# Estructuras de control: bucle foreach



---

## El bucle foreach

El bucle foreach es un tipo especial de bucle que permite recorrir estructuras que contienen varios elementos (como matrices, recursos u objetos) sin necesidad de preocuparse por el número de elementos.

Volver al principio de la página

---

## Problemas al recorrer matrices con el bucle for

En otros lenguajes de programación en los que los índices de las matrices son números naturales correlativos, las matrices se pueden recorrer fácilmente con bucles for. Pero como las matrices de PHP son 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.

- En el ejemplo siguiente, el bucle llega más allá de los límites de la matriz, generando mensajes de error:

**Paso 1**

```
<?php
$matriz = array("a", "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 3; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

Se ejecuta la primera instrucción del programa.

En este caso, se crea la matriz $matriz.

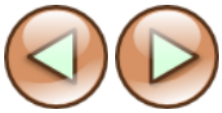
**Nota**: Este problema se podría evitar fácilmente, puesto que los índices son correlativos. La solución sería en este caso expresar la condición de continuación en función del tamaño de la matriz, utilizando $i < count($matriz) en vez de $i <= 3.

## Paso 2

```php
<?php
$matriz = array("a", "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 3; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
```

A continuación se abre la etiqueta *<pre>*.

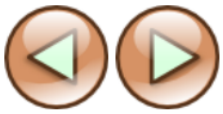


## Paso 3

```php
<?php
$matriz = array("a", "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 3; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
)
```

A continuación se imprimen los valores de la matriz $matriz.

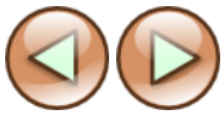


## Paso 4

```php
<?php
$matriz = array("a", "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 3; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
)
</pre>
```

A continuación se cierra la etiqueta *</pre>*.

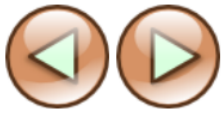


## Paso 5

```php
<?php
$matriz = array("a", "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 3; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
)
</pre>
```

A continuación se ejecuta el bucle. El primer paso es dar el valor inicial a la variable de control.
En este caso, la variable de control es $i y toma el valor 0.



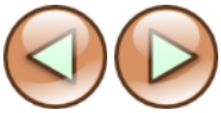
## Paso 6

```php
<?php
$matriz = array("a", "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 3; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
)
</pre>
```

A continuación se comprueba que la condición de continuación se cumple.

En este caso, $i es inferior o igual a 3 (vale 0), así que se pasa a ejecutar las instrucciones del bloque de sentencias.



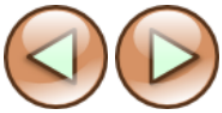
## Paso 7

```php
<?php
$matriz = array("a", "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 3; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
)
</pre>
<p>a</p>
```

A continuación se ejecutan las instrucciones del bloque.

En este caso, se imprime el texto a ya que $i tiene el valor 0 y $matriz[0] tiene el valor "a".



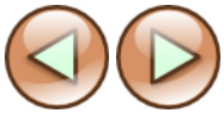
## Paso 8

```php
<?php
$matriz = array("a", "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 3; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
)
</pre>
<p>a</p>
```

A continuación se ejecuta de nuevo la instrucción de paso.

En este caso, la variable $i aumenta una unidad, por lo que ahora vale 1.



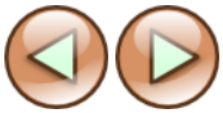
## Paso 9

```php
<?php
$matriz = array("a", "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 3; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
)
</pre>
<p>a</p>
<p>bb</p>
```

A continuación se comprueba de nuevo que la condición de continuación se cumple.

En este caso, $i es inferior o igual a 3 (vale 1), así que se pasa a ejecutar las instrucciones del bloque de sentencias.

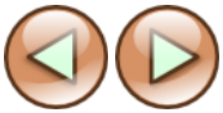


## Paso 10

```php
<?php
$matriz = array("a", "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 3; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
)
</pre>
<p>a</p>
<p>bb</p>
```

A continuación se ejecutan de nuevo las instrucciones del bloque.
En este caso, se imprime el texto bb ya que $i tiene el valor 1 y $matriz[1] tiene el valor "bb".

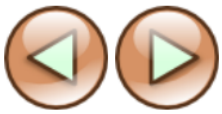


## Paso 11

```php
<?php
$matriz = array("a", "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 3; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
)
</pre>
<p>a</p>
<p>bb</p>
```

A continuación se ejecuta de nuevo la instrucción de paso.
En este caso, la variable $i aumenta una unidad, por lo que ahora vale 2.



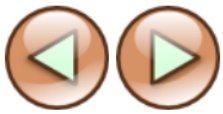
## Paso 12

```php
<?php
$matriz = array("a", "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 3; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
)
</pre>
<p>a</p>
<p>bb</p>
```

A continuación se comprueba de nuevo que la condición de continuación se cumple.

En este caso, $i es inferior o igual a 3 (vale 2), así que se pasa a ejecutar las instrucciones del bloque de sentencias.

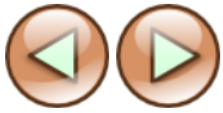


## Paso 13

