Vectores en PHP

Ing. Alberto MOreno

ÍNDICE

- **Introducción**

  - Cómo se declaran y construyen
  - Cómo se estructuran
  - EJEMPLOS

- **Índices**

  - Tipos de indexado y funcionamiento
  - EJEMPLOS

ÍNDICE

- Funciones nativas de PHP para su manejo

- Funciones de recorrido
- Funciones de ordenación
- Funciones de manipulación
- Funciones especiales para vectores asociativos

INTRODUCCIÓN

Se declaran y se accede a los elementos mediante corchetes. [ ]

La primera componente es cero.

Dentro de un mismo vector puede haber elementos de tipos distintos.

Se puede acceder a un elemento mediante un índice asociativo.

Pueden ser multidimensionales, añadiendo más subíndices.

Se pueden construir mediante array()

EJEMPLOS

```
$vector1[0]=1;
$vector1[1]='hola'
$vector1["nombre"]='Carlos'
```

TIPO DE LOS ELEMENTOS

```
$vector2=array(1,"Jordi",3);
$vector2=array(0=>1,1=>"Jordi",2=>3);
$vector3=array(0=>33,1=>"Juan",
"nombre"=>"Sara",3=>5);
```

CONSTRUCCIÓN

EJEMPLOS

```
$a[0][1]='Hola';
$a[0]["clave"]='una cosa';
$a["clave1"]["clave2"][0]='otra cosa';
```

MULTIDIMENSIONALES

```
$a[0]='nada';
$a[1]='hola';
$a[]='mundo';
```

AUTOASIGNACION

Asigna a $a[2] el valor 'mundo'.

EJEMPLOS

```
$a[]='a';
$a[]='b'; equivale a: $a=array('a','b','c');
$a[]='c';
```

CONSTRUCCIÓN

```
$cliente1=array(
   "nombre" => "Juan",
   "edad" => 23,
    "profesion" => "estudiante"
);
```

VECTORES ASOCIATIVOS

ÍNDICES (CLAVES)

Un índice puede ser un número entero o una cadena de texto.

Si el índice es una cadena no existe un índice entero correspondiente.

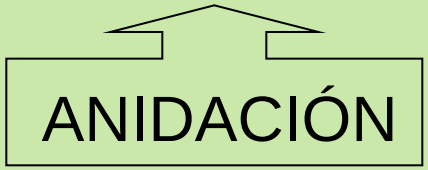
Si se omite el índice, este se genera automáticamente comenzando por cero.

Si el vector ya contiene índices enteros y se omite alguno, el siguiente generado será el mayor índice entero existente +1.

Si definimos dos índices enteros iguales, el último definido sobreescribirá al anterior.

EJEMPLOS

```
$frutas = array(
    “nispero"=>array("color"=>“amarillo",“sabor"=>“dulce",
        “forma"=>“redonda"),
    “pomelo" => array("color" => “amarillo",“sabor" => “agrio",
        “forma" =>“redonda"),
    );
```



print_r($frutas); Mostrará:

```
Array(
    [nispero]=>Array([color]=>naranja [sabor] =>dulce
        [forma] =>redonda)
    [pomelo]=>Array([color]=>amarillo [sabor]=>agrio
        [forma]=>redonda)
    )
```

EJEMPLOS

```
$cosas=array(
    “frutas”=>array(“a”=>”naranja”,”b”=>”platano”,
        “c”=>”manzana”)
    “numeros”=>array(1,2,3,4,5,6)
    “equipos” =>array(“primero”,5=>”segundo”,
        ”tercero”)
);
```

ANIDACIÓN

print_r($cosas); Mostrará:

```
Array(
    [frutas]=>Array([a]=>naranja[b] =>platano[c] =>manzana)
    [numeros]=>Array([0]=>1[1]=>2[2]=>3[3]=>4[4]=>5[5]=>6)
    [equipos]=>Array([0]=>primero[5] =>segundo[6] =>tercero)
)
```

EJEMPLOS

$mivector=array(1,1,1,1,1,8=>1,4 =>1,22,3 =>64);

print_r($mivector); Mostrará:

Array([0]=>1[1]=>1[2]=>1[3]=>64[4]=>1[8]=>1[9]=>22)

El 64 sobreescribe al anterior valor del elemento 3.
El 22 se aloja en el elemento 9 que es el índice de valor máximo(8) más uno.

