ILERNA Online

Videotutoría 8 (UF2): Programación modular

Módulo 03A: Programación

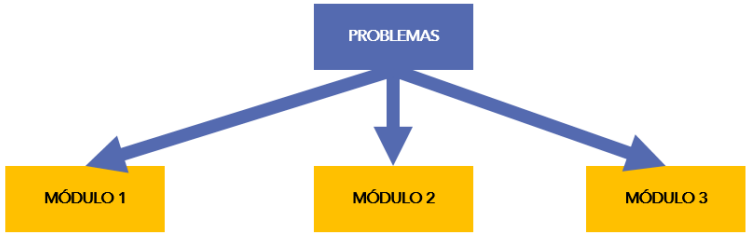
RESUMEN

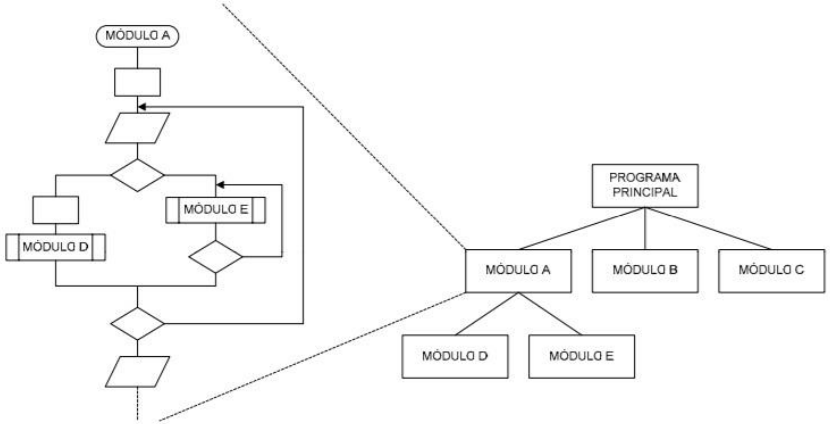
Ventajas

- Facilita el mantenimiento, la modificación y la documentación.
- Testing. Con los módulos por separados, y luego se vuelven a unir al programa principal.
- Reutilización de módulos. Podemos copiar el módulo y llevarlo a otro programa.
- Independencia de fallos.

Desventajas

- No se dispone de algoritmos formales de modularidad, por lo que a veces los programadores no tienen claras las ideas de los módulos → experiencia





FUNCIÓN	

Ámbito de la declaración Tipo Función Nombre_función (parámetros) {
 	
//declaración de variables locales
//Instrucciones
//retorno del tipo
}
	Utilizamos la palabra reservada *function* con una lista de parámetros (variables) que vamos a utilizar en ellas
	Cuando la función llega a su fin, retornará un valor del mismo tipo de la función con la directiva *return*

PROCEDIMIENTO	

Ámbito de la declaración void Nombre_procedimento{
 	
// Instrucciones
 	
}
	Un procedimiento no devuelve un valor y por tanto no incluye la directiva return

Ejemplo procedimiento

Realiza un programa conocer si un número introducido por el usuario es par o impar

```
0 referencias
static void Main(string[] args)
{
    Console.WriteLine("Introduce un número");
    int var = Convert.ToInt32(Console.ReadLine());
    par(var);
    Console.ReadKey();
}
```

Parámetro real, lo podré usar durante el resto del programa cuando quiera.

```
1 referencia
static void par(int num) {

    if (num % 2 == 0) {

        Console.WriteLine("Es Par");
    }
    else {

        Console.WriteLine("Es Impar");

    }
}
```

Si declaro el parámetro dentro de la función o método, será un parámetro local o formal.

ILERNA Online

Ejemplo función

Realiza un programa conocer si un número introducido por el usuario es par o impar

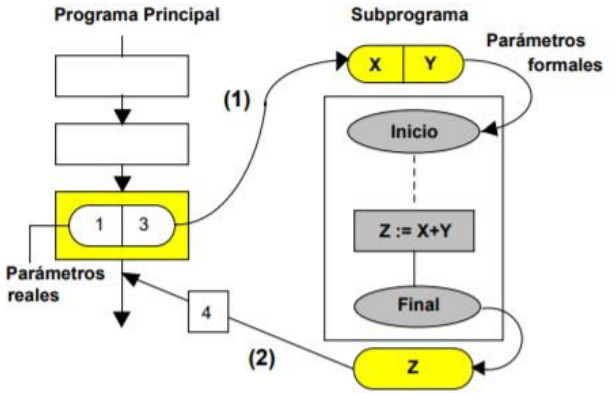
```
static void Main(string[] args)
{

    Console.WriteLine("Dame un número");
    int num = int.Parse(Console.ReadLine());
    if (EsPar(num))
    {

        Console.WriteLine("El número es par");
    }
    else
    {
        Console.WriteLine("El número es impar");
    }

}
```

```
1 referencia
static bool EsPar(int var) {

    if (var % 2 == 0)
    {

        return true;

    }else
    {

        return false; }

}
```

Ejemplo función

Parámetro real: Cuando utilizamos la función Par en el programa principal, vemos que la llamada la hacemos con una variable. Este parámetro es el que denominamos parámetro actual.

