Ciclo de Grado Superior: Desarrollo de Aplicaciones WEB

MÓDULO 2ºDAW:
Desarrollo Web en entorno Servidor.

Autor: Jorge López.

Tema 01: Sintaxis básica en PHP - Parte 5ª

12 Arrays

- ¿Qué es un Array?
- Creación de arrays en PHP
- Funciones de arrays
- Interactuar con arrays
- Arrays multidimensionales
- Arrays irregulares

12 Arrays.

¿Qué es un Array?

Un **array** es una colección de valores con un único nombre. Para acceder a los distintos valores de la variable se utiliza un índice **numérico** o **alfanumérico**.

Nos vamos a servir de la definición de un **array** en el **lenguaje C**, para ver la potencia de este tipo de datos en PHP. La definición es la siguiente:

```
int mi_array[100]; //Esto es C
```

- Lo primero que debe llamarnos la atención es que la variable está predefinida como int (entero). En **PHP** los **arrays** no tienen que **definirse** de una **forma** concreta, sino que pueden tomar distintos tipos de valores: enteros, caracteres, objetos, etcétera.

- Lo siguiente es que, en C, se necesita saber de antemano el número de valores máximo que podrá tomar el array. Esto **no** es **necesario en PHP**, ya que podrá ir creando valores nuevos a medida que lo vaya necesitando.

- Lo último es que en C el índice para acceder a los 100 valores distintos debe ser **numérico**, en cambio, en **PHP** el **índice** puede ser **numérico** o **alfanumérico**.

Creación de arrays en PHP

Vamos a ver **tres formas distintas** de **crear** un **array** dentro de un **script** de **PHP**.

(1) Asignación directa:

El camino más simple y, por otro lado lógico, es asignar valores cuando se necesiten. La primera vez que asignemos un valor, el **array** se creará en el entorno:

```
<?php
$mi_array[1] = 23; // Asignación directa
?>
```

De esta forma tenemos un valor asignado al índice 1 del array.

Puede asignar **cualquier índice** en la creación de este tipo de dato, e incluso **no asignar ninguno**, de forma que PHP se encarga de asociar un índice distinto para cada valor.

Veamos un **ejemplo**:

```php
<?php
    $mi_array[] = 23; // Empieza en el índice 0
    $mi_array[] = 54; // sigue en el índice 1
    $mi_array[] = 'pepe'; // sigue en el índice 2
    echo $mi_array[0]."<br>";
    echo $mi_array[1]."<br>";
    echo $mi_array[2]."<br><br>";

    // ahora almaceno información en el índice que yo quiera

    $mi_array[34] = 23;
    $mi_array[12] = 54;
    $mi_array[127] = 'pepe';
    // oberva que el íncide no es obligatorio que sea numérico
    $mi_array['luis'] = 'ejemplo';

    echo $mi_array[34]."<br>";
    echo $mi_array[12]."<br>";
    echo $mi_array[127]."<br>";
    echo $mi_array['luis']."<br>";
?>
```

Resultado:

```
23
54
pepe

23
54
pepe
ejemplo
```

(2) función array():

Esta función **crea un array con los valores que pase como datos de entrada**. Los índices serán añadidos automáticamente empezando desde 0. Si no asigna parámetros a **array( )**, la función le devolverá un array vacío.

```php
<?php
$mi_array = array(23,45,76,23);
?>
```

La función “array” es similar a esto:

```php
<?php
$mi_array[0] = 23;
$mi_array[l] = 45;
$mi_array[2] = 76;
$mi_array[3] = 23;
?>
```

Otro **ejemplo**:

```php
<?php
    $cars = array("Volvo", "BMW", "Toyota");
    echo "I like " .$cars[0].",".$cars[1]." and ".$cars[2].".";
?>
```

