Tema 4. ARRAYS

Els arrays o **matrius** formen una part molt important de la programació en PHP ja que permeten manegar grups de valors relacionats: ens permeten emmagatzemar múltiples valors en una sola estructura i, d'aquesta forma, associar-los sota una mateixa denominació. En especial, són molt utilitzats en les funcions lligades a les bases de dades.

En aquesta unitat veurem diferents tipus d'arrays que es poden definir, escalars i associatius, tant unidimensionals com multidimensionals.

4.1. ARRAYS ESCALARS

Un *array escalar* o simple, està format per un conjunt de valors ordenats respecte a un **índex de tipus enter**. Aquest índex indicarà la posició de l'element dins d'aquesta col·lecció ordenada, de manera que, a cada posició marcada per l'índex dins de l'array, hi hagi un valor.

Existeixen diferents formes de crear un array. La més senzilla és assignant el valor de cada element de forma explícita, és a dir, indicant cadascun dels valors que el composen i, fins i tot, la posició que ocupen dins de l'array.

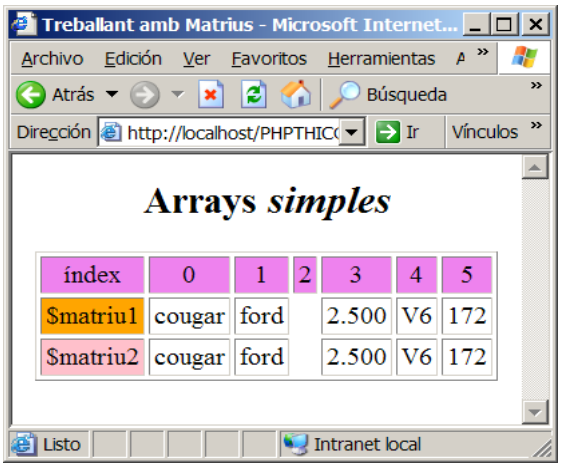
En el següent exemple podem veure dos formes equivalents de definir el mateix array:

```
<?php

  $matriz1[0]="cougar";
  $matriz1[1]="ford";
  $matriz1[3]="2.500";
  $matriz1[4]="V6";
  $matriz1[5]=172;

  $matriz2[]="cougar";
  $matriz2[]="ford";
  $matriz2[]="";
  $matriz2[]="2.500";
  $matriz2[]="V6";
  $matriz2[]=172;

?>
```



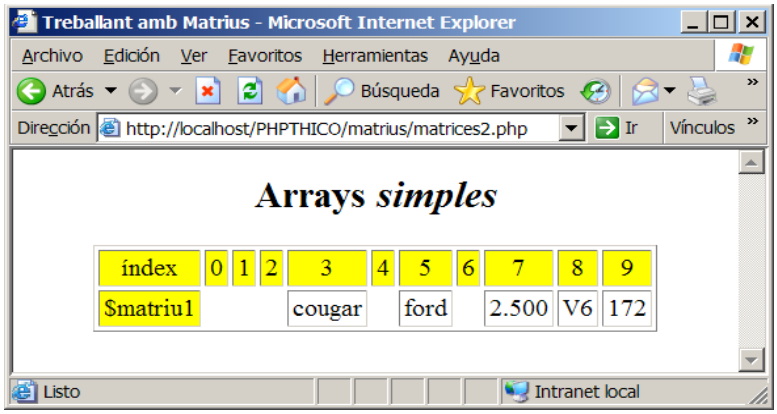
Es pot veure en aquest codi que un array pot combinar elements de diferent tipus, com ara, valors enters i cadenes de caràcters o, inclús, elements buits.

Quan, en generar un array, no s'indica la posició dels seus elements, aquests es van situant seqüencialment respecte a la última assignació realitzada sobre l'array. L'assignació numèrica de posicions dins de l'array no té perquè ser seqüencial, és a dir, es pot definir l'ordre numèric que interessi.

```php
<?php

  $matriz1[3]="cougar";
  $matriz1[5]="ford";
  $matriz1[7]="2.500";
  $matriz1[ ]="V6";
  $matriz1[ ]=172;

?>
```

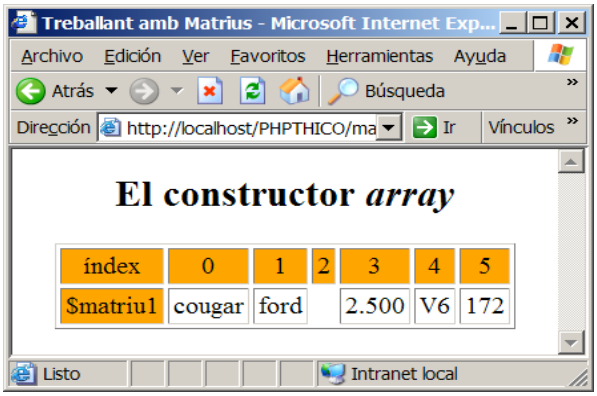


No s'ha de confondre el fet de que es tinguin valors guardats a la posició 9 de l'array i pensar que s'ha definit un array de nou elements, doncs seria un error, ja que amb la definició l'array és de 5 elements.

