
	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Sistemas de gestión empresarial				CURSO: 2
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	
	UNIDAD	COMPETENCIA	PHP			

## Índice

1.	¿Qué es PHP?	3
1.1.	Funcionamiento	3
2.	Estructura de una página PHP	4
3.	Programación en PHP	5
3.1.	Sintaxis básica PHP	5
3.1.1.	Comentarios	5
3.1.2.	Uso de mayúsculas y minúsculas	5
3.1.3.	Instrucciones	5
3.1.4.	Funciones y librerías	5
3.2.	Variables y constantes en PHP	6
3.2.1.	Variables	6
3.2.2.	Declaración de las variables	7
3.2.3.	Tipos de datos	7
3.2.4.	Ámbito de las variables	8
3.2.5.	Constantes	9
3.2.6.	Consideraciones	9
3.2.7.	Definición de constantes	10
3.3.	Entrada y salida por pantalla	10
4.	Operadores aritméticos y lógicos	11
4.1.	Operadores de comparación	11
4.2.	Operadores lógicos	11
4.3.	Operadores aritméticos	11
5.	Sentencias condicionales	12
5.1.	Sentencias if-else	12
5.2.	Sentencias if-elseif-else	13
5.3.	Sentencia switch	13
6.	Bucles	14
6.1.	Bucles numéricos for	14
6.2.	Bucles lógicos while	15
6.3.	Bucles lógicos do...while	15
7.	Cadenas de caracteres	15
8.	Matrices o arrays de datos	19
8.1.	Matrices numéricas y asociativas	19
8.2.	Inicializar un array	20
8.3.	Arrays multidimensionales	20

	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Sistemas de gestión empresarial				CURSO: 2
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	
	UNIDAD	COMPETENCIA	PHP			

8.4.	Recorrer un array numérico.....	21
8.5.	Recorrer un array asociativo.....	22
8.6.	Eliminación de datos de un array .....	22
8.7.	Ordenar un array .....	23
8.8.	Otras funciones de array .....	24

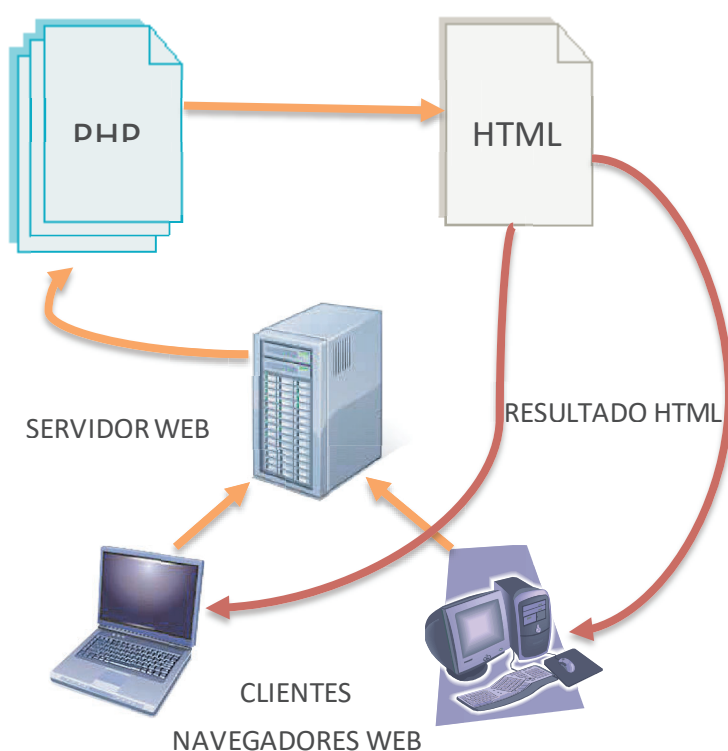
COLEXIO <b>VIVAS</b> S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Sistemas de gestión empresarial			CURSO:	2
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	
	UNIDAD	COMPETENCIA	PHP			

## 1. ¿Qué es PHP?

PHP (acrónimo recursivo de PHP: Hypertext Preprocessor) es un lenguaje de programación dirigido a la creación de páginas web dinámicas.

Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página Web resultante.

