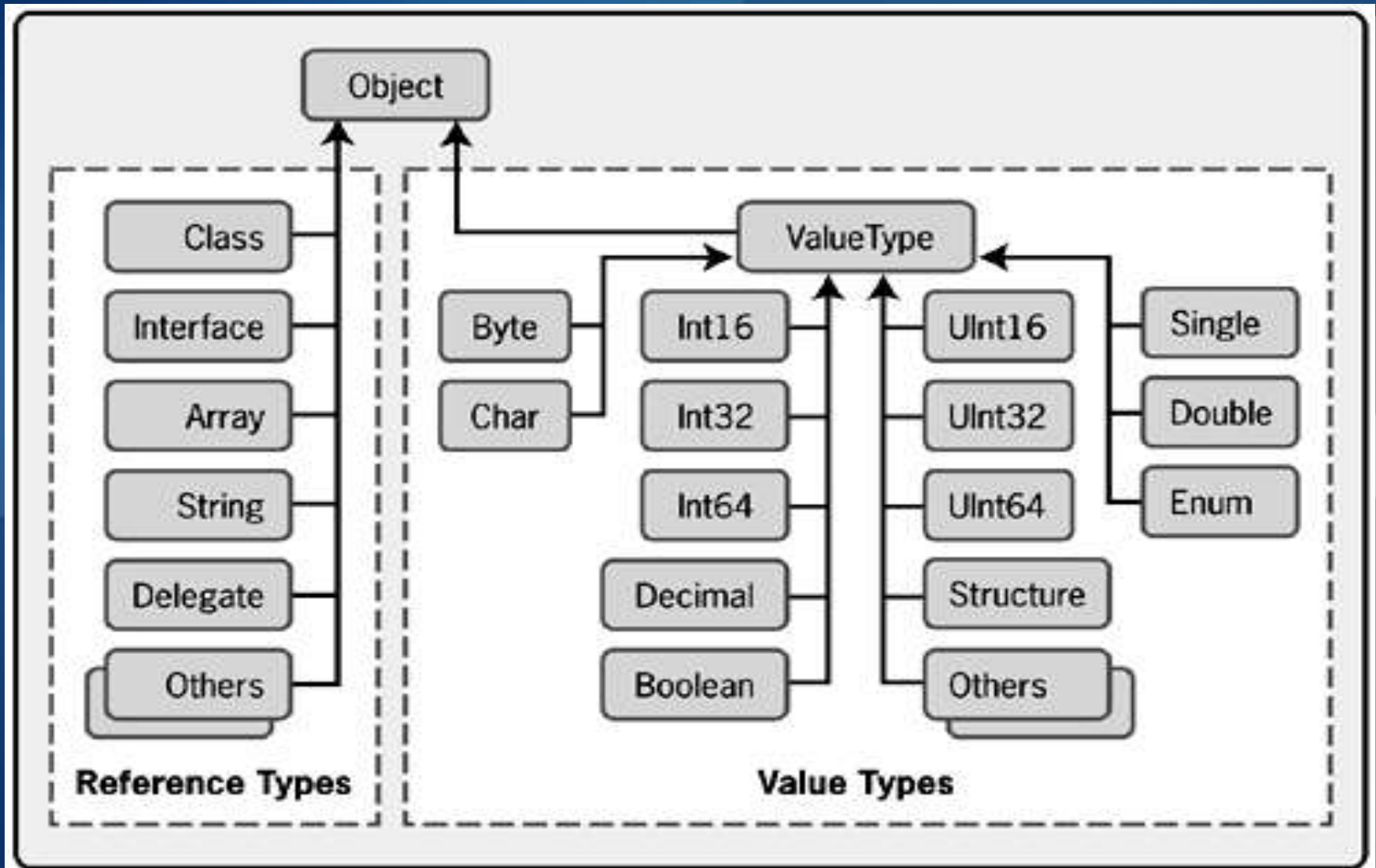
## Estructuras de Control

## Entrada y Salida de Datos

# Common Type System (1/2)

- ✚ Define un conjunto común de “tipos” de datos orientados a objetos.
- ✚ Todo lenguaje de programación .NET debe implementar los tipos definidos por el CTS.
- ✚ Todo tipo hereda directa o indirectamente del tipo `System.Object`.
- ✚ El CTS define tipos de VALOR y de REFERENCIA

# Common Type System (2/2)



# Tipos de Datos

Categoría	Clase	Descripción	C# Alias
Enteros	Byte	Un entero sin signo (8-bit)	byte
	SByte	Un entero con signo (8-bit)	sbyte
	Int16	Un entero con signo (16-bit)	short
	Int32	Un entero con signo (32-bit)	int
	Int64	Un entero con signo (64-bit)	long
Punto Flotante	Single	Un número de punto flotante de simple precisión (32-bit)	float
	Double	Un número de punto flotante de doble precisión (64-bit)	double
	Decimal	Un número decimal de 96-bit	decimal
Lógicos	Boolean	Un valor booleano (true o false)	bool
Otros	Char	Un caracter Unicode (16-bit)	char
	Object	La raíz de la jerarquía de objetos	object
	String	Una cadena de caracteres unicode inmutable y de tamaño fijo	string

# Tipos de Datos

- ✖ Las variables escalares son constantes o variable que contiene un dato atómico y unidimensional.
- ✖ Las variables no escalares son array (vector), lista y objeto, que pueden tener almacenado en su estructura más de un valor.



