Unidad 4 Funciones

Cátedra Introducción a la Programación
Tecnicaturas Universitarias
Facultad de Informática
Buenos Aires 1400 (8300) Neuquén
Universidad Nacional del Comahue, Argentina

Funciones

1. DEFINICIÓN DE FUNCIÓNES Y PARÁMETROS

Una de las herramientas más importantes en cualquier lenguaje de programación son las funciones. Una función es un conjunto de instrucciones que a lo largo del programa van a ser ejecutadas las veces que se requiera. Es por ello, que este conjunto de instrucciones se agrupan en una función. Las funciones pueden ser llamadas y ejecutadas desde cualquier punto del programa.

Los **nombres** de las funciones siguen las mismas reglas que otras etiquetas de PHP. Un nombre de función válido comienza con una letra o guión bajo, seguido de cualquier número de letras, números, o guiones bajos. Como expresión regular se expresaría así: [a-zA-Z_][a-zA-Z0-9_]*.

A continuación se expresa la sintaxis general para declarar una función en PHP:

function nombre (*parametro*1,parametro2,..., *parametroN*){
instrucción1
instrucción2
.
.
...
instrucciónN
}

Donde *$parametro1, $parametro2, . . . , $parametroN* es la lista de parámetros o argumentos de la función. Una función recibe parámetros actuales (valores externos) en sus parámetros formales o de entrada, de los cuales va a depender el resultado de dicha función. Es decir, **según el parámetro o parámetros con los que se invoque a la función, ésta devolverá un resultado u otro**.

Para invocar a (hacer que se ejecute) la función usaremos esta sintaxis: ***nombre($par1, $par2, $par3, . . . , $parN);*** donde *$par1, $par2, $par3, . . . , $parN* son los parámetros (información) que le pasamos a la función. Una función puede necesitar ningún, uno o varios parámetros para ejecutarse, lo que va a depender de la funcionalidad que implemente la función.

1.1. Retornando resultado

Las funciones pueden retornar un valor como resultado de su ejecución, para esto se requiere incluir la sentencia ***return*** seguida del resultado de la función. La sentencia return indica que cuando se alcanza se ha llegado al final de la función y se devuelve como resultado de la misma el contenido especificado a continuación del return. Después de un return puede devolverse una variable, un número, una cadena de texto, etc.

Por ejemplo: ***return "No dispone de permisos";*** /*significa que la función devuelve esta cadena de texto.*/

Otro ejemplo: ***return $calculo;*** /*indica que la función devuelve el contenido que se encuentre almacenado en la variable $calculo.*/

Otro ejemplo: ***return "Lo sentimos ".$usuario." pero no dispone de permisos. Para solicitar información puede escribir a ".$emailAdministrador;*** /*haría que la función devuelva una cadena de texto donde intervienen diversas variables.*/

1.2. Funciones predefinidas

En PHP, además de las funciones creadas por el usuario, existen múltiples funciones para realizar distintas tareas. Todas ellas están ya predefinidas, por lo que lo único que tenemos que hacer es llamarlas, pasarles los parámetros necesarios, y ellas realizan la tarea.

Para poder usarlas debemos conocer, el nombre de la función, los parámetros que debemos pasar, y por supuesto saber que tarea realizan.

Ya hemos visto algunas de estas funciones por ejemplo: ***trim($param)*** y ***fgets(STDIN)*** ,que nos permiten leer los valores ingresados por el usuario. Por supuesto, hay muchas más funciones predefinidas que vamos a ir viendo con el transcurso de la materia.

La totalidad de las funciones predefinidas que pueden usarse en PHP la podemos ver desde la página `http://php.net/manual/es/funcref.php`. Desde ahi se encuentran los distintos enlaces para ver las funciones predefinidas, agrupadas por categorías .

Entre la categoría de funciones predefinidas podemos consultar las funciones matemáticas `http://php.net/manual/es/book.math.php`.

1.3. Documentación

Cada función implementada por un programador realiza una tarea específica. Cuando la cantidad de funciones disponibles para ser utilizadas es grande, puede ser difícil saber exactamente qué hace una función. Es por eso que es extremadamente importante documentar en cada función cuál es la tarea que realiza, cuáles son los parámetros que recibe y qué es lo que devuelve.