### 1.1. Funcionamiento



El usuario escribe una URL en su navegador y éste envía una petición HTTP al servidor web. El servidor lee la página PHP (la interpreta) y envía al navegador del usuario un documento compilado en formato HTML.

El código PHP se ejecuta en el servidor y lo que se envía al navegador es una página HTML, por tanto el código fuente PHP nunca se verá en el navegador del cliente. Esto implica que es necesario un servidor con un intérprete de PHP para poder ejecutar los Scripts.

COLEXIO <b>VIVAS</b> S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Sistemas de gestión empresarial			CURSO:	2
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	
	UNIDAD	COMPETENCIA	PHP			

## 2. Estructura de una página PHP

Una página PHP es un archivo de texto que contiene uno o varios fragmentos de código PHP y que también puede contener fragmentos de código HTML, CSS y JavaScript. Las páginas PHP se guardarán con la extensión .php

Los fragmentos de código PHP están delimitados por las etiquetas:

**<?php** (etiqueta inicial) y **?>** (etiqueta final).

Todas las sentencias terminan por ;

No puede ir ninguna etiqueta HTML, CSS o JavaScript en medio de las etiquetas php. Si queremos insertar alguna etiqueta HTML/CSS/ código JavaScript en el medio (saltos de línea, columnas, código JavaScript...) deberá hacerse dentro de las instrucciones PHP.


Ejemplos:

<pre>&lt;!DOCTYPE html&gt; &lt;html&gt;   &lt;head&gt;     &lt;title&gt;Ejemplo bien hecho&lt;/title&gt;   &lt;/head&gt;   &lt;body&gt;     &lt;?php echo "Hola mundo &lt;br/&gt;"; ?&gt;   &lt;/body&gt; &lt;/html&gt;</pre>	<pre>&lt;!DOCTYPE html&gt; &lt;html&gt;   &lt;head&gt;     &lt;title&gt;Ejemplo mal hecho&lt;/title&gt;   &lt;/head&gt;   &lt;body&gt;     &lt;?php echo "Hola mundo"; &lt;br/&gt; ?&gt;   &lt;/body&gt; &lt;/html&gt;</pre>
---	--

En el código de la derecha la etiqueta HTML <br/> está integrada dentro del comando PHP echo. En el código de la izquierda la etiqueta HTML <br/> va fuera de echo por lo que el servidor intentará ejecutar la instrucción <br/> como si fuese PHP y al no existir como tal generará un error y no se ejecutará.

El código en PHP no tiene que ir todo junto en una página web. Se pueden insertar tantas líneas de código como hagan falta siempre que vayan entre las etiquetas <?php y ?>. Ejemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>HTML y PHP</title>
  </head>
  <body>
    <p>Esta línea es HTML</p>
    <?php echo "<p>Esta línea está generada con php </p>" ?>
    <a href="http://www.google.es">Enlace HTML </a> <br/>
    <?php echo "<a href='http://www.google.es'>Enlace generado con php </a>" ?>
  </body>
</html>
```

	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Sistemas de gestión empresarial				CURSO: 2
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	
	UNIDAD	COMPETENCIA	PHP			

### 3. Programación en PHP

#### 3.1. Sintaxis básica PHP

##### 3.1.1. Comentarios

PHP permite comentarios de una sola línea (de dos formas) y comentarios multilineas.

Las etiquetas para comentarios en php son: "//", "#", /\*....\*/ Ejemplo:

```
<?php
//Comentario de una línea, la siguiente línea es leída como código de programación
echo 'Linea de programacion';
#Comentario de una línea, tiene la misma utilidad que "//".
echo 'Linea de programacion';
/* Comentario multilinea
abarca varias líneas,
posee una etiqueta de apertura y otra de cierre */
echo 'Linea de programacion';
?>
```

##### 3.1.2. Uso de mayúsculas y minúsculas

En PHP las instrucciones pueden ir indistintamente en mayúsculas o minúsculas, pero los nombres de las variables no. PHP diferencia entre mayúsculas y minúsculas para los nombres de las variables y las claves (campos de los arrays). Esto significa que la variable \$A es distinta que la variable \$a, por lo que hay que tener cuidado a la hora de utilizar los nombres de las variables.

**Por convenio los nombres de las variables se utilizan en minúsculas.**

##### 3.1.3. Instrucciones

Un fragmento de código PHP va a estar compuesto por una varias instrucciones, para que el servidor sepa dónde termina una instrucción y donde comienza la siguiente, las instrucciones en PHP deben terminar en ;

Ejemplo:


```
<?php
$a = 11;
echo "Hola Mundo <br/>" ;
echo "a vale $a <br/>" ;
echo "a vale . $a. <br/>" ;
?>
```

##### 3.1.4. Funciones y librerías

Las funciones en PHP siguen la misma estructura que en otros lenguajes de programación; son conjuntos de instrucciones a las que se le pasan los parámetros que se desean y que pueden devolver (o no), un valor.

Estructura:

```
function Nombre (parámetro1, parámetro2,,){
    instrucción1;
    instrucción2;
    ...
    return valor;
}
```

	RAMA:	Informática	CICLO:	DAM			
	MÓDULO	Sistemas de gestión empresarial				CURSO:	2
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:		
	UNIDAD		COMPETENCIA	PHP			

Con el fin de hacer el código lo más modular y limpio posible, es conveniente utilizar el mayor número de funciones posibles y agrupar estas en archivos de extensión .php llamados **librerías**.

Para tener acceso a las funciones contenidas en estas librerías es necesario insertar la directiva **include** al inicio del archivo, por ejemplo en el encabezado <head>

Ejemplo:

<pre>&lt;!DOCTYPE html&gt; &lt;html&gt;   &lt;head&gt;     &lt;?php include 'misfunciones.php'?&gt;     &lt;title&gt;Tabla&lt;/title&gt;   &lt;/head&gt;   &lt;body&gt;     &lt;?php generaTabla(3,4); ?&gt;   &lt;/body&gt; &lt;/html&gt;</pre>	<pre>&lt;?php function generaTabla (\$filas, \$columnas){   echo "&lt;table border='1'&gt;";   for (\$i=0; \$i&lt;\$filas; \$i++){     echo "&lt;tr&gt;";     for(\$j=0; \$j&lt;\$columnas; \$j++){       echo"&lt;td&gt; [\$i][\$j] &lt;/td&gt;";     }     echo "&lt;/tr&gt;";   }   echo "&lt;/table&gt;"; } ?&gt;</pre>
a) Index.php	b) misfunciones.php

Los archivos y librerías que sólo contengan código en PHP se definen en el NetBeans como PHP Files, y los archivos HTML que contienen código en PHP se definen en el NetBeans como PHP Web Files.

Cuando se pasan datos a una función lo que realmente se le está pasando es una **copia** de la variable, por lo que los cambios que sufra esa variable dentro de la función no se verán reflejados en el valor real de la variable. Si deseamos modificar la variable real, es necesario pasarle a la función la dirección de memoria donde está almacenada la variable, es decir, habrá que pasar la variable por **REFERENCIA**. El paso por referencia de una variable se realiza poniendo el símbolo **&** delante de la variable en cuestión.

<pre>&lt;head&gt;   &lt;?php include 'misfunciones.php' ?&gt;   &lt;title&gt;Media&lt;/title&gt; &lt;/head&gt; &lt;body&gt;   &lt;?php     echo media (20, 50, \$res), "&lt;br/&gt;";     echo "resultado: " . \$res   ?&gt; &lt;/body&gt;</pre>	<pre>function media (\$a, \$b, &amp;\$resultado){   \$media=(\$a+\$b)/2;   \$resultado=\$media;   return \$resultado; }</pre>
a) index.php	b) misfunciones.php

## 3.2. Variables y constantes en PHP

### 3.2.1. Variables

Como vimos antes todas las variables deben estar precedidas por el signo dólar (\$), y se le asigna contenido con el signo igual (=).

Con las variables, PHP distingue entre mayúsculas y minúsculas, por lo que \$Variable y \$variable no serían lo mismo.

COLEXIO <b>VIVAS</b> S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Sistemas de gestión empresarial			CURSO:	2
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	
	UNIDAD	COMPETENCIA	PHP			

Los nombres de las variables en PHP siguen una serie de reglas:

- No pueden llevar espacios en blanco.
- Sólo pueden llevar números, letras y el símbolo \_ (guión bajo).
- No pueden empezar por un número.

Ejercicio. Di si las siguientes variables están bien escritas:

- \$dia semana
- SdiaSemana
- \$dia\_semana
- \$dia#semana
- \$1diasemana

### 3.2.2. Declaración de las variables

Las variables en PHP se crean cuando le asignamos un valor a las mismas, para una mejor comprensión del código conviene definir e inicializar las variables al inicio de los scripts en PHP.

Ejemplo:

```
<?php
    $a = 1,25;           //Las variables $a y $A son variables distintas
    $A = 0;
    $dato=3456;          //La variable $dato es contiene un entero
    $dato= false;        //Sobreescribo la variable $dato y almaceno un booleano
    $dato= 'Hola mundo'; //Sobreescribo $dato y almaceno un string
    //instrucciones PHP
    ...
?>
```

### 3.2.3. Tipos de datos

En una variable podemos almacenar diferentes tipos de información, es decir diferentes tipos de datos, éstos pueden ser:

- **Numéricos:** Los números pueden ser
  - Enteros (INTEGER) \$dato\_entero = 34;
  - Decimales (FLOAT) \$dato\_decimal = 3.222;
- **Cadenas de caracteres:** Datos que sirven para almacenar texto, ya sea una única letra (carácter), o una cadena de caracteres. El texto almacenado debe ir entre comillas dobles o simples  
\$texto = 'Hola mundo !!!'

Dentro del texto entrecomillado podemos escribir una serie de caracteres especiales llamados **caracteres de escape**. Los más comunes son:

\n	Salto de línea <sup>1</sup>	\\$	Signo de \$
\'	Comilla simple	\"	Comillas dobles
\r	Retorno de carro	\t	Tabulador horizontal
\\	Barra invertida		

<sup>1</sup> El salto de línea \n no se refiere a los saltos de línea del navegador (no es un <br />), sino que se refiere a los saltos de línea que inserta PHP en el código HTML cuando lo genera.

COLEXIO <b>VIVAS</b> S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Sistemas de gestión empresarial				CURSO: 2
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	
	UNIDAD	COMPETENCIA	PHP			

Cuando una cadena de caracteres va entre comillas simples, los únicos caracteres de escape que funcionan como tal son `\\` y `\'`. El resto se mostrarán por pantalla como `\n`, `\$`,...

- **Datos booleanos.** A partir de PHP4 se pueden usar tipos de datos booleanos. Los valores que pueden tomar son verdadero (true) o falso (false).

Ejemplo: `$validado = true;`

Además de los valores true o false, generará un valor true cualquier dato numérico distinto de 0 (positivo o negativo), y generará un valor false cualquier dato numérico de valor igual a 0. Es decir:

`$validado = 5;` //equivale a un true

`$validado = -5;` //equivale a un true

`$validado = 0;` //equivale a un false

En cualquier caso PHP es un lenguaje de programación **no tipado**, lo que significa que una misma variable puede almacenar un dato de tipo entero y después pasar a almacenar una cadena de caracteres.

### 3.2.4. Ámbito de las variables

En PHP podemos tener variables locales y variables globales.


Serán **variables locales** todas aquellas variables que se definen dentro de una función, y serán **variables globales** todas las variables que se definan fuera de una función.

Las variables globales son accesibles desde cualquier punto del código y las variables locales sólo dentro de la función en la que se definen.

Dentro de una función no se puede acceder a una variable global a través de su nombre. Para hacerlo se puede proceder de dos formas:

- Mediante el array o matriz del sistema `$GLOBALS` (índice), utilizando como índice de la matriz el nombre de la variable global a la que se desea acceder.
- Indicando al inicio de la función que la variable que se va a utilizar es global mediante la palabra `global` y el nombre de la variable global detrás.



	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Sistemas de gestión empresarial				CURSO: 2
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	
	UNIDAD	COMPETENCIA	PHP			

```

<?php
//Definición de variables globales
$dato = 11;
$cadena = 'Hola mundo';

function acesoVarGlobales(){
    //Definición de las variables globales
    global $cadena;
    //Definición de las variables locales
    $dato= 'HOLA!';
    //$dato se llama igual que una variable global, pero como
    //no se pone global delante, PHP la crea como si fuese una
    //variable local a esta función.

    echo "variable global cadena: $cadena <br/>"; // muestra Hola Mundo
    echo "variable local dato: $dato <br/>"; //Muestra HOLA!