# Temas a Tratar



## Tipos de datos

- CTS (Common Type System)
- Valores predeterminados
- Conversiones básicas en C#



## Operadores



## Estructuras de Control



## Entrada y Salida de Datos



# Valores Predeterminados (\*)



## Enteros

- 0 (cero)



## Punto flotante

- 0 (cero)



## Lógicos

- False



## Referencias

- Null

# Temas a Tratar

## Tipos de datos

- CTS (Common Type System)
- Valores predeterminados
- Conversiones básicas en C#

## Operadores

## Estructuras de Control

## Entrada y Salida de Datos

# Conversiones Básicas en C#

Existen dos tipos de conversiones:

Implícitas:

✘ No interviene el programador.

Explícitas:

✘ Interviene el programador, porque puede haber pérdida de datos.

# Conversiones Implícitas

Tipo	Conversiones permitidas
sbyte	short, int, long, float, double, decimal
byte	short, ushort, int, uint, long, ulong, float, double, decimal
short	int, long, float, double, decimal
ushort	int, uint, long, ulong, float, double, decimal
int	long, float, double, decimal
uint	long, ulong, float, double, decimal
long	float, double, decimal
ulong	float, double, decimal
float	double
double	Ninguna
decimal	Ninguna

# Conversiones Explícitas

## Ejemplo:

```
double pesoMax = 55.73;  
float limite = pesoMax; //Error. Hay posible pérdida de datos.  
  
// Hay que utilizar el 'casting' o casteo.  
limite = (float) pesoMax; //Correcto.  
  
// El programador se hace 'responsable' de la posible pérdida de datos.
```

# Temas a Tratar

- ✖ Tipos de Datos
- ✖ Operadores
  - Aritméticos
  - Lógicos
- ✖ Estructuras de Control
- ✖ Entrada y Salida de Datos

# Operadores Aritméticos

Descripción	C#
Asignación	=
Adición	+
Sustracción	-
Multiplicación	*
División	/
Negación	!
Módulo (Parte entera de la división)	%
Mayor	>
Menor	<
Mayor o Igual	>=
Menor o Igual	<=



# Temas a Tratar

- ✖ Tipos de Datos
- ✖ Operadores
  - Aritméticos
  - Lógicos
- ✖ Estructuras de Control
- ✖ Entrada y Salida de Datos

# Operadores Lógicos

Descripción	C#
Operador lógico Y	&&
Operador lógico O	
Negación lógica	!
Igualdad	==
Desigualdad	!=

- En C# todas las evaluaciones se hacen por “cortocircuito”

```
//Si Hacer1() es True, entonces  
//NO se evalua Hacer2()  
if (Hacer1() || Hacer2())  
{  
}
```

```
//Si Hacer1() es False, entonces  
//NO se evalua Hacer2()  
if (Hacer1() && Hacer2())  
{  
}
```

# Temas a Tratar

- ✚ Tipos de Datos
- ✚ Operadores
- ✚ Estructuras de Control
  - Condicionales
  - Repetitivas
- ✚ Entrada y Salida de Datos

# Sentencias Condicionales (1/2)

- C#: sentencia if con varios formatos

```
if (x > 10)  
    HacerAlgo();
```

```
if (x < 10)  
{  
    Hacer1();  
    Hacer2();  
}
```

```
if (x < 10)  
{  
    Hacer1();  
}  
else  
{  
    Hacer2();  
}
```

```
if (x < 10)  
{  
    Hacer1();  
}  
else if (x > 20)  
{  
    Hacer2();  
}  
else  
{  
    Hacer3();  
}
```

# Sentencias Condicionales (2/2)

- C#: sentencia switch

```
int a = 0;

switch (a)
{
    case 1:
        //CODIGO 1
        break;
    case 2:
        //CODIGO 2
        break;
    default:
        //CODIGO DEFAULT
        break;
}
```

# Temas a Tratar

- ✘ Tipos de Datos
- ✘ Operadores
- ✘ Estructuras de Control
  - Condicionales
  - Repetitivas
- ✘ Entrada y Salida de Datos

# Sentencias Repetitivas (1/2)

- C#: la sentencia for consta de tres partes

//Partes: declaración, prueba, acción

```
for (int i = 1; i < 10; i++)  
{  
}
```

- C#: la sentencia foreach permite recorrer arreglos y colecciones

```
string[] nombres = new string[5];  
  
foreach (string auxNombre in nombres)  
{  
    //auxNombre es un elemento de nombres.  
}
```