Crea un fichero llamado **prueba.php** y prueba este código:

```php
<?php
   header('Content-Type: text/html; charset=UTF-8');
   $coches = array("Volvo", "BMW", "Toyota","Mercedes","Lexus");
   echo "<br>";

   // *****forma 1*****
   // print_r() imprime información legible sobre una variable
   // esta forma nos sirve de bien poco
   // servirá para ver el contenido del array cuando hagamos ejercicios
   echo "Imprimir array FORMA-1:<br>";
   print_r(array_values($coches));
   echo "<br><br>";
   // *****forma 2*****
   echo "Imprimir array FORMA-2:<br>";
   // calculo el número de elementos de un array con count()

   $longitud = count($coches);

   // recorro todos los elementos y los imprimo
   for($i=0; $i<$longitud; $i++)
       {
       //visualizo el valor de cada elemento
       echo "posición nº".$i."=".$coches[$i];
       echo "<br>";
       }
   echo "<br>";

   // *****forma 3*****
   echo "Imprimir array FORMA-3:<br>";
   // de esta forma podemos recorrer arrays con índices numéricos o alfanuméricos
   // fíjate que $posicion y $coche son variables que no existen y van tomando valores en cada
iteración del bucle
   foreach($coches as $posicion=>$coche)
      {
      echo "En la posición nº" .$posicion. "=" . $coche;
      echo "<br>";
      }
   echo "<br>";

   // ahora hacemos lo mismo pero con un array con índices alfanuméricos
   $equipo = array('portero'=>'Pepe', 'defensa'=>'Luis', 'medio'=>'Juan', ' delantero'=>'Manolo');

   foreach($equipo as $posicion=>$jugador)
      {
      echo "El " . $posicion . " es " . $jugador;
      echo "<br>";
      }
?>
```

La función **array()** permite también añadir índices a los valores que se introducen.
Para ello se utiliza el operador **=>** de esta forma:

```
$mi_array = array(0 => 23, 1 => 45, 2 => 76);
```

También es posible añadir índices que no sean correlativos o índices alfanuméricos, incluso mezclar los dos tipos, vamos a ver un **ejemplo** para que quede claro esta forma de trabajar:

Crea un fichero llamado **prueba.php** y prueba este código:

```
<?php
$mi_array = array(0 => 1000, 1 => 2000, 2 => 3000,"cero" => 4000, "uno" => 5000, "dos"
=> "pepe");
// calculo nº elementos del array
$elementos=count($mi_array);
echo $elementos."<br><br>";

//veremos que no sale nada en las posiciones con índices alfanuméricos
$x=0;
while (isset($mi_array[$x]))
{
  echo $mi_array[$x]."<br>";
  $x++;
}

// no se han visualizado todos los valores
// los imprimo a mano
echo $mi_array["cero"]."<br>";
echo $mi_array["uno"]."<br>";
echo $mi_array["dos"]."<br>";

// por tanto el recorrido habría que hacerlo así
// para que se visualicen todos los valores
echo "<br><br>";

foreach($mi_array as $indice=>$valor)
  {
  echo "En la posición " . $indice . " está el valor: " . $valor;
  echo "<br>";
  }
?>
```

Como hemos visto en el ejemplo, en el **caso** en que como índice utilicemos un **índice alfanumérico**, en este caso para acceder a la información que contiene el array de forma manual, en esa posición estamos obligados a escribir el nombre del índice que habrá de ser una cadena y debe ponerse entre comillas (puedes verlo en el ejemplo anterior).

(3) funciones que devuelven arrays:

La última forma de obtener un array es utilizando alguna de las funciones que devuelven este tipo de datos. Es muy frecuente que las funciones que manejan bases de datos devuelvan las ocurrencias dentro de un array.