FUNCIONES DE RECORRIDO

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);

Mueve el puntero al último elemento del array.

next($array);

Mueve el puntero al proximo elemento del array.

prev($array);

Mueve el puntero al elemento previo respecto al actual.

current($array);

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

FUNCIONES DE RECORRIDO

key($array);

Devuelve el índice del elemento apuntado actualmente por el puntero interno del array, si es un vector asociativo devuelve la clave del elemento actual.

$array1=**each**($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.

Ejemplo:

$pareja=**each**($vec);

list($clave,$valor)=$pareja;

echo("La pareja clave-valor actual de vec es $clave = $valor");

FUNCIONES DE ORDENACION

sort($array);

Ordena un vector según los valores de sus elementos, si este es 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.

FUNCIONES DE ORDENACION

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.

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”=>”aguacate”);

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

Padding:

array=**array_pad**($mivector,tamaño,valor);

Rellena $mivector con valor hasta que tenga tamaño elementos, si tamaño es positivo completa agregando elementos hacia la derecha, si es negativo completa hacia la izquierda.

Ejemplo:

$entrada = array (12, 10, 9);

$resultado = **array_pad** ($entrada, 5, 0); // resultado: (12, 10, 9, 0, 0)

$resultado = **array_pad** ($entrada, -7, -1); // resultado:

(-1, -1, -1, -1, 12, 10, 9)

FUNCIONES DE MANIPULACIÓN

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 él (la asignación se hace de izquierda a derecha).

FUNCIONES DE MANIPULACIÓN

Merge:

$vec1=**array_merge**($array1,$array2,....);

Si los vectores son asociativos hace una unión 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 concatenados.

Sub-Vectores:

$vec1=**array_slice**($array,offset,cantidad);

Devuelve un sub-vector de $array 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

FUNCIONES DE MANIPULACIÓN

Count:

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

Devuelve la cantidad de elementos de un vector.

Splice:

$vec1=**array_splice**($vec,offset,cantidad,$vec_reemplazo);

Sustituye los elementos de $vec por los de $vec_reemplazo a partir del offset y hasta cantidad, si no le pasamos vec_reemplazo elimina los elementos a partir del offset y hasta cantidad.

Si no se pasa cantidad se eliminan o reemplazan todos los elementos desde el offset indicado hasta el fin del vector.

FUNCIONES DE MANIPULACIÓN

Shuffle:

shuffle(array);

Desordena en forma aleatoria los elementos de un vector.

Pertenencia:

$boolean = **in_array**($elem,$miarray,$strict);

Devuelve verdadero o falso según $elem pertenezca o no a $miarray, si $strict es *true* tendrá en cuenta el tipo de los valores.

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);

FUNCIONES DE MANIPULACIÓN

Reverse:

array=**array_reverse**(array);

Devuelve el vector invertido.

Compact:

array=**compact**(nombre_var1,nombre_var2,....,nombre_varN);

Crea un vector asociativo cuyas 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”);

FUNCIONES DE MANIPULACIÓN

Función:

array_walk permite aplicar una función a todos y cada uno de los elementos de un vector. La sintaxis es:

array_walk($array1,funcion,variable_extra);

Nos sirve para aplicar una función pasada como parámetro a cada uno de los elementos del vector $array1, la función recibirá como parámetro en primer lugar el “valor” del elemento de $array1 y en segundo lugar la “clave”, si el vector no es asociativo la clave es el numero de índice (0,1,2...). variable_extra es opcional.

Si se pasa variable_extra que puede ser cualquier tipo de PHP incluyendo un objeto, la función recibe dicha variable como tercer parámetro.

FUNCIONES DE MANIPULACIÓN

Otras funciones:

array_diff() Calcula las diferencias entre dos arrays.

array_fill() Rellena un array con valores.

array_search() Busca un valor y devuelve su posicion.

array_sum() Calcula la suma de todos los valores.

FUNCIONES PARA VECTORES ASOCIATIVOS

&array1=**array_keys**($array)

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

&array1=**array_values**($array)

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

Bibliografía

- Apuntes de clase:Tema 7(PHP)
- ajo.thinknerd.com/docs/cursophp/curso_php_cap_09.PDF

----------¿PREGUNTAS?