# Sentencias Repetitivas (2/2)

## ● C#: sentencia while

```
bool condicion = true;

while (condicion == true){
    //En algún momento poner condicion = false
}
```

## ● C#: sentencia do-while

```
bool condicion = true;

do{
    //En algún momento poner condicion = false
} while (condicion == true);
```

# Temas a Tratar

- ✖ Tipos de datos
- ✖ Operadores
- ✖ Estructuras de Control
- ✖ Entrada y Salida de Datos
  - Entry Point
  - La Clase Console

# Entry Point (1/2)

## Primer programa en C#

```
class HolaMundo
{
    static void Main(string[] args)
    {
        System.Console.WriteLine("Hola mundo C#");
        // nombre completamente cualificado.

        System.Console.ReadKey();
    }
}
```

# Entry Point (2/2)

- El punto de entrada para los programas en C# es:

```
static void Main(string[] args) { ... }
```

Dónde:

**static:** Es un modificador que permite ejecutar un método sin tener que instanciar a una variable (sin crear un objeto). El método Main() debe ser estático.

**void:** Indica el tipo de valor de retorno del método Main(). No necesariamente tiene que ser void.

**string [] args:** Es un Array de tipo string que puede recibir el método Main() como parámetro. Este parámetro es opcional.

# Temas a Tratar

- ✖ Tipos de datos
- ✖ Operadores
- ✖ Estructuras de Control
- ✖ Entrada y Salida de Datos
  - Entry Point
  - La Clase Console
    - Generalidades
    - Métodos
    - Propiedades
    - Formatos

# Generalidades

- ❖ `public static class Console`
- ❖ Representa la entrada, salida y errores de Streams para aplicaciones de consola.
- ❖ Es miembro del Namespace **System**.

# Temas a Tratar

- ✖ Tipos de datos
- ✖ Operadores
- ✖ Estructuras de Control
- ✖ Entrada y Salida de Datos
  - Entry Point
  - La Clase Console
    - Generalidades
    - Métodos
    - Propiedades
    - Formatos



# Métodos (1/2)

## Clear()

- Limpia el buffer de la consola. Equivalente a `clrscr()` de C.

## Read()

- Lee el próximo carácter del stream de entrada. Devuelve un entero.

## ReadKey(bool)

- Obtiene el carácter presionado por el usuario. La tecla presionada puede mostrarse en la consola. Equivalente a `getch()` / `getche()` de C.

# Métodos (2/2)

## ReadLine()

- Lee la siguiente línea de caracteres de la consola. Devuelve un **string**.  
Equivalente a **gets()** de C.

## Write()

- Escribe el string que se le pasa como parámetro a la salida estándar.  
Equivalente a **printf()** de C.

## WriteLine()

- Ídem método Write, pero introduce un salto de línea al final de la cadena.

# Temas a Tratar

- ✖ Tipos de datos
- ✖ Operadores
- ✖ Estructuras de Control
- ✖ Entrada y Salida de Datos
  - Entry Point
  - La Clase Console
    - Generalidades
    - Métodos
    - **Propiedades**
    - Formatos

# Propiedades

## `BackColor`

- Obtiene o establece el color de fondo de la consola.

## `ForeColor`

- Obtiene o establece el color del texto de la consola.

## `Title`

- Obtiene o establece el título de la consola.

# Temas a Tratar

- ✖ Tipos de datos
- ✖ Operadores
- ✖ Estructuras de Control
- ✖ Entrada y Salida de Datos
  - Entry Point
  - La Clase Console
    - Generalidades
    - Métodos
    - Propiedades
    - **Formatos**

# La Clase Console - Formatos (1/3)

## Console.Write() o Console.WriteLine()

Con los marcadores (“{}”), además de indicar el número de parámetro que se usará, podemos indicar la forma en que se mostrará.

Cuantos caracteres se mostrarán y si se formatearán a la derecha o la izquierda o también se pueden indicar otros valores de formato.

La forma de usar los marcadores o las especificaciones de formato será la siguiente:



# La Clase Console - Formatos (2/3)

$\{ N [, M ][: Formato ] \} (*)$

Dónde:

**N** será el número del parámetro, empezando por cero.

**M** será el ancho usado para mostrar el parámetro, el cual se rellenará con espacios. Si M es negativo, se justificará a la izquierda, y si es positivo, se justificará a la derecha.

**Formato** será una cadena que indicará un formato extra a usar con ese parámetro.



# La Clase Console - Formatos (3/3)

## Ejemplo:

```
Console.WriteLine("{0,10}{1,-10}{2}", 10, 15, 23);
```

```
Console.WriteLine("-----");
```

```
Console.WriteLine("{0,10:##.##}, {1,10:##.##}", 10.476, 15.355);
```

El resultado es:

```
10 15    23
```

```
-----
```

```
10.48    15.355
```

# Bibliografía

## **C# Al descubierto**

- Autor: Joseph Mayo.
- Editorial: Prentice Hall
- ISBN: 84-205-3477-3

 Disponible en la biblioteca de U.T.N. f.r.a.  
(Avellaneda).