```php
<?php
$matriz = array("a", "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 3; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
)
</pre>
<p>a</p>
<p>bb</p>
<br />
<b>Notice</b>:  Undefined offset: 2 in <b>ejemplo
<p></p>
```

Esta instrucción genera un aviso (puesto que intenta imprimir el valor $matriz[2], que no existe) y escribe un párrafo vacío (puesto que no hay valor de $matriz[2].



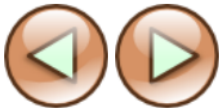
## Paso 14

```php
<?php
$matriz = array("a", "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 3; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
)
</pre>
<p>a</p>
<p>bb</p>
<br />
<b>Notice</b>:  Undefined offset: 2 in <b>ejemplo.
<p></p>
```

A continuación se ejecuta de nuevo la instrucción de paso.

En este caso, la variable $i aumenta una unidad, por lo que ahora vale 3.



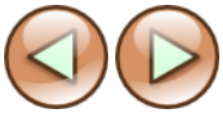
## Paso 15

```php
<?php
$matriz = array("a", "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 3; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
)
</pre>
<p>a</p>
<p>bb</p>
<br />
<b>Notice</b>:  Undefined offset: 2 in <b>ejemplo.
<p></p>
```

A continuación se comprueba de nuevo que la condición de continuación se cumple.

En este caso, $i es inferior o igual a 3 (vale 3), así que se pasa a ejecutar las instrucciones del bloque de sentencias.

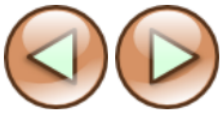


## Paso 16

```php
<?php
$matriz = array("a", "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 3; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
)
</pre>
<p>a</p>
<p>bb</p>
<br />
<b>Notice</b>:  Undefined offset: 2 in <b>ejemplo.
<p></p>
<br />
<b>Notice</b>:  Undefined offset: 3 in <b>ejemplo
<p></p>
```

Esta instrucción genera un aviso (puesto que intenta imprimir el valor $matriz[3], que no existe) y escribe un párrafo vacío (puesto que no hay valor de $matriz[3].



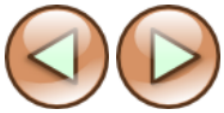
## Paso 17

```php
<?php
$matriz = array("a", "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 3; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
)
</pre>
<p>a</p>
<p>bb</p>
<br />
<b>Notice</b>:  Undefined offset: 2 in <b>ejemplo.
<p></p>
<br />
<b>Notice</b>:  Undefined offset: 3 in <b>ejemplo.
<p></p>
```

A continuación se ejecuta de nuevo la instrucción de paso.

En este caso, la variable $i aumenta una unidad, por lo que ahora vale 4.



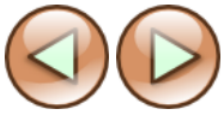
## Paso 17

```php
<?php
$matriz = array("a", "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 3; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
)
</pre>
<p>a</p>
<p>bb</p>
<br />
<b>Notice</b>:  Undefined offset: 2 in <b>ejemplo.
<p></p>
<br />
<b>Notice</b>:  Undefined offset: 3 in <b>ejemplo.
<p></p>
```

A continuación se ejecuta de nuevo la instrucción de paso.

En este caso, la variable $i aumenta una unidad, por lo que ahora vale 4.



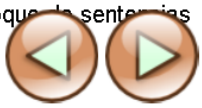
## Paso 18

```php
<?php
$matriz = array("a", "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 3; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
)
</pre>
<p>a</p>
<p>bb</p>
<br />
<b>Notice</b>:  Undefined offset: 2 in <b>ejemplo.php</b> o
<p></p>
<br />
<b>Notice</b>:  Undefined offset: 3 in <b>ejemplo.php</b> o
<p></p>
```

A continuación se comprueba de nuevo que la condición de continuación se cumple.

En este caso, $i ya no es inferior o igual a 3 (vale 4), así que no se ejecutan las instrucciones del bloque de sentencias y el bucle se termina.



**Paso 19**

```php
<?php
$matriz = array("a", "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 3; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
)
</pre>
<p>a</p>
<p>bb</p>
<br />
<b>Notice</b>:  Undefined offset: 2 in <b>ejemplo.php</b> o
<p></p>
<br />
<b>Notice</b>:  Undefined offset: 3 in <b>ejemplo.php</b> o
<p></p>
<p>Final</p>
```

Una vez terminado el bucle, se ejecuta la instrucción que sigue al bucle.
En este caso, imprime el párrafo final.

**Nota**: Este problema se podría evitar fácilmente, puesto que los índices son correlativos. La solución sería en este caso expresar la condición de continuación en función del tamaño de la matriz, utilizando $i < count($matriz) en vez de $i <= 3.

- En el ejemplo siguiente, a la matriz le faltan valores intermedios, por lo que también se generarían mensajes de error al recorrerla:

**Paso 1**

```php
<?php
$matriz = array(0 => "a", 2 => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 2; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

Se ejecuta la primera instrucción del programa.
En este caso, se crea la matriz $matriz.

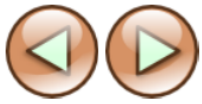
**Nota**: los avisos de error de este ejemplo se podría evitar incluyendo una comprobación de existencia del elemento antes de imprimirlo (if isset($matriz[$i]) ...), aunque tendríamos el problema de no saber a priori hasta qué valor debemos ejecutar el bucle para recorrer todos los valores.

**Paso 2**

```php
<?php
$matriz = array(0 => "a", 2 => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 2; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
```

A continuación se abre la etiqueta *<pre>*.

