# Índice

# 3. Programación de código embebido en linguaxes de marcas

## 3.3 Funcións

Unha **función** é un **conxunto de instrucións para levar a cabo unha tarefa determinada**. A función pode levar un conxunto de **parámetros de entrada e devolver un valor como resultado** da súa execución. En PHP existen **funciones integradas**, é dicir, funciones que **forman parte da librería de PHP**. Tamén existe a **posibilidade** de **definir funciones por parte do usuario**. A continuación estudaranse estes dous tipos de funcións.

Os **nomes das funcións** seguen as mesmas regras dos identificadores de PHP, é dicir, deben comezar por unha letra ou un guión baixo (_) e o resto de caracteres poden ser letras, números ou guións baixos (pódense utilizar caracteres non ingleses como acentos, eñes, etc), pero os nomes de funcións non distinguen entre maiúsculas ou minúsculas.

### Sintaxe

Unha función defínese coa seguinte sintaxe:

```
<?php
function foo ($arg_1, $arg_2, ..., $arg_n)
{
    echo "Funcion de exemplo.\n";
    return $retval;
}
?>
```

### 3.3.1. Funcións integradas

Son un amplo conxunto de funcións que forman parte da librería PHP. Estas funcións utilízanse indicando o seu nome e especificando a lista de parámetros, en caso de requirilos, e procesando o valor devolto, en caso de devolvelo. As funcións integradas poden ser de distinto tipo: manexo de arrays ou cadeas, manexo de datas, funcións HTTP ou de acceso a bases de datos, entre outras.

Un exemplo sinxelo de uso de funcións integradas podería ser a chamada á función strlen pasándolle como parámetro unha cadea, e procesando o valor devolto, é dicir, o número de caracteres da cadea, imprimíndoo por pantalla mediante unha sentencia echo, tal como se viu na actividade 2 da presente unidade didáctica.

Dado que o número de funcións integradas é elevado, e que se engaden novas funcións en cada versión de PHP, neste módulo soamente se estudarán as funcións máis importantes

involucradas nunha aplicación web con acceso a bases de datos. Polo tanto, ao longo das restantes actividades das unidades didácticas iranse describindo e utilizando as funcións na medida que fagan falta.

### 3.3.2. Funcións definidas polo usuario

O programador pode definir as súas propias funcións, indicando o código PHP que desexe que se execute cando sexan chamadas.

Para a creación dunha función usarase a seguinte sintaxe:

**Sintaxe da creación dunha función**

```
function nomeFunción()
{
  Código que será executado
}
```

O **nome da función debe reflectir o que a función fai**. O nome da función pode comezar cunha **letra ou un guión baixo**, pero non un número.

No seguinte exemplo defínese unha función que imprime o nome dunha persoa:

**Exemplo de creación dunha función**

```
<?php
function imprimeNome()
{
  echo "Víctor";
}

echo "O meu nome é ";
imprimeNome();
?>
```

Resultado do exemplo anterior:

**Resultado do exemplo de creación dunha función**

```
O meu nome é Víctor
```

#### Parámetros das funcións

A información pode suministrarse ás funcións mediante a lista de parámetros, unha lista de variables e/ou constantes separadas por comas.

PHP soporta pasar parámetros por valor (o comportamento por defecto), por referencia, e parámetros por defecto.

No seguinte exemplo mostraranse distintos nomes co mesmo apelido:

**Exemplo de función con parámetros**

```
<?php
function imprimeNomeApel($fnome)
{
  echo $fnome . " López.<br>";
}

echo "A miña amiga chámase ";
imprimeNomeApel("Carmen");
echo "A irmá da miña amiga chámase ";
imprimeNomeApel("Leticia");
echo "O irmán da miña amiga chámase ";
imprimeNomeApel("Javier");
```

```
?>
```

O exemplo anterior produce o seguinte resultado:

**Resultado do exemplo de función con parámetros**

```
A miña amiga chámase Carmen López
A irmá da miña amiga chámase Leticia López
O irmán da miña amiga chámase Javier López
```

Exemplo de función con dous parámetros, onde o segundo parámetro é un signo ortográfico:

**Exemplo de función con dous parámetros**

```
<?php
function imprimeNomeApel($fnome, $signoOrtografico)
{
  echo $fnome . " López" . $signoOrtografico . "<br>";
}

echo "A miña amiga chámase ";
imprimeNomeApel("Carmen", ".");
echo "A irmá da miña amiga chámase ";
imprimeNomeApel("Leticia", "!");
echo "O irmán da miña amiga chámase ";
imprimeNomeApel("Javier", "?");
?>
```

O exemplo anterior produce o seguinte resultado:

**Resultado do exemplo de función con dous parámetros**

```
A miña amiga chámase Carmen López.
A irmá da miña amiga chámase Leticia López!
O irmán da miña amiga chámase Javier López?
```

## Pasar parámetros por referencia

Por defecto, os parámetros dunha función pásanse por valor (de maneira que se cambias o valor do argumento dentro da función, non se ve modificado fóra dela)/dela). Se desexas permitir a unha función modificar os seus parámetros, debes pasalos por referencia.