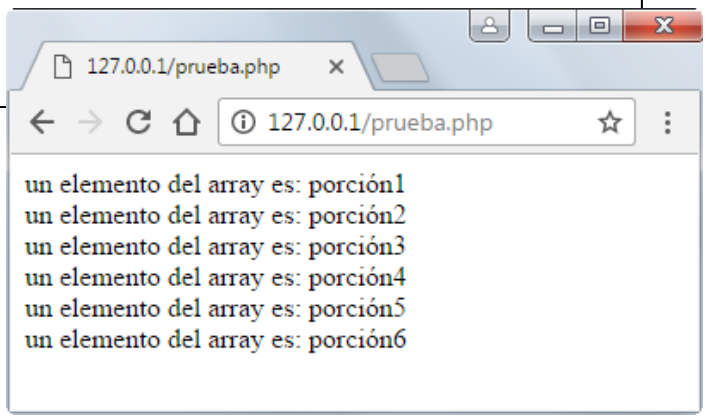
Por ejemplo: la función **range()** devuelve un array con valores numéricos, que van desde un número de inicio hasta un número final tal y como se muestra en el ejemplo:

```
<?php
$mi_array = range(120,130) ;
?>
```

Esta función crea un array, empezando desde el **índice 0** y **el valor 120**, hasta el **índice 10** y el **valor 130**.

Por ejemplo: la función **explode()** nos devuelve un array a partir de una cadena de texto, tendremos que indicarle el **carácter de división** a la hora de crear el array:

```
<?php

$pizza  = "porción1,porción2,porción3,porción4,porción5,porción6";
$porciones = explode(",", $pizza);

foreach ($porciones as $valor)
 {
  echo ("un elemento del array es: $valor<br>") ;
 }

?>
```



Arrays multidimensionales

Hasta aquí hemos visto ejemplos de **arrays** de **una sola dimensión**. **PHP** soporta el uso de arrays de **varias dimensiones** fácilmente, aunque suelen ser **complejos de entender y de utilizar**.

En un **array** normalmente hablamos de filas y columnas y decimos que el array tiene **m filas** y **n columnas**. Así, cada elemento queda identificado por su **fila i** y su **columna j** (también podríamos decir **fila f** y **columna c**).

Ejemplo: el siguiente array (**$valor**) tiene **2 dimensiones**:

4 COLUMNAS

3 FILAS

1	14	8	3
6	19	7	2
3	13	4	1

Accedemos a una posición de esta forma:

$valor [f] [c]
$valor [1] [2]=7

Por ejemplo, en la matriz que hemos mostrado anteriormente podríamos decir que tenemos **3 filas** (fila 0, fila 1 y fila 2) y **4 columnas** (columna 0, columna 1, columna 2, columna 3). Podríamos decir que el elemento de la fila 1, columna 3 **es igual a 2**.

Crea un fichero llamado **prueba.php** y prueba este código:

```php
<?php
   $valor [0] [0] = 1; $valor [0] [1] = 14; $valor [0] [2] = 8; $valor [0] [3] = 3;
   $valor [1] [0] = 6; $valor [1] [1] = 19; $valor [1] [2] = 7; $valor [1] [3] = 2;
   $valor [2] [0] = 3; $valor [2] [1] = 13; $valor [2] [2] = 4; $valor [2] [3] = 1;
    // imprimimos los valores del array
   for ($f=0; $f<=2; $f++)
   {
      for ($c=0; $c<=3; $c++)
      {
         echo $valor [$f] [$c]."----";
      }
      echo "<br><br>";
   }
?>
```

El **array** también lo podríamos haber creado de **estas 2 formas**:

```php
$valor= array
(
   array(1 , 14 , 8 , 3 ),
   array(6 , 19 , 7 , 2 ),
   array(3 , 13 , 4 , 1 )
);
```

```php
$valor[0]=array(1,14,8,3);
$valor[1]=array(6,19,7,2);
$valor[2]=array(3,13,4,1);
```

¿Cómo podríamos saber el **nº de filas** y el **nº de columnas** en un **array** de **2 dimensiones**?