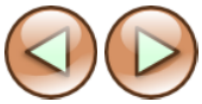


**Paso 3**

```php
<?php
$matriz = array(0 => "a", 2 => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 2; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [2] => bb
)
```

A continuación se imprimen los valores de la matriz $matriz.



**Paso 4**

```php
<?php
$matriz = array(0 => "a", 2 => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 2; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [2] => bb
)
</pre>
```

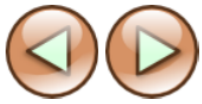
A continuación se cierra la etiqueta *</pre>*.

**Paso 5**

```php
<?php
$matriz = array(0 => "a", 2 => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 2; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [2] => bb
)
</pre>
```

A continuación se ejecuta el bucle. El primer paso es dar el valor inicial a la variable de control.
En este caso, la variable de control es $i y toma el valor 0.

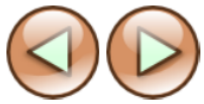


**Paso 6**

```php
<?php
$matriz = array(0 => "a", 2 => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 2; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [2] => bb
)
</pre>
```

A continuación se comprueba que la condición de continuación se cumple.
En este caso, $i es inferior o igual a 2 (vale 0), así que se pasa a ejecutar las instrucciones del bloque de sentencias.

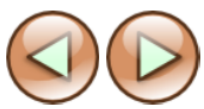


**Paso 7**

```php
<?php
$matriz = array(0 => "a", 2 => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 2; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [2] => bb
)
</pre>
<p>a</p>
```

A continuación se ejecutan las instrucciones del bloque.
En este caso, se imprime el texto a ya que $i tiene el valor 0 y $matriz[0] tiene el valor "a".

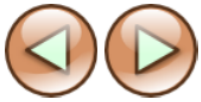


**Paso 8**

```php
<?php
$matriz = array(0 => "a", 2 => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 2; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [2] => bb
)
</pre>
<p>a</p>
```

A continuación se ejecuta de nuevo la instrucción de paso.
En este caso, la variable $i aumenta una unidad, por lo que ahora vale 1.

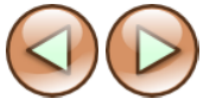


**Paso 9**

```php
<?php
$matriz = array(0 => "a", 2 => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 2; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [2] => bb
)
</pre>
<p>a</p>
```

A continuación se comprueba de nuevo que la condición de continuación se cumple.
En este caso, $i es inferior o igual a 2 (vale 1), así que se pasa a ejecutar las instrucciones del bloque de sentencias.

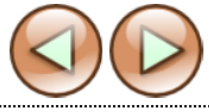


**Paso 10**

```php
<?php
$matriz = array(0 => "a", 2 => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 2; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [2] => bb
)
</pre>
<p>a</p>
<br />
<b>Notice</b>:  Undefined offset: 1 in <b>ejemplo.php</
<p></p>
```

A continuación se ejecutan de nuevo las instrucciones del bloque.
Esta instrucción genera un aviso (puesto que intenta imprimir el valor $matriz[1], que no existe) y escribe un párrafo vacío (puesto que no hay valor de $matriz[1].

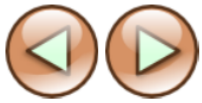


**Paso 11**

```
<?php
$matriz = array(0 => "a", 2 => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 2; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [2] => bb
)
</pre>
<p>a</p>
<br />
<b>Notice</b>:  Undefined offset: 1 in <b>ejemplo.php</
<p></p>
```

A continuación se ejecuta de nuevo la instrucción de paso.
En este caso, la variable $i aumenta una unidad, por lo que ahora vale 2.

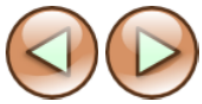


**Paso 12**

```
<?php
$matriz = array(0 => "a", 2 => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 2; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [2] => bb
)
</pre>
<p>a</p>
<br />
<b>Notice</b>:  Undefined offset: 1 in <b>ejemplo.php</
<p></p>
```

A continuación se comprueba de nuevo que la condición de continuación se cumple.
En este caso, $i es inferior o igual a 2 (vale 2), así que se pasa a ejecutar las instrucciones del bloque de sentencias.

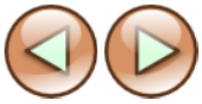


**Paso 13**

```
<?php
$matriz = array(0 => "a", 2 => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 2; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [2] => bb
)
</pre>
<p>a</p>
<br />
<b>Notice</b>:  Undefined offset: 1 in <b>ejemplo.php</
<p></p>
<p>bb</p>
```

En este caso, se imprime el texto bb ya que $i tiene el valor 2 y $matriz[2] tiene el valor "bb".

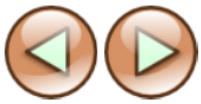


**Paso 14**

```
<?php
$matriz = array(0 => "a", 2 => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 2; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [2] => bb
)
</pre>
<p>a</p>
<br />
<b>Notice</b>:  Undefined offset: 1 in <b>ejemplo.php</
<p></p>
<p>bb</p>
```

A continuación se ejecuta de nuevo la instrucción de paso.
En este caso, la variable $i aumenta una unidad, por lo que ahora vale 3.

