PHP-5

ESTRUCTURAS DE CONTROL

if ... elseif ... else

- Permite condicionar la ejecución de un bloque de sentencias al cumplimiento de una condición.
- La sintaxis de la construcción if más sencilla es la siguiente:

```
if (expresión) { bloque_de_sentencias }
```

La expresión se evalúa siempre, si el resultado es true se ejecuta el bloque de sentencias y si el resultado es false no se ejecuta el bloque de sentencias.

if ... elseif ... else

- A la construcción if se le puede añadiendir la instrucción else:

```
if (expresión) { bloque_de_sentencias_1 }
else { bloque_de_sentencias_2 }
```

- La expresión se evalúa siempre, si el resultado es true se ejecuta solamente el bloque de sentencias 1 y si el resultado es false se ejecuta solamente el bloque de sentencias 2.
- La construcción if ... else se puede extender añadiendo la instrucción elseif:

```
if (expresión_1) { bloque_de_sentencias_1 } elseif
(expresión_2) { bloque_de_sentencias_2 } else {
bloque_de_sentencias_3 }
```

if ... elseif ... else

- La construcción:

```
if (expresión_1) {
    $variable = expresion1;
} else
    $variable = expresion2;
}
```

- Se puede sustituir por:

```
$variable = (expresión_1)? expresion1 : expresion2;
```

- Se suele escribir en varias líneas para facilitar la legibilidad:

```
$variable = (expresión_1)
    ? expresion1
    : expresion2;
```

switch

- Equivalente a una construcción if ... elseif ... en las que las expresiones son comparaciones de igualdad de la misma expresión con valores distintos.

Mismo comportamiento que en C, sólo que la expresión del case puede ser integer, float o string.

```
switch (expresión_1) {
    case valor_1:
        bloque_de_sentencias_1;
        break;
    case valor_2:
        bloque_de_sentencias_2;
        break;
    ...
    case valor_n:
        bloque_de_sentencias_n;
        break;
}
```

- Que es equivalente a :

```
if (expresión_1 == valor_1) {
    bloque_de_sentencias_1
} elseif (expresión_1 == valor_2) {
    bloque_de_sentencias_2
} elseif (expresión_1 == valor_3) {
    bloque_de_sentencias_3
  ...
} elseif (expresión_1 == valor_n) {
    bloque_de_sentencias_n
}
```

for

- La sintaxis del bucle for es la siguiente:

```
for (expresión_inicial; condición_continuación; expresión_paso) {
    bloque_de_sentencias
}
```

- La expresión inicial se evalúa siempre. La condición de continuación se evalúa al principio de cada iteración: si el resultado es true se ejecuta primero el bloque de sentencias, después la expresión de paso y finalmente se evalúa nuevamente la condición de continuación; si el resultado es false el bucle se termina.

Bucles for anidados

- Es cuando un bucle se encuentra en el bloque de sentencias de otro bloque. Los bucles pueden tener cualquier nivel de anidamiento (un bucle dentro de otro bucle dentro de un tercero, etc.).
- En los bucles anidados es importante utilizar variables de control distintas, para no obtener resultados inesperados.

```php
<?php
print "<p>Comienzo</p>\n";
for ($i = 1; $i < 3; $i++) {                    // Bucle exterior

    for ($j = 10; $j < 12; $j++) {              // Bucle interior
        print "<p>i: $i -- j: $j</p>\n";
    }

}

print "<p>Final</p>\n";
?>
```

- En estos casos, si el bucle exterior se ejecuta M veces y el bucle interior se ejecuta N veces cada vez que se ejecuta el bucle exterior, en total el bloque de instrucciones se ejecutará M x N veces.

while

- Mismo comportamiento que en C y java. Sintaxis del bucle:

```
while (expresión) {
    bloque_de_sentencias
}
```

- La expresion se evaiua ai principio de cada iteración: si el resultado es true se ejecuta el bloque de sentencias; si el resultado es false el bucle se termina.

```
<?php
$i = 0;
while ($i < 5) {
    print "<p>$i</p>\n";
    $i++;
}
?>
```

```
<p>0</p>
<p>1</p>
<p>2</p>
<p>3</p>
<p>4</p>
```

do ... while

```
do {
     bloque_de_sentencias
} while (expresión)
```

- La expresión se evalúa al final de cada iteración: si el resultado es true se ejecuta el bloque de sentencias; si el resultado es false el bucle se termina.
- Mismo comportamiento que en C y java.

foreach

- Es muy útil para recorrer matrices cuyo tamaño se desconoce o matrices cuyos índices no son correlativos o numéricos (matrices asociativas).

```
foreach ($matriz as $valor) {
    bloque_de_sentencias
}
```

- El bucle ejecuta el bloque de sentencias tantas veces como elementos contenga la matriz $matriz y, en cada iteración, la variable $valor toma uno de los valores de la matriz.
- Mismo comportamiento que en java.

```php
<?php
$matriz = array(0, 1, 10, 100, 1000);
foreach ($matriz as $valor) {
    print "<p>$valor</p>\n";
}
?>
```

```
<p>0</p>
<p>1</p>
<p>10</p>
<p>100</p>
<p>1000</p>
```

- La sintaxis del bucle foreach puede también ser la siguiente:

```php
foreach ($matriz as $indice => $valor) {
    bloque_de_sentencias
}
```

- El bucle ejecuta el bloque de sentencias tantas veces como elementos contenga la matriz $matriz y, en cada iteración, la variable $valor toma uno de los valores de la matriz y la variable $indice toma como valor el índice correspondiente. Esta construcción es muy útil.

```php
<?php
$matriz = array ("red" => "rojo", "green" => "verde", "blue" => "azul");
foreach ($matriz as $indice => $valor) {
    print "<p>$indice : $valor</p>\n";
}
?>
```

```
<p>red : rojo</p>
<p>green : verde</p>
<p>blue : azul</p>
```

- **Problemas al recorrer matrices con el bucle for**

como las matrices de PHP pueden ser matrices asociativas y los índices de las matrices no tienen por qué ser valores numéricos correlativos, nos podemos encontrar con problemas. Por ejemplo, podemos hacer involuntariamente referencia a un término inexistente o algunos términos pueden resultar inaccesibles.