Crea un fichero llamado **prueba.php** y prueba este código:

```php
// así calculamos las filas del array
echo "FILAS en el array: ".count($valor)."<br>";

// así calculamos las columnas del array
echo "COLUMNAS en el array: ".count($valor[0]);
```

Otras formas de **recorrer** el array:

Añade este código al código anterior:

```php
<?php
    // imprimimos los valores del array
   for ($f=0; $f<count($valor); $f++)
   {
      for ($c=0; $c<count($valor[$f]); $c++)
      {
         echo $valor [$f] [$c]."----";
      }
      echo "<br><br>";
   }
?>
```

Añade este código al código anterior:

```php
<?php
// imprimimos los valores del array
// sin utilizar count()
$f=0; $c=0;
while (isset($valor[$f]))
{
   while (isset($valor[$f][$c]))
   {
      echo $valor [$f] [$c]."----";
      $c++;
   }
   $f++; $c=0;
   echo "<br><br>";
}
?>
```

Ejemplo:

- Esto **se podría hacer** en **PHP** sin problemas: (fíjate en el **índice** de las **filas** y **columnas**).
- No tenemos que empezar en la **fila 0** obligatoriamente.

	20	21	22	23
16	1	14	8	3
17	6	19	7	2
18	3	13	4	1

Crea un fichero llamado **prueba.php** y prueba este código:

```php
<?php
$valor [16] [20] = 1; $valor [16] [21] = 14; $valor [16] [22] = 8; $valor [16] [23] = 3;
$valor [17] [20] = 6; $valor [17] [21] = 19; $valor [17] [22] = 7; $valor [17] [23] = 2;
$valor [18] [20] = 3; $valor [18] [21] = 13; $valor [18] [22] = 4; $valor [18] [23] = 1;
// imprimimos los valores del array
$f=16; $c=20;
while (isset($valor[$f]))
{
   while (isset($valor[$f][$c]))
   {
      echo $valor [$f] [$c]."----";
      $c++;
   }
   $f++; $c=20;
   echo "<br><br>";
}
?>
```

Ejemplo:

- recorrido de un **array** de **2** **dimensiones** con la instrucción **foreach()**:

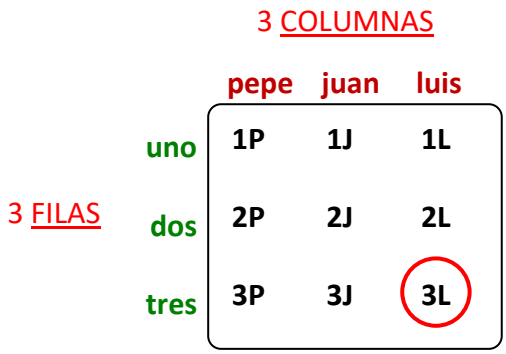
	0	1	2	3
0	1	14	8	3
1	6	19	7	2
2	3	13	4	1

Crea un fichero llamado **prueba.php** y prueba este código:

```php
<?php
$tabla [0] [0] = 1; $tabla [0] [1] = 14; $tabla [0] [2] = 8; $tabla [0] [3] = 3;
$tabla [1] [0] = 6; $tabla [1] [1] = 19; $tabla [1] [2] = 7; $tabla [1] [3] = 2;
$tabla [2] [0] = 3; $tabla [2] [1] = 13; $tabla [2] [2] = 4; $tabla [2] [3] = 1;

foreach($tabla as $indice=>$contenido)
{
    //$contenido -> coge como valor una fila entera;
    //por tanto -> $contenido es un array unidimensional;
    foreach($contenido as $indice2=>$contenido2)
    {
       echo "En la posición ".$indice."-".$indice2." es: " . $contenido2;
       echo "<br>";
    }
    echo "<br><br>";
}

// otra forma de visualización
foreach($tabla as $indice=>$contenido)
{
    foreach($contenido as $indice2=>$contenido2)
    {
       echo $contenido2."--";
    }
    echo "<br>";
}
?>
```

Ejemplo:

- recorrido de un **array** de **2 dimensiones** donde los **índices** del array son **alfanuméricos**.
- vamos a utilizar la instrucción **foreach().**