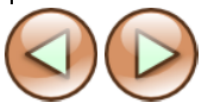


**Paso 15**

```
<?php
$matriz = array(0 => "a", 2 => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 2; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [2] => bb
)
</pre>
<p>a</p>
<br />
<b>Notice</b>:  Undefined offset: 1 in <b>ejemplo.php</
<p></p>
<p>bb</p>
```

A continuación se comprueba de nuevo que la condición de continuación se cumple.
En este caso, $i ya no es inferior o igual a 2 (vale 3), así que no se ejecutan las instrucciones del bloque de sentencias y el bucle se termina.



**Paso 16**

```
<?php
$matriz = array(0 => "a", 2 => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 2; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [2] => bb
)
</pre>
<p>a</p>
<br />
<b>Notice</b>:  Undefined offset: 1 in <b>ejemplo.php</
<p></p>
<p>bb</p>
<p>Final</p>
```

Una vez terminado el bucle, se ejecuta la instrucción que sigue al bucle.
En este caso, imprime el párrafo final.

**Nota**: los avisos de error de este ejemplo se podría evitar incluyendo una comprobación de existencia del elemento antes de imprimirlo (if isset($matriz[$i]) ...), aunque tendríamos el problema de no saber a priori hasta qué valor debemos ejecutar el bucle para recorrer todos los valores.

- Finalmente, en el ejemplo siguiente, la matriz no tiene índices numéricos, por lo que sólo se generarían mensajes de error al recorrerla:

**Paso 1**

```
<?php
$matriz = array("uno" => "a", "dos" => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 1; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

Se ejecuta la primera instrucción del programa.
En este caso, se crea la matriz $matriz.

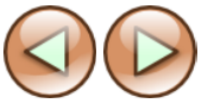
**Nota**: Esta situación no se podría evitar pues los índices de la matriz no son valores numéricos.

**Paso 2**

```
<?php
$matriz = array("uno" => "a", "dos" => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 1; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
```

A continuación se abre la etiqueta *<pre>*.

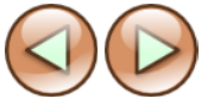


## Paso 3

```php
<?php
$matriz = array("uno" => "a", "dos" => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 1; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [uno] => a
    [dos] => bb
)
```

A continuación se imprimen los valores de la matriz $matriz.

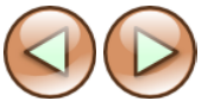


## Paso 4

```php
<?php
$matriz = array("uno" => "a", "dos" => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 1; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [uno] => a
    [dos] => bb
)
</pre>
```

A continuación se cierra la etiqueta *</pre>*.

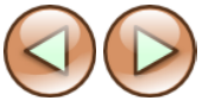


## Paso 5

```php
<?php
$matriz = array("uno" => "a", "dos" => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 1; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [uno] => a
    [dos] => bb
)
</pre>
```

A continuación se ejecuta el bucle. El primer paso es dar el valor inicial a la variable de control.
En este caso, la variable de control es $i y toma el valor 0.

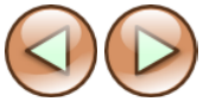


**Paso 6**

```php
<?php
$matriz = array("uno" => "a", "dos" => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 1; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [uno] => a
    [dos] => bb
)
</pre>
```

A continuación se comprueba que la condición de continuación se cumple.
En este caso, $i es inferior o igual a 1 (vale 0), así que se pasa a ejecutar las instrucciones del bloque de sentencias.

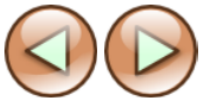


**Paso 7**

```php
<?php
$matriz = array("uno" => "a", "dos" => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 1; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [uno] => a
    [dos] => bb
)
</pre>
<br />
<b>Notice</b>:  Undefined offset: 0 in <b>ejemp
<p></p>
```

A continuación se ejecutan las instrucciones del bloque.
Esta instrucción genera un aviso (puesto que intenta imprimir el valor $matriz[0], que no existe) y escribe un párrafo vacío (puesto que no hay valor de $matriz[0].

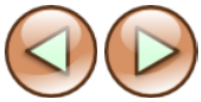


**Paso 9**

```php
<?php
$matriz = array("uno" => "a", "dos" => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 1; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [uno] => a
    [dos] => bb
)
</pre>
<br />
<b>Notice</b>:  Undefined offset: 0 in <b>ejemp
<p></p>
```

A continuación se comprueba de nuevo que la condición de continuación se cumple.
En este caso, $i es inferior o igual a 1 (vale 1), así que se pasa a ejecutar las instrucciones del bloque de sentencias.



**Paso 10**

```php
<?php
$matriz = array("uno" => "a", "dos" => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 1; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [uno] => a
    [dos] => bb
)
</pre>
<br />
<b>Notice</b>:  Undefined offset: 0 in <b>ejemp
<p></p>
<br />
<b>Notice</b>:  Undefined offset: 1 in <b>ejemp
<p></p>
```

A continuación se ejecutan de nuevo las instrucciones del bloque.

Esta instrucción genera otro aviso (puesto que intenta imprimir el valor $matriz[1], que no existe) y escribe un párrafo vacío (puesto que no hay valor de $matriz[1].

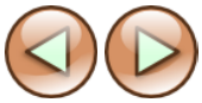
**Paso 11**

```php
<?php
$matriz = array("uno" => "a", "dos" => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 1; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [uno] => a
    [dos] => bb
)
</pre>
<br />
<b>Notice</b>:  Undefined offset: 0 in <b>ejemp
<p></p>
<br />
<b>Notice</b>:  Undefined offset: 1 in <b>ejemp
<p></p>
```

A continuación se ejecuta de nuevo la instrucción de paso.

En este caso, la variable $i aumenta una unidad, por lo que ahora vale 2.



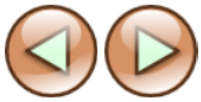
**Paso 12**

