Unidad 7: Arreglos

Cátedra Introduccón a la Programación
Tecnicatura en Desarrollo de Aplicaciones Web
Facultad de Informática
Buenos Aires 1400 (8300) Neuquén
Universidad Nacional del Comahue, Argentina

---

1. ARREGLOS

Conceptualmente un arreglo es una colección de elementos de un mismo tipo.

Una variable de tipo arreglo contiene elementos de un mismo tipo, que pueden ser accedidos mediante un número entero, denominado índice. El valor 0 es el índice o localizador del primer elemento y n-1 es el índice del último elemento localizado en el Arreglo. Por lo tanto la dimensión, tamaño o longitud del arreglo está dada por el valor de n.

Un arreglo de enteros almacenará valores enteros, un arreglo de booleanos almacenará valores booleanos (true o false), un arreglo de strings almacenará valores string (cadena de caracteres), un arreglo de reales almacenará números reales, un arreglo de arreglos, donde cada elemento almacenará un nuevo arreglo.

Es decir, los elementos almacenados dentro de un Arreglo pueden ser de cualquier tipo, como los visto hasta el momento: string, entero, punto flotante, booleano y hasta otro Arreglo.

Por ejemplo podemos implementar un arreglo que almacena información de los n empleados de una empresa. De esta manera $empleado[23] podría tomar como valor *"Juan Pérez Suárez"* (i.e: $empleado[23] = "Juan Pérez Suárez"). En este caso se trataría de un arreglo de cadenas de texto, es decir, un arreglo de string. En otro caso $empleado[23] podría contener el número de legajo del empleado dentro de la empresa, por ejemplo: $empleado[23] = 2312. En este caso se trataría de un arreglo de valores numéricos.

Según el lenguaje de programación, la implementación de las estructuras de datos es diferente.

PHP cuenta con una gran variedad de tipos de datos que permiten representar la información según como esté estructurada. En esta unidad se estudian los arreglos existentes en PHP, que **son tipos de datos utilizados cuando se quiere agrupar elementos**.

1.1. Arrays unidimensionales

PHP soporta tanto arrays escalares como asociativos. De hecho, no hay diferencias entre los dos. Se puede crear una array usando la funcion array(), o se puede asignar el valor de cada elemento del array de manera explícita. Además, como PHP es un lenguaje dinámico, sin declaración de tipos, en cada elemento del arreglo puede ser de un tipo distinto.

```
1  <?php
2  $a[0] = "abc";
3  $a[1] = "def";
4  $b["foo"] = 13;
5  ?>
```

1.2. Arrays multidimensionales

Dentro de un arreglo podemos almacenar información estructurada en otro arreglo. Por ejemplo, en la estructura que almacena la información correspondientes a los empleados de una empresa, podemos almacenar datos vinculados a cada empleado como el nombre, apellido, antiguedad, fecha nacimiento etc.

```
1  $emp1 = array( "nombre" => "Juan",
```

```
2                   "apellido" => "Suarez",
3                   "antiguedad" => 3,
4                   "fechaNacimiento" => "22/20/1979",
5                   );
6   $emp2 = array(  "nombre" => "Pedro",
7                   "apellido" => "Martinez",
8                   "antiguedad" => 13,
9                   "fechaNacimiento" => "22/20/1969",
10                  );
11  $empleados = Array();
12  $empleados[0] = $emp1;
13  $empleados[1] = $emp2;
```

1.3. Recorrer los elementos de un arreglo

Para recorrer un arreglo es posible utilizar las estructura de control "*Repetitivas*" vistas hasta el momento: ***for*** , ***while***. Si requerimos acceder a TODOS los elementos del arreglo desde el elemento 0 al elemento N-1, siendo N la cantidad de elementos de un arreglo, la estructura mas adecuada es el ***for***. En cambio, si necesitamos recorrer el arreglo hasta encontrar un elemento con determinadas características, la estructura mas adecuada es el ***While,*** siendo la condición de corte la obtención del elemento buscado. Para obtener la cantidad de elementos que se encuentran en un arreglo, se debe invocar de la función ***count()***.

