Un array guarda múltiples valores en la misma variable.

Arrays numéricos

- La clave del array es un índice numérico
- hay 3 maneras de rellenar el array

en cadena

```
$cars=array("Saab","Volvo","BMW","Toyota");
```

por índice

```
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
```

uno tras otro

```
$cars[]="Saab";
$cars[]="Volvo";
$cars[]="BMW";
$cars[]="Toyota";
```

- Para consultar un valor

```
echo $cars[0];
```

Otro ejemplo:

```
A | B | C | D | E | F
$array[0] = A
```

- Conocer si un array no está vacío: `if (!empty($array))`

Arrays Multidimensionales

- Cada elemento del array principal puede ser un array

Ejemplo

```
A | B | C | D | E | F
G | H | I | J | K | L
M | N | O | P | Q | R
S | T | U | V | W | X

$array[0][0] = A
$array[0] = array que contiene los valores A | B | C | D | E | F
$array[1] = array que contiene los valores G | H | I | J | K | L
$array[0][1] = B
```

- es decir $array[fila][columna]

Ejemplo de rellenar una tabla de multiplicar como matriz usando **for**

```
$columnas = 7;
$filas = 3;
for($i=0; $i < $columnas; $i++)
{
    for($j=0; $j < $filas; $j++)
    {
        $matriz[$i][$j] = $i * $j;
    }
}
```

Arrays asociativos

- Las llaves son strings
- Para rellenar el array

En cadena

```
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
```

Por índice

```
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";
```

- Para consultar un valor

```
echo "Peter is " . $ages['Peter'] . " years old.";
```

- Ojo: no se puede hacer $ages[0]

Array asociativo multidimensional

Ejemplo

```
$usuarios=array(
array('nombre'=>'nombre1',
      'apellido'=>'apellido1',
      'edad'=>22,
      'email'=>'nombre1@lclis.com',
      'mascotas'=>array('gato','perro','araña')
      ),
array('nombre'=>'nombre2',
      'apellido'=>'apellido2',
      'edad'=>25,
      'email'=>'nombre2@lclis.com',
      'mascotas'=>array('pez','vaca','conejo')
      )
);
```

Para consultar un valor

```
echo $usuarios[1]['mascotas'][2]; // dara conejo
```

Foreach

Permite iterar sobre arrays sin saber el inicio si el final, lo recorre. No conozco cuantos elementos hay en el array. También servira sobre objetos.

- Dos maneras de decirlo
  - `foreach (expresion_array as $value) sentencia;`
  - `foreach (expresion_array as $key=>$value) sentencia;`
- En **$key** pone la llave
  En **$value** pone el valor

Ejemplo

- para cada uno de los elementos del array `$usuarios` hay un array llamado `$key` y un array llamado `$values`
  en el ejemplo anterior key=0,1 values=array('nombre...

```
foreach($usuarios as $key=>$usuario)
{
    echo"<pre>";
    print_r($usuario);
    echo"</pre>";
}
```

array_keys

- funcion que devuelve un array con las llaves (las cabeceras)
  `$keys_arr=array_keys($usuarios[0]);`

Recorrer un array con While

Con while se puede hacer lo mismo que con foreach, tal como vemos en este ejemplo

```
<?php
$arr = array("one", "two", "three");

reset($arr);
while (list(, $value) = each($arr)) {
    echo "Value: $value<br />\n";
}
foreach ($arr as $value) {
    echo "Value: $value<br />\n";
}

reset($arr);
while (list($key, $value) = each($arr)) {
    echo "Key: $key; Value: $value<br />\n";
}
foreach ($arr as $key => $value) {
    echo "Key: $key; Value: $value<br />\n";
}

?>
```

- **list($var1,$var2..) = $array**
  - coloca en cada variable un elemento del array

```
$info = array('coffee', 'brown', 'caffeine');
list($drink, $color, $power) = $info;
echo "$drink is $color and $power makes it special.\n";
```

- **each($array)**
  - devuelve la clave y el valor de un array (en un array de 4 elementos), y avanza una posición

```
$fruit = array('a' => 'apple', 'b' => 'banana', 'c' => 'cranberry');
while (list($key, $val) = each($fruit)) {
    echo "$key => $val\n";
}
```

Ver ejemplo de PHP incrustado

PHP incrustado en HTML

Se suele **incrustar código PHP en una vista HTML** para rellenar una tabla a partir de una matriz multidimensional.

Imaginemos que tenemos un array de usuarios $usuarios así:

```
Array
(
    [0] => Array
        (
            [idusuarios] => 1
            [name] => pepe
            [email] => pepe@pepe.com
            [password] => pepe
            [fecha] => 2010-10-01 00:00:00
            [estado] => 1
        )

    [1] => Array
        (
            [idusuarios] => 2
            [name] => juan
            [email] => juan@juan.com
            [password] => juan
            [fecha] => 2010-09-01 00:00:00
            [estado] => 0
        )

)
```

Podemos tener una vista donde definimos la tabla, y para cada fila le incrustamos un código PHP que recorre el array de

usuarios $usuarios, mostrándonos cada fila $row:

```
<table border=1>
    <tr>
        <th>id</th>
        <th>name</th>
        <th>email</th>
        <th>password</th>
        <th>fecha</th>
        <th>status</th>
        <th>options</th>
    </tr>

<?php while (list(, $row) = each($usuarios)): ?>
    <tr>
        <td><?=$row["idusuarios"]?></td>
        <td><?=$row["name"]?></td>
        <td><?=$row["email"]?></td>
        <td><?=$row["password"]?></td>
        <td><?=$row["fecha"]?></td>
        <td><?=$row["estado"]?></td>
        <td><a href=#>update</a> <a href=#>delete</a></td>
    </tr>
<?php endwhile;?>
</table>
```

Funciones con arrays

reset($array)

- pone el puntero al inicio

array_count_values()

array_search()

count()

sort(), rsort()

ksort(), krsort()

- ordena por el valor de la llave