Accedemos a una posición de esta forma:

$valor [f] [c]
$valor [tres] [luis]=3L

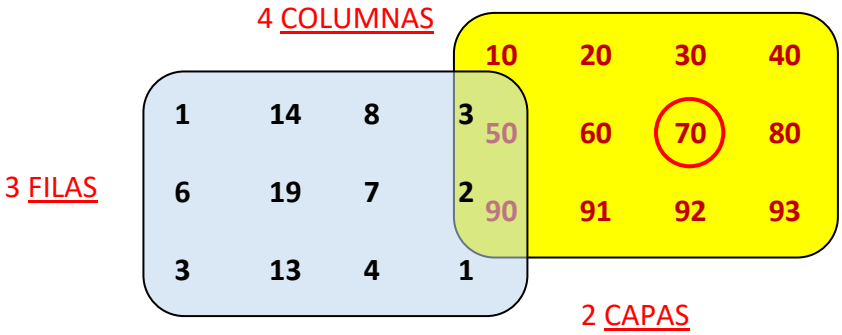
Crea un fichero llamado **prueba.php** y prueba este código:

```php
<?php
$tabla['uno']['pepe']="1P";
$tabla['uno']['juan']="1J";
$tabla['uno']['luis']="1L";

$tabla['dos']['pepe']="2P";
$tabla['dos']['juan']="2J";
$tabla['dos']['luis']="2L";

$tabla['tres']['pepe']="3P";
$tabla['tres']['juan']="3J";
$tabla['tres']['luis']="3L";

foreach($tabla as $indice=>$contenido)
{
    foreach($contenido as $indice2=>$contenido2)
    {
       echo $contenido2."--";
    }
    echo "<br>";
}
?>
```

Resultado:

1P--1J--1L--
2P--2J--2L--
3P--3J--3L--

Ejemplo: el siguiente **array** tiene **3 dimensiones**:

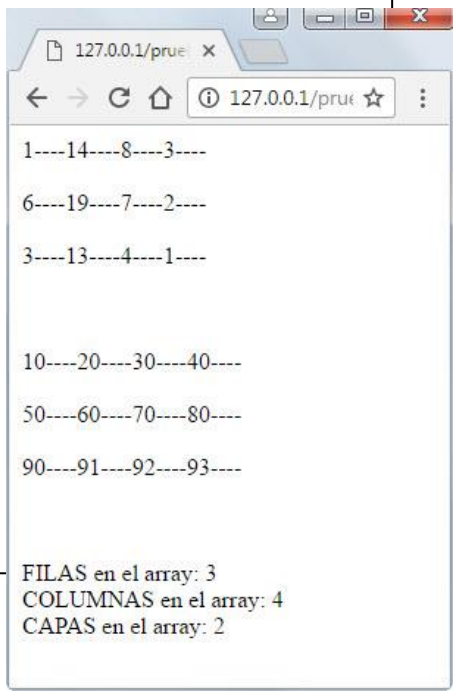


Accedemos a una posición de esta forma:

$valor [f] [c] [ca]
$valor [1] [2] [1]=70

Crea un fichero llamado **prueba.php** y prueba este código:

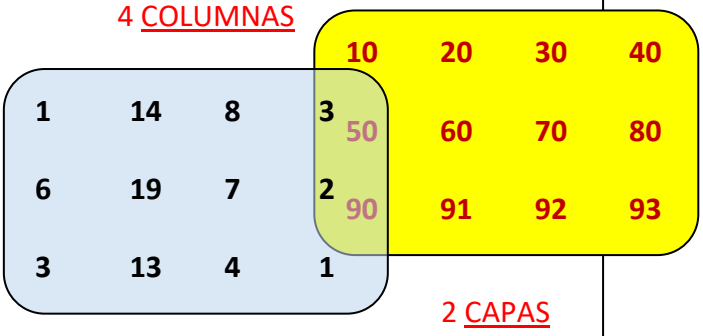
```php
<?php

$valor [0] [0] [0]= 1; $valor [0] [1] [0]= 14; $valor [0] [2] [0]= 8; $valor [0] [3] [0]= 3;
$valor [1] [0] [0]= 6; $valor [1] [1] [0]= 19; $valor [1] [2] [0]= 7; $valor [1] [3] [0]= 2;
$valor [2] [0] [0]= 3; $valor [2] [1] [0]= 13; $valor [2] [2] [0]= 4; $valor [2] [3] [0]= 1;

$valor [0] [0] [1]= 10; $valor [0] [1] [1]= 20; $valor [0] [2] [1]= 30; $valor [0] [3] [1]= 40;
$valor [1] [0] [1]= 50; $valor [1] [1] [1]= 60; $valor [1] [2] [1]= 70; $valor [1] [3] [1]= 80;
$valor [2] [0] [1]= 90; $valor [2] [1] [1]= 91; $valor [2] [2] [1]= 92; $valor [2] [3] [1]= 93;

for ($ca=0; $ca<count($valor[0][0]); $ca++)
{
   for ($f=0; $f<count($valor); $f++)
   {
         for ($c=0; $c<count($valor[0]); $c++)
         {
         echo $valor [$f] [$c] [$ca]."----";
         }
         echo "<br><br>";
   }
   echo "<br><br>";
}

echo "FILAS en el array: ".count($valor)."<br>";
echo "COLUMNAS en el array: ".count($valor[0])."<br>";
echo "CAPAS en el array: ".count($valor[0] [0]);

?>
```



El **array** también lo podríamos haber creado de **estas 2 formas**:

```
$valor= array(
    array( // fila [0]
        array(1,10),    // columna [0]
        array(14,20),   // columna [1]
        array(8,30),    // columna [2]
        array(3,40),    // columna [3]
    ),
    array( // fila [1]
        array(6,50),    // columna [0]
        array(19,60),   // columna [1]
        array(7,70),    // columna [2]
        array(2,80),    // columna [3]
    ),

    array( // fila [2]
        array(3,90),    // columna [0]
        array(13,91),   // columna [1]
        array(4,92),    // columna [2]
        array(1,93),    // columna [3]
    ),
);
```

3 FILAS

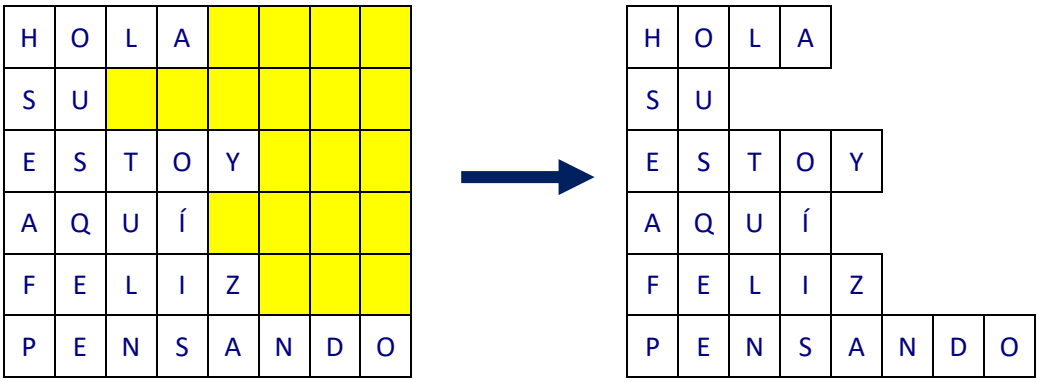



```
$valor[0]=array(array(1,10),array(14,20),array(8,30),array(3,40));
$valor[1]=array(array(6,50),array(19,60),array(7,60),array(2,80));
$valor[2]=array(array(3,90),array(13,91),array(4,92),array(1,93));
```

Arrays irregulares

Hasta aquí hemos visto ejemplos de **arrays regulares**.