FOREACH

```
<?php
   print "<B><U>Sentencia foreach</U></B><BR>";
   print "<BR>Primer ejemplo de foreach<BR>";
   $matriz1 = array("PHP 3", "PHP 4", "PHP 5");
   foreach ($matriz1 as $var1) {
      print "Elemento de matriz 1: $var1<br>";
    }
    print "<BR>Segundo ejemplo de foreach<BR>";
    $matriz2["PHP 3"] = 1998;
    $matriz2["PHP 4"] = 2000;
    $matriz2["PHP 5"] = 2004;
    foreach ($matriz2 as $clave => $var1) {
            print "Elemento de matriz 2: clave  $clave
    año $var1<br>";
    }
?>
```

ESTRUCTURAS DE CONTROL: BREAK

- Sentencia break:
  - break n;
- Nos permite salir de una estructura de control o bucle de manera directa.
- Si la acompañamos de un número entero (opcional), indicamos el número de niveles de la estructura anidada que queremos saltar.

ESTRUCTURA DE CONTROL: CONTINUE

- Sentencia continue:

  –continue [n];

- Nos permite abandonar la iteración vigente de unaestructura de control.

- Si la acompañamos de un número entero (opcional), indicamos el número de niveles de la estructura anidada que queremos saltar.

Diferencias entre break y continue en PHP

- Son dos de las sentencias más utilizadas en PHP para **manipular el flujo de las iteraciones en las estructuras de control .**
- Ambas cortan el ciclo actual pero con una importante diferencia:
  - break finaliza la ejecución de la estructura control en curso.
  - continue finaliza la iteración actual de la estructura control y se inicia una nueva iteración.

Diferencias entre break y continue en PHP

- En otras palabras, continue le dice a PHP que la iteración actual se ha acabado y debe empezar en la evaluación de la condición que abre la estructura de control.
- Por su parte, break le dice a PHP que la evaluación de la estructura control actual ha terminado y que no siga haciendo iteraciones.

EJEMPLOS

```php
$letters = [ 'A', 'B', 'C' ];
foreach ( $letters as $letter) {
    if( 'A' == $letter ) {
        continue;
        echo 'Esto nunca se imprimirá';
    }
    echo $letter;
}
```

Se imprimirá la cadena BC ya que cuándo $letter es igual a A la iteración no alcanza la sentencia echo $letter; sino que vuelve al principio del foreach.

EJEMPLOS

```php
$letters = [ 'A', 'B', 'C' ];
foreach ( $letters as $letter) {
    if( 'A' == $letter ) {
        break;
        echo 'Esto nunca se imprimirá';
    }
    echo $letter;
}
```

- No se imprimirá nada ya que en la primera iteración, cuándo $letter es igual a A, se finaliza la ejecución de la estructura foreach y ninguno de los echo es alcanzado.

EJEMPLOS-ANIDADOS

- Cuándo break o continue se utilizan en una estructura de control anidada en otra se puede **especificar el número de estructuras a las que afectan**. El número por omisión es 1 y afecta sólo a la estructura actual

```php
while ( $foo ) {
    $items = [ 1, 2, 3 ];
    foreach( $items as $item ) {  <--------------------┐
        continue;                     --- vuelve aquí -----┘
    }
}

// Esta estructura es igual a la anterior
while ( $foo ) {
    $items = [ 1, 2, 3 ];
    foreach( $items as $item ) {  <--------------------┐
        continue 1;                   --- vuelve aquí -----┘
    }
}
```

Pero podemos hacer que vuelva a la estructura `while` externa con `c`

```php
while ( $foo ) {                  <--------------------┐
    $items = [ 1, 2, 3 ];
    foreach( $items as $item ) {
        continue 2;                   --- vuelve aquí ----┘
    }
}
```

EJEMPLOS-ANIDADOS

O con `break`:

```
while ( $foo ) {
    $items = [ 1, 2, 3 ];
    foreach( $items as $item ) {
        break 2;                --- salta aquí -----┐
    }
}
                                <--------------------┘
```

Ayuda para ejercicios

Php2-4

Matrices

Matrices de una dimensión

- **Matrices de índices numéricos**

Una matriz es un tipo de variable que puede almacenar varios valores a la vez. En las matrices de una dimensión (que se suelen llamar vectores) cada elemento se identifica por un índice que se escribe entre corchetes.

formato : $matriz[$indice]

Matrices de una dimensión

- Cada elemento de la matriz se comporta como una variable independiente y se pueden almacenar datos de tipos distintos en una misma matriz.
- Los valores de los índices numéricos suelen ser siempre positivos, y no permite valores decimales en los índices numéricos(En caso de usarlos, PHP los trunca automáticamente a enteros y como en el caso de los negativos, hay que tener cuidado al imprimirlos).

ejemplos

- Datos del mimo tipo:

```php
<?php
$nombres[0] = "Ana";
$nombres[1] = "Bernardo";
$nombres[2] = "Carmen";

print "<p>$nombres[2]<p>\n";
print "<p>$nombres[0]<p>\n";
print "<p>$nombres[1]<p>\n";
?>
```

Carmen
Ana
Bernardo

- Datos de diferente tipo:

```php
<?php
$datos[0] = "Santiago";
$datos[1] = "Ramón y Cajal";
$datos[2] = 1852;

print "<p>$datos[0] $datos[1] nació en $datos[2].<p>\n";
?>
```

Santiago Ramón y Cajal nació en 1852.

ejemplos

- Los valores de los índices numéricos si son negativos o decimales hay que sacarlos de la cadena:

```php
<?php
$nombres[-3] = "Hola";

print "<p>$nombres[-3]<p>\n";
?>
```

Parse error: syntax error, unexpected '-', expecting identifier (T_STRING) or variable (T_VARIABLE) or number (T_NUM_STRING) in **ejemplo.php** on line **3**

```php
<?php
$nombres[-3] = "Alba";

print "<p>{$nombres[-3]}<p>\n";
print "<p>" . $nombres[-3] . "<p>\n";
?>
```

