Las Arrays en PHP

Concepto de Array

- Las variables que hemos visto hasta ahora eran escalares.
- Una variable escalar almacena un solo valor.
- Un array almacena un conjunto o secuencia de valores escalares (u otro array), no necesariamente del mismo tipo.
- Para hacer debug de un array podemos usar print_r() o var_dump()
- El array puede considerarse como un valor que se asocia a uno o varios índices.
- Ejemplo:

```
<?php
$el_array = array('Enero', 10.3 , TRUE);
print_r($el_array);
?>
```

Ejecución del ejemplo:

```
Array ( [0] => Enero [1] => 10.3 [2] => 1 )
```

Tipos de arrays

Criterio	Tipos	Ejemplo
Según su dimensión	vectores: arrays unidimensionales	`$a[1]= "rojo";` `$a[3]= "azul";`
	matrices: arrays multidimensionales	`$a[3]["color"]="azul";` `$a[5]["color"][7]= "verde";`

Según el índice	Escalares (indexados numéricamente)	`$a[0]="rojo";`
		`$a[1]="verde";`
		`$a[2][3]="azul";`

	Asociativos	`$a["nom"]="Carlos";`
		`$a["apellido"]="Mas";`
		`$a["hobby"]="leer";`
		
		o
		
		`$a= array ("nom" => "Alfonso",`
		`"apellido"=>"Mas",`
		`"hobby" => "leer");`
		
		`$b["horiz"]["vert"]="XA";`

Las Arrays en PHP

Ejemplos de Arrays

```
<?php
$frutas = array ("n"=>"naranja","p"=>"platano","m"=>"manzana");
$numeros = array(1, 2, 3, 4, 5, 6);
$puestos = array("primero", 5 => "segundo", "tercero")
?>

Salida de debug:
print_r($frutas);print_r($numeros);print_r($puestos);

Array ( [n] => naranja [p] => platano [m] => manzana )
Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5 [5] => 6 )
Array ( [0] => primero [5] => segundo [6] => tercero )
```

Acceso a los elementos

```
echo $frutas["n"] . '<br/>'; //naranja
echo $frutas['n'] . '<br/>'; //naranja
echo $frutas[n] . '<br/>'; //naranja
echo $frutas[0] . '<br/>'; //
echo $numeros[0] . '<br/>'; //1
echo $puestos[5] . '<br/>'; //segundo
echo $puestos["5"] . '<br/>'; //segundo
echo $puestos['5'] . '<br/>'; //segundo
```

Arrays Asociativos de una dimensión

- Una de las particularidades de PHP es la posibilidad de poder sustituir los números de índice por alguna palabra que se asocie a cada elemento.
- Los arrays asociativos permiten asociar una clave a cada valor almacenado.
- Ejemplo. Son inicializaciones equivalentes:

```
$precios=array("Cebolla"=>100,"Aceite"=>10,"Tomate"=>4;

$precios = array("Cebolla"=>100);
$precios["Aceite"] =10;
$precios["Tomate"] =4;

$precios["Cebolla"] =100;
$precios["Aceite"] =10;
$precios["Tomate"] =4;
```

Arrays indexados numéricamente de una dimensión

En PHP, los arrays indexados numéricamente comienzan en cero.

Ejemplos:

```
$animales = array("Tigre","Gato","Perro");
```

Equivalente a:

```
$animales[0] = "Tigre";
$animales[1] = "Gato";
$animales[2] = "Perro";
```

Añadir nuevos campos:

```
$animales[3]= "Conejo";
$animales[]= "León";
```

<table>
<tr><td>Funciones: range, unset y array_values</td><td>

- Define un rango de valores:

$numeros = range(2,5); // $numeros = array (2, 3, 4, 5);
$numeros = range(5,2); // $numeros = array (5, 4, 3, 2);

- La función unset() permite eliminar un elemento de una matriz o la matriz entera.

$matriz = array(5 => 1, 12=> 2);
$matriz[] = 56; // Esto es igual que $matriz[13] = 56;
unset($matriz[5]); // Esto elimina el primer elemento de la matriz
unset($matriz); // Esto elimina la matriz completa

- Con unset() de un elemento de la matriz, la matriz NO es re-indexada. Hay que usar array_values().