Una altra forma de definir arrays és mitjançant el constructor **array()** proporcionat per PHP. Aquest constructor no és una funció regular; té la següent definició:

array array (mixed valors,...)

```php
<?php
  $matriz1 = array("cougar","ford",null,"2.500","V6",172);
?>
```



Exercici 1
Mostra els arrays vistos fins a aquest punt.

El constructor **array()** també permet assignar els elements de l'array en l'ordre que es vulgui. Per això s'indica l'índex, seguit del símbol **=>** i el valor de l'element.

```php
<?php
  $matriz1 = array("cougar","ford",null,"2.500","V6",172);
  $matriz2 = array(2=>"cougar","ford",1=>null,0=>"2.500","V6",172);
?>
```

Als elements als quals no se'ls hi assigna explícitament un índex, prenen la posició seqüencial relativa a la última assignació de posició dins de l'array.

4.2. ARRAYS ASSOCIATIUS

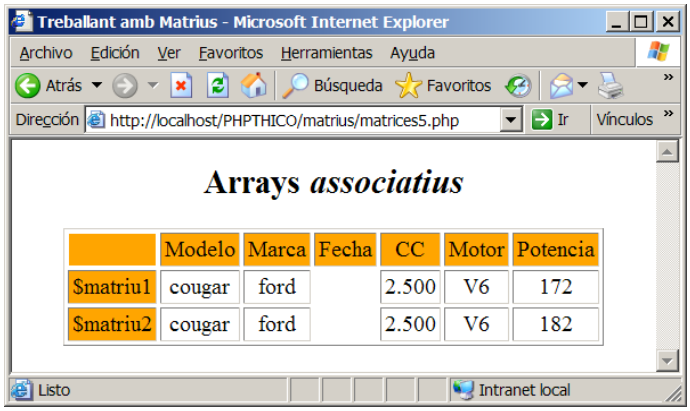
A diferència dels arrays simples, els **arrays associatius** (també coneguts com a *taules hash* o *arrays indexats per cadena*) estan formats per un conjunt de valors que estan ordenats respecte a un índex de tipus *string*, és a dir, una cadena de caràcters. D'aquesta forma, aquest array estarà composat per parells **clau-valor**, fent-se necessari proporcionar la clau per poder accedir al valor emmagatzemat a l'array.

D'igual forma que amb els arrays simples, es pot utilitzar el constructor del llenguatge **array()** per a definir-los, o bé, fer-ho especificant de forma explícita cadascun dels seus components.

```
<?php
 $matriz1 = array("modelo"=>"cougar","marca"=>"ford",
            "fecha"=>null,"cc"=>"2.500","motor"=>"V6",
            "potencia"=>172);

 $matriz2['modelo']="cougar";
 $matriz2['marca']="ford";
 $matriz2['fecha']=null;
 $matriz2['cc']="2.500";

 $matriz2['motor']="V6";
 $matriz2['potencia']=182;
?>
```



Exercici 2
Mostra els arrays associatius vistos fins al moment.

4.3. ARRAYS MULTIDIMENSIONALS

PHP permet definir arrays multidimensionals mitjançant la combinació d'arrays unidimensionals (que poden ser tant de tipus escalar com associatius). A continuació podem veure diferents formes de fer-ho.