Alba

Alba

Matrices de una dimensión

- **Matrices asociativas**: En php a diferencia de otros lenguajes los índices no tienen por qué ser numéricos y cuando lo son no tienen por qué ser consecutivos.

```php
<?php
$datos["nombre"] = "Santiago";
$datos["apellidos"] = "Ramón y Cajal";

print "<p>$datos[nombre] $datos[apellidos]<p>\n";
?>
```

Santiago Ramón y Cajal

```php
<?php
$nombres[5] = "Fernando";
$nombres[9] = "Jacinta";

print "<p>$nombres[9]<p>\n";
print "<p>$nombres[5]<p>\n";
?>
```

Jacinta

Fernando

Sintaxis_Resumen:

- Sintaxis:
  - array ([clave =>] valor, ...)
  - La clave es una cadena o un entero no negativo.
  - El valor puede ser de cualquier tipo válido en PHP, incluyendo otro array
  - Ejemplos:

    $color = array ('rojo'=>101, 'verde'=>51, 'azul'=>255);

    $medidas = array (10, 25, 15);
  - Acceso:

    $color['rojo'] // No olvidar las comillas

    $medidas[0]
  - El primer elemento es el 0

Imprimir todos los valores de una matriz: la función print_r()

- La función print_r(variable) permite imprimir todos los valores de una matriz de forma estructura. En general, print_r() imprime cualquier variable variable de forma legible.
- La notación con llaves ({}) **no admite la función** print_r().
- Aunque print_r() genera espacios y saltos de línea que pueden verse en el código fuente de la página, print_r() no genera etiquetas html, por lo que el navegador no muestra esos espacios y saltos de línea.

```php
<?php
$datos["nombre"] = "Santiago";
$datos["apellidos"] = "Ramón y Cajal";

print_r($datos);
?>
```

```
Array ( [nombre] => Santiago [apellidos] => Ramón y Cajal )
```

- Para mejorar la legibilidad una solución es añadir la etiqueta <pre>, que fuerza al navegador a mostrar los espacios y saltos de línea.

```php
<?php
$datos["nombre"] = "Santiago";
$datos["apellidos"] = "Ramón y Cajal";

print "<pre>\n"; print_r($datos); print "</pre>\n";
?>
```

```
Array
(
    [nombre] => Santiago
    [apellidos] => Ramón y Cajal
)
```

Imprimir todos los valores de una matriz: la función print_r()

- Si se añade el segundo argumento true, print_r() no imprime nada pero devuelve el texto que se imprimiría si no estuviera el argumento true . Se utiliza para guardar la respuesta en una variable que se puede imprimir posteriormente:

```php
<?php
$datos["nombre"] = "Santiago";
$datos["apellidos"] = "Ramón y Cajal";

$tmp = print_r($datos, true);
print "<p>La matriz es $tmp</p>\n";
?>
```

```php
<?php
$datos["nombre"] = "Santiago";
$datos["apellidos"] = "Ramón y Cajal";

print "<p>La matriz es " . print_r($datos, true) . "</p>\n";
?>
```

La matriz es Array ([nombre] => Santiago [apellidos] => Ramón y Cajal)

Imprimir todos los valores de una matriz: la función print_r()

- La notación con llaves ({}) no admite la función print_r().

```php
<?php
$datos["nombre"] = "Santiago";
$datos["apellidos"] = "Ramón y Cajal";

print "<p>La matriz es {print_r($datos, true)}</p>\n";
?>
```

Notice: Array to string conversion in **ejemplo.php** on line **5**

La matriz es {print_r(Array, true)}

Matrices de más de una dimensión

- Los elementos se identifican por varios índices que se escriben cada uno entre corchetes , su formato es:

$matriz[$indice1][$indice2]...

```php
<?php
$datos["pepe"]["edad"] = 25;
$datos["pepe"]["peso"] = 80;
$datos["juan"]["edad"] = 22;
$datos["juan"]["peso"] = 75;

print "<pre>\n"; print_r($datos); print "</pre>\n";
?>
```

```
Array
(
    [pepe] => Array
        (
            [edad] => 25
            [peso] => 80
        )

    [juan] => Array
        (
            [edad] => 22
            [peso] => 75
        )

)
```

Matrices de más de una dimensión

- La forma más adecuada de referenciar a los elementos de una matriz es de manera similar a como se guardaría en una base de datos, es decir, el primer índice correspondería al número de registro y el segundo índice iría tomando los valores de los campos de la tabla (nombre, edad, peso, etc).

```php
<?php
$datos[0]["nombre"] = "pepe";
$datos[0]["edad"] = 25;
$datos[0]["peso"] = 80;
$datos[1]["nombre"] = "juan";
$datos[1]["edad"] = 22;
$datos[1]["peso"] = 75;

print "<pre>\n"; print_r($datos); print "</pre>\n";
?>
```

```
Array
(
    [0] => Array
        (
            [nombre] => pepe
            [edad] => 25
            [peso] => 80
        )

    [1] => Array
        (
            [nombre] => juan
            [edad] => 22
            [peso] => 75
        )

)
```

Matrices de más de una dimensión

- No se pueden imprimir directamente valores de una matriz de más de una dimensión dentro de cadenas con print.

```php
<?php
$datos[0]["nombre"] = "pepe";
$datos[0]["edad"] = 25;
$datos[0]["peso"] = 80;

print "<p>$datos[0][nombre] tiene $datos[0][edad] años.</p>\n";
?>
```

Notice: Array to string conversion in **ejemplo.php** on line **6**

Notice: Array to string conversion in **ejemplo.php** on line **6**

Array[nombre] tiene Array[edad] años.