```
<?php
$matriz = array("uno" => "a", "dos" => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 1; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [uno] => a
    [dos] => bb
)
</pre>
<br />
<b>Notice</b>:  Undefined offset: 0 in <b>ejemp
<p></p>
<br />
<b>Notice</b>:  Undefined offset: 1 in <b>ejemp
<p></p>
```

A continuación se comprueba de nuevo que la condición de continuación se cumple.

En este caso, $i ya no es inferior o igual a 1 (vale 2), así que no se ejecutan las instrucciones del bloque de sentencias y el bucle se termina.



**Paso 13**

```
<?php
$matriz = array("uno" => "a", "dos" => "bb");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
for ($i = 0; $i <= 1; $i++) {
    print "<p>$matriz[$i]</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [uno] => a
    [dos] => bb
)
</pre>
<br />
<b>Notice</b>:  Undefined offset: 0 in <b>ejemp
<p></p>
<br />
<b>Notice</b>:  Undefined offset: 1 in <b>ejemp
<p></p>
<p>Final</p>
```

Una vez terminado el bucle, se ejecuta la instrucción que sigue al bucle.

En este caso, imprime el párrafo final.

### Sintaxis del bucle foreach

La sintaxis del bucle foreach puede ser una de las siguientes:

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

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

El bucle se ejecuta tantas veces como elementos tiene la matriz. En cada iteración, las variables $indice y $valor van tomando los valores de los índices y de la matriz para ese índice.

Volver al principio de la página

---

### Ejemplos de bucles

**Matrices**

- En el ejemplo siguiente, el bucle recorre una matriz imprimiendo sus valores

**Paso 1**

```php
<?php
$matriz = array("a", "bb", "ccc");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $valor) {
    print "<p>$valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

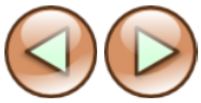
Se ejecuta la primera instrucción del programa.
En este caso, se crea la matriz $matriz.

**Paso 2**

```php
<?php
$matriz = array("a", "bb", "ccc");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $valor) {
    print "<p>$valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
```

A continuación se abre la etiqueta *<pre>*.

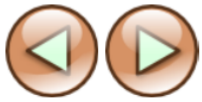


## Paso 3

```php
<?php
$matriz = array("a", "bb", "ccc");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $valor) {
    print "<p>$valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
    [2] => ccc
)
```

A continuación se imprimen los valores de la matriz $matriz.

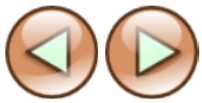


## Paso 4

```php
<?php
$matriz = array("a", "bb", "ccc");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $valor) {
    print "<p>$valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
    [2] => ccc
)
</pre>
```

A continuación se cierra la etiqueta *</pre>*.

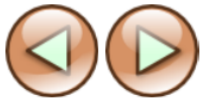


**Paso 5**

```php
<?php
$matriz = array("a", "bb", "ccc");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $valor) {
    print "<p>$valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
    [2] => ccc
)
</pre>
```

A continuación se ejecuta el bucle. La variable $valor va a ir tomando sucesivamente los valores de la matriz.
En este caso, la variable $valor toma el valor "a".

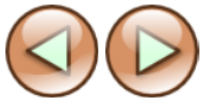


**Paso 6**

```php
<?php
$matriz = array("a", "bb", "ccc");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $valor) {
    print "<p>$valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
    [2] => ccc
)
</pre>
<p>a</p>
```

A continuación se ejecutan las instrucciones del bloque.
En este caso, se imprime el texto a ya que $valor tiene el valor "a".

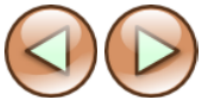


**Paso 7**

```php
<?php
$matriz = array("a", "bb", "ccc");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $valor) {
    print "<p>$valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
    [2] => ccc
)
</pre>
<p>a</p>
```

A continuación $valor pasa a tomar el valor siguiente de la matriz.
En este caso, la variable $valor toma el valor "bb".

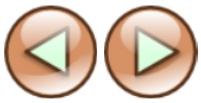


**Paso 8**

```php
<?php
$matriz = array("a", "bb", "ccc");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $valor) {
    print "<p>$valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
    [2] => ccc
)
</pre>
<p>a</p>
<p>bb</p>
```

A continuación se ejecutan de nuevo las instrucciones del bloque.
En este caso, se imprime el texto bb ya que $valor tiene el valor "bb".

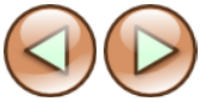


**Paso 9**

```php
<?php
$matriz = array("a", "bb", "ccc");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $valor) {
    print "<p>$valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
    [2] => ccc
)
</pre>
<p>a</p>
<p>bb</p>
```

A continuación $valor pasa a tomar el valor siguiente de la matriz.
En este caso, la variable $valor toma el valor "ccc".

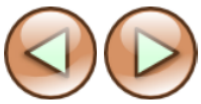


**Paso 10**

```php
<?php
$matriz = array("a", "bb", "ccc");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $valor) {
    print "<p>$valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
    [2] => ccc
)
</pre>
<p>a</p>
<p>bb</p>
<p>ccc</p>
```

A continuación se ejecutan de nuevo las instrucciones del bloque.
En este caso, se imprime el texto ccc ya que $valor tiene el valor "ccc".