¿**Qué ocurre** cuando el array tiene esta forma?
(las posiciones amarillas del dibujo **no existen**)



¿**Cómo** imprimimos (**recorremos**) el array?

Crea un fichero llamado **prueba.php** y prueba este código:

```php
<?php
   //SOLUCIÓN1-(utilizando count)
   // array original
   $original[0]=array('H','O','L','A');
   $original[1]=array('S','U');
   $original[2]=array('E','S','T','O','Y');
   $original[3]=array('A','Q','U','I');
   $original[4]=array('F','E','L','I','Z');
   $original[5]=array('P','E','N','S','A','N','D','O');

   echo "el array original es : <br><br>";
   for($fila=0; $fila< count($original); $fila++)
   {
      // calculamos el nº de columnas en cada fila que tratamos
      // ya que las filas no tienen el mismo nº de columnas
      for($columna=0; $columna< count($original[$fila]); $columna++)
      {
         echo $original[$fila][$columna]."*";
      }
      echo " columnas: ".count($original[$fila]);
      echo "<br>";
   }
   echo "<br>";
?>
```

Resultado:

el array original es :

H*O*L*A* columnas: 4
S*U* columnas: 2
E*S*T*O*Y* columnas: 5
A*Q*U*I* columnas: 4
F*E*L*I*Z* columnas: 5
P*E*N*S*A*N*D*O* columnas: 8

¿Pero **realmente es necesario** calcular el nº de columnas de cada fila?

Crea un fichero llamado **prueba.php** y prueba este código:

```php
<?php
//SOLUCIÓN2-(sin utilizar count)
$fila=0;
echo "el array original es : <br><br>";
while(isset($original[$fila]))
{
   $columna=0;
   while(isset($original[$fila][$columna]))
   {
      echo $original[$fila][$columna]."*";
      $columna++;
   }
   $fila++;
   echo " columnas: ".$columna;
   echo "<br>";
}
?>
```

Resultado:

el array original es :

H*O*L*A* columnas: 4
S*U* columnas: 2
E*S*T*O*Y* columnas: 5
A*Q*U*I* columnas: 4
F*E*L*I*Z* columnas: 5
P*E*N*S*A*N*D*O* columnas: 8

- recorrido de un **array** de **2 dimensiones irregular** con la instrucción **foreach()**:

Crea un fichero llamado **prueba.php** y prueba este código:

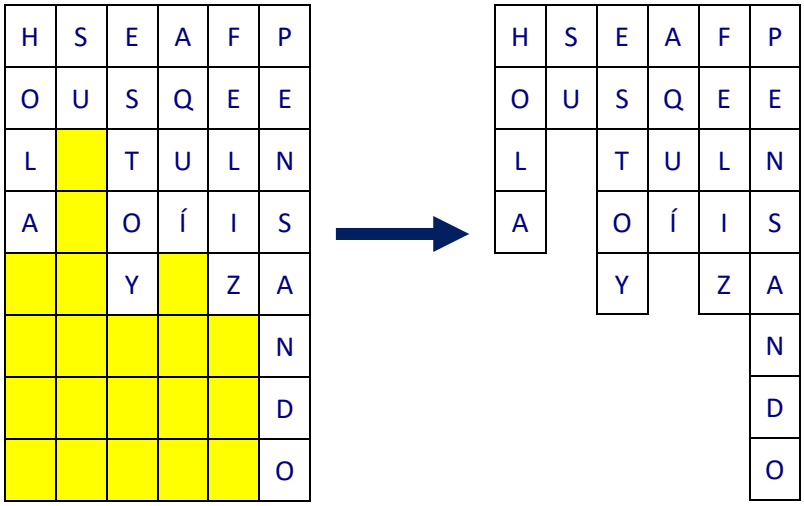
```php
<?php
//observa que el algoritmo es igual, no cambia
foreach($tabla as $indice=>$contenido)
{
    foreach($contenido as $indice2=>$contenido2)
    {
       echo $contenido2."--";
    }
    echo "<br>";
}
```

