03

ARRAYS EN PHP

Contenido

Qué es un array 3

Crear un array 3

Imprimir todos los valores de un array: la función print_r() 5

Borrar un array o elementos de un array 6

Contar elementos de un array 7

Máximo y mínimo 9

Encontrar un valor en un array 10

Los arrays como listas 11

Operaciones con arrays 11

- Recorrido de un array usando foreach 12
- Recorrido manejando el cursor interno del array 12
- Recorrido recuperando los índices y valores de un array 14
- Recorrido utilizando array_walk() 14
- Búsqueda de elementos 15
- Recuperación aleatoria de valores de un array 16
- Inserción de elementos 17
- Eliminación de un elemento de un array 18
- Obtención de subarrays 19
- Ordenación de arrays 19

Arrays multidimensionales o matrices 21

Qué es un array

Un array es un tipo de variable que puede **almacenar varios valores a la vez**. En los arrays de una dimensión (que a veces se llaman vectores) cada elemento se identifica por un índice que se escribe entre corchetes ($v[$indice]). Pero los arrays pueden tener más **dimensiones** (como las matrices matemáticas) y entonces los elementos se identifican por varios índices que se escriben cada uno entre corchetes ($v[$indice1][$indice2]...).

En otros lenguajes de programación, los índices de los arrays tienen que ser números enteros positivos y tienen que estar todos definidos, pero en PHP se dice que los arrays son **asociativos** porque **los índices no tienen por qué ser números** y cuando son números no tienen por qué ser valores correlativos.

```php
<?php
$v[5] = 25;
$v[3] = 12;

print "<pre>"; print_r($v); print "</pre>\n";
?>
```

```
Array
(
    [5] => 25
    [3] => 12
)
```

```php
<?php
$v["pepe"]["edad"] = 25;
$v["juan"]["peso"] = 75;

print "<pre>"; print_r($v); print "</pre>\n";
?>
```

```
Array
(
    [pepe] => Array
        (
            [edad] => 25
        )

    [juan] => Array
        (
            [peso] => 75
        )
)
```

Crear un array

Un array se puede crear definiendo algún valor del array.

```php
<?php
$v[5] = 25;

print "<pre>"; print_r($v); print "</pre>\n";
?>
```

```
Array
(
    [5] => 25
)
```

o utilizando la **función array**($indice => $valor, ...):

```php
<?php
$v = array( 5 => 25);

print "<pre>"; print_r($v); print "</pre>\n";
?>
```

```
Array
(
    [5] => 25
)
```

Los arrays en PHP son arrays **asociativos**, es decir, que **los índices no tienen por qué ser números enteros** positivos:

```php
<?php
$v[5] = 25;
$v[-1] = "negativo";
$v["número 1"] = "cinco";

print "<pre>"; print_r($v); print "</pre>\n";
?>
```

```php
<?php
$v = array( 5 => 25, -1 => "negativo", "número 1"
=> "cinco");

print "<pre>"; print_r($v); print "</pre>\n";
?>
```

```
Array
(
    [5] => 25
    [-1] => negativo
    [número 1] => cinco
)
```

Los arrays en PHP pueden ser **multidimensionales**:

```php
<?php
$v[5][3] = 25;
$v["letras"][1] = "letra A";

print "<pre>"; print_r($v); print "</pre>\n";
?>
```

```php
<?php
$v = array(5 => array(3 => 25), "letras" =>
array(1 => "letra A"));

print "<pre>"; print_r($v); print "</pre>\n";
?>
```

```
Array
(
    [5] => Array
        (
            [3] => 25
        )
    [letras] => Array
        (
            [1] => letra A
        )
)
```

Imprimir todos los valores de un array: la función print_r()

