

# **Clase 01**

## **Introducción a .NET y C#**

# ¿Qué es una PLATAFORMA DE DESARROLLO?

Una plataforma de desarrollo es un entorno de software que cuenta con un conjunto de herramientas que nos permite construir determinadas aplicaciones de software.



# Características de .NET

- ✓ Multiplataforma
- ✓ Open Source
- ✓ Multi-lenguaje



# Componentes de .NET

- Common Language Runtime (CLR)
- Base Class Library (BCL)
- Componentes de infraestructura común (lenguajes, compiladores, sistema de proyectos, etc)
- Frameworks (Windows Forms, WPF, ASP. NET)
- Herramientas de desarrollo (editores de código, IDEs, línea de comandos)

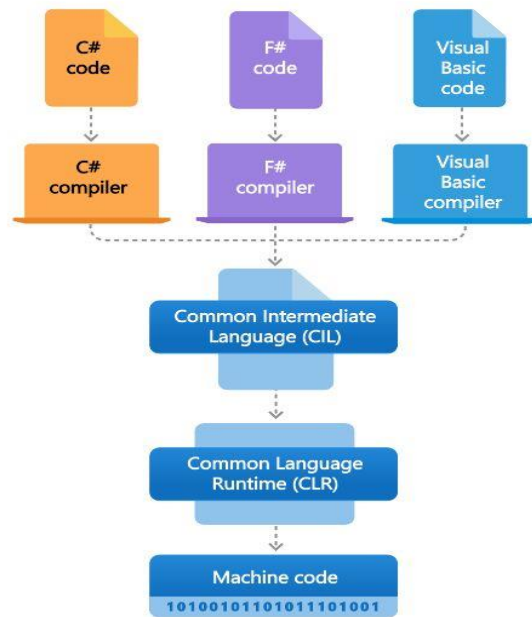
# Proceso de compilación

1. Se compilan los archivos que contienen el **código fuente** a **lenguaje intermedio**.
2. Al ejecutarse la aplicación, el lenguaje intermedio se compila a **lenguaje nativo (máquina)** por el **CLR**.

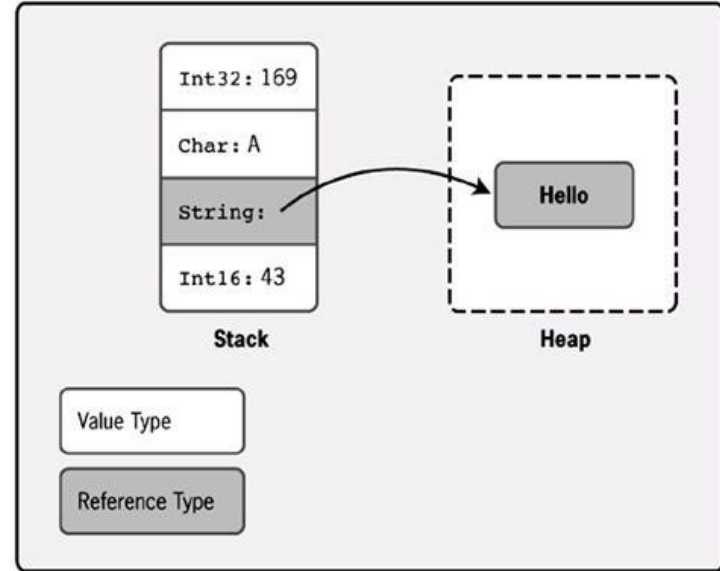
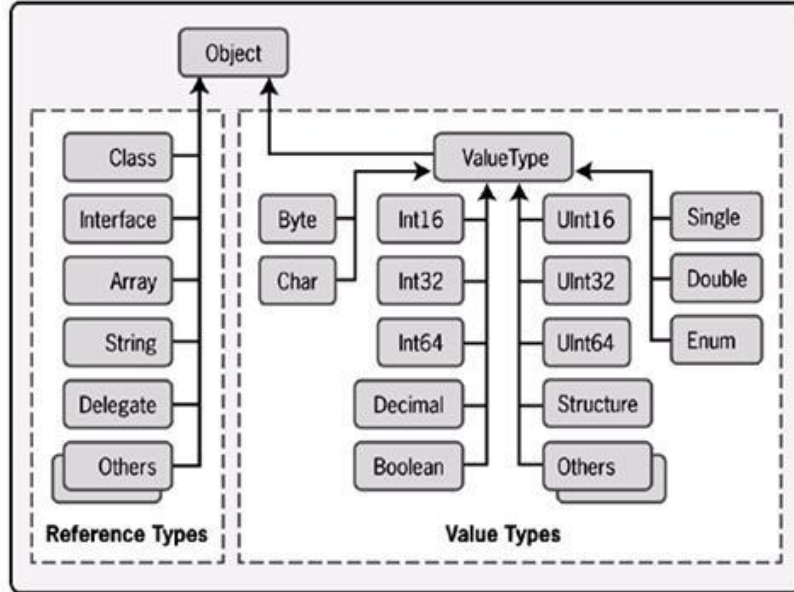
## Conceptos clave:

Archivos con lenguaje intermedio (.exe, .dll)

Tiempo de ejecución y tiempo de compilación.



# Common Type System (CTS)



| 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        | float    |
|                | Double  | Un número de punto flotante de doble         | 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 | 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.

```
//escalares
const int NUMERO = 10;
decimal dec = 2;
float puntoFlotante = 5.5F;

//no escalares
int[] numeros = new int[25];
Persona persona = new Persona();
System.Collections.Generic.List<int> lista = new System.Collections.Generic.List<int>();
```



# Tipos especiales: Object y dynamic



```
1 object nombre = "Esteban";  
2 int longitud = ((string) nombre).Length;  
3  
4 Console.WriteLine("{0} tiene {1} caracteres.", nombre, longitud);
```



```
1 dynamic nombre = "Esteban";  
2 dynamic promedio = 9.99;  
3 int longitud = nombre.Length;  
4  
5 Console.WriteLine("{0} tiene {1} caracteres y un promedio de {2}.", nombre, longitud, promedio);
```

# Inferencia de tipos

```
1  var producto = "Alfajor Capitán del Espacio";
2
3  Console.WriteLine("{0} es de tipo {1}", nameof(producto), producto.GetType().Name);
4
5  var capas = 3;
6
7  Console.WriteLine("{0} es de tipo {1}", nameof(capas), capas.GetType().Name);
8
9  var precio = 99.99M;
10
11 Console.WriteLine("{0} es de tipo {1}", nameof(precio), precio.GetType().Name);
12
13 var peso = 40F;
14
15 Console.WriteLine("{0} es de tipo {1}", nameof(peso), peso.GetType().Name);
16
17 var stock = 1000L;
18
19 Console.WriteLine("{0} es de tipo {1}", nameof(stock), stock.GetType().Name);
20
21 var glaseado = true;
22
23 Console.WriteLine("{0} es de tipo {1}", nameof(glaseado), glaseado.GetType().Name);
24
25 var codigoGusto = 'C';
26
27 Console.WriteLine("{0} es de tipo {1}", nameof(codigoGusto), codigoGusto.GetType().Name);
```

Microsoft Visual Studio Debug Console

```
producto es de tipo String
capas es de tipo Int32
precio es de tipo Decimal
peso es de tipo Single
stock es de tipo Int64
glaseado es de tipo Boolean
codigoGusto es de tipo Char
```

# Conversiones de tipos de datos

## Implicitas

No interviene el programador (no requieren casteo)

No deberían implicar pérdida de datos.



```
1 // Los float pueden almacenar números más grandes que los int.  
2 // No hay pérdida de datos.  
3  
4 float entero = 15;
```

## Explicitas

Interviene el programador (se quiere un casteo).