$a = array(1 => 'uno', 2 => 'dos', 3 => 'tres');
unset($a[2]); // $a = array(1 => 'uno', 3 => 'tres');
$b = array_values($a);
print_r ($b); // Array(0 => 'uno', 1 =>'tres')

</td></tr>
<tr><td>Conversiones a Array</td><td>

Para cualquiera de los tipos: (integer, float, string, boolean, resource) si convierte un valor a un array, se obtiene un array con un elemento (con índice 0), el cual es el valor escalar con el que inició.

<?php
$mi_array = 20;
$a=(array) $mi_array;
print_r ($a);
?>

Resultado: Array ([0] ⇒ 20)

Si convierte un valor NULL a matriz, obtiene un array vacío.

</td></tr>
<tr><td>Paso de arrays como parámetros en funciones</td><td>Se pueden modificar los valores de los arrays dentro de las funciones, pasando los arrays como parámetros por referencia.</td></tr>
</table>

Acceso de Arrays – Funciones de recorrido de un array

Cada vector tiene un contador interno que permite recorrer los elementos del mismo.
No son funciones seguras si el vector contiene elementos vacíos.

Funciones:

reset()	Hace que el puntero interno apunte al primer elemento del array.
end()	Hace que el puntero interno apunte a la última posición del array.
each()	Recupera el par formado por la clave y el valor del elemento actual y además avanza una posición el puntero del array. Devuelve false cuando no hay más elementos que tratar.
next()	Permite ir al siguiente elemento. En el caso de estar en el último elemento devuelve false.
prev()	Permite ir al elemento anterior. En el caso de estar al principio, devuelve false.
current	Devuelve el contenido del elemento actual. Esta función no desplaza el puntero interno de posición y devuelve el valor false cuando se encuentre después del último elemento o si la matriz no tiene elementos.
list()	Asigna los valores del elemento actual de una matriz a las variables que se hayan pasado como parámetro.
key()	Devuelve la clave o índice del elemento actual.

Función each()

Devuelve el par clave/valor actual para el vector y avanza el cursor del mismo.
Esta pareja se devuelve en un vector de 4 elementos, con las claves 0, 1, key, y value.
Los elementos 0 y key contienen el nombre de clave del elemento del vector, y 1 y value contienen los datos.

Si el puntero interno para el vector apunta pasado el final del contenido de la matriz, each() devuelve false.

Ejemplo 1:

```
$animales = array ("perro", "gato", "paloma", "hormiga");
$animal = each ($animales);
print_r($animal);

Array
(
[1] => perro
[value] => perro
[0] => 0
[key] => 0
)
```

<table>
<tr><td></td><td>Ejemplo 2:

<pre>$animales = array ("Mamifero" => "Ballena","Insecto" => "Hormiga");
$animal = each ($animales);
print_r($animal);

Array
(
[1] => Ballena
[value] => Ballena
[0] => Mamifero
[key] => Mamifero
)</pre></td></tr>
<tr><td>Función list()</td><td>Se usa para asignar una lista de variables en una sola operación.
Como array(), ésta no es realmente una función, sino una construcción del lenguaje.

Ejemplo 1:

<pre>list($a, $b, $c) = array (1, 2, 3); //$a=1 $b=2 $c=3</pre>

Ejemplo 2:

<pre>$precios = array("Cebolla"=>100,"Aceite"=>10,"Tomate"=>4);
reset ($precios);
while (list($producto,$precio)=each($precios))
   echo "$producto - $precio
";

//Cebolla - 100
//Aceite - 10
//Tomate - 4</pre>

Asigna los campos 0 y 1 devueltos por each() a las variables $producto y $precio
Ejemplo 3:
<pre>$precios = array("Cebolla"=>100,"Aceite"=>10,"Tomate"=>4);
reset ($precios);
while($elemento = each($precios))
{
echo $elemento["key"];
echo " - " ;
echo $elemento["value"];
echo "<br/>";
}

//Cebolla - 100
//Aceite - 10
//Tomate - 4</pre></td></tr>
</table>

foreach	foreach (expresion_array as $key => $value) sentencia
	
	Recorre la lista de claves y valores devueltos por la expresión expresion_array.
	La construcción foreach hace una copia del vector devuelto.
	Comienza siempre desde el principio del vector.
	No hace falta hacer reset antes de comenzar.
	No modifica el puntero interno sobre el vector original.
	Ejemplo:
	$arr = array ("Mamifero" => "Ballena","Insecto" => "Hormiga");
	reset($arr);
	while(list(,$valor)=each($arr)) echo "Valor: $valor<br \>\n";
	Resultado:
	Valor: Ballena
	Valor: Hormiga
	
	foreach($arr as $value) echo "Valor: $value<br \>\n";
	Resultado:
	Valor: Ballena
	Valor: Hormiga
	
	Ejercicio 1: Mostrar sólo los índices (usando el while) del ejer anterior.
	