La función **print_r**($variable [, $devolver]) escribe la variable $variable de forma legible, incluso aunque se trate de un array. Aunque print_r() genera espacios y saltos de línea que pueden verse en el código fuente de la página, print_r() no genera etiquetas HTML, por lo que el navegador no muestra esos espacios y saltos de línea.

```
<?php
$v = array("nombre" => "Pepito", "apellidos"
=> "Conejo");
print_r($v);
?>
```

```
Array ( [nombre] => Pepito
[apellidos] => Conejo )
```

Para mejorar la legibilidad una solución es añadir la etiqueta *<pre>*, que **fuerza al navegador a mostrar los espacios y saltos de línea**.

```
<?php
$v = array("nombre" => "Pepito", "apellidos" =>
"Conejo");
print '<pre>'; print_r($v); print '</pre>\n';
?>
```

```
Array
(
    [nombre] => Pepito
    [apellidos] => Conejo
)
```

Si el argumento $devolver toma el valor **true**, print_r() no escribe nada pero devuelve el texto que se escribe cuando el argumento no está o toma el valor **false**.

Si el argumento $devolver no está o toma el valor **false**, print_r() devuelve 1 (**true**).

```
<?php
$v = array("nombre" => "Pepito", "apellidos" =>
"Conejo");
$tmp = print_r($v, true);
print "<p>El array es $tmp</p>\n";
?>
```

```
El array es Array ( [nombre] => Pepito
[apellidos] => Conejo )
```

```
<?php
$v = array("nombre" => "Pepito", "apellidos" =>
"Conejo");
$tmp = print_r($v, false);
print "<p>El array es $tmp</p>\n";
?>
```

```
Array ( [nombre] => Pepito [apellidos] =>
Conejo )

La array es 1
```

```
<?php
$v = array("nombre" => "Pepito", "apellidos" =>
"Conejo");
$tmp = print_r($v);
print "<p>El array es $tmp</p>\n";
?>
```

```
Array ( [nombre] => Pepito [apellidos] =>
Conejo )

La array es 1
```

Borrar un array o elementos de un array

La función **unset**() permite **borrar** un array o elementos de un array.

```php
<?php
$v = array( 5 => 25, -1 => "negativo", "número 1" => "cinco");

print "<pre>"; print_r($v); print "</pre>\n";

unset ($v[5]);

print "<pre>"; print_r($v); print "</pre>\n";
?>
```

```
Array
(
    [5] => 25
    [-1] => negativo
    [número 1] => cinco
)

Array
(
    [-1] => negativo
    [número 1] => cinco
)
```

```php
<?php
$v = array( 5 => 25, -1 => "negativo", "número 1" => "cinco");

print "<pre>"; print_r($v); print "</pre>\n";

unset ($v);

print "<pre>"; print_r($v); print "</pre>\n";
?>
```

```
Array
(
    [5] => 25
    [-1] => negativo
    [número 1] => cinco
)

Notice:  Undefined variable: $v in prueba.php on line 8
```

Contar elementos de un array

La función **count**($v) permite contar los elementos de un array.

```php
<?php
$v[3] = 25;
$v[4] = 30;
$v[5] = 35;
$v["letra"] = "letra A";

$elementos = count($v);

print "<p>El array tiene $elementos elementos.</p>\n";
print "<pre>"; print_r($v); print "</pre>\n";
?>
```

```
El array tiene 4 elementos.
```

```
Array
(
    [3] => 25
    [4] => 30
    [5] => 35
    [letra] => letra A
)
```

En un array multidimensional, la función count($v) devolvería simplemente el número de elementos del primer índice:

```
El array tiene 2 elementos.
```

```php
<?php
$v[5][3] = 25;
$v[5][4] = 30;
$v[5][5] = 35;
$v["letra"][1] = "letra A";

$elementos = count($v);

print "<p>El array tiene $elementos elementos.</p>\n";
print "<pre>"; print_r($v); print "</pre>\n";
?>
```

```
Array
(
    [5] => Array
        (
            [3] => 25
            [4] => 30
            [5] => 35
        )
    [letra] => Array
        (
            [1] => letra A
        )
)
```

