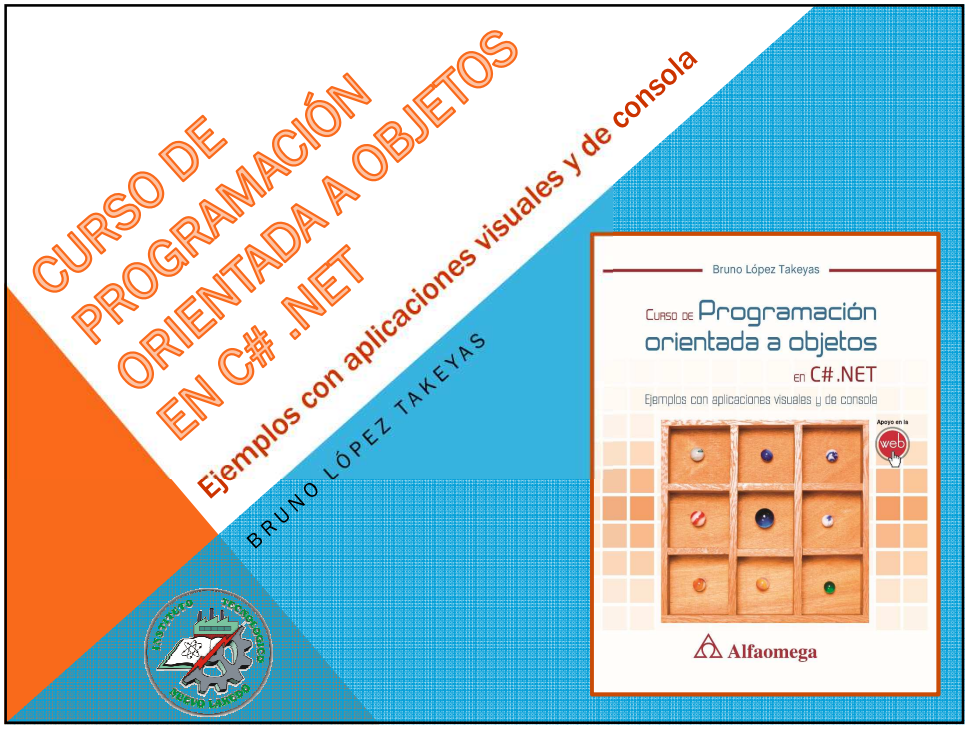

CURSO DE PROGRAMACIÓN ORIENTADA A OBJETOS EN C# .NET
Ejemplos con aplicaciones visuales y de consola
BRUNO LÓPEZ TAKEYAS
Bruno López Takeyas
Curso de Programación orientada a objetos en C#.NET
Ejemplos con aplicaciones visuales y de consola
Alfaomega


CAPÍTULO 2
DISEÑO E IMPLEMENTACIÓN DE MÉTODOS EN C#
2

Preguntas detonadoras

- ¿Qué es un método?
- ¿Cuáles son los tipos de métodos? ¿En qué se parecen? ¿En qué difieren?
- ¿Cómo se envían datos a los métodos?
- ¿Cuándo se recomienda enviar parámetros por valor? ¿Cuándo por referencia?
- ¿Por qué son importantes los métodos para los objetos?

3

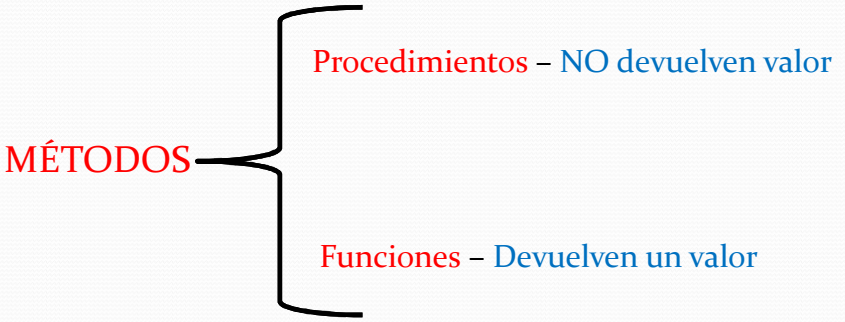
Métodos

- Contienen instrucciones para ejecutar al momento de ser invocados.
- Un método contiene:
  - Modificador de Acceso (Determina su visibilidad)
  - Tipo de dato (Devuelto al finalizar su ejecución)
  - Identificador (Nombre con el cual se invoca)
  - Parámetros (Cero o mas variables que recibe el método)

