Ejercicios Funciones y Procedimientos (Métodos)

Índice

Ejercicio 1

> Escribe el resultado de ejecutar este programa y comenta el motivo de la salida.
>
> **Nota:** Puedes ayudarte con la traza.

```
class Program
{
    static void Eleva(int a)
    {
        a = a * a;
    }
    static void Main()
    {
        for (int a = 0; a < 10; a = a + 2)
        {
            Eleva(a);
            Console.WriteLine(a);
        }
    }
}
```

Ejercicio 2

Diseña una función denominada Eleva, que calcule **x elevado a n**, dados ambos como parámetro.
Una vez definido, integra dicha función en un programa que calcule e imprima el resultado de la siguiente expresión, siendo x, y, m tres números enteros introducidos por teclado.

$$\frac{(x^4 + y^m)}{2}$$

Ejercicio 3

Escribe un algoritmo que, dados por teclado cinco días de la semana, escritos en forma de número, muestre el nombre del día asociado a cada uno de ellos.
Para ello, implementa el procedimiento **DiaSemana**, que dado un número escriba en pantalla el día correspondiente **`(utilizando switch)..`** .

Nota: el parámetro del procedimiento será de **entrada** y por **referencia**. Además deberá comprobar que el número de entrada esté en el rango de 1 a 7 indicando, si es necesario, que la entrada no ha sido valida.

Ejercicio 4

Implementa un programa en C# con una función que admita como argumentos **dos números enteros (m y n)** y devuelva como valor asociado al nombre de la función, el número combinatorio.
Este método debe llamar a su vez a la función **factorial** (que devolverá el factorial de un número pasado como argumento).

$$\binom{m}{n} = \frac{m!}{n! * (m-n)!}$$

Ejercicio 5

> Escribe el resultado de ejecutar este programa y comenta el motivo de la salida.
>
> **Nota:** Puedes ayudarte con la traza.

```
class Program
{
    public const int n = 5;
    public static int b = 2, a = 3;
    static int funcion(int b)
    {
        int c;
        c = b + a;
        b++;
        return c;
    }
    static void Main()
    {
        int i;
        for (i = 0; i < n; i++)
            b = funcion(i);
        Console.WriteLine(b);
    }
}
```

Ejercicio 6

> Crea un método que cambie de formato una fecha. Dados el día, mes y año devuelva un número entero tipo long `(ver nota)` .
> Dada esta función, diseña un programa que solicite dos fechas de nacimiento y averigüe cual de las dos personas es mayor.
>
> **Nota:** Para una fecha que corresponda con 2/4/1997 el entero largo devuelto sería 19970402.

☑ Ejercicio 7

Construye un programa que dados tres números enteros correspondientes a la hora, minutos y segundos actuales, calcula la hora (en el mismo formato) que será un segundo más tarde. Para ello, se deben diseñar dos métodos:

• **HoraASegundos** que dados tres argumentos de entrada correspondientes a hora, minutos y segundos, devuelva la conversión de dicha hora a segundos desde las 00:00:00.

• **SegundosAHora**, que dado un argumento de entrada correspondiente a una hora en segundos desde las 00:00:00, la convierta en horas, minutos y segundos y la devuelva. **Devolverás la información mediante parámetros de salida.**

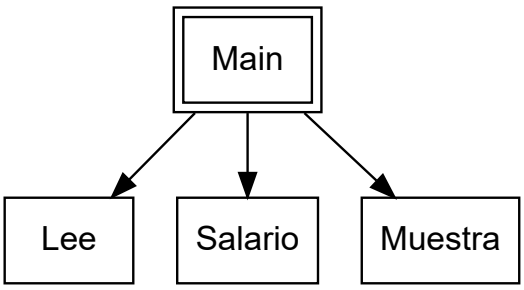
Nota: El algoritmo debe leer la hora en formato HH, MM y SS, después transformarla a segundos (con HoraASegundos), sumarle uno a dichos segundos y después volver a transformarla en HH, MM y SS (con HoraASegundos).

☑ Ejercicio 8

Escribe un método llamado **Lee** que que obtenga los siguientes datos de un usuario: número de departamento, coste por hora y horas trabajadas. **Usarás tuplas para resolverlo**

Escribe un método llamado **Salario** que para calcular el salario semanal multiplique el coste por hora por el número de horas trabajadas.

Escribe un método llamado **Muestra** que muestre el salario semanal, el número del departamento, el coste por hora y las horas trabajadas. Podéis fijaros en el siguiente DEM:



☑ Ejercicio 9

> Escribe un programa para jugar a adivinar números. El programa tiene que seguir los siguientes pasos:
>
> 1. Calcular de forma aleatoria el **número a adivinar** por el jugador. El número debe hallarse entre 0 y 50 (ambos inclusive).
> 2. Preguntar un número al jugador y dar una **pista** indicando si el número introducido es mayor o menor que el número a adivinar.
> 3. Si el jugador acierta el número, la partida terminará indicando la **cantidad de tentativas** hechas por este jugador para acertar.
> 4. Habrá un máximo de tentativas dependiendo del **nivel** elegido para jugar:
>   `fácil =10, medio = 6, difícil = 4` .
> 5. El programa preguntará si se desea **seguir jugando**. Si se responde que sí el juego seguirá pidiendo un nuevo nivel y generando otro número.
> 6. Para salir abra que pulsar **ESC**.
>
> **Nota:** Será necesario realizar los métodos y el paso de parámetros que consideres adecuado para una correcta programación.

```
//Forma de controlar la pulsación en consola de la tecla  ESC con C#...
// El programa espera una tecla y la guarda en continuar.
        ConsoleKeyInfo continuar = Console.ReadKey();
// Comprobación de que la tecla pulsada es ESC.
        Bool esESC = continuar.Key == ConsoleKey.Escape;
```

☑ Ejercicio 10

Programa que implementará un juego con las siguientes características:

1. El programa pedirá que introduzcas el número de participantes a jugar.
2. Cada participante tirará 3 veces un dado con valores entre 1 y 100 (electrónico se entiende), sumándose el valor de las 3 jugadas. Ganará el participante que obtenga mayor puntuación según las siguientes reglas:
   - Si el nº obtenido es múltiplo de 3 o de 5 sumara 10 pts.
   - Si el nº obtenido es múltiplo de 4 o de 6 sumara 5 pts.
   - Si el nº obtenido es mayor de 80 sumara 2 pts.
   - Si el nº obtenido es mayor de 50 sumar 1 pts.
   - Si el nº obtenido es menor de 50 restará 2 pts.
   - Si el nº obtenido es menor de 20 restará 1 pts.
3. El DEM para el juego será más o menos el siguiente.