Para contar todos los elementos de un array multidimensional, habría que utilizar la función **count**($v, **COUNT_RECURSIVE**).

```php
<?php
$v[5][3] = 25;
$v[5][4] = 30;
$v[5][5] = 35;
$v["letra"][1] = "letra A";

$elementos = count($v,
COUNT_RECURSIVE);

print "<p>El array tiene $elementos
elementos.</p>\n";
print "<pre>"; print_r($v); print "</pre>\n";
?>
```

```
El array tiene 6 elementos.
Array
(
    [5] => Array
        (
            [3] => 25
            [4] => 30
            [5] => 35
        )
    [letra] => Array
        (
            [1] => letra A
        )
)
```

Es importante fijarse en que en este caso la función count() está contando también los dos vectores fila. Si quisiéramos contar únicamente los elementos de un array bidimensional habría que restar el número de vectores fila:

```php
<?php
$v[5][3] = 25;
$v[5][4] = 30;
$v[5][5] = 35;
$v["letra"][1] = "letra A";

$elementos = count($v,
COUNT_RECURSIVE) - count($v);

print "<p>El array tiene $elementos
elementos.</p>\n";
print "<pre>"; print_r($v); print "</pre>\n";
?>
```

```
El array tiene 4 elementos.
Array
(
    [5] => Array
        (
            [3] => 25
            [4] => 30
            [5] => 35
        )
    [letra] => Array
        (
            [1] => letra A
        )
)
```

Máximo y mínimo

La función **max**($v, ...) devuelve el valor máximo de un array (o varias). La función **min**($v, ...) devuelve el valor mínimo de un array (o varias).

```php
<?php
$valores = array (10, 40, 15, -1);
$maximo = max($valores);
$minimo = min($valores);

print "<pre>"; print_r($valores); print "</pre>\n";
print "<p>El máximo del array es $maximo.</p>\n";
print "<p>El mínimo del array es $minimo.</p>\n";
?>
```

```
Array
(
    [0] => 10
    [1] => 40
    [2] => 15
    [3] => -1
)
```

El máximo del array es 40. El mínimo del array es -1.

Los valores no numéricos se tratan como 0, pero si 0 es el mínimo o el máximo, la función devuelve la cadena.

```php
<?php
$valores = array (10, 40, 15, 'abc');
$maximo = max($valores);
$minimo = min($valores);

print "<pre>"; print_r($valores); print "</pre>\n";
print "<p>El máximo del array es $maximo.</p>\n";
print "<p>El mínimo del array es $minimo.</p>\n";
?>
```

```
Array
(
    [0] => 10
    [1] => 40
    [2] => 15
    [3] => abc
)
```

El máximo del array es 40.
El mínimo del array es abc.

Encontrar un valor en un array

La función booleana **in_array**($elemento, $v[, $tipo]) devuelve **true** si el elemento se encuentra en el array. Si el argumento booleano **$tipo** es **true**, in_array() comprueba además que los tipos coincidan.

```
<?php
$valores = array (10, 40, 15, -1);


print "<pre>"; print_r($valores); print
"</pre>\n";
if (in_array(15, $valores)) {
  print "<p>15 está en el array
  \$valores.</p>\n";
}
if (!in_array(25, $valores)) {
  print "<p>25 no está en el array
  \$valores.</p>\n";
}
if (!in_array("15", $valores, true)) {
  print "<p>\"15\" no está en el array
  \$valores.</p>\n";
}
?>
```

```
Array
(
[0] => 10 [1] => 40
[2] => 15 [3] => -1
)
```

```
15 está en dl array $valores.
25 no está en el array $valores.
"15" no está en el array $valores
```

Los arrays como listas

Una lista es una secuencia de elementos en la que cada uno de ellos ocupa una posición determinada dentro de la misma.

Cuando con PHP definimos los elementos de un *array*, no sólo asociamos valores a índices, sino que también estamos definiendo una lista de elementos, donde cada elemento es un par formado por el índice y su valor asociado.

