Php2-4

Matrices

Matrices de una dimensión

Una matriz es un tipo de variable que puede almacenar varios valores a la vez, para poder organizar los datos y acceder a ellos de formar más fácil.

En las matrices de una dimensión (que se suelen llamar vectores) cada elemento se identifica por un índice que se escribe entre corchetes.

Matrices de una dimensión

- **Hay dos tipos de matrices en función del tipo de índices:**
  - **Matrices de índices numéricos**
  - **Matrices asociativas**
- **La forma de referenciar al elemento en ambos casos es la misma:**
  - 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

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.
- 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 ya que $tmp contiene ahora la salida de print_r,. Se utiliza cuando se necesita 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";
?>
```

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

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

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

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.
```

Funciones De Arrays

- Según la **documentación de PHP**, existen un total de 79 funciones de arrays. En este apartado se recogen algunas de las más importantes divididas por secciones (no se incluyen las ya utilizadas para la iteración de arrays).

Crear una matriz

- **Declaración explícita:**
- 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..

Crear una matriz

- 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
)
```

Añadir elementos a una matriz

- **Declaración implícita:**
- 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
)
```

Añadir elementos a una matriz

- 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
)
```

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
```

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.
```

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
)
```

Buscar un valor en una matriz

- La **función booleana in_array($valor, $matriz [,$tipo])** devuelve true si el valor se encuentra en la matriz. Si el argumento booleano $tipo es true, in_array() comprueba además que los tipos coincidan.

```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 en una matriz

- **La función array_search($valor,$matriz[,$tipo])** busca el valor en la matriz y, si lo encuentra, devuelve el índice correspondiente, pero si hay varios valores coincidente sólo devuelve el primero que encuentra.
- **La función array_key($matriz,$valor[,$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
)
```