Ordenaciones de arrays

Los elementos de un array se van ordenando según se van definiendo. Por tanto, su orden *no es el mismo* que el de los *valores de sus índices*.

Las funciones PHP que ordenan los elementos de un array permiten dos opciones.

Con una de ellas es posible la ordenación de los elementos **sin modificar los valores de los índices**, mientras que la otra **sí modifica los índices**.

En el segundo de los casos la modificación puede afectar incluso al *tipo de índices* dado que **los resultados** de las ordenaciones – tanto si hemos partido de un array *escalar* como si lo hemos hecho desde uno *asociativo*– es siempre un array **escalar**.

Ordenación por valores sin mantener índices

sort(*array***)**

Ordena los valores del *array* en sentido *creciente* y lo *reindexa* asignando índice CERO al menor de los valores.

rsort(*array***)**

Ordena la matriz en sentido *decreciente* de sus valores y la *reindexa* asignando índice CERO al mayor de estos.

Ordenación por índices

ksort(*array***)**

Ordena la matriz según sus índices y en sentido creciente de estos.

krsort(*array***)**

Ordena la matriz por *índices* en sentido *decreciente* de los mismos.

Ordenación por valores manteniendo índices

asort(*array***)**

Ordena la matriz según sus *valores* en sentido *creciente* y *mantiene* los índices del array original.

arsort(*array***)**

Ordena la matriz por valores en sentido decreciente y sigue **manteniendo** los índices originales.

Ordenación mediante función definida por usuario

PHP permite que el usuario pueda defina **funciones** en las que establezca sus criterios particulares de ordenación. Las funciones PHP que permiten usar esta característica son las siguientes:

uasort(*array, funcion***)**

Ordena la matriz utilizando los criterios establecidos por la **función** definida por el usuario y mantiene los índices del array.

Ordenaciones de *arrays*

```
<?
$a=array(1,2,3,1,1,2,3,3,4,4,4,0,1);
$b=array("blanco","azul","blanco","blanco","azul","Blanco","Azul");
$c=array(
    "b" =>"verde",
    "c" =>"rojo",
    "e" =>"verde",
    "f" =>"Rojo",
    "g" =>"Verde",
    "a"=>"rojo",
    "d" =>"rojo",);

sort ($a);

echo "<h3>Ordenación por valores usando sort</h3>";
foreach ($a as $clave=>$valor){
echo "Clave: ",$clave," Valor: ",$valor, "<br>";
}

sort ($c);

echo "<h3>Ordenación por valores usando sort</h3>";
foreach ($c as $clave=>$valor){
echo "Clave: ",$clave," Valor: ",$valor, "<br>";
}

rsort($a);

echo "<h3>Ordenación inversa por valores usando rsort</h3>";
 foreach ($a as $clave=>$valor){
echo "Clave: ",$clave," Valor: ",$valor, "<br>";
}

ksort($b);

echo "<h3>Ordenación por claves usando ksort</h3>";
 foreach ($b as $clave=>$valor){
echo "Clave: ",$clave," Valor: ",$valor, "<br>";

}

krsort($b);

echo "<h3>Ordenación inversa por claves usando krsort</h3>";
 foreach ($b as $clave=>$valor){
echo "Clave: ",$clave," Valor: ",$valor, "<br>";

}

asort($c);

echo "<h3>Ordenación por valores manteniendo indices </h3>";
 foreach ($c as $clave=>$valor){
echo "Clave: ",$clave," Valor: ",$valor, "<br>";

}

arsort($c);

echo "<h3>Ordenación inversa por valores manteniendo indices arsort</
h3>";
 foreach ($c as $clave=>$valor){
echo "Clave: ",$clave," Valor: ",$valor, "<br>";

}
echo "<h3>Ordenación por valores mediante
            función de usuario  manteniendo indices</h3>";

/*  esta funcion recoge el valor de la variable $a
y aplicar el operador de comparación ternario
de forma que si el valor de la variable es impar
devuelve como valor -2 y si es par devuelve 2
el 2 y el menos 2 unicamente establecen criterios de
comparacion de modo que los valores -2 serán considerados
anteriores a los valores +2 */


function micomparar (&$a) {
          return ($a%2!=0) ? -2 : 2;
}
```

usort(*array, funcion***)**

Ordena la matriz por **valores** utilizando los criterios definidos en la **función** de usuario y modifica los índices.

uksort(*array, funcion***)**

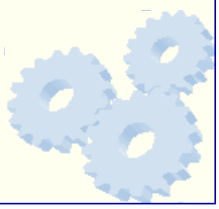
Ordena la matriz por **claves** utilizando los criterios definidos en la **función**.

En el ejemplo hemos definido una función de comparación siguiendo el criterio de ser o no ser múltiplo de 2.

Trataremos las **funciones** en un tema aparte. La utilidad de la que hemos incluido en el ejemplo es la siguiente: Si el valor de la variable es par le asignamos un número negativo como respuesta y en caso contrario uno positivo.

De esta forma los valores del array que devuelven negativos se consideran anteriores en la ordenación a los que dan como resultado un número positivo.

```
uasort ($a, micomparar);

   foreach ($a as $clave=>$valor){
echo "Clave: ",$clave," Valor: ",$valor, "<br>";

}
echo "<h3>Ordenación por clave mediante función de usuario </h3>";

uksort ($a, micomparar);

   foreach ($a as $clave=>$valor){
echo "Clave: ",$clave," Valor: ",$valor, "<br>";

}

echo "<h3>Ordenación por valores mediante función de usuario </h3>";

usort ($a, micomparar);

   foreach ($a as $clave=>$valor){
echo "Clave: ",$clave," Valor: ",$valor, "<br>";

}

   ?>
```



ejemplo62.php