La documentación de una función se coloca antes del encabezado de la función, en un párrafo encerrado entre /** */. Así, para la función vista en el ejemplo anterior:

```
1  <?php
2  /**  * Imprime por pantalla un saludo,
3  *dirigido a la persona que se indica por parámetro.
4  * @param string $alguien
5  */
6  function  saludo($alguien){
7                  echo  "\n Hola ". $alguien. " !";
8                  echo  "\n Estoy programando en PHP.";
9  }
10 echo " Ingrese su nombre:";
11 $nombre = trim(fgets(STDIN));
12 saludo($nombre);
13 ?>
```

Cuando una función está correctamente documentada, es posible acceder a su documentación desde el entorno de programación que estamos utilizando

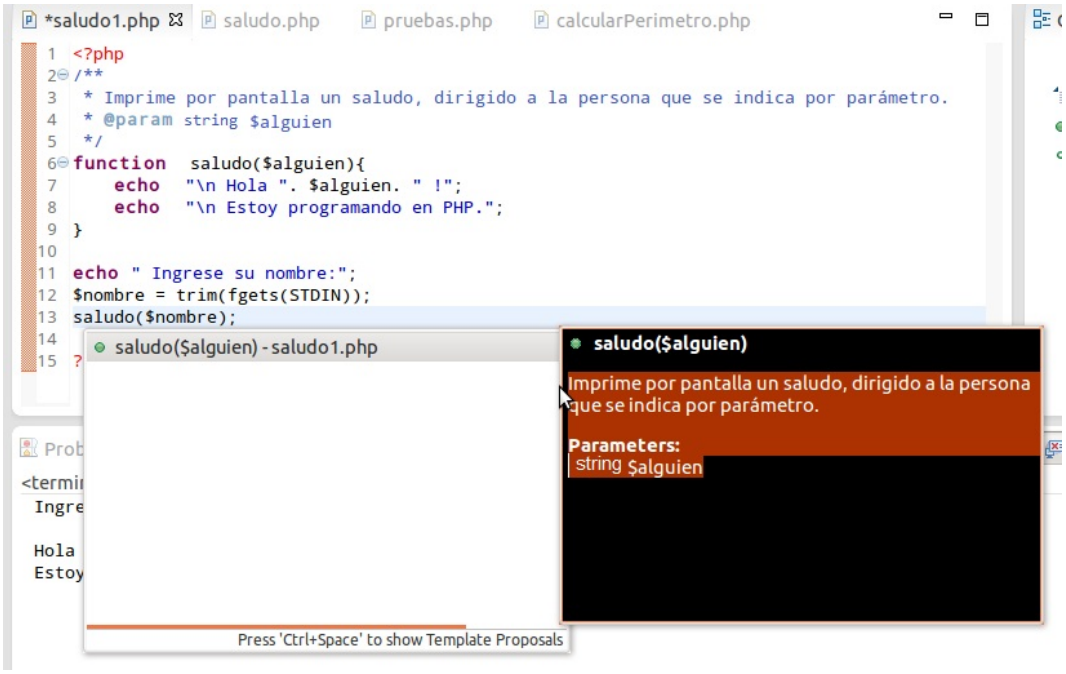

Fig. 1. comentarios desde el entorno de programación Eclipse

1.4. Imprimir versus devolver

A continuación se define una función print_asegundos (horas, minutos, segundos) con tres parámetros (horas, minutos y segundos) que imprime por pantalla la transformación a segundos de una medida de tiempo expresada en horas, minutos y segundos. Para ver si realmente funciona, podemos invocar a la funcion de la siguiente manera: print_asegundos (1, 10, 10), el resultado obtenido de la invocación: *Son 4210 segundos*

```
1  <?php
2  /**  * Transforma en segundos una medida de tiempo
3        *expresada en horas, minutos y segundos
4  * @param int $horas
5  * @param int $minutos
6  * @param int $segundos
7  */
8  function print_asegundos ($horas, $minutos, $segundos){
9          // regla de transformación
10         $segsal = 3600 * $horas + 60 * $minutos + $segundos;
11         echo "Son: ", $segsal, " segundos";
12 }
13 print_asegundos(1,10,10);
14 ?>
```

Contar con funciones es de gran utilidad, ya que nos permite ir armando una biblioteca de instrucciones con problemas que vamos resolviendo, y que se pueden reutilizar para resolver nuevos problemas. Sin embargo, *más útil que tener una biblioteca donde los resultados se imprimen por pantalla, es contar con una biblioteca donde los resultados se devuelven*, para que la gente que usa esas funciones manipule los resultados a su voluntad: los imprima, los use para realizar cálculos más complejos, etc.

```
1  <?php
2  /**  * Transforma en segundos una medida de tiempo
3       * expresada en horas, minutos y segundos
4  * @param  int $horas
5  * @param  int $minutos
6  * @param  int $segundos
7  * @return int
8  */
```

```
9  function  asegundos ($horas, $minutos, $segundos){
10     // regla de transformación
11         $segsal = 3600 * $horas + 60 * $minutos + $segundos;
12 return $segsal;
13 }
14
15 $cantSeg = asegundos(1,10,10);
16 echo  "La cantidad de segundos es:" . $cantSeg;
17 $cantSeg2 =  asegundos(2,20,20);
18 echo "\nLa cantidad de segundos es:". $cantSeg2;
19 $sumSeg = $cantSeg + $cantSeg2;
20 echo "\nLa suma de los segundos es:". $sumSeg;
21 ?>
```

De esta forma, es posible realizar distintas operaciones con el valor obtenido de la función y no solo obtener una visualización de un resultado.