4

Métodos

- En C# las subrutinas se conocen como métodos, se codifican como parte de una clase y se clasifican en …



5

Sintaxis de los métodos

```
< tipoValorDevuelto > < nombreMétodo > (< parámetros >)
{
      < cuerpo >
}
```

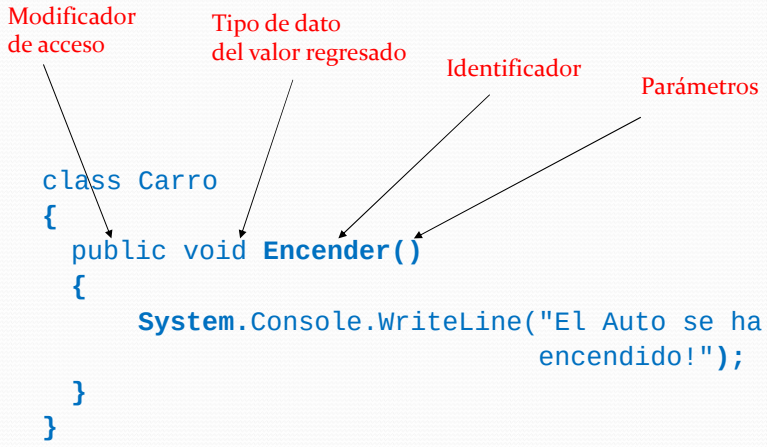
void significa que NO devuelve valor (procedimiento)

- *Ejemplo:*

```
void Saludo( )
{
    Console.Write("Hola");
}
```

6

Ejemplo de un método (en la clase)




```
class Carro
{
  public void Encender()
  {
      System.Console.WriteLine("El Auto se ha
                                encendido!");
  }
}
```

7

Procedimientos

```
static void Imprimir()
{
   Console.WriteLine(Nombre);
   Console.WriteLine(Edad);
   Console.WriteLine(Sueldo);
}
```

8

Funciones

```
static int Sumar() // Devuelve un valor de tipo numérico entero

static double Calcular() // Devuelve un valor de tipo numérico real
static string Comparar() // Devuelve un valor de tipo cadena
```

```
static double CalcularArea()
{
  return(Math.PI * Math.Pow(Radio,2));
}
```

9

Llamadas a los métodos

```
class Program
    {
        static void Main(string[] args)
        {
            Metodo(); // Se invoca (llamada)
        }

        static void Metodo( )
        {
                . . . // Codificación
        }
     }
```

10

Llamadas de procedimientos

```
class Program
    {
        static void Main(string[] args)
        {
            Procedimiento(); // Llamada
        }

        static void Procedimiento( )
        {
            Console.Write("Tec Laredo");
            return();// Fin del Procedimiento
        }
    }
```

El Procedimiento NO devuelve valor

11

Métodos que reciben parámetros

- Entre los paréntesis se especifican una o mas variables (separadas por comas) con sus respectivos tipos de datos.
- Esas variables estarán accesibles dentro del método.

```
public void CambiarEstado( string nuevoestado )
{
   estado = nuevoestado;
}
```

- Al momento de invocar el método, se deben incluir esos valores en la llamada:

```
miCarro.CambiarEstado("Apagado");
```

12

Parámetros recibidos por los métodos

Parámetros

- Por valor – Se envía una copia del valor de la variable
- Por referencia – Se envía la dirección de la variable