Por ejemplo:

```
$a = array("Lunes" => 1,
           100     => "Cien",
           "Ten"   => "Diez");
```

Estamos definiendo una lista de elementos que, gráficamente, se puede representar como aparece a continuación:

Índice	"Lunes"
Valor	1

Primer elemento

Índice	100
Valor	"Cien"

Segundo elemento

Índice	"Ten"
Valor	"Diez"

Tercer elemento

Operaciones con arrays

Recorrido de un array escalar secuencial

Un *array* escalar secuencial se caracteriza porque los índices de sus elementos son los valores comprendidos entre 0 y N-1, siendo N el número de elementos del *array*. PHP nos ofrece la función ***count()*** que determina el número de elementos de un *array* y que en combinación con el bucle *for* permite procesar todos los elementos.

Ejemplo:

```
<html>
<head><title>Recorrido de un array</title></head>
<body>
<?php
   $a = array("Enero", "Febrero", "Marzo", "Abril", "Mayo");
   echo "<table border='1' align='center'> \n";
   echo "<tr bgcolor='orange'>\n",
        "<th>Índice</th>\n",
        "<th>Valor</th>\n",
        "</tr>\n";
   // Generamos las filas con un bucle for
   for ($i=0; $i < count($a); $i++)
```

```
            echo "<tr aling='center'>\n",
                 "<td>", $i, "</td>\n",
                 "<td>", $a[$i], "</td>\n",
                 "</tr>\n";
    echo "</table>\n";
?>
</body>
</html>
```

El resultado sería el siguiente:

Índice	Valor
0	Enero
1	Febrero
2	Marzo
3	Abril
4	Mayo

Recorrido de un array usando foreach

La construcción *foreach* de PHP permite recorrer un *array* como se fuese una lista, procesando desde el primero al último elemento de la misma. Maneja un cursor interno que va accediendo a cada uno de los elementos siguiendo el orden en que se han definido:

Índice	"Lunes"
Valor	1

Índice	100
Valor	"Cien"

Índice	"Ten"
Valor	"Diez"

Por ejemplo:

```
foreach($a as $mivalor)
   echo $mivalor;
```

Recorrido manejando el cursor interno del array

PHP maneja un cursor o puntero interno para el recorrido de los *arrays*. Además proporciona una serie de funciones relacionadas con la manipulación de dicho cursor:

- **reset($a)**. Sitúa el cursor en el primer elemento del *array* y devuelve el valor de dicho elemento. Si el *array* está vacío devuelve *false*.
- **next($a)**. Avanza el cursor al siguiente elemento del *array* y devuelve el valor de dicho elemento. Si el *array* está vacío o no hay más elementos devuelve *false*.

- **prev($a)**. Retrocede el cursor al anterior elemento del *array* y devuelve el valor de dicho elemento. Si el *array* está vacío o no hay más elementos devuelve *false*.
- **end($a)**. Avanza el cursor al último elemento del *array* y devuelve el valor de dicho elemento. Si el *array* está vacío devuelve *false*.
- **current($a)** o **pos($a)**. Devuelve el valor del elemento situado en la posición actual del cursor. Si el *array* está vacío o no hay más elementos devuelve *false*.
- **key($a)**. Devuelve el índice del elemento situado en la posición actual del cursor. Si el *array* está vacío o no hay más elementos devuelve *false*.

