PROMETEO

Funciones

Índice

1

Introducción

PROMETEO

01

Introducción

- Una **función** es un fragmento de código que **implementa una determinada funcionalidad**: encapsula una serie de instrucciones que resuelven un cálculo o tarea.
- El uso de funciones **facilita el mantenimiento** y **optimiza** y **simplifica el código.**
- El código de una función se puede **reutilizar** invocando a la función siempre que se necesite resolver el cálculo que tenga implementado.

01

Introducción

- **Llamada a la función** : instrucción que se utiliza para invocar a la función. Es el **nombre** de la función seguida de **()**
- **El prototipo de la función** consta de:
  - Modificador (si procede)
  - Tipo del valor que devuelve.
    - '**void**' si no devuelve nada.
    - 'tipo' del dato que retorne. En este caso se utiliza '**return**' para devolver el valor
  - Nombre de la función
  - Parámetros entre paréntesis, si los tiene.
  - Ejemplos:

    public static void func1 ()

    public static String func2 (tipo param1, tipo param2)

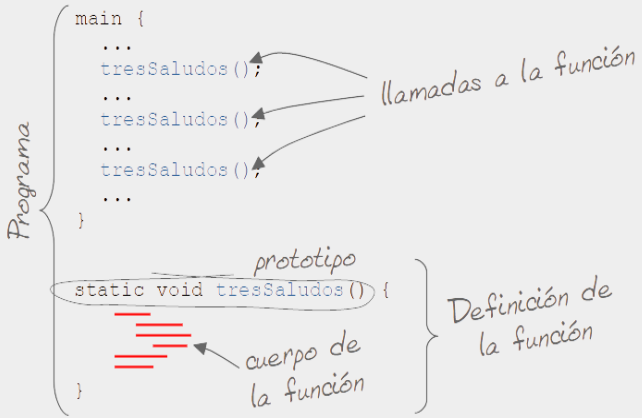
01

Introducción

- El **cuerpo** de la función son las instrucciones que se incluyen entre llaves tras el prototipo.

- **Definición** de una función:
  prototipo + cuerpo.

- La definición de las funciones, en java, puede hacerse antes o después del main.

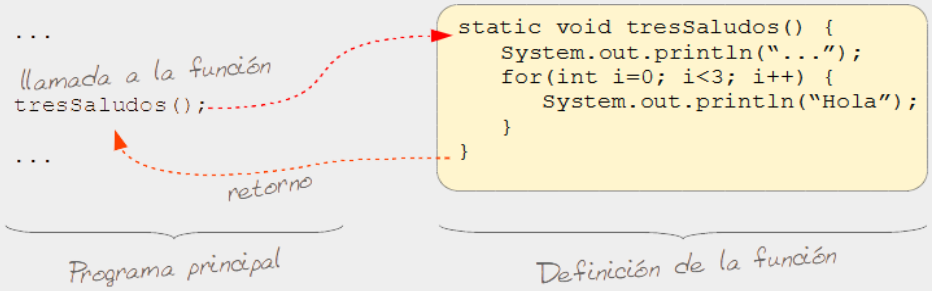


01

Introducción

Llamada a una función:

1. Se ejecutan las instrucciones del programa hasta llegar a la invocación de la función
2. El flujo de ejecución salta a la definición de la función
3. Se ejecuta el cuerpo de la función
4. Cuando se llega al final de la función, se retorna al programa principal.
5. Continúa la ejecución desde la instrucción siguiente a la llamada a la función.



2

Parámetros y retorno de una función

PROMETEO

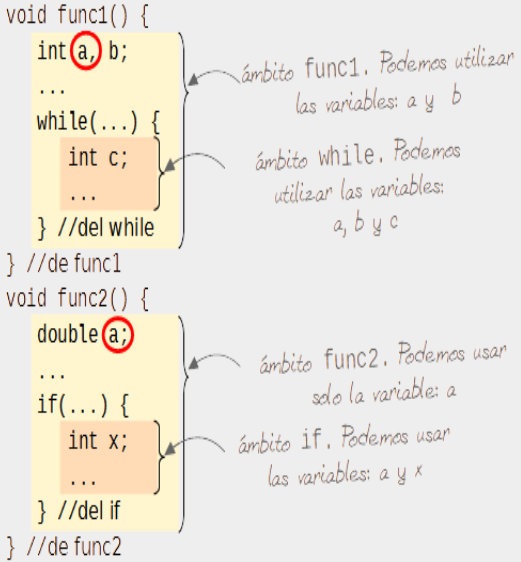
02

Ámbito de las variables

- En una función se pueden declarar variables cuyo ámbito se limita a la propia función. Se conocen como **variables locales**.