Podrían implicar pérdida de datos.



```
1 // Los double pueden almacenar números más grandes que los int.  
2 // Además los enteros no guardan los decimales.  
3 // Puede haber pérdida de datos.  
4  
5 int entero = (int)15.2;
```

# Operadores

| Operadores aritméticos |                |
|------------------------|----------------|
| Operador               | Nombre         |
| +                      | Suma           |
| -                      | Resta          |
| *                      | Multiplicación |
| /                      | División       |
| %                      | Módulo / Resto |
| ++                     | Incremento     |
| --                     | Decremento     |

| Operadores de asignación |   |
|--------------------------|---|
| Operador                 | Nombre  |
| =                        | Asignación  |
| +=                       | Suma y asignación   |
| -=                       | Resta el valor de la izquierda al de la variable de la derecha y almacena el resultado en la misma variable.          |
| *=                       | Multiplica el valor de la izquierda por el de la variable de la derecha y almacena el resultado en la misma variable. |
| /=                       | Divide el valor de la izquierda por el de la variable de la derecha y almacena el resultado en la misma variable.     |

# Operadores

| Operadores de comparación |                 |
|---------------------------|-----------------|
| Operador                  | Nombre          |
| <                         | Menor que       |
| >                         | Mayor que       |
| <=                        | Menor o igual a |
| >=                        | Mayor o igual a |

| Operadores de igualdad |             |
|------------------------|-------------|
| Operador               | Nombre      |
| ==                     | Igualdad    |
| !=                     | Desigualdad |

| Operadores lógicos |  |
|--------------------|--|
| Operador           | Nombre                                 |
| !                  | Negación lógica                        |
| &                  | AND lógico                             |
| &&                 | AND condicional lógico / cortocircuito |
|                    | OR lógico                              |
|                    | OR condicional lógico / cortocircuito  |

```
1 //Si MetodoUno() es True, entonces NO se evalua MetodoDos()
2
3 if (MetodoUno() || MetodoDos())
4 { }
5
6 //Si MetodoUno() es False, entonces NO se evalua MetodoDos()
7
8 if (MetodoUno() && MetodoDos())
9 { }
```

# Sentencia Condicionales

```
int numero;  
numero = 10;  
  
if(numero > 10)  
{  
    //Hacer 1  
}  
else  
{  
    //Hacer 2  
}
```

```
int numero;  
numero = 10;  
  
if(numero > 10)  
{  
    //Hacer 1  
}  
else if(numero <= 20)  
{  
    //Hacer 2  
}  
else  
{  
    //Hacer 3  
}
```

# Sentencia Condicionales

```
string nombre;  
nombre = "Federico";  
  
switch (nombre)  
{  
    case "Juan":  
        //Hacer 1  
        break;  
    case "Pedro":  
        //Hacer 2  
  
        break;  
    case "Federico":  
        //Hacer 3  
        break;  
}
```

```
switch (numero)  
{  
    case 1:  
        //Hacer 1  
        break;  
    case 2:  
        //Hacer 2  
        break;  
    default:  
        //Hacer 3  
        break;  
}
```

# Sentencia Repetitivas

```
// Partes: declaración, prueba, acción
for (int i = 0; i < 10; i++)
{
    ...
}
```

```
string[] nombres = new string[5];

foreach (string item in nombres)
{
    //item es un elemento de nombres
}
```

```
bool condicion = true;
while (condicion == true)
{
    //En algun momento poner condicion en false
}
```

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



# Entry Point

El punto de entrada para los programas en C# es la función Main

**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.

# Console

- Es una clase pública y estática.
- Las aplicaciones de .NET pueden usar la clase System.Console para leer y escribir caracteres en la consola.
- Es miembro del NameSpace System.

## Metodos

- **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.