Ejemplo:

```
<html>
<head><title>Recorrido de un array manejando el cursor interno</title></head>
<body>
<?php
   $a = array("Enero", "Febrero", "Marzo", "Abril", "Mayo");
   echo "<table border='1' align='center'> \n";
   echo "<tr bgcolor='orange'>\n",
         "<th>Índice</th>\n",
         "<th>Valor</th>\n",
         "</tr>\n";
   // Generamos las filas con un bucle do .. while
   do {
          $indice = key($a);
          $valor = current($a);
          echo "<tr align='center'>\n",
                "<td>", $indice, "</td>\n",
                "<td>", $valor , "</td>\n",
                "</tr>\n";
   } while (next($a));
   echo "</table>\n";
?>
</body>
</html>
```

Recorrido recuperando los índices y valores de un array

Existen dos funciones:

- **array_values($a)**. Devuelve todos los valores del *array* guardándolos en un *array* escalar secuencial.
- **array_keys($a)**. Devuelve todos los índices del *array* guardándolos en un *array* escalar secuencial.

Ejemplo:

```
<html>
<head><title>Recorrido de un array</title></head>
<body>
<?php
   $a = array("Enero", "Febrero", "Marzo", "Abril", "Mayo");
   echo "<table border='1' align='center'> \n";
   echo "<tr bgcolor='orange'>\n",
         "<th>Índice</th>\n",
         "<th>Valor</th>\n",
         "</tr>\n";
   // Generamos las filas con un bucle for que recorre $indices
   $indices = array_keys($a);
   $valores = array_values($a);
   for($i=0; $i < count($indices); $i++)
          echo "<tr align='center'>\n",
                "<td>", $indices[$i], "</td>\n",
                "<td>", $valores[$i] , "</td>\n",
                "</tr>\n";
   echo "</table>\n";
?>
</body>
</html>
```

Recorrido utilizando array_walk()

La función ***array_walk()*** permite aplicar una función, que ha sido definida por el usuario, a cada uno de los elementos del *array*. Para poder usar este recurso de PHP es necesario proporcionar dos parámetros: una variable de tipo *array* y un identificador de una función. El siguiente código aplicará la función *mifuncion()* a todos los elementos de $a:

```
array_walk($a, 'mifuncion')
```

donde *mifuncion()* debe recibir dos parámetros que se corresponden con el valor y el índice de cada elemento del array. Por ejemplo:

```
function mifuncion($valor, $indice) {
   echo "Índice: ", $indice, ", valor: ", $valor, "<br />\n";
}
```

Ejemplo:

```
<html>
<head><title>Recorrido de un array</title></head>
<body>
<?php
    function escribir_elemento($valor, $indice){
            echo "<tr align='center'>\n",
                    "<td>", $indice, "</td>\n",
                    "<td>", $valor , "</td>\n",
                    "</tr>\n";
    }
    $a = array("Enero", "Febrero", "Marzo", "Abril", "Mayo");
    echo "<table border='1' align='center'> \n";
    echo "<tr bgcolor='orange'>\n",
            "<th>Índice</th>\n",
            "<th>Valor</th>\n",
            "</tr>\n";
    // Generamos las filas con la función array_walk() que llama a
    // escribir_elemento para cada elemento del array
    array_walk($a, 'escribir_elemento');
    echo "</table>\n";
?>
</body>
</html>
```

Para poder modificar los valores contenidos en el *array* es necesario que en la función definida por el usuario se especifique que el parámetro donde se proporciona cada valor es pasado por referencia:

```
function mifuncion(&$valor, $indice) {
    $valor = $valor + 5;
}
```

Búsqueda de elementos

Para ello utilizamos la función ***in_array()***, que recibe dos parámetros (el valor buscado y el *array*) y devuelve un valor *booleano* (*true* si está, *false* si no está). Hay otro tercer parámetro opcional que si toma el valor *true* hace que las comparaciones sean más estrictas y también se tenga en cuenta el tipo de los datos.

```
$a = array(1, 2, 3, 4);
$encontrado1 = in_array('2', $a);          // Devuelve true
$encontrado2 = in_array('2', $a, true);    // Devuelve false
```

La función *array_search()* devuelve el índice correspondiente al valor buscado:

```
$a = array(1, 2, 3, 4);
$encontrado1 = array_search('2', $a);          // Devuelve 1
$encontrado2 = array_search('2', $a, true);    // Devuelve false
```

La función *array_key_exists()* comprueba si el índice, que se pasa como primer parámetro, pertenece al array, que se pasa como segundo parámetro.