- Por otro lado, dentro de los bloques de instrucciones de estructuras if-else o bucles, también se pueden declarar variables que limitan su ámbito al propio bloque de instrucciones. Son las **variables de bloque**.

- Las variables de bloque deben tener nombre distinto al de las variables declaradas fuera del bloque

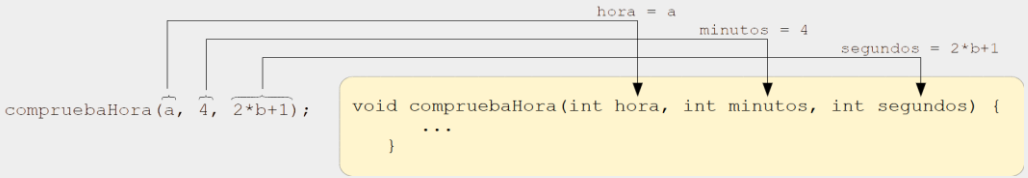


02

Parámetros

Los **parámetros** de una función son los elementos que permiten **pasar datos desde el programa principal a la función**, para que esta realice los cálculos o tratamiento que corresponda sobre sobre los mismos.

- Son **variables locales a la función**. Se declaran entre los paréntesis del prototipo.
- Tienen el **tipo** indicado en la definición de la función.
- **Su valor se inicializa en el momento de la llamada a la función**. En cada invocación se pueden asignar valores diferentes a los parámetros.
- Se pueden utilizar tantos como se necesiten.
- Los valores de los parámetros pueden asignarse con literales, expresiones o variables.

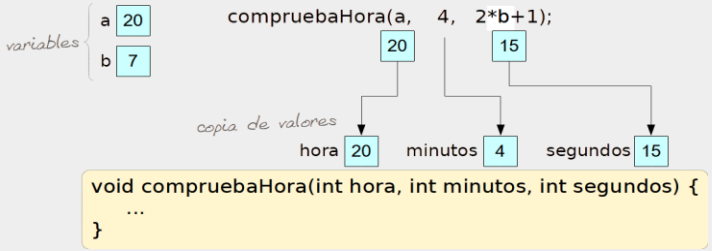


02

Parámetros

- Toman su valor como una **copia del valor** de la expresión o variable utilizada en la llamada: **paso de parámetros por valor** o por copia.

- 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.



En el cuerpo de la función se pueden modificar las variables 'hora', 'minutos', 'segundos', pero no se modificará el valor de las variables 'a' o 'b'.

02

Retorno de una función

El cálculo o tratamiento que realiza una función puede devolverse al programa principal como resultado o información de salida.
Para devolver un valor al programa principal:

- Se utiliza la directiva **return** (return valor). Es deseable utilizar un único 'return' para mayor claridad.
- return será la **última instrucción** ya que fuerza el fin de ejecución de la función.
- En el prototipo de la función se indica el tipo de variable que se devolverá.
  - Si se indica **void** , la función no devuelve ningún valor
  - En otro caso, se debe garantizar en el bloque de instrucciones que se calculará el valor resultado a devolver.

En tiempo de ejecución, la llamada a la función se convierte en un valor determinado, resultado del cálculo implementado en la función.

3

Sobrecarga de funciones

PROMETEO

03

Sobrecarga

- La **sobrecarga** de funciones se da cuando dos o más funciones tienen el **mismo nombre** en un mismo programa (clase)
- Se **distinguen** entre sí por la **lista de parámetros**: diferentes en número o tipo.
- Pueden devolver distintos tipos de datos
- Agrupan funcionalidades de uso similar bajo un mismo identificador.

```
//función sobrecargada
   } public static int suma(int a, int b){
   int suma;
   suma = a+b;
   return suma;
}

//función sobrecargada
public static double suma(int a, double pesoA,
                          int b, double pesoB){
   double suma;
   suma = a*pesoA /(pesoA + pesoB) +b*pesoB/(pesoA+pesoB);
   return suma;
```

4

Recursividad

PROMETEO

04

Recursividad

- Una **función recursiva** es aquella que incluye **llamadas a sí misma** en el cuerpo de instrucciones.
- Para garantizar que la ejecución de las llamadas recursivas finaliza en algún momento, es imprescindible identificar un **caso base**.
- En cada llamada recursiva **la talla del problema a resolver se reduce**, hasta llegar al caso base.

04

Recursividad

- Para identificar si se ha alcanzado el caso base se añadirá un control de selección (if) que marcará si se continua la recursión o no (caso base verdadero).

- El caso base es el escenario en el que la resolución al problema llega a su mínima expresión (trivial) y se puede resolver sin realizar más llamadas recursivas.

- Al resolver el caso base, se va retornando la ejecución a las llamadas anteriores hasta llegar a la inicial.


PROMETEO