13

Envío de parámetros por valor

```
class Program
    {
        static void Main(string[] args)
        {
               int x=10;
            Metodo( x ); // Se envia el valor de x
            Console.Write("x="+x.ToString()); // x=10
        }
        static void Metodo(int y)
        {
             y+=5;
             Console.Write("y="+y.ToString()); // y=15
        }
     }
```

14

Envío de parámetros por referencia

```
static void Main(string[] args)
        {
            int x = 10;
            Metodo(ref x);  // Se envia la referencia de x
            Console.WriteLine("x=" + x); // x=15
            Console.ReadKey();
        }
        static void Metodo(ref int y)
        {
            y += 5;
            Console.WriteLine("\n\ny=" + y);  // y=15
        }
```

15

Métodos que retornan valores

- El "Tipo de dato" del método NO es "void".
- Dentro del método debe haber una sentencia "return" con algún valor del tipo de dato del método.
- Ejemplo (Al declararlo):

```
public string ConsultarEstado()
{
    return estado;
}
```

- Al llamar al método (desde el programa):

```
string estado_actual = miCarro.ConsultarEstado();
```

16

16

Llamadas de métodos que retornan valor (funciones)

```
static void Main(string[ ] args)
        {
            double Radio = 10, Area;
            Area = Funcion(Radio);
            Console.WriteLine("Area=" + Area);
            Console.ReadKey();
        }

        static double Funcion(double r)
        {
            return (Math.PI * r * r);
        }
```

Parámetro enviado a la función

Variable receptora

Valor devuelto por la Función

17

Invocando al método (en el programa)

```
Carro miCarro = new Carro();
miCarro.Encender();
```

Nombre del objeto

Nombre del método

Parámetros

18

Invocando métodos

```
class Arbol
{
    int Altura;
    public void Podar( )
    {
        Console.WriteLine("Podando …");
    }
}

Arbol Pino = new Arbol(); // Se crea el objeto Pino
Pino.Podar();  //Se invoca el método Podar() del
  objeto Pino
```

19

Ámbito de las variables

- El ámbito de una variable define dónde puede usarse esa variable
- Una variable local declarada en un bloque de programa, sólamente puede ser usada en ese bloque
- El ámbito de una variable también aplica a los métodos y a los ciclos

Ámbito de las variables

Ámbito de variables

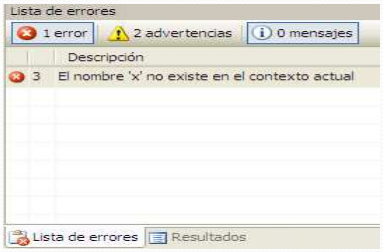
Locales – Se declaran y utilizan dentro de un contexto específico. No se puede hacer referencia a ellas fuera de la sección de código donde se declara.

Globales – Se declaran fuera del cuerpo de cualquier método

21

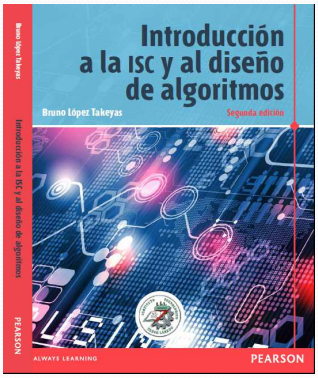
Ámbito de variables en un ciclo for

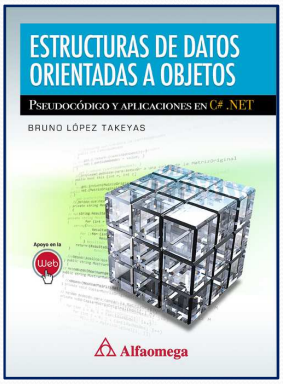
```
for(int x = 1; x<=10; x++)
{
  Console.Write(x);
}

Console.Write(x);
```



Otros títulos del autor

http://www.itnuevolaredo.edu.mx/Takeyas/Libro





takeyas@itnuevolaredo.edu.mx

Bruno López Takeyas