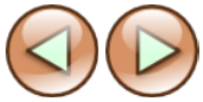
**Paso 11**

```php
<?php
$matriz = array("a", "bb", "ccc");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $valor) {
    print "<p>$valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
    [2] => ccc
)
</pre>
<p>a</p>
<p>bb</p>
<p>ccc</p>
```

A continuación $valor pasa a tomar el valor siguiente de la matriz.

Pero como en la matriz no quedan más valores, no se ejecutan las instrucciones del bloque de sentencias y el bucle se termina.



**Paso 12**

```php
<?php
$matriz = array("a", "bb", "ccc");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $valor) {
    print "<p>$valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [0] => a
    [1] => bb
    [2] => ccc
)
</pre>
<p>a</p>
<p>bb</p>
<p>ccc</p>
<p>Final</p>
```

Una vez terminado el bucle, se ejecuta la instrucción que sigue al bucle.

En este caso, imprime el párrafo final.

- En el ejemplo siguiente, el bucle recorre una matriz imprimiendo sus índices y valores

**Paso 1**

```php
<?php
$matriz = array("uno" => "a", 2 => "bb", "tres" => "ccc");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $indice => $valor) {
    print "<p>$indice - $valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

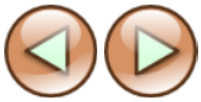
Se ejecuta la primera instrucción del programa.
En este caso, se crea la matriz $matriz.

**Paso 2**

```php
<?php
$matriz = array("uno" => "a", 2 => "bb", "tres" => "ccc");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $indice => $valor) {
    print "<p>$indice - $valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
```

A continuación se abre la etiqueta *<pre>*.

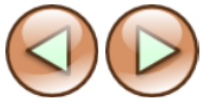


**Paso 3**

```php
<?php
$matriz = array("uno" => "a", 2 => "bb", "tres" => "ccc");
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $indice => $valor) {
    print "<p>$indice - $valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [uno] => a
    [2] => bb
    [tres] => ccc
)
```

A continuación se imprimen los valores de la matriz $matriz.

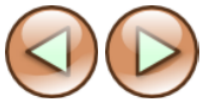


**Paso 4**

```php
<?php
$matriz = array("uno" => "a", 2 => "bb", "tres" => "ccc")
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $indice => $valor) {
    print "<p>$indice - $valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [uno] => a
    [2] => bb
    [tres] => ccc
)
</pre>
```

A continuación se cierra la etiqueta *</pre>*.



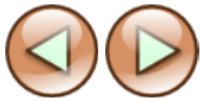
**Paso 5**

```php
<?php
$matriz = array("uno" => "a", 2 => "bb", "tres" => "ccc")
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $indice => $valor) {
    print "<p>$indice - $valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [uno] => a
    [2] => bb
    [tres] => ccc
)
</pre>
```

A continuación se ejecuta el bucle. Las variables $indice y $valor van a ir tomando sucesivamente los valores del índice y del elemento de la matriz, respectivamente.

En este caso, la variable $indice toma el valor "uno" y $valor toma el valor "a".



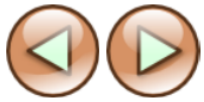
**Paso 6**

```php
<?php
$matriz = array("uno" => "a", 2 => "bb", "tres" => "ccc")
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $indice => $valor) {
    print "<p>$indice - $valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [uno] => a
    [2] => bb
    [tres] => ccc
)
</pre>
<p>uno - a</p>
```

A continuación se ejecutan las instrucciones del bloque.

En este caso, se imprime el texto uno - a ya que $indice tiene el valor "uno" y $valor tiene el valor "a".

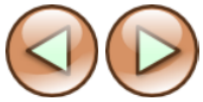


**Paso 7**

```php
<?php
$matriz = array("uno" => "a", 2 => "bb", "tres" => "ccc")
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $indice => $valor) {
    print "<p>$indice - $valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [uno] => a
    [2] => bb
    [tres] => ccc
)
</pre>
<p>uno - a</p>
```

A continuación $indice y $valor pasan a tomar los valores del elemento siguiente de la matriz.
En este caso, la variable $indice toma el valor "2" y $valor toma el valor "bb".

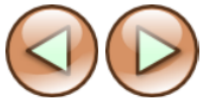


**Paso 8**

```php
<?php
$matriz = array("uno" => "a", 2 => "bb", "tres" => "ccc")
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $indice => $valor) {
    print "<p>$indice - $valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [uno] => a
    [2] => bb
    [tres] => ccc
)
</pre>
<p>uno - a</p>
<p>2 - bb</p>
```

A continuación se ejecutan de nuevo las instrucciones del bloque.
En este caso, se imprime el texto 2 - bb ya que $indice tiene el valor "2" y $valor tiene el valor "bb".

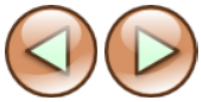


## Paso 9

```php
<?php
$matriz = array("uno" => "a", 2 => "bb", "tres" => "ccc")
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $indice => $valor) {
    print "<p>$indice - $valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [uno] => a
    [2] => bb
    [tres] => ccc
)
</pre>
<p>uno - a</p>
<p>2 - bb</p>
```

A continuación $indice y $valor pasan a tomar los valores del elemento siguiente de la matriz.
En este caso, la variable $indice toma el valor "tres" y $valor toma el valor "ccc".

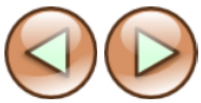


## Paso 10