Se queres que un parámetro dunha función sempre se pase por referencia, podes antepor un ampersand (&) ao nome do parámetro na definición da función:

```
<?php
function add_some_extra(&$string)
{
    $string .= ' e algo mais.';
}
$str = 'Esto é unha cadea, ';
add_some_extra($str);
echo $str;  // Mostra 'Esto é una cadena, e algo mais.'
?>
```

### Parámetros por defecto

En PHP pódense definir funcións con argumentos predeterminados, de maneira que se na chamada á función non se envía un argumento, a función asume un valor predeterminado. Loxicamente, os argumentos predeterminados deben ser os últimos na lista de argumentos, para evitar ambigüidades.

Os argumentos predeterminados establécense na definición da función, igualando o nome do argumento ao seu valor predeterminado.

O exemplo seguinte mostra unha función que calcula diferentes tipos de media (aritmética, xeométrica, harmónica). Os argumentos da función son os números cuxa media se debe calcular e o tipo de media a calcular. No exemplo, o tipo de media predeterminado é a media aritmética.

```
<?php
// ESTA ES LA DEFINICIÓN DE LA FUNCIÓN calculaMedia
function calculaMedia($arg1, $arg2, $arg3 = "aritmética")
{
    if ($arg3 == "aritmética") {
        $media = ($arg1 + $arg2) / 2;
    } elseif ($arg3 == "geométrica") {
        $media = sqrt($arg1 * $arg2) / 2;
    } elseif ($arg3 == "armónica") {
        $media = 2 * ($arg1 * $arg2) / ($arg1 + $arg2);
    }
    return $media;
}
// ESTO SON EJEMPLOS DE USO DE LA FUNCIÓN calculaMedia
$dato1 = 12;
$dato2 = 16;
// EL TERCER ARGUMENTO INDICA EL TIPO DE MEDIA A CALCULAR
$media = calculaMedia($dato1, $dato2, "geométrica");
print "<p>La media geométrica de $dato1 y $dato2 es $media.</p>\n";
// AL NO PONER EL TERCER ARGUMENTO, LA FUNCIÓN DEVUELVE LA MEDIA ARITMÉTICA
$media = calculaMedia($dato1, $dato2);
print "<p>La media aritmética de $dato1 y $dato2 es $media.</p>\n";
?>
```

## Devolvendo valores

Os valores retórnanse usando a instrución opcional return. Pode devolverse calquera tipo de valor, incluíndo listas e obxectos.

```
<?php
function square ($num)
{
    return $num * $num;
}
echo square (4);   // mostra '16'.
?>
```

Non se pode devolver múltiples valores desde unha función, pero un efecto similar pódese conseguir devolvendo unha lista.

### 3.3.3. Funcións dentro de funcións

Calquera instrución válida de PHP pode aparecer no corpo da función, incluso outras funciones e definicións de clases.

Por exemplo funcións dentro de funcións:

```
<?php
function foo()
{
  function bar()
  {
    echo "I don't exist until foo() is called.\n";
  }
}
/* We can't call bar() yet
   since it doesn't exist. */
foo();
/* Now we can call bar(),
   foo()'s processesing has
   made it accessible. */
bar();
?>
```

PHP non soporta a redefinición (sobrecarga) de funcións previamente declaradas.

### 3.3.4. Funcións variables

PHP soporta o concepto de funcións variable, isto significa que se unha variable ten un paréntese engada ao final, PHP buscará unha función co mesmo nome que a avaliación da variable, e tentará executala. Entre outras cousas, isto permite implementar retrochamadas (callbacks), táboas de funcións e demais.

As funcións varibles non funcionarán con construcións da linguaxe, tal como *echo(), print(), unset(), isset(), empty(), include(), require()* e derivados. Necesitarase usar unha función propia para utilizar calquera destes construtores como funcións variables.

```
<?php
function foo()
{
    echo "In foo()<br>\n";
}
function bar($arg = '')
{
    echo "In bar(); argument was '$arg'.<br>\n";
}
```

```
// This is a wrapper function around echo
function echoit($string)
{
    echo $string;
}
$func = 'foo';
$func();        // This calls foo()
$func = 'bar';
$func('test');  // This calls bar()
$func = 'echoit';
$func('test');  // This calls echoit()
?>
```

### 3.3.5. Incluír funcións: include e require

**Cando as funcións se atopan nun ficheiro** e **se queren utilizar noutro ficheiro diferente**, é necesario referenciar ao ficheiro que contén as funcións mediante a **función integrada require_once()**. Por exemplo, **se as funcións están no ficheiro a.php, e se desexa utilizalas no ficheiro b.php, en b.php será necesario chamar á función require_once('a.php')**;.

Exercicios.