Parámetro formal: Como podéis ver en el ejemplo, hemos definido e implementado una función Par. Esta función tiene un parámetro formal que como hemos dicho anteriormente solo tiene validez dentro de la función.



Otro ejemplo

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ConsoleApp35
8   {
9       public class Program
10      {
11
12          static int CalcularMedia(int[] newarray)
13          {
14              int notaMedia = 0;
15              for (int i = 0; i < newarray.Length; i++)
16              {
17                  notaMedia += newarray[i];
18              }
19              notaMedia /= newarray.Length;
20              return notaMedia;
21          }
22
23          static void Main(string[] args)
24          {
25              int[] notas = new int[5];
26              for (int i = 0; i < notas.Length; i++)
27              {
28                  Console.Write("Introduce la nota en la posición " + (i + 1) + ": ");
29                  notas[i] = Convert.ToInt32(Console.ReadLine());
30              }
31              Console.WriteLine("Su nota media es " + CalcularMedia(notas));
32              Console.ReadKey();
33
34          }
35      }
36  }
```

Parámetro formal

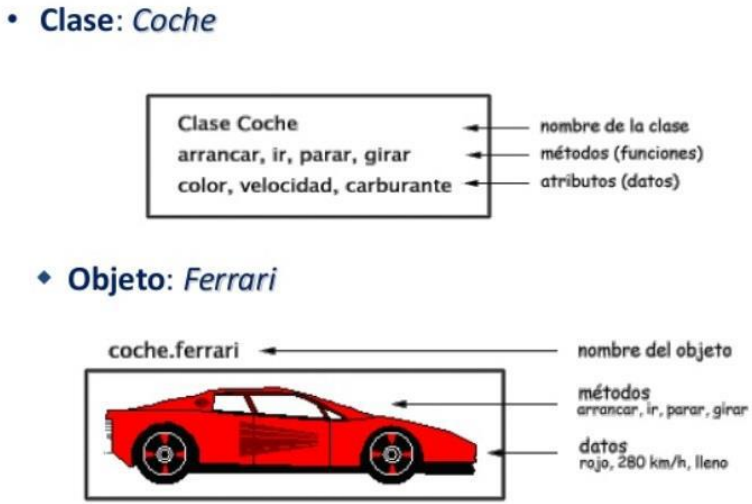
Parámetro real

ILERNA Online

Clases y objetos

- ***Una clase*** es una plantilla para la creación de objetos de datos según un modelo predefinido. Las clases se utilizan para representar entidades o conceptos, como los sustantivos en el lenguaje.

- Cada clase es un modelo que define un conjunto de variables el estado, y métodos apropiados para operar con dichos datos el comportamiento. Cada objeto creado a partir de la clase se denomina ***instancia de la clase***



Clases estáticas

- **Una clase estática** es básicamente lo mismo que una clase no estática, con la diferencia de que no se pueden crear instancias de una clase estática.

La siguiente lista contiene las características principales de una clase estática:

- Solo contiene miembros estáticos.
- No se pueden crear instancias de ella.
- Está sellada

- https://docs.microsoft.com/es-es/dotnet/csharp/programming-guide/classes-and-structs/static-classes-and-static-class-members

Accesibilidad a las clases

Public

Puede obtener acceso al tipo o miembro cualquier otro código del mismo ensamblado o de otro ensamblado que haga referencia a éste.

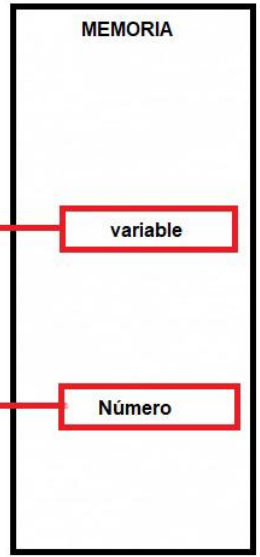
Private

Solamente el código de la misma clase o estructura puede acceder al tipo o miembro

Paso por valor

Cuando ejecutamos una función que tiene parámetros pasados por valor, se realiza una copia del parámetro que se ha pasado, es decir, que todas las modificaciones y/o cambios que se realicen se están haciendo en esta copia que se ha creado. El original no se modifica, de manera que no se altera su valor en la función.

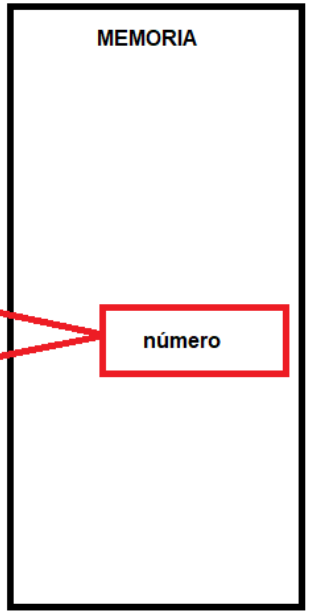
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp36
{
    class Program
    {
        static void MiMetodo(int variable)
        {
            variable = variable + 5;
        }

        static void Main(string[] args)
        {

            int numero = 1;
            MiMetodo(numero);
            Console.WriteLine(numero);
            Console.ReadKey();

        }
    }
}
```

