



RELACIÓN DE EJERCICIOS DE DART (II)

EJERCICIOS CON FUNCIONES

Para la resolución de estos ejercicios, seguiremos la [documentación oficial](#)

1- Vamos a declarar en un programa Dart una función sin parámetros ni valores devueltos, que únicamente muestre por pantalla la fecha y la hora actuales (**DateTime.now()**).

2- Haremos una nueva versión del programa y la función anterior para que ésta última reciba un parámetro: las horas hacia adelante o atrás a partir del momento actual, y calcule y muestre esa fecha/hora. Investiga en la librería **dart:core** los métodos disponibles para el manejo de **DateTime**. Utiliza el método add en este caso, pero experimenta otros.

3- Vamos a hacer un pequeño programa Dart con una función que muestre el nombre y el id de un empleado que se les pasa como parámetros, pero sabiendo que, tanto uno como otro podrían no saberse, en cuyo caso, se mostrará el nombre 'desconocido' y/o el id 999, que serán una suerte de valores por defecto. (Función con parámetros opcionales, y sin devolución de valor de retorno)

4- Hagamos esta vez un programa con una función que muestre el nombre de un empleado que se le pasa como parámetro y los apellidos, pero esta vez, puede que no se pase apellido, en cuyo caso se considerará null (**nullable parameter**), cosa que deberá procesar la función adecuadamente. (Función con parámetros posicionales y sin devolución de valor de retorno)

5- Repitamos el ejercicio de la hora actual y el desplazamiento horario como parámetro entero normal, pero esta vez, la función devolverá la fecha y hora al punto de llamada.

6- Haz un programa en Dart que contenga una función que calcule la suma de los n primeros números naturales. La función tendrá un parámetro n y devolverá el valor de la suma al punto de la llamada

7- Haz el pequeño programa en Dart que calcule el factorial de un número n inicializado a un valor entero y positivo. La función tendrá a n como parámetro de entrada y devolverá el valor de la suma al punto de la llamada

8- Haz en Dart el programa que invoca a una función que calcula el MCD de dos números enteros positivos utilizando el algoritmo de Euclides. Los dos números serán parámetros de entrada, y la función devolverá el resultado al punto de la llamada. Ambos números serán inicializados en el propio programa.

9- Haz en Dart el programa que invoca a dos funciones, una que calcula el cociente de la división entre dos números enteros positivos utilizando el algoritmo de Euclides y la otra que

calcula el resto. Los dos números serán parámetros de entrada, y la función devolverá el resultado al punto de la llamada. Ambos números serán inicializados en el propio programa.

Pregunta 1: qué estructura de datos nos permitiría que una función pudiera devolver dos valores simultáneamente?

Pregunta 2: ¿Por qué no sería recomendable hacer una función así? Investiga el **principio SRP**

10- Haz un programa que declare e inicialice un número entero positivo e invoque a una función que lo convierta en binario y lo devuelva en forma de cadena (**String**) de ceros y unos. El programa deberá mostrar el resultado por pantalla.

Una vez que funcione adecuadamente, modifica la función para que tenga un segundo parámetro, la base de conversión, que será 2 por defecto, pero que pueda ser cualquier otra, hasta la base 8.

Pregunta: qué habría que añadirle a la función para que también pudiera convertir a base 16?

11- Haz un pequeño programa en Dart que declare una lista de temperaturas máximas. A continuación, deberá invocar a una función que reciba como parámetro a esa lista y calcule y devuelva cual es la temperatura máxima. Haz otra con el mismo esquema de parámetros y devolución, que devuelva la mínima, y otra para la media.

Pregunta: estas funciones devuelven un valor numérico que queda descontextualizado en el caso de la máxima y la mínima, perdiendo el dato de en qué día se produjeron. ¿Cómo lo solucionarías? Hay varias opciones

12- Soluciona el problema anterior mediante la declaración de un **Map** y haciendo que cada función, en lugar de devolver valores, muestren los resultados por pantalla ellas mismas.

13- Haz un pequeño programa Dart que defina la función `esPrimo(int n)`, a la que invocará pasándole un número entero positivo, y que devolverá un valor booleano tras buscarle divisores al número: true si lo es, false si no lo es.

14- Haz una nueva versión del ejercicio 6, esta vez con una función **RECURSIVA**

15- Lo mismo con el ejercicio 7

16- Ídem con el ejercicio 8

17- Ídem con el ejercicio 9

18- Haz un programa Dart que declare una lista, y la inicializa con los valores devueltos por una función que devuelva los números de la serie de Fibonacci hasta llegar al término n -ésimo, siendo n un valor entero declarado e inicializado al principio del programa

19- Haz lo mismo en otro programa con una función que genere una lista de todos los números primos que hay entre 1 y n, siendo n un valor entero declarado e inicializado en el programa. Ten en cuenta que puedes reutilizar la función esPrimo ya trabajada en el ejercicio 13

20- Teniendo en cuenta que la creación de librerías de funciones favorece la reusabilidad de las mismas, recoge todas las funciones del ejercicio 11 y ponlas en un archivo. A continuación haz una versión de ese ejercicio, en el que importes la librería y te ahorres de declarar dichas funciones. [Ver documentación](#)