Recorriendo un arreglo con una estructura de control for:

```
1  <?php
2  $arreglo = array(12,24,67,34);
3  for ($i=0 ; $i< count($arreglo);$i++) {
4          echo "Elemento de la  posicion $i = $arreglo[$i]\n";
5  }
6  ?>
```

Recorriendo un arreglo con una estructura de control while:

```
1  <?php
2  // Visualizar los elementos del arreglo hasta encontrar un elemento >60
3  $arreglo = array(12,24,67,34);
4  $i=0 ;
5  while ($i< count($arreglo) and $arreglo[$i]<=60 ){
6          echo "Elemento de la  posicion $i = $arreglo[$i]\n";
7          $i++;
8  }
9  ?>
```

Otra forma habitual de recorrer un array es utilizando la estructura de control repetitiva ***foreach()*** que nos permite acceder a cada elemento y clave del arreglo.

Recorriendo un arreglo con una estructura de control foreach:

```
1  <?php
2  $arreglo = array(12,24,67,34);
3  foreach ($arreglo as $indice => $unElemento) {
4          echo "Elemento de la  posicion $indice = $unElemento\n";
5  }
6  ?>
```

Recorriendo un arreglo asociativo con una estructura de control foreach:

```
1  <?php
2  $array = array('a' => 4, 'b' => 8, 'c' => -1, 'd' => -9);
3  foreach ($arreglo as $indice => $unElemento) {
4          echo "Elemento con clave $indice = $unElemento\n";
5  }
6  ?>
```

1.4. Funciones para ordenar los elementos de un arreglo

Nos puede interesar que los elementos de un arreglo estén ordenados: una vez que finalizó la inscripción en un curso, tener a los padrones de los alumnos por orden de inscripción puede ser muy incómodo, siempre será preferible tenerlos ordenados por número para realizar cualquier comprobación.

Los arrays se pueden ordenar usando las funciones asort(), arsort(), ksort(), rsort(), sort(), uasort(), usort(), y uksort() dependiendo del tipo de ordenación que se desee. A continuación vamos a ver alguna de ellas:

—**sort:** esta función ordena los elementos de un arreglo. Los elementos estarán ordenados de menor a mayor cuando la función haya terminado.

```
1  <?php
2  $frutas = array("l" => "limon",
3                  "n" => "naranja",
4                        "m" => "manzana",
5                        "d" => "durazno");
6  sort($frutas);
7  foreach ($frutas as $indice => $unafruta) {
8              echo "$indice = $unafruta\n";  }
9  }
10 ?>
```

—**rsort**: Ordena los elementos de un arreglo en orden inverso

```
1  <?php
2  $frutas = array("l" => "limon",
3                  "n" => "naranja",
4                        "m" => "manzana",
5                        "d" => "durazno");
6  rsort($frutas);
7  foreach ($frutas as $indice => $unafruta) {
8              echo "$indice = $unafruta\n";  }
9  }
10 ?>
```

—**asort**: esta función ordena un array manteniendo la correlación de los índices del array con los elementos con los que están asociados. Esta función se utiliza principalmente para ordenar arrays asociativos en los que el orden es importante.

```
1  <?php
2  $frutas = array("l" => "limon",
3                  "n" => "naranja",
4                        "m" => "manzana",
5                        "d" => "durazno");
6  asort($frutas);
7  foreach ($frutas as $indice => $unafruta) {
8              echo "$indice = $unafruta\n";  }
9  }
10 ?>
```

—**arsort:** Ordena un array en orden inverso y mantiene la asociación de índices

```
1  <?php
2  $frutas = array("l" => "limon",
3                  "n" => "naranja",
4                        "m" => "manzana",
5                        "d" => "durazno");
6  arsort($frutas);
7  foreach ($frutas as $indice => $unafruta) {
8              echo "$indice = $unafruta\n";
9  }
10 ?>
```