Recuperación aleatoria de valores de un array

La función ***array_rand()*** recibe como parámetro un *array* y devuelve uno de sus índices elegido al azar. Si se quiere obtener más de un índice habrá que proporcionar a la función el número deseado en un segundo parámetro. En este caso, el resultado se guardará en un *array* auxiliar.

En ejemplo siguiente desarrollamos la página inicial de un portal de salas de cine y deseamos que en ella aparezcan las imágenes de presentación de cuatro de las películas que se exhiben actualmente. Además estas imágenes cambian aleatoriamente cada vez que se carga la página:

```
<html>
<head><title>Incorporación aleatoria de imágenes</title></head>
<body bgcolor="orange">
<?php
   // Guardamos en una variable el directorio donde están las imágenes
   $directorio="imagenes";
   // Guardamos en un array los nombres de las imágenes
   $a=array("pelicula1.jpg",  "pelicula2.jpg",  "pelicula3.jpg",
            "pelicula4.jpg",  "pelicula5.jpg",  "pelicula6.jpg",
            "pelicula7.jpg", "pelicula8.jpg", "pelicula9.jpg");
   // Obtenemos aleatoriamente 4 claves del array
   $claves = array_rand($a, 4);
?>
<!-- Para realizar el diseño de la página usamos una tabla -->
<table width="100%" height="100%" cellspacing=0 cellspacing=0>
   <tr>
      <!-- imagen superior izquierda -->
      <td align='left' valign="top">
         <img width="180" height="250" src="<?php echo $directorio, '/', $a[$claves[0]]; ?>" />
      </td>
      <!-- Contenido central -->
      <td rowspan=2 align='center' valign="middle">
         <h1>GRAN CINE</h1><br />
         <h2>ILUSION</h2><br />
         <h3>Pase a conocer nuestra Cartelera</h3>
      </td>
         <!-- imagen superior derecha -->
         <td align='right' valign="top">
            <img width="180" height="250" src="<?php echo $directorio, '/', $a[$claves[1]]; ?>" />
      </td>
   </tr>
   <tr>
      <!-- imagen inferior izquierda -->
      <td align='left' valign="bottom">
         <img width="180" height="250" src="<?php echo $directorio, '/', $a[$claves[2]]; ?>" />
      </td>
      <!-- imagen inferior derecha -->
```

```
        <td align='right' valign="bottom">
            <img width="180" height="250" src="<?php echo $directorio, '/', $a[$claves[3]]; ?>" />
        </td>
    </tr>
</table>
</body>
</html>
```

Inserción de elementos

Sabemos que un *array* es una lista de elementos y sabemos que podemos incluir nuevos elementos al final de la lista simplemente asignando un valor a un elemento de un *array* sin especificar un índice.

```
$a[] = "nuevo valor"
```

También se puede hacer con la función ***array_push()***. Se pueden insertar varios elementos a la vez.

```
int array_push(array a, mixed var [, ...])
```

Para insertar elementos al principio de la lista:

```
int array_unshift(array a, mixed var [, ...])
```

Otra posibilidad es la de rellenar un *array* por la izquierda o la derecha con un determinado valor de por medio:

```
array array_pad(array entrada, int tamanio, mixed relleno)
```

El primer parámetro es un *array,* el segundo es el tamaño hasta donde hay que rellenar y el tercero es el valor de relleno. Si el tamaño es positivo el *array* se rellena por la derecha y si es negativo por la izquierda.

```
<?php
$input=array(12,10,9);
$result=array_pad($input,5,0);                    // El resultado es array(12, 10, 9, 0, 0)
$result=array_pad($input,-7,-1);                  // El resultado es array(-1, -1, -1, -1, 12, 10, 9)

?>
```

Eliminación de un elemento de un array

En PHP la función ***unset()*** elimina elementos de un *array*:

```
unset($a[1]);
```