Matrices de más de una dimensión

- Para imprimir valores, debemos utilizar llaves ({}) o sacar los elementos de la cadena. En ambos casos, los índices no numéricos deben escribirse entre comillas.

```php
<?php
$datos[0]["nombre"] = "pepe";
$datos[0]["edad"] = 25;
$datos[0]["peso"] = 80;

print "<p>{$datos[0]["nombre"]} tiene {$datos[0]["edad"]} años.</p>\n";
print "<p>" . $datos[0]["nombre"] . " pesa " . $datos[0]["peso"] . " kg.</p>\n";
?>
```

```
pepe tiene 25 años.
pepe pesa 80 kg.
```

Crear una matriz

- Una matriz se puede crear definiendo explícitamente sus valores y definiciones posteriores añaden nuevos términos a la matriz.

```php
<?php
$cuadrados[5] = 25;
$cuadrados[3] = 9;

print "<pre>\n"; print_r($cuadrados); print "</pre>\n";
?>
```

```
Array
(
    [5] => 25
    [3] => 9
)
```

- **La función array()** también permite crear matrices, no añade valores a la matriz existente, sino que **la crea desde cero..**
- Si el argumento es una lista de valores, array($valor, ...), PHP crea una matriz de índices numéricos correlativos:

```php
<?php
$nombres = array("Ana", "Bernardo", "Carmen");

print "<pre>\n"; print_r($nombres); print "</pre>\n";
?>
```

```
Array
(
    [0] => Ana
    [1] => Bernardo
    [2] => Carmen
)
```

Crear una matriz

- La función array() sin argumentos crea una matriz vacía.
- La función array($indice => $valor, ...) también permite crear matrices asociativas:

```php
<?php
$cuadrados = array(3 => 9, 5 => 25, 10 => 100);

print "<pre>\n"; print_r($cuadrados); print "</pre>\n";
?>
```

```
Array
(
    [3] => 9
    [5] => 25
    [10] => 100
)
```

```php
<?php
$edades = array("Andrés" => 20, "Bárbara" => 19, "Camilo" => 17);

print "<pre>\n"; print_r($edades); print "</pre>\n";
?>
```

```
Array
(
    [Andrés] => 20
    [Bárbara] => 19
    [Camilo] => 17
)
```

Borrar una matriz o elementos de una matriz

- **La función unset() permite borrar una matriz o elementos de una matriz, si se intenta borrar un elemento no definido, PHP no genera ningún aviso.**

```php
<?php
$matriz = array(5 => 25, -1 => "negativo", "número 1" => "cinco");

print "<pre>\n"; print_r($matriz); print "</pre>\n";

unset($matriz[5]);

print "<pre>\n"; print_r($matriz); print "</pre>\n";
?>
```

```
Array
(
    [5] => 25
    [-1] => negativo
    [número 1] => cinco
)

Array
(
    [-1] => negativo
    [número 1] => cinco
)
```

```php
<?php
$matriz = array(5 => 25, -1 => "negativo", "número 1" => "cinco");

print "<pre>\n"; print_r($matriz); print "</pre>\n";

unset($matriz);

print "<pre>\n"; print_r($matriz); print "</pre>\n";
?>
```

```
Array
(
    [5] => 25
    [-1] => negativo
    [número 1] => cinco
)


Notice:  Undefined variable: matriz in prueba.php on line 8
```

CREAR-BORRAR ARRAY RESUMEN

- Hay dos formas de crear un array:
  - Asignación directa.
    - Se añaden sus elementos uno a uno, indicando el índice (mediante []).
    - Si el array no existía se crea.

```
$vec[5] = '1° elemento'; $vec[1] = "2° elemento";
$vec[] = '3° elemento'; $vec[6]= "3° elem..
```

- Utilizando el constructor **array().**
  - Se añaden entre paréntesis los primeros elementos del array. El primero de todos tiene asignado el índice cero.

```
$vec = array ( 3, 9, 2.4, 'Juan');
  // $vec[0] = 3; $vec[1] = 9; $vec[2] = 2.4;...
```

  - Se pueden fijar el índice con el operador **=>**

```
$vec = array ( 2=>3 ,9, 2.4, 'nombre'=>'Juan');
// $vec[2] = 3;$vec[3]=9;.. $vec['nombre']="Juan
```

CREAR-BORRAR ARRAY RESUMEN

- Ejemplo2:

```
$unarray = array("dia" => 15, 1 => "uno");
```

- Ejemplo3:

```
$otro = array("unarray" => array(0=>14, 4=>15),
              "nombre" => "Una tabla");
```

- Para eliminar un elemento del array hay que emplear unset()
  - unset($miarray['nombre']);
  - unset($miarray);
- Los arrays no se imprimen con echo, sino con *print_r*:

```
print_r ($unarray) ;
```

Añadir elementos a una matriz

- La notación $matriz[] = $valor permite añadir un elemento a una matriz. Si la matriz no existe previamente, la matriz se crea con el elemento indicado. El índice toma automáticamente valores numéricos consecutivos.

```php
<?php
$numeros[] = 66;
$numeros[] = 44;

print "<pre>\n"; print_r($numeros); print "</pre>\n";
?>
```

```
Array
(
    [0] => 66
    [1] => 44
)
```

- El índice del elemento creado es el entero siguiente al mayor de los valores existentes.

```php
<?php
$numeros = array(10 => 25, 3 => 23);
$numeros[] = 66;
$numeros[] = 44;

print "<pre>\n"; print_r($numeros); print "</pre>\n";
?>
```

```
Array
(
    [10] => 25
    [3] => 23
    [11] => 66
    [12] => 44
)
```

Añadir elementos a una matriz

- La función **array_push($matriz,$valor1,$valor2…)** también permite añadir uno o varios elementos a una matriz ya existente. Los índices de los valores añadidos no se pueden especificar, sino que toma automáticamente valores numéricos consecutivos
- El índice del elemento creado es el entero siguiente al mayor de los valores existentes.

```php
<?php
$numeros = array(10 => 25, 3 => 23);
array_push($numeros, 66, 44);

print "<pre>\n"; print_r($numeros); print "</pre>\n";
?>
```

```
Array
(
    [10] => 25
    [3] => 23
    [11] => 66
    [12] => 44
)
```