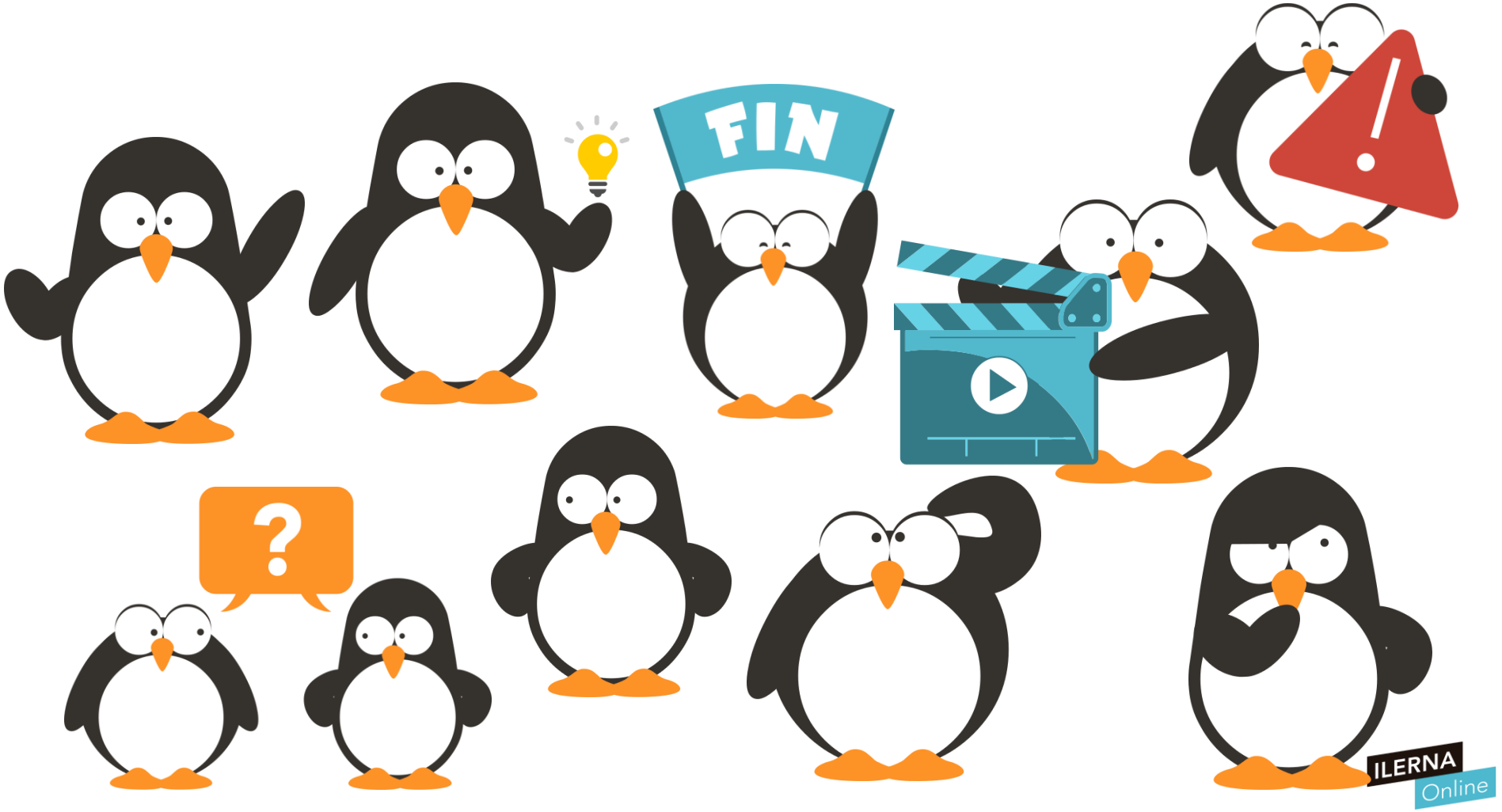


Paso por referencia

Sin embargo, cuando ejecutamos una función que tiene parámetros pasados por referencia, todas aquellas modificaciones que se realicen en la función van a afectar a sus parámetros, ya que se trabaja con los originales.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp36
{
    class Program
    {
        static void MiMetodo(ref int variable)
        {
            variable = variable + 5;
        }

        static void Main(string[] args)
        {

            int numero = 1;
            MiMetodo(ref numero);
            Console.WriteLine(numero);
            Console.ReadKey();

        }
    }
}
```




FIN
?
ILERNA
Online