4.3.1. ARRAY MULTIDIMENSIONAL DE TIPUS ESCALAR

```
<?php
 $matriu1[0][0] = "peseta";
 $matriu1[0][1] = 166.386;
 $matriu1[1][0] = "dolar";
 $matriu1[1][1] = 0.96;

 $matriu2[0] = array("peseta",166.386);
 $matriu2[1] = array("dolar",0.96);

 $matriu3 = array(array("peseta",166.386),array("dolar",0.96));
?>
```

4.3.2. ARRAY MULTIDIMENSIONAL DE TIPUS ASSOCIATIU

```
<?php
 $matriz1['peseta']['moneda'] = "peseta";
 $matriz1['peseta']['cambio'] = 166.386;
 $matriz1['dolar']['moneda'] = "dolar";
 $matriz1['dolar']['cambio'] = 0.96;


 $matriz2['peseta']=array("moneda"=>"peseta","cambio"=>166.386);
 $matriz2['dolar']=array("moneda"=>"dolar","cambio"=>0.96);

 $matriz3 = array("peseta"=>array("moneda"=>"peseta","cambio"=>166.386),
            "dolar"=>array("moneda"=>"dolar","cambio"=>0.96));
?>
```

4.3.3. ARRAY MULTIDIMENSIONAL COMBINAT

```
<?php
 $matriz1[0]['moneda'] = "peseta";
 $matriz1[0]['cambio'] = 166.386;
 $matriz1[1]['moneda'] = "dolar";
 $matriz1[1]['cambio'] = 0.96;

 $matriz2[0]=array("moneda"=>"peseta","cambio"=>166.386);
 $matriz2[1]=array("moneda"=>"dolar","cambio"=>0.96);

 $matriz3 = array(array("moneda"=>"peseta","cambio"=>166.386),
            array("moneda"=>"dolar","cambio"=>0.96));
?>
```

Exercici 3
Mostra els arrays vistos en aquest apartat.

4.4. RECÒRRER UN ARRAY

Una de les operacions més habituals quan es treballa amb arrays és recorre'l per obtenir els seus elements, ja sigui per modificar-los o per treballar amb ells.

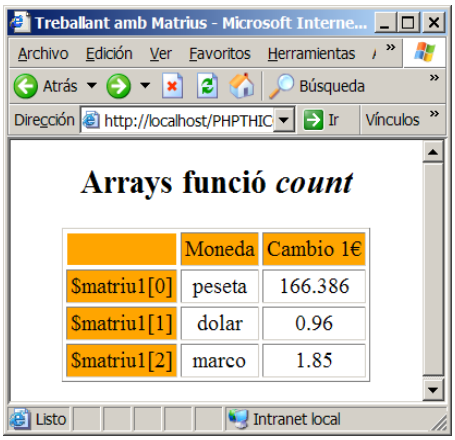
4.4.1. RECORREGUT EN ARRAYS SEQÜENCIALS

Recórrer un array seqüencial és relativament senzill fent ús d'un bucle i iterant en funció del valor de l'índex. El problema principal en aquest tipus de recorregut és conèixer a priori el nombre d'elements que composen l'array. Precisament per solucionar aquest problema, PHP proporciona la funció **count()** que retorna el nombre d'elements que té la variable que se li indica com a argument.

Una vegada conegut el nombre d'elements de l'array, es pot utilitzar un bucle per anar recorrent els seus elements:

```
<?php
  $matriu1[0][0] = "peseta";
  $matriu1[0][1] = 166.386;
  $matriu1[1][0] = "dolar";
  $matriu1[1][1] = 0.96;
  $matriu1[2][0] = "marco";
  $matriu1[2][1] = 1.85;
?>
<TABLE BORDER="1" CELLPADDING="2" CELLSPACING="2">
   <TR ALIGN="center" BGCOLOR="orange">
      <TD></TD>
      <TD>Moneda</TD>
      <TD>Cambio 1€</TD>
  </TR>
  <?php
     for($i=0;$i<count($matriu1);$i++){
        echo "<TR ALIGN='center'>";
        echo "<TD BGCOLOR='orange'>\$matriu1[$i]</TD>";
        for($j=0;$j<count($matriu1[$i]);$j++){
           echo "<TD>".$matriu1[$i][$j]."</TD>";
        }
        echo "</TR>";
     }

  ?>
</TABLE>
```



Una altra funció relacionada amb el recorregut d'arrays és **sizeof()**, que obté el nombre d'elements de l'array passat com a argument en la crida a la funció. A l'exemple anterior es podria haver utilitzat aquesta funció substituint la línia:

for ($i=0; $i<count($matriz1); $i++) {

per aquesta altra: **for ($i=0; $i<sizeof($matriz1); $i++) {**

4.4.2. RECORREGUT EN ARRAYS NO SEQÜENCIALS

Un array no seqüencial no sembla fàcil de Recórrer, donat que no només necessitem saber el nombre d'elements que composen l'array, sinó també la seva posició dins de l'array. Aquest mateix problema es presenta amb els arrays associatius, donat que, a més del nombre d'elements que composen l'array, es necessiten conèixer les claus per poder accedir als valors emmagatzemats.