—**ksort:** Ordena un array por clave, manteniendo la correlación entre la clave y los datos. Esto es útil principalmente para arrays asociativos.

```php
<?php
$frutas = array("l" => "limon",
                "n" => "naranja",
                "m" => "manzana",
                "d" => "durazno");
ksort($frutas);
echo ">>> Orden alfabetico basado en las claves \n ";
foreach ($frutas as $indice => $unafruta) {
        echo "$indice = $unafruta\n";
}
?>
```

—**uasort:** Ordena un array con una función de comparación definida por el usuario y mantiene la asociación de índices.

```php
<?php

// Funci n de comparaci n
function comparar($a, $b){
    if ($a == $b){
        return 0;
    }
    return ($a < $b) ? -1 : 1;
}


$miArray = array(
        "a" => 4,
                "b" => 8,
                "c" => -1,
                "d" => -9);

uasort($miArray, 'comparar');

echo ">>> Ordenado seg n funci n de comparaci n \n ";
foreach ($miArray as $indice => $miArray) {
        echo "$indice = $miArray\n";
}
?>
```

1.5. Arreglos y cadenas

A partir de una cadena de caracteres, podemos obtener un arreglo con sus componentes usando la función ***explode***.

Por ejemplo, si queremos obtener las palabras (separadas entre sí por espacios) que componen la cadena *$c* escribiremos simplemente explode(" ",$c):

```php
<?php
$c = "Una cadena con espacios";
$arrPalabra = explode(" ",$c) ;
print_r($arrPalabra);
?>
```

De manera inversa, para obtener los elementos de un arreglo en una cadena de caracteres, es posible utilizar la función ***implode.*** La función retorna la cadena que resulta de unir todas las componentes separadas entre sí por medio del separador que es indicado como parámetro de la función

```php
<?php
$array = array('apellido', 'email', 'tel fono');
$cadena_separada_por_comas = implode(",", $array);
echo $cadena_separada_por_comas; // apellido,email,tel fono
?>
```

1.5.1. Ejercicios con arreglos y cadenas

Ejercicio 7.9.. Escribir una función que reciba como parámetro una cadena de palabras separadas por espacios y devuelva, como resultado, cuántas palabras tiene la cadena dada.

Ejercicio 7.10.. Escribir una función que recibe por parámetro los correos electrónicos de una lista de correo separados por coma y retorna un arreglo de direcciones de correo.

1.6. Resumen

—PHP nos provee varias estructuras que nos permiten agrupar diferentes tipos datos. En particular, los arreglos son estructuras mutables que permiten agrupar valores, con la posibilidad de agregar, quitar o reemplazar sus elementos.

—Los arreglos se utilizan en situaciones en las que los elementos a agrupar pueden ir variando a lo largo del tiempo. Por ejemplo, para representar las notas de un alumno en diversas materias, los inscriptos para un evento o la clasificación de los equipos en una competencia.

REFERENCIA DEL LENGUAJE PHP

$Arreglo :

Los arreglos son estructuras que nos permiten manipular una colección de datos de un mismo tipo. Los arreglos se acceden a partir de indices numéricos o asociativos

Casos particulares:

$Arreglo= []	Arreglo vacio
$Arreglo[1]= $elemento	$elemento es el valor asignado a la posición 1 del arreglo
$Arreglo[$i]= $elemento	$elemento es el valor asignado a la posición $i del arreglo
$Arreglo[“nombre”] = “Andrea”	Arreglo asociativo cuya clave *nombre* tiene el valor *Andrea*

count

Función que retorna la cantidad de elementos de un arreglo.

asort(), arsort(), ksort(), rsort(), sort(), uasort(), usort(), y uksort():

Funciones que permiten ordenar los elementos de un arreglo ya sea a partir de sus elemento, sus claves o funciones implementadas por los programadores.