Capítulo 8: Manejo de vectores.

En el capítulo introductorio al lenguaje se estudiaron los vectores como tipos de datos en PHP, se vio como crear y utilizar vectores y vectores asociativos, en este capítulo se estudian funciones nativas de PHP que facilitan la manipulación de vectores y matrices.

Colas y Pilas usando vectores en PHP.

PHP cuenta con instrucciones nativas para manipular vectores que permiten utilizar los mismos como pilas o colas, las instrucciones son:

variable=array_pop(array);

Elimina del vector el último elemento y lo devuelve en una variable.

array_push(array, variable);

Agrega un elemento al final del vector con el valor de la variable que se le pasa como segundo parámetro.

variable=array_shift(array);

Elimina del vector el primer elemento y lo devuelve en la variable.

array_unshift(array,variable);

Agrega el elemento pasado al principio del vector desplazando todos los valores.

Usando array_shift y array_pop se pueden implementar colas, usando array_push y array_pop tenemos una pila mientras que usando las 4 instrucciones podemos manejar facilmente una cola doble.

Funciones de sort

PHP dispone de varias modalidades de sort para vectores, las funciones son:

sort(array);

Ordena un vector según los valores de sus elementos, si es un vector asociativo considera claves y valores como elementos comunes (no los distingue). Ordena en orden ascendiente.

rsort(array);

Idem anterior pero ordena en orden descendiente.

asort(array);

Ordena un vector según los valores de sus elementos pero manteniendo las asociaciones clave-valor. Ordena los pares ordenados clave-valor según “valor”

arsort(array);

Idem anterior pero en orden descendiente.

ksort(array);

Ordena un vector asociativo por los valores de sus “claves” teniendo en cuenta las asociaciones clave-valor.

krsort(array);

Idem anterior pero en orden descendiente.

usort(array,funcion);

Dado un array y una función de comparación ordena los “valores” del array usando la funcion provista para comparar elementos. La función debe recibir dos elementos y devolver 1 si el primero es mayor, -1 si el segundo es mayor o bien 0 si los elementos son iguales. Ejemplo:

```
function cmp ($a, $b) {
   if ($a == $b) return 0;
   return ($a > $b) ? -1 : 1;
}
$a = array (3, 2, 5, 6, 1);
usort ($a, cmp);
```

uksort(array,funcion);

Ordena un vector asociativo por “clave” usando para comparar las claves la función pasada como parámetro.

uasort(array,funcion);

Ordena un vector por los “valores” de sus elementos preservando la relación clave-valor de un array asociativo usando para ordenar la funcion provista por el usuario.

Ejemplos:

$vector=array(“d”=>”banana”, “a”=>”limon”,”c”=>”pera”, “b”=>”anana”);

Función	Resultado
sort($vector)	“a”,“anana”,”b”,”banana”,”c”,”d”,”limon”,”pera”
rsort($vector)	“pera”,”limon”,”d”,”c”,”banana”,”b”,”anana”,”a”
asort($vector)	“b”,”anana”,”d”,”banana”,”a”,”limon”,”c”,”pera”
arsort($vector)	“c”,”pera”,”a”,”limon”,”d”,”banana”,”b”,”anana”
ksort($vector)	“a”,”limon”,”b”,”anana”,”c”,”pera”,”d”,”banana”
krsort($vector)	“d”,”banana”,”c”,”pera”,”b”,”anana”,”a”,”limon”

Funciones para manipular vectores:

Padding:

array=array_pad(array,pad_size,pad_value);

Completa el array pasado con pad_value hasta que el vector tenga pad_size elementos, si pad_size es positivo completa agregando elementos hacia la derecha, si es negativo completa hacia la izquierda. El vector no es modificado, devuelve el vector resultado.

```
$input = array (12, 10, 9);
$result = array_pad ($input, 5, 0);  // result is array (12, 10, 9, 0, 0)
$result = array_pad ($input, -7, -1); // result is array (-1, -1, -1, -1, 12, 10, 9)
$result = array_pad ($input, 2, "noop");  // not padded
```

List:

List en realidad no es una instrucción sino una construcción especial del lenguaje que permite asignar a un grupo de variables los elementos de un vector.

Ejemplo:

```
$vector=array(1,2);
list($a,$b)=$vector;   //$a=1, $b=2
```

Si el vector tiene más elementos que las variables que se usan en list entonces el último elemento de list será un vector con todos los elementos que quedaban en el array (la asignación se hace de izquierda a derecha).