# Propiedades de Console

- `BackColor`: Obtiene o establece el color de fondo de la consola.
- `ForegroundColor`: Obtiene o establece el color del texto de la consola.
- `Title`: Obtiene o establece el título de la consola.

```
Console.BackgroundColor = ConsoleColor.Green;  
Console.ForegroundColor = ConsoleColor.Blue;  
Console.Title = "Mi Consola de prueba";
```

# Formato de salida de Texto

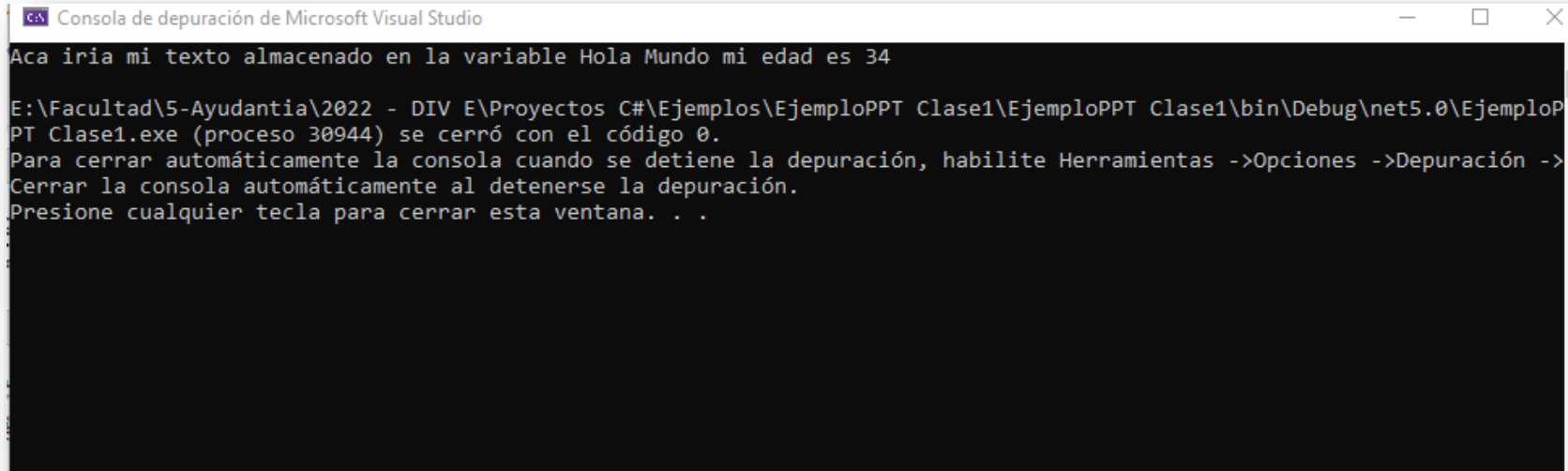
Con los marcadores “{}” podemos imprimir por consola el valor de nuestras variables.

```
string saludo = "Hola Mundo";  
  
Console.WriteLine("Aca iria mi texto almacenado en la variable {0}", saludo);
```

```
string saludo = "Hola Mundo";  
int edad = 34;  
  
Console.WriteLine("Aca iria mi texto almacenado en la variable {0} mi edad es {1}", saludo, edad);
```

```
string saludo = "Hola Mundo";  
int edad = 34;  
  
Console.WriteLine($"Aca iria mi texto almacenado en la variable {saludo}, mi edad es {edad}");
```

