ILERNA Online

Videotutoría 8 (UF2): Programación modular

Módulo 03A: Programación

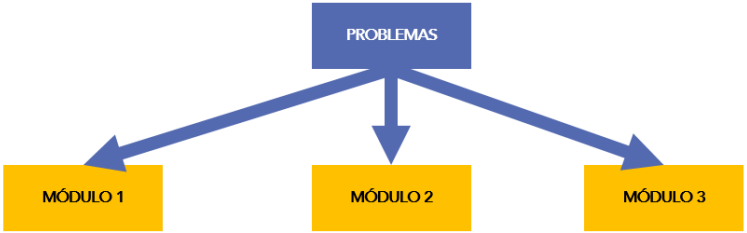
RESUMEN

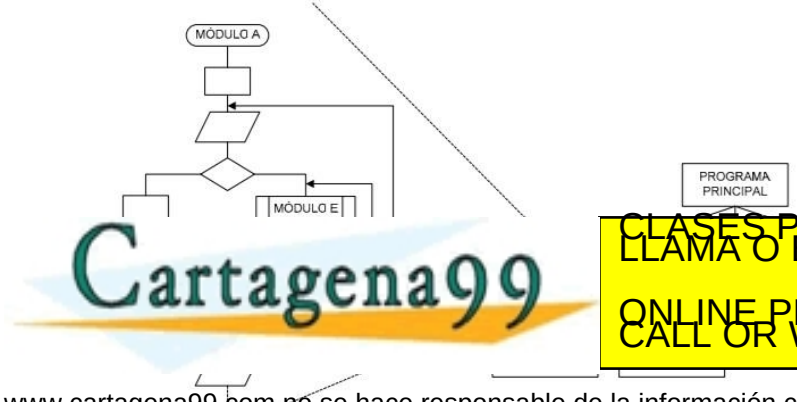
Ventajas

- 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*

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.

No tienen que tener el mismo nombre estas dos variables (si el mismo tipo). Al cambiar el nombre, la estoy declarando local.

```
1 referencia
static void par(int num) {
    if (num % 2 == 0) {
        Console.WriteLine("Es Par");
    }
    else {
        Console.WriteLine("Es Impar");
```

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

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");
```

Al ponerlo así directamente nos coge el valor true.
También seria correcto if (EsPar(num)== true)

Si declarase la función privada no podría acceder a ella desde el main (programa).

1 referencia

```
static bool EsPar(int var) {

    if (var % 2 == 0)
    {

        return true;

    }else
    {

        return false; }
```

Como tenemos dos opciones, los resumo, si no detallo.

Otro ejemplo

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

namespace ConsoleApp35
{
    public class Program
    {

        static int CalcularMedia(int[] newarray)
        {
            int notaMedia = 0;
            for (int i = 0; i < newarray.Length; i++)
            {
                notaMedia += newarray[i];
            }
            notaMedia /= newarray.Length;
            return notaMedia;
        }
        static void Main(string[] args)
        {
            int[] notas = new int[5];
            for (int i = 0; i < notas.Length; i++)
            {
                Console.Write("Introduce la nota en la posición " + (i + 1) + ": ");
```

Parámetro formal

No sirve para nada, simplemente se ha heredado de anteriores lenguajes de programación.

Parámetro real

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*** Instanciar: rellenar una plantilla. Cuando instanciamos esa clase, la convertimos en un objeto.

- **Clase**: *Coche*

Clase persona:
- Edad
- Estatura
- Apellido
- Sexo
- Profesión

4 extremidades no porqué todos las tenemos.

Objeto
- André
- Hombre
- Militar
- etc

- **Objeto**: *Ferrari*

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