PROGRAMACIÓN UD-4

FUNCIONES

ÍNDICE

[INTRODUCCIÓN 3](#_Toc118517495)

[DEFINICIÓN 3](#_Toc118517496)

[EJECUCIÓN 3](#_Toc118517497)

[ÁMBITO DE LAS VARIABLES 4](#_Toc118517498)

[PARÁMETROS DE ENTRADA 4](#_Toc118517499)

# INTRODUCCIÓN

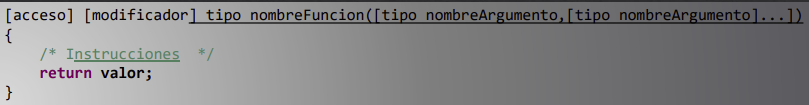
Si el código de nuestro programa es amplio, es probable que tengamos que implementar una misma funcionalidad en varios sitios lo cual nos puede llevar a tener distintos problemas:

* **Duplicidad de código.** Aumenta el tamaño del programa haciéndolo menos comprensible.
* **Dificultad de mantenimiento.** Cualquier modificación en una funcionalidad, habrá que aplicarla tantas veces en el código, como veces se use esa funcionalidad en nuestro código.

En algunos casos los bucles pueden evitar la duplicación de código, pero si la funcionalidad es requerida en distintos sitios, esto no será posible.

Una función es un conjunto de instrucciones agrupadas bajo un nombre y con un objetivo común, que se ejecuta al ser invocada.

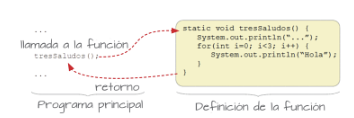
# DEFINICIÓN



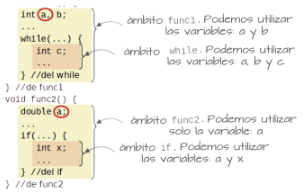
* Acceso. Es opcional. Si no se pone se quedará el questé por defecto.
* Modificado. Es opcional.
* Tipo. Si no se quiere devolver nada, se utilizará void; y si se quiere devolver algo, se utilizará el tipo de lo que devolvamos con la instrucción return.
* nombreFuncion. Es el nombre que usaremos para invocar a la función.
* Parámetros. Son opcionales. Valores que le pasamos a la función para que pueda realizar su cometido.
* Instrucciones. Son las instrucciones que ejecutará nuestra función. Podemos definir la variable, realizar asignaciones, condiciones, bucles…

# EJECUCIÓN

* Las instrucciones del programa principal se ejecutan hasta que encuentra la llamada a la función.
* La ejecución salta a la definición de la función.
* Se ejecuta el cuerpo de la función.
* Cuando la ejecución del cuerpo termina, retornamos al punto del programa desde donde se invocó la función. Si tiene un return devolverá un valor.
* El programa continúa su ejecución.



# ÁMBITO DE LAS VARIABLES

Las variables definidas en una función se denominan variables locales y sólo serán utilizables dentro de la función.

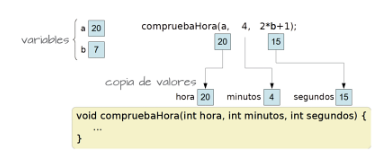
# PARÁMETROS DE ENTRADA

Los parámetros son variables locales de la función que han sido inicializadas en la llamada.

En java el paso de valores a los parámetros se hace por valor, se generan unas variables locales (parámetros formales) que copiarán el valor con el qu se hizo la llamada. Si al llamar usamos variables a estas variables se les llama parámetros reales.

En java los parámetros reales se asignan uno a uno a los parámetros formales en función de su ubicación.

Cualquier cambio en un parámetro de entrada que se efectúe dentro del cuerpo de la función no repercute en la variable o expresión utilizada en la llamada, ya que lo que se modifica es una copia y no el dato original.



Cuando una función devuelve un valor, la llamada a la función se puede usar dentro de una expresión. Por ejemplo:

* int intValor = potencia(2, 3); 🡪 se asignará a intValor lo que devuelva función potencia.
* if (intValor > meda (intA, intB, intC)); 🡪 en este caso se ejecutará el if si intValor es mayor que la media que devuelva la función.

Si una función devuelve un valor, hay que cambiar el void de la definición por el valor devuelto. Dentro de la función utilizaremos la instrucción return para devolver un valor.

# SOBRECARGA DE FUNCIONES

Java permite que 2 o más funciones compartan el mismo identificar, siempre que el número o el tipo de los parámetros del prototipo sean distintos. Para distinguirlas no sirve con el parámetro de vuelta.

Para invocar a cada una de las instancias de la función, se utilizarán los parámetros de la llamada.

Por ejemplo, las siguientes llamadas son a funciones distintas:

* sumar(1,2,3) 🡪 3 valores de tipo int.
* sumar(1,2) 🡪 2 valores de tipo int.
* sumar(1.5,2) 🡪 2 valores, 1 de tipo double y 1 de tipo int.

# RECURSIVIDAD

Si una función se invoca a sí misma, diremos que es recursiva directa.

Si una función invoca a otras funciones y antes de dar una respuesta a la llamada lagunas de las funciones vuelven a llamar a la principal, se llamará recursividad indirecta.

Al igual que pasa en los bucles hay que evitar las llamadas infinitas por lo que hay que hacer una devolución en función de un caso base. Si la función del caso base es falsa, se llamará de nuevo a la función recursiva, en caso contrario se devolverá un valor.