1.5. Cómo usar una función en un programa

Una función es útil porque nos permite repetir la misma instrucción con diferente valores en sus argumentos (si es que los tiene), todas las veces que las necesitemos en un programa.

Para utilizar las funciones definidas anteriormente, escribiremos un programa que pida dos duraciones, y en los dos casos las transforme a segundos y las muestra por pantalla.

1. **Análisis**: El programa debe pedir dos duraciones expresadas en horas, minutos y segundos, y las tiene que mostrar en pantalla expresadas en segundos.
2. **Especificación**:
   —Entradas: dos duraciones leídas de teclado y expresadas en horas, minutos y segundos.
   —Salidas: Mostrar por pantalla cada una de las duraciones ingresadas, convertidas a segundos. Con los valores de las entradas (h, m, s) se calcula 3600 * h + 60 * m + s, y luego se visualiza el resultado por pantalla.
3. **Diseño**:
   —Se tienen que leer dos duraciones (expresadas en horas, minutos y segundos) y para cada una de ellas invocar a la funcion. Para ello repetimos 2 veces lo siguiente:

```
1 Leer cuántas horas tiene el tiempo dado
2     (y referenciarlo con la variable hs)
3 Leer cuántos minutos tiene tiene el tiempo dado
4     (y referenciarlo con la variable min)
5 Leer cuántos segundos tiene el tiempo dado
6     (y referenciarlo con la variable seg)
7 Mostrar por pantalla 3600 * hs + 60 * min + seg
```

Convertir y mostrar por pantalla es exactamente lo que hace nuestra función print_asegundos, por lo que podemos invocar a la función: *print_asegundos*

```
1 Leer cuántas horas tiene el tiempo dado
2     (y referenciarlo con la variable hs)
3 Leer cuántos minutos tiene tiene el tiempo dado
4     (y referenciarlo con la variable min)
5 Leer cuántos segundos tiene el tiempo dado
6     (y referenciarlo con la variable seg)
```

a) El pseudocódigo final queda:

```
1 repetir 2 veces:
2 Leer cuántas horas tiene el tiempo dado
3     (y referenciarlo con la variable hs)
4 Leer cuántos minutos tiene tiene el tiempo dado
5     (y referenciarlo con la variable min)
6 Leer cuántos segundos tiene el tiempo dado
```

```
7       (y referenciarlo con la variable seg)
8  Invocar la función print_asegundos(hs, min, seg)
```

4. **Implementación**: A partir del diseño, se escribe el programa PHP que se muestra en el Código 3.1, que se guardará en el archivo *dos_tiempos.php*.
5. **Prueba**: Probamos el programa con las ternas *(0,1,0)* y *(0,0,1)*:

```php
<?php /**  *
Transforma en segundos una medida de tiempo expresada:
 * en horas, minutos y segundos
 * @param int $horas
 * @param int $minutos
 * @param int $segundos
 * @return int
*/
function  asegundos ($horas, $minutos, $segundos){
    // regla de transformación
        $segsal = 3600 * $horas + 60 * $minutos + $segundos;
        return $segsal;
}
```

```php
echo "␣Cuantas␣horas?";
$hs = trim(fgets(STDIN));
echo "␣Cuantos␣minutos?";
$min = trim(fgets(STDIN));
echo "␣Cuantos␣segundos?";
$se = trim(fgets(STDIN));
$cantSeg = asegundos($hs,$min,$se);
echo  "La␣cantidad␣de␣segundos␣es:" . $cantSeg;

echo "\n␣Cuantas␣horas?";
$hs2 = trim(fgets(STDIN));
echo "␣Cuantos␣minutos?";
$min2 = trim(fgets(STDIN));
echo "␣Cuantos␣segundos?";
$se2 = trim(fgets(STDIN));
$cantSeg = asegundos($hs2,$min2,$se2);
echo  "La␣cantidad␣de␣segundos␣es:" . $cantSeg;
```