Merge:

```
array=array_merge(array1,array2,....);
```

Si los vectores son asociativos hace un merge de los vectores en donde si 2 o más vectores tienen la misma clave sólo una queda en el vector resultado. Si los vectores no son asociativos (indexados por número) entonces el resultado tiene todos los elementos de los "n" vectores pasados concatenados.

Sub-Vectores:

```
array=array_slice(array,offset[,cantidad]);
```

Devuelve un sub-vector del vector pasado a partir del offset indicado y con la cantidad de elementos indicada, si cantidad no se especifica devuelve todos los elementos desde offset hasta el fin del vector.

```
$vec=array(10,6,7,8,23);

$res=array_slice($vec,1,3);   //deja en la variable $res 6,7,8
```

Count:

```
$cantidad=count($vector);
```

Devuelve la cantidad de elementos de un vector.

Splice:

```
array=array_splice(array,offset,[cantidad,array_reemplazo]);
```

Si array_reemplazo no se pasa como parámetro entonces elimina del vector pasado la cantidad de elementos indicada a partir del offset indicado, si array_reemplazo existe dichos elementos son reemplazados por aquellos del vector array_reemplazo.
Si no se pasa cantidad se eliminan o reemplazan todos los elementos desde el offset indicado hasta el fin del vector.

Shuffle:

```
shuffle(array);
```

Desordena en forma aleatoria los elementos de un vector.

Pertenencia:

Boolean = in_array(variable,array);

Devuelve true/false según el elemento pasado pertenezca o no al vector.

Range:

array=range(low,high);

Crea un vector con los números correspondientes desde low hasta high.

Ejemplo:

$vec=range(6,12); //$vec=(6,7,8,9,10,11,12);

Reverse:

array=array_reverse(array);

Devuelve el vector revertido.

Compact:

array=compact(nombre_var1,nombre_var2,....,nombre_varn);

Crea un vector asociativo con las variables y sus valores donde las claves son los nombres de las variables y los valores el contenido de las mismas.

Ejemplo:
$ciudad=”miami”;
$edad=”23”;

$vec=compact(“ciudad”,”edad”);

Es equivalente a:

$vec=array(“ciudad”=>”miami”,”edad”,”23”);

Funcion Alfa:

PHP soporta una función muy usada en programación funcional que es conocida como función alfa, en php se denomina array_walk y permite aplicar una función a todos y cada uno de los elementos de un vector. La sintaxis es:

array_walk(array,funcion,variable_extra);

variable_extra es opcional, se aplica la función pasada como parámetro a cada uno de los elementos del vector, la función recibirá como parámetro en primer lugar el “valor” del elemento del array y en segundo lugar la “clave”, si el vector no es asociativo la clave es el numero de índice (0,1,2...). Si se pasa variable_extra que puede ser cualquier tipo de PHP incluyendo un objeto la función recibe dicha variable como tercer parámetro.

Funciones especiales para vectores asociativos:

array=array_keys(array)

Devuelve un vector con todas las claves de un vector asociativo.

array=array_values(array)

Devuelve un vector con todos los valores de un vector asociativo.

Funciones para recorrer vectores:

En PHP cada vector tiene asociado un puntero interno que apunta a un elemento del vector y que puede ser usado para recorrer vectores y otras operaciones, las funciones que operan con el puntero interno son:

reset(array);

Resetea el puntero interno al principio del array.

end(array);

Setea el puntero interno apuntando al último elemento del array

next(array);

Mueve el puntero al proximo elemento del array

prev(array);

Mueve el puntero al último elemento del array

current(array);

Devuelve el elemento apuntado actualmente por el puntero interno del array.

key(array);

Devuelve la clave (índice) del elemento apuntado actualmente por el puntero interno del array, si es un vector asociativo devuelve la clave del elemento actual. Si es un vector común devuelve el numero de índice del elemento actual.

array=each(array)

Devuelve un vector clave-valor con los valores correspondientes al elemento actual del array y además mueve el puntero al elemento siguiente, si es un vector asociativo devuelve clave-valor, si es un vector común devuelve indice-valor.

Each suele usarse en conjunto con list para recorrer un vector:

Ejemplo:

```
while(list($clave,$valor)=each($vector)) {
  //Hacer algo
}
```