```php
<?php
$matriz = array("uno" => "a", 2 => "bb", "tres" => "ccc")
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $indice => $valor) {
    print "<p>$indice - $valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [uno] => a
    [2] => bb
    [tres] => ccc
)
</pre>
<p>uno - a</p>
<p>2 - bb</p>
<p>tres - ccc</p>
```

A continuación se ejecutan de nuevo las instrucciones del bloque.
En este caso, se imprime el texto tres - ccc ya que $indice tiene el valor "tres" y $valor tiene el valor "ccc".



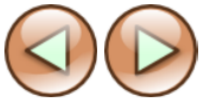
**Paso 11**

```php
<?php
$matriz = array("uno" => "a", 2 => "bb", "tres" => "ccc")
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $indice => $valor) {
    print "<p>$indice - $valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [uno] => a
    [2] => bb
    [tres] => ccc
)
</pre>
<p>uno - a</p>
<p>2 - bb</p>
<p>tres - ccc</p>
```

A continuación $indice y $valor pasan a tomar los valores del elemento siguiente de la matriz.

Pero como en la matriz no quedan más valores, no se ejecutan las instrucciones del bloque de sentencias y el bucle se termina.



**Paso 12**

```php
<?php
$matriz = array("uno" => "a", 2 => "bb", "tres" => "ccc")
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $indice => $valor) {
    print "<p>$indice - $valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
    [uno] => a
    [2] => bb
    [tres] => ccc
)
</pre>
<p>uno - a</p>
<p>2 - bb</p>
<p>tres - ccc</p>
<p>Final</p>
```

Una vez terminado el bucle, se ejecuta la instrucción que sigue al bucle.

En este caso, imprime el párrafo final.

- Si la matriz es una matriz vacía, el bucle simplemente no se ejecuta, como muestra el siguiente ejemplo:

**Paso 1**

```php
<?php
$matriz = array();
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $indice => $valor) {
    print "<p>$indice - $valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

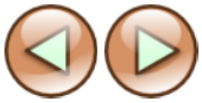
Se ejecuta la primera instrucción del programa.
En este caso, se crea la matriz $matriz.

**Paso 2**

```php
<?php
$matriz = array();
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $indice => $valor) {
    print "<p>$indice - $valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
```

A continuación se abre la etiqueta *<pre>*.

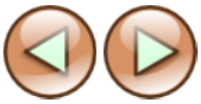


**Paso 3**

```php
<?php
$matriz = array();
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $indice => $valor) {
    print "<p>$indice - $valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
)
```

A continuación se imprimen los valores de la matriz $matriz.

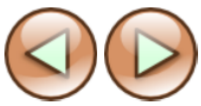


**Paso 4**

```php
<?php
$matriz = array()
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $indice => $valor) {
    print "<p>$indice - $valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
)
</pre>
```

A continuación se cierra la etiqueta *</pre>*.

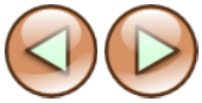


**Paso 5**

```php
<?php
$matriz = array()
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $indice => $valor) {
    print "<p>$indice - $valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
)
</pre>
```

A continuación $indice y $valor deberían pasan a tomar los valores del primer elemento de la matriz.
Pero como en la matriz no hay valores, no se ejecutan las instrucciones del bloque de sentencias y el bucle se termina sin haber ejecutado ninguna vez las instrucciones del cuerpo del bucle..



**Paso 6**

```php
<?php
$matriz = array()
print "<pre>\n";
print_r($matriz);
print "</pre>\n";
foreach ($matriz as $indice => $valor) {
    print "<p>$indice - $valor</p>\n";
}
print "<p>Final</p>\n";
?>
```

```
<pre>
Array
(
)
</pre>
<p>Final</p>
```

Una vez terminado el bucle, se ejecuta la instrucción que sigue al bucle.
En este caso, imprime el párrafo final.

# Matrices



---

## Qué es una matriz

Una matriz es un tipo de variable que puede almacenar varios valores a la vez. En las matrices de una dimensión (que a veces se llaman vectores) cada elemento se identifica por un índice que se escribe entre corchetes ($matriz[$indice]). Pero las matrices pueden tener más dimensiones (como las matrices matemáticas) y entonces los elementos se identifican por varios índices que se escriben cada uno entre corchetes ($matriz[$indice1][$indice2]...).