Resultado:

H--O--L--A--
S--U--
E--S--T--O--Y--
A--Q--U--I--
F--E--L--I--Z--
P--E--N--S--A--N--D--O--

Llegados a este punto tendrías que **SABER** diseñar un algoritmo que imprima (**recorra**) el siguiente array:

(las posiciones amarillas del dibujo **no existen**)

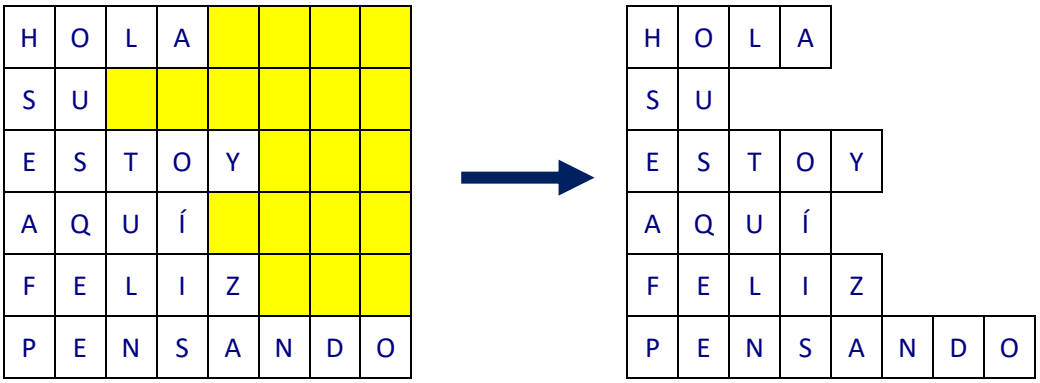


Te lo propongo como **ejercicio** en la hoja de ejercicios.
(puedes utilizar esta definición del array)

```php
<?php
$original[0][0]= "H"; $original[0][1]= "S"; $original[0][2]= "E"; $original[0][3]= "A";
$original[0][4]= "F";$original[0][5]= "P";

$original[1][0]= "O"; $original[1][1]= "U"; $original[1][2]= "S"; $original[1][3]= "Q";
$original[1][4]= "E";$original[1][5]= "E";

$original[2][0]= "L"; $original[2][2]= "T"; $original[2][3]= "U"; $original[2][4]= "L";
$original[2][5]= "N";

$original[3][0]= "A"; $original[3][2]= "O"; $original[3][3]= "I"; $original[3][4]= "I";
$original[3][5]= "S";

$original[4][2]= "Y"; $original[4][4]= "Z"; $original[4][5]= "A";

$original[5][5]= "N";
$original[6][5]= "D";
$original[7][5]= "O";
?>
```

Ejemplo:

- recorrido de un **array** de **2 dimensiones** **irregular** donde **leemos** las filas de **golpe**:



Crea un fichero llamado **prueba.php** y prueba este código:

```php
<?php

// array original
$original[0]=array('H','O','L','A');
$original[1]=array('S','U');
$original[2]=array('E','S','T','O','Y');
$original[3]=array('A','Q','U','I');
$original[4]=array('F','E','L','I','Z');
$original[5]=array('P','E','N','S','A','N','D','O');

echo "el array original es : <br><br>";

for($fila=0; $fila< count($original); $fila++)
{
   // vamos a leer una fila de golpe
   // en "$tabla_auxiliar" tendré una tabla unidimensional
   $tabla_auxiliar=$original[$fila];

   // recorro esa tabla
   $c=0;
   for($c; $c< count($tabla_auxiliar); $c++)
   {
      echo $tabla_auxiliar[$c]."*";
   }
   echo "<br>";
}
?>
```

Resultado:

el array original es :

H*O*L*A*
S*U*
E*S*T*O*Y*
A*Q*U*I*
F*E*L*I*Z*
P*E*N*S*A*N*D*O*