Copiar una matriz

- **Se puede copiar una matriz creando una nueva variable. Modificar posteriormente cualquiera de las dos no afecta a la otra.**

```php
<?php
$cuadrados = array(5 => 25, 9 => 81);
$cuadradosCopia = $cuadrados;
$cuadrados[] = 100;

print "<p>Matriz inicial (modificada):</p>\n";
print "<pre>\n"; print_r($cuadrados); print "</pre>\n\n";

print "<p>Copia de la matriz inicial (sin modificar):</p>\n";
print "<pre>\n"; print_r($cuadradosCopia); print "</pre>\n";
?>
```

Matriz inicial (modificada):

```
Array
(
    [5] => 25
    [9] => 81
    [10] => 100
)
```

Copia de la matriz inicial (sin modificar):

```
Array
(
    [5] => 25
    [9] => 81
)
```

Contar elementos de una matriz

- La función count($matriz)permite contar los elementos de una matriz.

```php
<?php
$nombres[1] = "Ana";
$nombres[10] = "Bernardo";
$nombres[25] = "Carmen";

$elementos = count($nombres);

print "<p>La matriz tiene $elementos elementos.</p>\n";
print "<pre>\n"; print_r($nombres); print "</pre>\n";
?>
```

```
La matriz tiene 3 elementos.
Array
(
    [1] => Ana
    [10] => Bernardo
    [25] => Carmen
)
```

Contar elementos de una matriz

- En una matriz multidimensional, la función count($matriz) considera la matriz como un vector de vectores y devuelve simplemente el número de elementos del primer índice.

```php
<?php
$datos["pepe"]["edad"] = 25;
$datos["pepe"]["peso"] = 80;
$datos["juan"]["edad"] = 22;
$datos["juan"]["peso"] = 75;
$datos["ana"]["edad"]  = 30;

$elementos = count($datos);

print "<p>La matriz tiene $elementos elementos.</p>\n";
print "<pre>\n"; print_r($datos); print "</pre>\n";
?>
```

```
La matriz tiene 3 elementos.
Array
(
    [pepe] => Array
        (
            [edad] => 25
            [peso] => 80
        )

    [juan] => Array
        (
            [edad] => 22
            [peso] => 75
        )

    [ana] => Array
        (
            [edad] => 30
        )

)
```

Contar elementos de una matriz

- Para contar todos los elementos de una matriz multidimensional, habría que utilizar la función count($matriz, COUNT_RECURSIVE).

```php
<?php
$datos["pepe"]["edad"] = 25;
$datos["pepe"]["peso"] = 80;
$datos["juan"]["edad"] = 22;
$datos["juan"]["peso"] = 75;
$datos["ana"]["edad"]  = 30;

$elementos = count($datos, COUNT_RECURSIVE);

print "<p>La matriz tiene $elementos elementos.</p>\n";
print "<pre>\n"; print_r($datos); print "</pre>\n";
?>
```

La matriz tiene 8 elementos.

```
Array
(
    [pepe] => Array
        (
            [edad] => 25
            [peso] => 80
        )

    [juan] => Array
        (
            [edad] => 22
            [peso] => 75
        )

    [ana] => Array
        (
            [edad] => 30
        )

)
```

Máximo y mínimo

- La función max($matriz, …)devuelve el valor máximo de una matriz (o varias). La función min($matriz,…..)devuelve el valor mínimo de una matriz (o varias).

```php
<?php
$numeros = array (10, 40, 15, -1);
$maximo = max($numeros);
$minimo = min($numeros);

print "<pre>"; print_r($numeros); print "</pre>\n";
print "<p>El máximo de la matriz es $maximo.</p>\n";
print "<p>El mínimo de la matriz es $minimo.</p>\n";
?>
```

```
Array
(
    [0] => 10
    [1] => 40
    [2] => 15
    [3] => -1
)
El máximo de la matriz es 40.
El mínimo de la matriz es -1.
```

Máximo y mínimo

- Los valores no numéricos se tratan como 0, pero si 0 es el mínimo o el máximo, la función devuelve la cadena.

```php
<?php
$valores = array(10, 40, 15, "abc");
$maximo = max($valores);
$minimo = min($valores);

print "<pre>\n"; print_r($valores); print "</pre>\n";
print "<p>El máximo de la matriz es $maximo.</p>\n";
print "<p>El mínimo de la matriz es $minimo.</p>\n";
?>
```

```
Array
(
    [0] => 10
    [1] => 40
    [2] => 15
    [3] => abc
)
El máximo de la matriz es 40.
El mínimo de la matriz es abc.
```

RECORRER UN ARRAY

- Recorrer un array no secuencial o asociativo:
  - A través de funciones que actúan sobre un puntero interno:
    - current() - devuelve el valor del elemento que indica el puntero
    - pos() - realiza la misma función que current
    - reset- mueve el puntero al primer elemento del array.
    - end() - mueve el puntero al último elemento del array
    - next()- mueve el puntero al elemento siguiente
    - prev() - mueve el puntero al elemento anterior
    - count() - devuelve el número de elementos de un array
    - key() – devuelve el índice de la posición actual

EJEMPLO

```
$semana = array("lunes", "martes", "miércoles", "jueves",  "viernes",
                "sábado", "domingo");
echo count($semana);        //7
reset($semana);             //situamos el puntero en el 1ºelemento
echo current($semana);      //lunes
next($semana);
echo pos($semana);          //martes
end($semana);
echo pos($semana);          //domingo
prev($semana);
echo current($semana);      //sábado
echo key($semana);          // 5
```

RECORRER UN ARRAY: OTRAS FUNCIONES

- **list($var1, $var2, $var3, ...)**
  - Asigna valor a una lista de variables en una sola operación. Solo arrays numéricos

```
list($var1, $var2)= array("Lunes", "Martes");
$var1="Lunes", $var2="Martes"
```

- **each($unarray)**
  - En cada iteración devuelve el par clave/valor actual y avanza el cursor. Devuelve un array de 4 elementos:

    0,key → la clave

    1,value → el valor

```
$semana = array("lunes", "martes");
foreach ($semana as $k=>$v){
        echo "<pre>";
        print_r(each($semana));
        echo "</pre>";}
```

```
Array
(
    [1] => lunes
    [value] => lunes
    [0] => 0
    [key] => 0
)

Array
(
    [1] => martes
    [value] => martes
    [0] => 1
    [key] => 1
)
```

RECORRER UN ARRAY: OTRAS FUNCIONES

- Otra forma de recorrer un array utilizando las funciones anteriores sería:

```
$unarray = array('uno', 'dos', 'tres');
reset($unarray);
while (list($clave, $valor) = each($unarray))
        echo "$clave => $valor\n";
```

RECORRER UN ARRAY: OTRAS FUNCIONES

- **array_values ($unarray)**
  - Devuelve todos los valores del array en orden numérico.

```
$matriz = array("talla" => "XL",
                "color" =>"dorado");
print_r(array_values($matriz));
```

```
Array
(
    [0] => XL
    [1] => dorado
)
```

BUSCAR UN VALOR

- **array_search(valor, $matriz)**
  - Permite buscar un valor en un array y si lo encuentra devuelve su clave, sino devuelve NULL.

- **in_array(valor, $matriz, $strict)**
  - Devuelve *True* o *False* en función de la existencia o no de un valor en el array. Si *$strict* es *True se* tendrá en cuenta el tipo de los valores.
  - Es case-sensitive.

```
$a = array('1.10', 12.4, 1.13);
if (in_array('12.4', $a, true)) {
        echo "'12.4' Encontrado con chequeo STRICT\n";}
if (in_array(1.13, $a, true)) {
        echo "1.13 Encontrado con chequeo STRICT\n";}
```

```php
<?php
$valores = array (10, 40, 15, -1);

print "<pre>\n"; print_r($valores); print "</pre>\n";

if (in_array(15, $valores)) {
  print "<p>15 está en la matriz \$valores.</p>\n";
} else {
  print "<p>15 no está en la matriz \$valores.</p>\n";
}

if (in_array(25, $valores)) {
  print "<p>25 está en la matriz \$valores.</p>\n";
} else {
  print "<p>25 no está en la matriz \$valores.</p>\n";
}

if (in_array("15", $valores, true)) {
  print "<p>\"15\" está en la matriz \$valores.</p>\n";
} else {
  print "<p>\"15\" no está en la matriz \$valores.</p>\n";
}
?>
```

```
Array
(
    [0] => 10
    [1] => 40
    [2] => 15
    [3] => -1
)
```

15 está en la matriz $valores.

25 no está en la matriz $valores.

"15" no está en la matriz $valores.

BUSCAR UN VALOR

- array_key($matriz,$vslor[,$tipo]) busca el valor en la matriz y, si lo encuentra, devuelve una matriz cuyos valores son los índices de todos los elementos coincidentes.

```php
<?php
$valores = array (10, 40, 15, 30, 15, 40, 15);
print "<pre>\n"; print_r($valores); print "</pre>\n";

$encontrado = array_search(15, $valores);
print "<p>$encontrado</p>\n";

$encontrados = array_keys($valores, 15);
print "<pre>\n"; print_r($encontrados); print "</pre>\n";
?>
```

```
Array
(
    [0] => 10
    [1] => 40
    [2] => 15
    [3] => 30
    [4] => 15
    [5] => 40
    [6] => 15
)
2
Array
(
    [0] => 2
    [1] => 4
    [2] => 6
)
```

array_count_values($matriz)

- Cuenta las veces que aparece cada elemento de un array en ese array

```
$matriz = array(1, "hola", 1, "mundo", "hola");
array_count_values($matriz);     // devuelve array(1=>2,"hola"=>2,"mundo"=>1)
```

MODIFICAR ARRAYS

- mixed **array_pop** (array &$matriz) extrae y devuelve el último valor de la *matriz* , acortando la *matriz* en un elemento. Si *matriz* está vacía (o no es una matriz), se regresará **NULL**.
- Esta función ejecutará un reset() en el puntero array después de su uso.

MODIFICAR ARRAYS

```php
<?php
$pila = array ("naranja", "plátano", "manzana", "frambuesa");
$fruta = array_pop ($pila);
print_r($pila);
?>
```

Después de esto, *$pila* tendrá sólo 3 elementos:

```
Array
(
    [0] => naranja
    [1] => plátano
    [2] => manzana
)
```

y *frambuesa* será asignada a *$fruta*.

MODIFICAR ARRAYS

- int array_push(array &$matriz, $var1, $var2, ...): trata array como si fuera una pila y coloca la variable que se le proporciona al final del array. El tamaño del array será incrementado por el número de variables insertados. Tiene el mismo efecto que:
- <?php
  $array[] = $var;
  ?>

MODIFICAR ARRAYS

- mixed **array_shift** (array &$array): extrae el primer valor de la *matriz* y lo devuele, acortando la *matriz* en un elemento y moviendo todo hacia abajo. Todos los índices numéricos de la matriz serán modificados para empezar desde cero, mientras que los índices literales no serán tocados. Si *array* está vacío (o no es una matriz), se regresará **NULL**
- Esta función ejecutará un reset() en el puntero array después de su uso.

MODIFICAR ARRAYS

```php
<?php
$pila = array ("naranja", "plátano", "manzana", "frambueza");
$fruta = array_shift ($pila);
print_r($pila);
?>
```

Esto resulta en *$pila* conteniendo 3 elementos:

```
Array
(
    [0] => plátano
    [1] => manzana
    [2] => frambueza
)
```

y *naranja* es asignada a *$fruta*.

MODIFICAR ARRAYS

- int **array_unshift** (array &$matriz , mixed $var [, mixed $...]): añade los elementos que se le pasan al frente de la *matriz* . La lista de elementos es añadida como un todo, de modo que los elementos añadidos mantienen su orden. Todos los ídices numéricos de la matriz serán modificados para iniciar a contar desde cero mientras que los índices alfanuméricos no serán modificados.
- Devuelve el número de elementos en la *matriz* .