Per poder treballar amb arrays estructurats de forma no seqüencial tant de tipus escalar com associatius, PHP compta amb el següent conjunt de funcions (també es poden aplicar a arrays estructurats seqüencialment):

- **current(**matriu**):** retorna el valor de la posició actual del punter dins de l'array: tots els array tenen un punter intern que apunta a la posició de l'element *actual* amb el que s'està treballant en un moment donat. Retorna *false* quan el punter està al final d'un array o quan l'array no té cap element.

- **pos(**matriu**):** és idèntica a la funció anterior.

- **key(**matriu**):** retorna l'índex de la posició actual de l'array passat com a argument: un número en el cas que l'array sigui de tipus escalar o una cadena de caràcters en el cas que l'array sigui associatiu.

- **next(**matriu**):** retorna el valor de l'element següent a l'actual (si existeix) i avança el punter intern una posició. En el cas que l'element actual sigui l'últim de l'array retorna *false*.

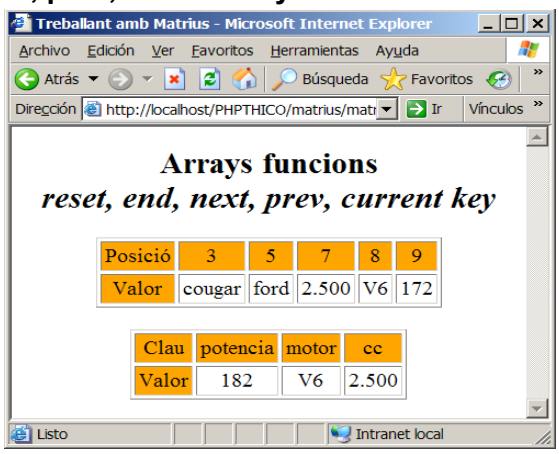
- **prev(**matriu**):** retorna el valor de l'element anterior a l'actual (si existeix) i retrocedeix el punter intern a una posició. En cas de que l'element actual sigui el primer de l'array, retorna *false*.

- **end(**matriu**):** col·loca el punter intern a l'últim element d'un array escalar.

- **reset(**matriu**):** retorna el valor del primer element de l'array i situa el punter intern a la seva primera possició.

```
<HTML>
 <HEAD>  <TITLE>Treballant amb Matrius</TITLE> </HEAD>
 <BODY>  <CENTER>
   <H2>Arrays funcions <BR><I>reset, end, next, prev, current  key</I></H2>
   <?php
    $matriu1[3]="cougar";
    $matriu1[5]="ford";
    $matriu1[7]="2.500";
    $matriu1[ ]="V6";
    $matriu1[ ]=172;

    $matriu2['modelo']="cougar";
    $matriu2['marca']="ford";
    $matriu2['fecha']=null;
    $matriu2['cc']="2.500";
    $matriu2['motor']="V6";
    $matriu2['potencia']=182;
   ?>
   <TABLE BORDER="1" CELLPADDING="2" CELLSPACING="2">
    <TR ALIGN="center" BGCOLOR="orange">
     <TD>Posició</TD>
    <?php
       echo "<TD BGCOLOR='orange'>".key($matriu1)."</TD>";
       while(next($matriu1)){
          echo "<TD BGCOLOR='orange'>".key($matriu1)."</TD>";
       }
    ?>     </TR>     <TR ALIGN="center">     <TD BGCOLOR="orange">Valor</TD>
    <?php
       echo "<TD>".reset($matriu1)."</TD>";
       while(next($matriu1)){
          echo "<TD>".current($matriu1)."</TD>";
       }
    ?>
   </TABLE><BR>
   <TABLE BORDER="1" CELLPADDING="2" CELLSPACING="2">
    <TR ALIGN="center" BGCOLOR="orange">
    <TD>Clau</TD>
    <?php
       end($matriu2);
       echo "<TD BGCOLOR='orange'>".key($matriu2)."</TD>";
       while(prev($matriu2)){
          echo "<TD BGCOLOR='orange'>".key($matriu2)."</TD>";
       }
    ?>     </TR>     <TR ALIGN="center">
    <TD BGCOLOR="orange">Valor</TD>
    <?php
       end($matriu2);
       echo "<TD>".current($matriu2)."</TD>";
       while(prev($matriu2)){
          echo "<TD>".current($matriu2)."</TD>";
       }
    ?>
   </TABLE>
  </CENTER>
 </BODY>
</HTML>
```



- **each(**matriu**):** s'utilitza per a recórrer arrays (sobre tot els associatius), però retorna un parell de valors corresponents a la clau i al valor associat a aquesta clau. A més, avança el punter intern fins al següent element. Si el punter apunta a l'última posició de l'array, l'execució d'aquesta funció retorna *false*.