En otros lenguajes de programación, los índices de las matrices tienen que ser números enteros positivos y tienen que estar todos definidos, pero en PHP se dice que las matrices son asociativas porque los índices no tienen por qué ser números y cuando son números no tienen por qué ser valores correlativos.

```php
<?php
$matriz[5] = 25;
$matriz[3] = 12;

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

```
Array
(
    [5] => 25
    [3] => 12
)
```

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

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

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

    [juan] => Array
        (
            [peso] => 75
        )
)
```

## Crear una matriz

Una matriz se puede crear definiendo algún valor de la matriz

```php
<?php
$matriz[5] = 25;

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

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

o utilizando la función array($indice => $valor, ...):

```php
<?php
$matriz = array( 5 => 25);

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

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

---

Las matrices en PHP son matrices asociativas, es decir, que los índices no tienen por qué ser números enteros positivos:

```php
<?php
$matriz[5] = 25;
$matriz[-1] = "negativo";
$matriz["número 1"] = "cinco";

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

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

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

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

Las matrices en PHP pueden ser multidimensionales:

```php
<?php
$matriz[5][3] = 25;
$matriz["letras"][1] = "letra A";

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

```php
<?php
$matriz = array(5 => array(3 => 25), "letras" => array(1 => "letra A"));

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

```
Array
(
    [5] => Array
        (
            [3] => 25
        )
    [letras] => Array
        (
            [1] => letra A
        )
)
```

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

La función print_r($variable [, $devolver]) escribe la variable $variable de forma legible, incluso aunque se trate de una matriz.

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
$matriz = array("nombre" => "Pepito", "apellidos" => "Conejo");
print_r($matriz);
?>
```

```
Array ( [nombre] => Pepito [apellidos] => Conejo )
```

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
$matriz = array("nombre" => "Pepito", "apellidos" => "Conejo");
print "<pre>"; print_r($matriz); print "</pre>\n";
?>
```

```
Array
(
    [nombre] => Pepito
    [apellidos] => Conejo
)
```

Si el argumento $devolver toma el valor **true**, print_r() no escribe nada pero devuelve el texto que se escribe cuando el argumento no está o toma el valor **false**.

Si el argumento $devolver no está o toma el valor **false**, print_r() devuelve 1 (**true**).

```php
<?php
$matriz = array("nombre" => "Pepito", "apellidos" => "Conejo");
$tmp = print_r($matriz, true);
print "<p>La matriz es $tmp</p>\n";
?>
```

La matriz es Array ( [nombre] => Pepito [apellidos] => Conejo )

```php
<?php
$matriz = array("nombre" => "Pepito", "apellidos" => "Conejo");
$tmp = print_r($matriz, false);
print "<p>La matriz es $tmp</p>\n";
?>
```

Array ( [nombre] => Pepito [apellidos] => Conejo )

La matriz es 1

```php
<?php
$matriz = array("nombre" => "Pepito", "apellidos" => "Conejo");
$tmp = print_r($matriz);
print "<p>La matriz es $tmp</p>\n";
?>
```

Array ( [nombre] => Pepito [apellidos] => Conejo )

La matriz es 1

## Borrar una matriz o elementos de una matriz

La función unset() permite borrar una matriz o elementos de una matriz .

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

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

unset ($matriz[5]);

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

unset ($matriz);

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

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

Notice:  Undefined variable: matriz
```

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

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

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

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

## Contar elementos de una matriz

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

```php
<?php
$matriz[3] = 25;
$matriz[4] = 30;
$matriz[5] = 35;
$matriz["letra"] = "letra A";

$elementos = count($matriz);

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

```
La matriz tiene 4 elementos.
Array
(
    [3] => 25
    [4] => 30
    [5] => 35
    [letra] => letra A
)
```

En una matriz multidimensional, la función `count($matriz)` devolvería simplemente el número de elementos del primer índice:

```php
<?php
$matriz[5][3] = 25;
$matriz[5][4] = 30;
$matriz[5][5] = 35;
$matriz["letra"][1] = "letra A";

$elementos = count($matriz);

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

```
La matriz tiene 2 elementos.
Array
(
    [5] => Array
        (
            [3] => 25
            [4] => 30
            [5] => 35
        )
    [letra] => Array
        (
            [1] => letra A
        )
)
```

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

```php
<?php
$matriz[5][3] = 25;
$matriz[5][4] = 30;
$matriz[5][5] = 35;
$matriz["letra"][1] = "letra A";

$elementos = count($matriz, COUNT_RECURSIVE);

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

```
La matriz tiene 6 elementos.
Array
(
    [5] => Array
        (
            [3] => 25
            [4] => 30
            [5] => 35
        )
    [letra] => Array
        (
            [1] => letra A
        )
)
```

Es importante fijarse en que en este caso la función `count()` está contando también las dos matrices fila. Si quisieramos contar únicamente los elementos de una matriz bidimensional habría que restar el número de matrices fila:

```php
<?php
$matriz[5][3] = 25;
$matriz[5][4] = 30;
$matriz[5][5] = 35;
$matriz["letra"][1] = "letra A";

$elementos = count($matriz, COUNT_RECURSIVE) - count($matriz);

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

```
La matriz tiene 4 elementos.
Array
(
    [5] => Array
        (
            [3] => 25
            [4] => 30
            [5] => 35
        )
    [letra] => Array
        (
            [1] => letra A
        )
)
```

## 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
$valores = array (10, 40, 15, -1);
$maximo = max($valores);
$minimo = min($valores);

print "<pre>"; 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] => -1
)
El máximo de la matriz es 40.
El mínimo de la matriz es -1.
```

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>"; 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.
```

## Encontrar un valor en una matriz

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

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

print "<pre>"; print_r($valores); print "</pre>\n";
if (in_array(15, $valores)) {
  print "<p>15 está en la matriz \$valores.</p>\n";
}
if (!in_array(25, $valores)) {
  print "<p>25 no está en la matriz \$valores.</p>\n";
}
if (!in_array("15", $valores, true)) {
  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.
```

Volver al principio de la página

Última modificación de esta página: 22 de septiembre de 2014