MODIFICAR ARRAYS

```php
<?php
$queue = array("orange", "banana");
array_unshift($queue, "apple", "raspberry");
?>
```

Esto resultará que *$queue* tenga los siguientes elementos:

```
Array
(
    [0] => apple
    [1] => raspberry
    [2] => orange
    [3] => banana
)
```

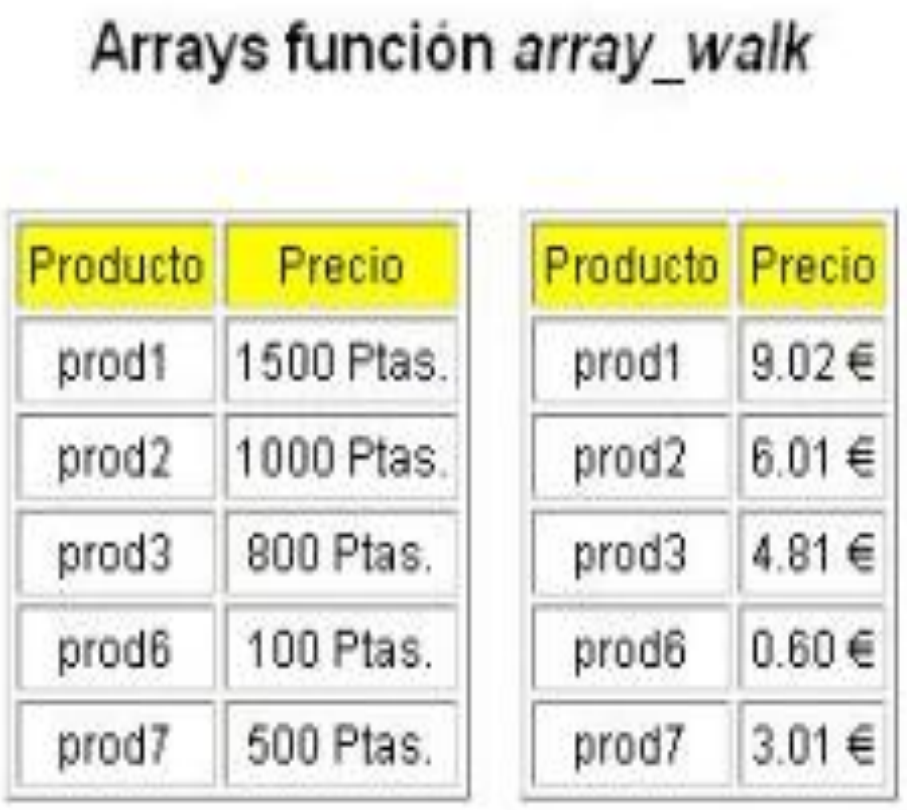
MODIFICAR ARRAYS

- *array_walk* permite aplicar una función definida por el usuario "func_usuario" a cada elemento de un array.
- la función *func_usuario()* recibe dos argumentos: a) el valor del elemento y b) su clave asociada. Si es necesario pasar otros parámetros, éstos se le pasarán previamente a la función *array_walk()*.

```
<BODY>
  <CENTER>
    <H2>Arrays función <I>array_walk</I></H2>
    <?php
      $precios['prod1']=1500;
      $precios['prod2']=1000;
      $precios['prod3']=800;
      $precios['prod6']=100;
      $precios['prod7']=500;
    ?>
   <TABLE BORDER="0" CELLPADDING="4" CELLSPACING="6">
    <TR ALIGN="center"><TD>
    <TABLE BORDER="1" CELLPADDING="2" CELLSPACING="2">
      <TR ALIGN="center" BGCOLOR="yellow">
      <TD>Producto</TD><TD>Precio</TD></TR>
      <?php
       while(list($pos,$valor)=each($precios)){
        echo "<TR ALIGN='center'><TD>".$pos."</TD><TD>";
        printf("%4d Ptas.",$valor);
        echo "</TD></TR>";
            }
      ?>
    </TABLE></TD><TD>
    <TABLE BORDER="1" CELLPADDING="2" CELLSPACING="2">
      <TR ALIGN="center" BGCOLOR="yellow">
      <TD>Producto</TD><TD>Precio</TD></TR>
      <?php
          function aEuros(&$valor,$clave){
              $valor=$valor/166.386;
          }

          array_walk($precios,'aEuros');
          reset($precios);
          while(list($pos,$valor)=each($precios)){
           echo "<TR ALIGN='center'><TD>".$pos."</TD><TD>";
           printf("%02.2f €",$valor);
           echo "</TD></TR>";
          }
```

MODIFICAR ARRAYS



- A parte de las funciones vistas en este capitulo, existen infinidad de funciones que se pueden consultar junto con el resto de teoria sobre php en : http://www.php.net/manual/es/

Ordenar una matriz

- Existen varias funciones de ordenamiento. Las más simples son las siguientes:
  - Asort($matriz, $opciones)ordena por orden alfabético / numérico de los valores

```php
<?php
$valores = array(5 => "cinco", 1 => "uno", 9 => "nueve");

print "<p>Matriz inicial:</p>\n\n";
print "<pre>\n"; print_r($valores); print "</pre>\n\n";

asort($valores);

print "<p>Matriz ordenada con asort():</p>\n\n";
print "<pre>\n"; print_r($valores); print "</pre>\n\n";
?>
```

Matriz inicial:

```
Array
(
    [5] => cinco
    [1] => uno
    [9] => nueve
)
```

Matriz ordenada con asort():

```
Array
(
    [5] => cinco
    [9] => nueve
    [1] => uno
)
```

Ordenar una matriz

- arsort($matriz, $opciones)ordena por orden alfabético / numérico inverso de los valores.

```php
<?php
$valores = array(5 => "cinco", 1 => "uno", 9 => "nueve");

print "<p>Matriz inicial:</p>\n\n";
print "<pre>\n"; print_r($valores); print "</pre>\n\n";

arsort($valores);

print "<p>Matriz ordenada con arsort():</p>\n\n";
print "<pre>\n"; print_r($valores); print "</pre>\n\n";
?>
```