	Ejercicio 2: Mostrar los índices y valores del ejercicio anterior usando el bucle while y foreach.
	Mostrar los resultados.

Recorrido de un array Unidimensional Escalar	

```php
<?php
//RECORRIDO DE UN ARRAY UNIDIMENSIONAL ESCALAR (INDEXADO).



//1.- RECORRIDO DE UN ARRAY SIMPLE MEDIANTE USO DE VARIABLES.
$productos = array ("MESA", "SILLA", "FLEXO", "ESTANTERIA");
echo "<br\>";
echo "$productos[0]<br \>";
echo "$productos[1]<br \>";
echo "$productos[2]<br \>";
echo "$productos[3]<br \>";
echo "<br>";echo "<br \>";




//2.-RECORRIDO DE UN ARRAY CON FOR, CONOCIENDO EL NÚMERO DE
ELEMENTOS A PRIORI Y SIN CONOCER.
for ($i=0;$i<=3;$i++) echo "$productos[$i] <br \>";
echo "<br>";
for ($i=0;$i<count($productos);$i++)
echo "$productos[$i] <br \>";
echo "<br \>";



//3. RECORRIDO DE UN ARRAY CON EL BUCLE WHILE.
reset($productos);
while( list($clave,$valor)=each($productos))
      echo "Clave: $clave; Valor: $valor<br \>\n";



//4.RECORRIDO DE UN ARRAY CON FOREACH
foreach($productos as $clave => $valor)
      echo "Clave: $clave; Valor: $valor<br \>\n";
echo "<br>";

?>
```

<table>
<tr><td>Recorrido de un Array Unidimensional Asociativo</td><td>

Ejercicio 3:

Realizar:

$productos = array ("M"=>"MESA","S"=>"SILLA","F"=>"FLEXO","E"=>"ESTANTERIA");

- Recorrido del array mediante el uso de variables.
- Recorrido del array mediante el uso del bucle while.
- Recorrido del array mediante el uso de foreach.

</td></tr>
<tr><td>Ordenación de Vectores Indexados numéricam.</td><td>

void sort (array vector)

Ordena los valores del vector de menor a mayor sin mantener la asociación de índices.

Útil para vectores con índices numéricos.

```
$productos = array("Peras", "Limones", "Melones", "Manzanas" );
sort($productos);
print_r($productos);
```

Resultado:

```
Array ( [0] => Limones [1] => Manzanas [2] => Melones [3] => Peras )
```

void rsort (array vector)

Ordena los valores del vector de mayor a menor sin mantener la asociación de índices.

```
$productos = array("Peras", "Limones", "Melones", "Manzanas" );
rsort($productos);
print_r($productos);
```

Resultado:

```
Array ( [0] => Peras [1] => Melones [2] => Manzanas [3] => Limones )
```

</td></tr>
<tr><td>Ordenación de Vectores Asociativos</td><td>

void asort (array vector)

Ordena una vector por sus valores, manteniendo la asociación de índices.

```
$precios= array("Peras"=>120, "Manzanas"=>150, "Limones"=>100);
asort($precios);
print_r($precios);
```

Resultado:

```
Array ( [Limones] => 100 [Peras] => 120 [Manzanas] => 150 )
```

</td></tr>
</table>

<table>
<tr><td></td><td>void ksort (array vector)

Ordena una vector por sus claves, manteniendo la asociación de índices.

$precios= array("Peras"=>120, "Manzanas"=>150, "Limones"=>100);
ksort($precios);
print_r($precios);

Resultado:
Array ([Limones] => 100 [Manzanas] => 150 [Peras] => 120)

arsort() y krsort() ordenan en sentido inverso.</td></tr>
<tr><td>array_walk</td><td>int array_walk (array vector, string func,mixed datos_usuario)

Aplica la función llamada func a cada elemento del vector.
La función func recibirá el valor de la matriz como primer parámetro y la clave como segundo.
Si se proporciona el parámetro datos_usuario se le pasará como tercer parámetro a la función de usuario.

