

Ejercicios Métodos 'Avanzados'

[Descargar estos ejercicios](#)

Índice

1. [Ejercicio 1](#)
2. ☒ [Ejercicio 2](#)
3. [Ejercicio 3](#)
4. [Ejercicio 4](#)
5. ☒ [Ejercicio 5](#)
6. [Ejercicio 6](#)
7. ☒ [Ejercicio 7](#)

Ejercicio 1

Escribe un método al se le pase un número y diga **si es primo o no**. Suponiendo ya definido el método anterior, escribe un programa que lea dos números enteros por teclado y los sume sólo si son primos, indicando si el resultado también es primo.
En caso contrario, debe decir cuál (o cuáles) de ellos no son primos.

✓ Ejercicio 2

Realiza un programa para calcular el número mágico de una persona. El número mágico de una persona debe considerarse como la suma de las cifras de su día de nacimiento, repitiendo el proceso hasta que la suma de las cifras devuelva un número menor de 10. De esta forma, alguien nacido el 7 de marzo de 1965 tendría como número mágico el 4.

👁 **Nota:** Deberás usar los métodos necesarios para que tu programa quede bien modularizado y con alta cohesión, evitando el acoplamiento.

Un ejemplo de ejecución podría ser:

```
Día nacimiento: 7
Mes nacimiento: 3
Año nacimiento: 1965
7 + 3 + 1 + 9 + 6 + 5 = 31
3 + 1 = 4
Tu número mágico es: 4
```

Ejercicio 3

Escribir en forma de **tabla** los cuadrados y los cubos de los 10 primeros números enteros positivos. Definir para ello un método **Cuadrado** y otra **Cubo**, que te eleven un número al cuadrado y otro al cubo.

Un ejemplo de ejecución podría ser:

Numero	Cuadrado	Cubo
1	1	1
2	4	8
3	9	27
...		
10	100	1000

Ejercicio 4

Escribe el resultado de ejecutar este programa y comenta el motivo de la salida.

👁 **Nota:** Puedes ayudarte con la traza.

```
class Program
{
    static public int a = 0, b = 3;

    static int Funcion1(int a)
    {
        b = b + Funcion2(a) + 1;
        return b;
    }
    static int Funcion2(int a)
    {
        return (b + a + 1);
    }
    static void Main()
    {
        int cont;
        for (cont = 1; cont <= 5; cont++)
        {
            b = Funcion1(cont + a + b) + 1;
            Console.WriteLine("{0} ", b);
        }
    }
}
```

✓ Ejercicio 5

Escribe un programa que lea un número en base 10 y que posteriormente muestre un menú que nos permita convertirlo a base dos, base ocho o hexadecimal.

Crea un método llamado **PasaABinario**, otro llamado **PasaAOctal** y otro llamado **PasaAHexadecimal** para realizar dichas operaciones.

Todos los métodos **devolverán un string** que contendrá los dígitos resultantes de la conversión.

👁 **Nota:** No se puede usar el método Convert existente ya en C#.

Ejercicio 6

En una tienda de piezas de repuesto se realizan una serie de ventas de las que tendremos que calcular el precio final a pagar. Escribe un programa teniendo en cuenta que los descuentos a efectuar están en función de:

1. La clase de comprador (clase A: 2%, clase B: 4% y clase C: 6%)
2. Del tipo de pieza (tipo 1: 8% y tipo 2: 10%).

Crea un método **LeeDatos** que recoja el precio de la pieza, la clase de comprador y el tipo de pieza, y los devuelva mediante **tupla**.

Escribe un método llamada **PrecioAPagar**, para calcular y devolver el precio final. A este método se le pasará la tupla y devolverá el valor del precio final, se hará a través de un switch C#8.

👁 **Nota:** Se termina la introducción de datos cuando un precio sea cero. El programa visualizara el precio parcial de cada venta y una vez terminadas estas, el importe total líquido obtenido en las ventas efectuadas (la suma de total de todas las ventas).

✓ Ejercicio 7

El **calendario Gregoriano actual** obedece a la reforma del calendario juliano que ordenó el papa Gregorio XIII en 1582. Se decidió, después de algunas modificaciones, que en lo sucesivo fuesen bisiestos todos los años múltiplos de cuatro, pero que de los años seculares (los acabados en dos ceros) sólo fuesen bisiestos aquellos que fuesen múltiplos de cuatrocientos.

En base a estos conceptos, deberemos construir un programa que dada una fecha (día, mes y año), nos devuelva como resultado el correspondiente día de la semana en texto (Lunes, Martes, Miércoles, etc.).

👉 **Importante:** La estructura del programa estará formada, además de por el método main, los métodos estáticos (Piensa cual sería el DEM lógico para estas definiciones):

- `(int dia, int mes, int año) LeeFecha()`
- `bool FechaValida(int dia, int mes, int año)`
- `bool Bisiesto(int año)`
- `string DiaSemana(int dia, int mes, int año)`

Consideraciones:

1. La entrada será válida si...
 - **año** → es mayor o igual que 1582.
 - **día** → cumple la condición de ser mayor que 1 y menor que 28, 29, 30 o 31 según corresponda.
 - **mes** → esté entre 1 y 12.
2. Antes de calcular el día de la semana, se tendrán que ajustar los datos de manera que:
Si `mes <= 2`, tendremos que **sumarle 12 al mes** y **restarle uno al año**. El día de la semana se calculará con la siguiente operación:

```
diaSemana = (dia + 2 * mes + 3 * (mes + 1) / 5 + año + año / 4 - año / 100 + año / 400 +
```

👁 **Nota:** Si `diaSemana` es 1 → Domingo, 2 → Lunes, 3 → Martes, ... y 0 → Sábado..