Matriz inicial:

```
Array
(
    [5] => cinco
    [1] => uno
    [9] => nueve
)
```

Matriz ordenada con arsort():

```
Array
(
    [1] => uno
    [9] => nueve
    [5] => cinco
)
```

Ordenar una matriz

- ksort($matriz,$opciones):ordena por orden alfabético / numérico de los índices:

```php
<?php
$valores = array(5 => "cinco", 1 => "uno", 9 => "nueve");

print "<p>Matriz inicial:</p>\n\n";
print "<pre>\n"; print_r($valores); print "</pre>\n\n";

ksort($valores);

print "<p>Matriz ordenada con ksort():</p>\n\n";
print "<pre>\n"; print_r($valores); print "</pre>\n\n";
?>
```

Matriz inicial:

```
Array
(
    [5] => cinco
    [1] => uno
    [9] => nueve
)
```

Matriz ordenada con ksort():

```
Array
(
    [1] => uno
    [5] => cinco
    [9] => nueve
)
```

Ordenar una matriz

- Sort($matriz,$opciones):ordena por orden alfabético / numérico de los valores y genera nuevos índices numéricos consecutivos a partir de cero:

```php
<?php
$valores = array(5 => "cinco", 1 => "uno", 9 => "nueve");

print "<p>Matriz inicial:</p>\n\n";
print "<pre>\n"; print_r($valores); print "</pre>\n\n";

sort($valores);

print "<p>Matriz ordenada con sort():</p>\n\n";
print "<pre>\n"; print_r($valores); print "</pre>\n\n";
?>
```

Matriz inicial:

```
Array
(
    [5] => cinco
    [1] => uno
    [9] => nueve
)
```

Matriz ordenada con sort():

```
Array
(
    [0] => cinco
    [1] => nueve
    [2] => uno
)
```

Ordenar una matriz

- Rsort($atriz,$opciones): ordena por orden alfabético / numérico inverso de los valores y genera nuevos índices numéricos consecutivos a partir de cero:

```php
<?php
$valores = array(5 => "cinco", 1 => "uno", 9 => "nueve");

print "<p>Matriz inicial:</p>\n\n";
print "<pre>\n"; print_r($valores); print "</pre>\n\n";

rsort($valores);

print "<p>Matriz ordenada con rsort():</p>\n\n";
print "<pre>\n"; print_r($valores); print "</pre>\n\n";
?>
```

Matriz inicial:

```
Array
(
    [5] => cinco
    [1] => uno
    [9] => nueve
)
```

Matriz ordenada con rsort():

```
Array
(
    [0] => uno
    [1] => nueve
    [2] => cinco
)
```

Reindexar una matriz

- La función array($matriz)devuelve los valores de una matriz en el mismo orden que en la matriz original, pero renumerando los índices desde cero:

```php
<?php
$nombres = array("a" => "Ana", "b" => "Bernardo", "c" => "Carmen", "d" => "David");

print "<p>Matriz inicial:</p>\n\n";
print "<pre>\n"; print_r($nombres); print "</pre>\n\n";

$matriz = array_values($nombres);

print "<p>Matriz barajada con shuffle():</p>\n\n";
print "<pre>\n"; print_r($nombres); print "</pre>\n\n";
?>
```

Matriz inicial:

```
Array
(
    [a] => Ana
    [b] => Bernardo
    [c] => Carmen
    [d] => David
)
```

Matriz barajada con shuffle():

```
Array
(
    [0] => Ana
    [1] => Bernardo
    [2] => Carmen
    [3] => David
)
```

Barajar una matriz

- La función shuffle($matriz)baraja los valores de una matriz. Los índices de la matriz original se pierden, ya que se renumeran desde cero.

```php
<?php
$numeros = array(0, 1, 2, 3, 4, 5);

print "<p>Matriz inicial:</p>\n\n";
print "<pre>\n"; print_r($numeros); print "</pre>\n\n";

shuffle($numeros);

print "<p>Matriz barajada con shuffle():</p>\n\n";
print "<pre>\n"; print_r($numeros); print "</pre>\n\n";
?>
```

Matriz inicial:

```
Array
(
    [0] => 0
    [1] => 1
    [2] => 2
    [3] => 3
    [4] => 4
    [5] => 5
)
```

Matriz barajada con shuffle():

```
Array
(
    [0] => 3
    [1] => 1
    [2] => 5
    [3] => 0
    [4] => 4
    [5] => 2
)
```

```php
<?php
$numeros = array("a" => 1, "b" => 2, "c" => 3, "d" => 4);

print "<p>Matriz inicial:</p>\n\n";
print "<pre>\n"; print_r($numeros); print "</pre>\n\n";

shuffle($numeros);

print "<p>Matriz barajada con shuffle():</p>\n\n";
print "<pre>\n"; print_r($numeros); print "</pre>\n\n";
?>
```

Matriz inicial:

```
Array
(
    [a] => 1
    [b] => 2
    [c] => 3
    [d] => 4
)
```

Matriz barajada con shuffle():

```
Array
(
    [0] => 2
    [1] => 1
    [2] => 4
    [3] => 3
)
```

Ayuda para el bloque-2 de ejercicios

EXPLICACIÓN DE CÓMO SE CALCULA LA LETRA QUE CORRESPONDE A UN NÚMERO DE D.N.I.

- Para la búsqueda de la letra de un DNI necesitamos tener dispuesto el abecedario de la siguiente manera:
- 'T','R','W','A','G','M','Y','F','P','D','X','B','N','J','Z','S','Q','V','H','L','C','K','E'
- Por ejemplo, si queremos calcular la letra que le correspondería al número de DNI 74568099:
  Dividimos el número entre 23: 74568099/23=3242091’2608…
  Nos quedamos con el resto obtenido que es 6, por lo que al número de ese DNI le corresponde la letra Y.

RESTO:	0 1 2 3 4 5 6 7 8 9 10 11
LETRA :	T R W A G M Y F P D X B

RESTO:	12 13 14 15 16 17 18 19 20 21 22
LETRA:	N J Z S Q V H L C K E