La eliminación de una elemento no altera el orden relativo del resto de elementos. También se posible eliminar todo el array:

```
unset($a);
```

También disponemos de otras funciones:

- **mixed array_pop(array a)**. Borra el último elemento devolviendo su valor. Si el *array* está vacío devuelve NULL.
- **mixed array_shift(array a)**. Borrar el primer elemento. Además realiza una reasignación de índices numéricos para empezar desde cero. Los índices de tipo cadena de caracteres no se modifican. Si el *array* está vacío devuelve NULL.
- **array array_splice(array a, int posicion [, int tamanio [, array sustitutos]])**. Elimina una parte del *array* o sustituye una parte por otra. Parámetros:
    - *a* es el *array* sobre el que se va a operar.
    - *posicion* indica la posición desde la cual se realiza el borrado o sustitución. Los valores positivos hacen referencia a posiciones del principio del *array* y los negativos a posiciones del final del *array*.
    - *tamanio* es opcional. Si no se indica se borran o sustituyen todos los elementos desde la posición indicada por el parámetro *posicion*. Si es positivo indica el número de elementos a borrar o sustituir. Si es negativo señala la posición desde el final del *array* donde hay que dejar de borrar o sustituir.
    - *sustitutos* es opcional y proporciona el *array* que va a sustituir a los elementos borrados.

      La función *array_splice()* devuelve la porción de *array* que ha sido eliminado reasignando los índices del mismo para comenzar en cero.

- **array_unique()**. Elimina los valores repetidos de un *array*. Devuelve un *array* donde se han eliminado los valores repetidos.

Obtención de subarrays

- **array array_filter(array entrada, 'filtro')**. Devuelve un *subarray* formado por los elementos que cumplen una determinada condición. *'filtro'* es el identificador de una función *filtro()* que hay que definirse y que recibe los valores del *array* y que devuelve verdadero o falso dependiendo de si el elemento cumple o no la condición.
- **array array_slice(array a, int posicion [, int tamanio [, bool conservar_indices]])**. Devuelve una parte del *array* considerando la posición que ocupan sus elementos. Consta de los siguientes parámetros:
    - *a* es el *array* sobre el que se va a operar.
    - *posicion* indica la posición desde la cual se obtiene el *subarray*. Los valores positivos hacen referencia a posiciones del principio del *array* y los negativos a posiciones del final del *array*.
    - *tamanio* es opcional. Si no se indica se devuelven todos los elementos desde la posición indicada por el parámetro *posicion*. Si es positivo indica el número de elementos que contendrá el *subarray*. Si es negativo la secuencia se detendrá a tantos elementos desde el final del *array* como indique.
    - *conservar_indices* es opcional y si toma el valor *true* hace que se conserven los índices originales. En caso contrario, se reasignan los índices numéricos comenzando con el cero.
- **array array_chunk(array a, int tamanio [, bool conservar_indices])**. Divide el *array* en un conjunto de segmentos del mismo tamaño.
    - *a* es el *array* sobre el que se va a operar.
    - *tamanio* indica el número de elementos de cada segmento o *subarray* generado.
    - *conservar_indices* es opcional y si toma el valor *true* hace que se conserven los índices originales. En caso contrario, se reasignan los índices numéricos comenzando con el cero en cada *subarray*.

Ordenación de arrays

Para ordenar un array por sus valores tenemos:

```
bool sort(array a[, int criterio]);
```

donde el primer parámetro es el array a ordenar y el segundo es el criterio de ordenación, que puede ser:

- SORT_REGULAR. Los elementos se comparan según las reglas definidas por PHP.
- SORT_NUMERIC. Los elementos se comparan como si fueran números.
- SORT_STRING. Los elementos se comparan como si fueran cadenas de caracteres.
- SORT_LOCALE_STRING. Los elementos se comparan según se haya definido en la máquina donde se ejecuta el script PHP.