# Ejemplos de salida



A screenshot of a Visual Studio debug console window. The title bar reads 'Consola de depuración de Microsoft Visual Studio'. The console output shows a program that prints 'Aca iria mi texto almacenado en la variable Hola Mundo mi edad es 34'. Below this, a message states that the application 'PT Clase1.exe (proceso 30944)' has closed with code 0. Further instructions in Spanish tell the user how to close the console automatically by going to 'Herramientas -> Opciones -> Depuración -> Cerrar la consola automáticamente al detenerse la depuración'. The window ends with 'Presione cualquier tecla para cerrar esta ventana. . .'. The console background is black with white text.

```
Consola de depuración de Microsoft Visual Studio

Aca iria mi texto almacenado en la variable Hola Mundo mi edad es 34

E:\Facultad\5-Ayudantia\2022 - DIV E\Proyectos C#\Ejemplos\EjemploPPT Clase1\EjemploPPT Clase1\bin\Debug\net5.0\EjemploPPT Clase1.exe (proceso 30944) se cerró con el código 0.

Para cerrar automáticamente la consola cuando se detiene la depuración, habilite Herramientas ->Opciones ->Depuración -> Cerrar la consola automáticamente al detenerse la depuración.

Presione cualquier tecla para cerrar esta ventana. . .
```

# Formato de salida de Texto

Además de indicar el número de parámetro que se usará, podemos indicar la forma en que se mostrará.

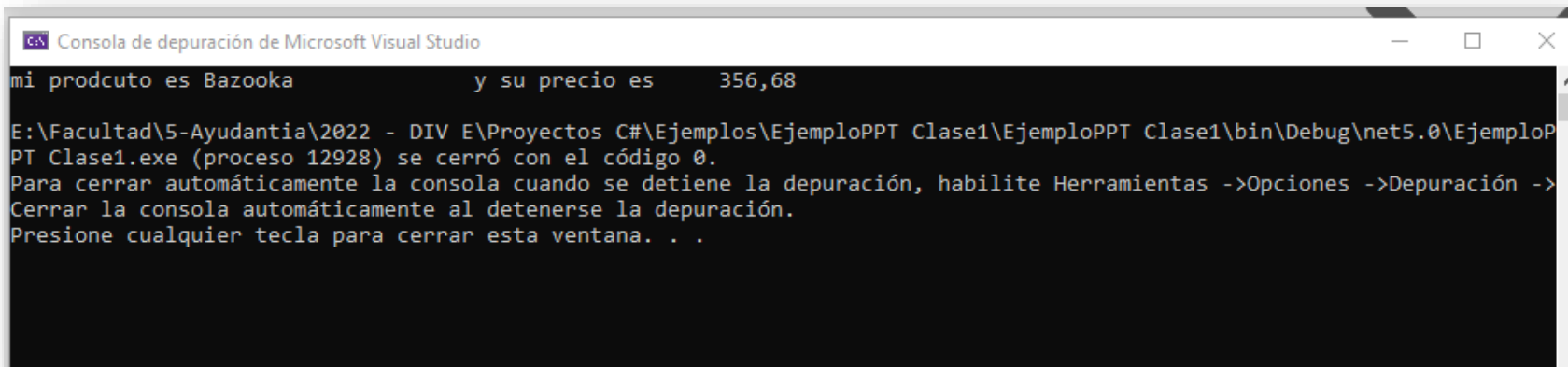
Cuantos caracteres se mostrarán y si se formatearán a la derecha o la izquierda o también se pueden indicar otros valores de formato.

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

- 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.

```
float precio = 356.678945f;  
string nombreProducto = "Bazooka";  
  
Console.WriteLine("mi producto es {0,-20} y su precio es {1,10:0.##}", nombreProducto, precio);
```

# Ejemplos de salida



```
Consola de depuración de Microsoft Visual Studio

mi prodcto es Bazooka          y su precio es      356,68

E:\Facultad\5-Ayudantia\2022 - DIV E\Proyectos C#\Ejemplos\EjemploPPT Clase1\EjemploPPT Clase1\bin\Debug\net5.0\EjemploPPT Clase1.exe (proceso 12928) se cerró con el código 0.
Para cerrar automáticamente la consola cuando se detiene la depuración, habilite Herramientas ->Opciones ->Depuración ->
Cerrar la consola automáticamente al detenerse la depuración.
Presione cualquier tecla para cerrar esta ventana. . .
```