Ejemplo:

$frutas = array ("d"=>"limon", "a"=>"naranja", "b"=>"platano", "c"=>"manzana");

function test_alterar (&$item1, $clave, $prefix)
{ $item1 = "$prefix: $item1";}

function test_ver ($item2, $clave)
{ echo "$clave. $item2
\n";}

echo "Antes...:\n
";
array_walk ($frutas, 'test_ver');
array_walk ($frutas, 'test_alterar', 'fruta');

echo "...y despues:\n
";
array_walk ($frutas, 'test_ver');

Antes...:
d. limon
a. naranja
b. platano
c. manzana
...y despues:
d. fruta: limon
a. fruta: naranja
b. fruta: platano
c. fruta: manzana</td></tr>
</table>

<table>
<tr><td>Extract</td><td>void extract (array vector_vars [, int tipo_extraccion [, string prefijo]])

Para cada par clave/valor creará una variable en la tabla de símbolos actual, sujeto a los parámetros tipo_extraccion y prefijo.

Controla las colisiones con las variables que ya existen. La forma de tratar éstas se determina por el tipo_ extracción.

Los valores útiles son:
• EXTR_OVERWRITE (predeterminado) Si hay colisión, sobrescribe la variable existente.
• EXTR_PREFIX_ALL Añade el prefijos todas las variables.

<pre>$array =
array("key1"=>"valuel","key2"=>"value2","key3"=>"value3");
extract($array);
echo "$key1 $key2 $key3"; //Resultado: valuel value2 value3</pre></td></tr>
<tr><td>Compact</td><td>array compact (mixed varname [, mixed ...])

A partir de los nombres de vars que se le pasan, crea una matriz que contiene variables y sus valores

Ejemplo:
<pre>$nombre = "Francisco";
$profesion = "Profesor";
$aficion = "Cine";
$resultado = compact ("nombre", "profesion","aficion");</pre>Resultado:
Array ([nombre] => Francisco [profesion] => Profesor [aficion] => Cine)</td></tr>
<tr><td>Array_count_values</td><td>array array_count_values (array entrada)

Devuelve un vector usando los valores de la matriz entrada como claves y su frecuencia de aparición en la entrada como valores.

Ejemplo:

<pre>$array = array(4, 5, 1, 2, 3, 1, 2, 1);
$ac = array_count_values($array);
print_r($ac);</pre>Resultado:
Array ([4] => 1 [5] => 1 [1] => 3 [2] => 2 [3] => 1)</td></tr>
</table>

Explode()

Divide una cadena por algún carácter o cadena, formando un array con los distintos elementos.

```php
$pizza = "trozo1 trozo2 trozo3";
$trozos = explode(" ", $pizza);
echo $trozos[0]; // trozo1
echo $trozos[1]; // trozo2
echo $trozos[2]; // trozo3
print_r($trozos);
```

Resultado:
Array ([0] => trozo1 [1] => trozo2 [2] => trozo3)

Implode()

Une los elementos de un array formando una cadena.

```php
$array = array('apellido', 'email', 'telefono');
$separado_por_comas = implode(",", $array);
echo $separado_por_comas; // apellido,email,telefono
print_r($separado_por_comas);//apellido,email,telefono
```

Split()

Divide la cadena en elementos de un array según una expresión regular

```php
$date = "29/01/1990";
//tambien valdrían $date = "29-01-1990"; $date = "29.01.1990";
list( $dia, $mes, $año ) = split( '[/.-]', $date );
echo "Dia: $dia; Mes: $mes; Año: $año<br>\n";
```

Más funciones

array = **array_keys**(array)
Devuelve un vector con todas las claves de un vector.

array = **array_values**(array)
Devuelve un vector con todos los valores de un vector.

boolean = **in_array**(variable,array)
Verifica si existe un elemento en un array.

array = **range**(min,max)
Crea un vector con un rango desde min hasta max.

array = **array_reverse**(array)
Invierte un array.

boolean = **shuffle**(array)
Desordena un array.
Devuelve true si desordena el array y false en caso contrario.

array = **array_merge**(array1, array2, ...)
Unión de arrays.

Ejercicio 4:
Realizar las pruebas de las funciones anteriores.
Mostrar los resultados.

Manejo de Pilas	variable=**array_pop**(array) Saca y elimina el último elemento. **array_push**(array, variable) Agrega al final. **Ejercicio 5:** Realizar un ejemplo de uso de Pilas mediante arrays. Mostrar el resultado.
Manejo de Colas	variable=**array_shift**(array) Saca y elimina el primer elemento. **Array_unshift**(array,variable) Agrega al principio desplazando el resto **Ejercicio 6:** Realizar un ejemplo de uso de Colas mediante arrays, usando algunas de las cuatro últimas funciones vistas. Mostrar el resultado.