- **list():** assigna una llista de variables en una sola operació. Normalment s'utilitza en combinació amb la funció anterior **each()**.

```
<?php
   while(list($pos,$valor)=each($matriz1)){
      echo "<TR ALIGN='center'><TD>".$pos."</TD>";
      echo "<TD>".$valor."</TD></TR>";
   }
 ?>
```

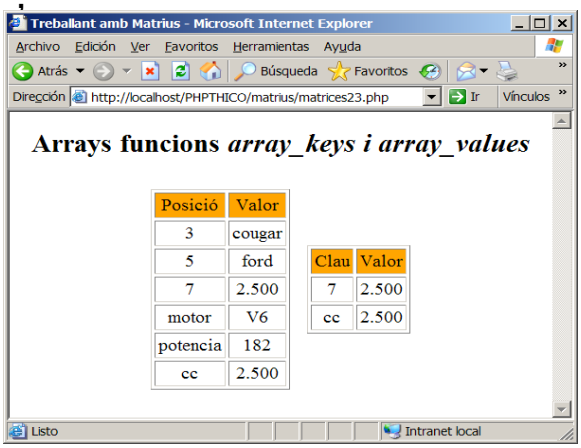
Exercici 4
Mostra l'array de l'exemple anterior mitjançant **each()** i **list()**

- **array_keys(**matriu**):** retorna les claus que formen l'array. Admet un paràmetre opcional que permet seleccionar només les claus el valor de les quals coincideix amb un patró donat.

- **array_values(**matriu**):** retorna els valors que formen part de l'array.

El següent exemple mostra el recorregut per 2 arrays no seqüencials fent ús d'aquestes instruccions:

```
<?php
   $claves=array_keys($matriu1);
   $valores=array_values($matriu1);
   for($i=0;$i<sizeof($claves);$i++){
      echo "<TR ALIGN='center'><TD>".$claves[$i]."</TD>";
      echo "<TD>".$valores[$i]."</TD></TR>";
   }
?>
<?php
   $claves=array_keys($matriu1,"2.500");
   for($i=0;$i<sizeof($claves);$i++){
      echo "<TR ALIGN='center'><TD>".$claves[$i]."</TD>";
      echo "<TD>".$matriu1[$claves[$i]]."</TD></TR>";
   }
?>
```



4.5. ORDENAR UN ARRAY

A vegades és necessari ordenar els elements d'un array per després, per exemple, poder llistar-los en ordre alfabètic.

- **sort(**matriu**):** ordena alfanumèricament els valors dels elements d'un array de menor a major. Per a ordenar de forma inversa es disposa de la funció **rsort(**matriu**).**

NOTA: Si s'apliquen aquestes funcions a arrays associatius, perdran les seves claus i es convertiran en array simples o escalars.

```
<?php
 $matriu1[0]="Madrid";
 $matriu1[1]="Zaragoza";
 $matriu1[2]="Bilbao";
 $matriu1[3]="Valencia";
 $matriu1[4]="Lerida";
 $matriu1[5]="Alicante";
?>

 <?php
    sort($matriu1);
    while(list($pos,$valor)=each($matriu1)){
       echo "<TR ALIGN='center'><TD>".$pos."</TD>";
       echo "<TD>".$valor."</TD></TR>";
    }
 ?>

 <?php
    rsort($matriu1);
    while(list($pos,$valor)=each($matriu1)){
       echo "<TR ALIGN='center'><TD>".$pos."</TD>";
       echo "<TD>".$valor."</TD></TR>";
    }
 ?>
```

Com es pot veure al resultat, les dues funcions, en ordenar els valors de l'array, ordenen en realitat els índexs:

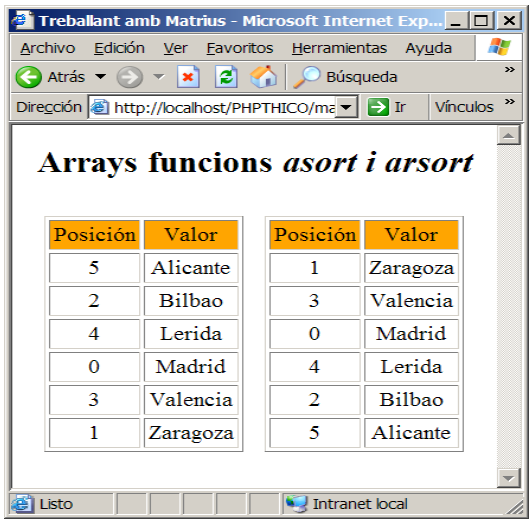
Per evitar aquest efecte lateral de redefinició d'índexs, tant per arrays escalars com associatius, es poden utilitzar les següents funcions:

- **asort(**matriu**):** ordena alfanumèricament els valors dels elements d'un array de major a menor, però mantenint la relació existent entre els índexs i els seus valors associats. Això és possible donat que la ordenació es fa sobre els elements de l'array i no sobre els índexs.

- **arsort(**matriu**):** realitza la ordenació inversa.

```
<?php
   asort($matriu1);
   while(list($pos,$valor)=each($matriu1)){
      echo "<TR ALIGN='center'><TD>".$pos."</TD>";
      echo "<TD>".$valor."</TD></TR>";
   }
?>

<?php
   arsort($matriu1);
   while(list($pos,$valor)=each($matriu1)){
      echo "<TR ALIGN='center'><TD>".$pos."</TD>";
      echo "<TD>".$valor."</TD></TR>";
   }
?>
```



- **ksort(**matriu**):** ordena alfanumèricament les claus d'un array de menor a major manteniment les correlacions entre clau i valor associat. La funció que realitza la ordenació inversa és **krsort(**matriu**).**

```
<?php
  $matriu1[d]="Madrid";
  $matriu1[c]="Zaragoza";
  $matriu1[e]="Bilbao";
  $matriu1[b]="Valencia";
  $matriu1[f]="Lerida";
  $matriu1[a]="Alicante";
?>

  <?php
     ksort($matriu1);
     while(list($pos,$valor)=each($matriu1)){
        echo "<TR ALIGN='center'><TD>".$pos."</TD>";
        echo "<TD>".$valor."</TD></TR>";
     }
  ?>
<?php
     krsort($matriu1);
     while(list($pos,$valor)=each($matriu1)){
        echo "<TR ALIGN='center'><TD>".$pos."</TD>";
        echo "<TD>".$valor."</TD></TR>";
     }
  ?>
```

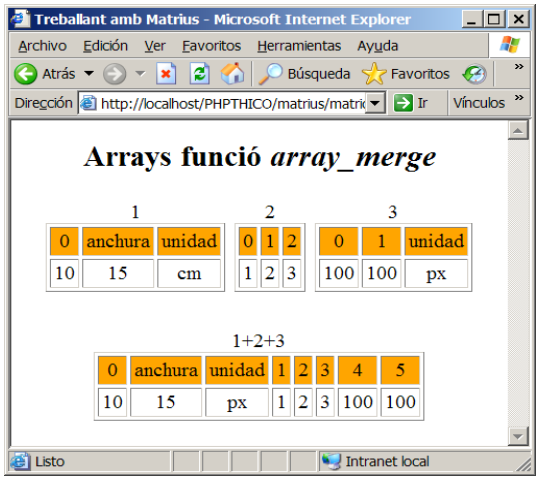
4.6. ALTRES OPERACIONS

4.6.1. MODIFICAR UN ARRAY

Les següents funcions permeten modificar el tamany d'un array afegint elements o unint diferents arrays en un de sol.

- **array_merge(**matriu1, matriu2**):** combina en un sol array els valors dels arrays que rep com a arguments. Els elements es van afegint al final del primer array. Si s'estan unint arrays associatius, s'ha de tenir en compte que les claus amb igual nom no s'afegeixen a l'array, sinó que s'actualitzen amb l'últim valor subministrat.

```
$matriu1=array(-"altura"=>"10","anchura"=>"15","unidad"=>"cm");
 $matriu2=array("1","2","3");
 $matriu3=array("100","100","unidad"=>"px");
 $matriuM=array_merge($matriu1,$matriu2,$matriu3);
```



- **array_pad(**matriu, tamany, farcit**):** retorna un array igual a *matriu* però modificant el seu nombre d'elements fins aconseguir la longitud desitjada (*tamany*) afegint nous elements per la dreta (si *tamany* és **positiu**) o per l'esquerra (si *tamany* és **negatiu**).

```
<?php
 $matriz1=array("a","b","c");
 $matrizP1=array_pad($matriz1,-5,"-?-");
 $matrizP2=array_pad($matriz1,5,"-?-");
?>
```