    //Definimos las variables locales
    echo $GLOBALS["dato"]."<br/>"; //Muestra 11;
    echo $GLOBALS["cadena"]; //Muestra 'Hola Mundo'
}
?>

```

### 3.2.5. Constantes

Las constantes son similares a las variables, con la salvedad de que no llevan el signo dólar delante, y sólo la podemos asignar una vez.

### 3.2.6. Consideraciones

En PHP las constantes:

- **No** van precedidas por el símbolo \$
- Los nombres de las constantes siguen los mismos criterios que los nombres de las variables, es decir, sólo admiten letras (sin contar caracteres especiales como ñ o ç), números y guiones bajos, no pueden empezar por número pero sí por guion bajo.
- **Por convenio suelen llevar nombres en mayúsculas**
- Se definen mediante la función define(). **Una vez creadas no se pueden modificar ni eliminar.**
- Sólo podemos tener constantes de tipo: booleanas, numéricas (enteras o decimales) y de tipo string (cadena de caracteres)
- Se pueden definir dentro o fuera de una función, una vez definidas, quedarán accesibles desde cualquier punto de código.

COLEXIO <b>VIVAS</b> S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Sistemas de gestión empresarial			CURSO:	2
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	
	UNIDAD	COMPETENCIA	PHP			

### 3.2.7. Definición de constantes

Para definir las constantes utilizamos la función **define ('NOMBRE\_CONSTANTE', valor)**, a la que le debemos pasar dos parámetros, el nombre de la constante y el valor que queremos que tenga. Para acceder a esa constante bastará con que pongamos su nombre.

<pre>&lt;!DOCTYPE html&gt; &lt;html&gt;   &lt;head&gt;     &lt;?php include 'misfunciones.php' ?&gt;     &lt;title&gt;Tabla&lt;/title&gt;   &lt;/head&gt;   &lt;body&gt;     &lt;p&gt; &lt;?php imprimeConstante1() ?&gt; &lt;/p&gt;     &lt;p&gt; &lt;?php imprimeConstante2() ?&gt; &lt;/p&gt;   &lt;/body&gt; &lt;/html&gt;</pre>	<pre>&lt;?php define ('CONST_1', 'Clase de php'); function imprimeConstante1(){     echo "&lt;br/&gt;". CONST_1; //Imprime 'Clase de PHP'     define ('CONST_2', 10); } function imprimeConstante2(){     echo "&lt;br/&gt;". CONST_2; //Imprime 10 } ?&gt;</pre>
a) index.php (Página en HTML)	b) misfunciones.php (Página en PHP)

### 3.3. Entrada y salida por pantalla

PHP se ejecuta en el servidor, y la única forma de insertar datos que afecten al servidor es a través de formularios.

La salida por pantalla en PHP consiste en escribir datos en la página HTML que se enviarán al usuario. Esto puede hacerse mediante dos instrucciones:

- **echo** e **print** que no permiten formatear la salida.
- **printf** que sí permite formatear la salida.

Para unir varias cadenas utilizamos **'.' ó ','**

Ejemplos de utilización:

```
<?php
$a = 2.5;
$b = 3;
$c = 'Hola Mundo ';
echo '<br/> Ejemplo de echo: a= ', $a, ' b= ', $b;
print '<br/> Ejemplo de print: a= ' . $a. ' b= ' . $b;
printf ('<br/> Ejemplo de printf: a=%f, b=%d c=%s', $a,$b,$c);
?>
```

Formatos más importantes en PHP:

%c	Carácter ASCII	%d	Número entero
%s	Cadena de caracteres	%f	Número decimal

COLEXIO <b>VIVAS</b> S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Sistemas de gestión empresarial			CURSO:	2
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	
	UNIDAD	COMPETENCIA	PHP			

El texto del echo y del print puede ponerse entre comillas dobles o entre comillas simples. Si insertamos una variable dentro de una cadena con comillas dobles, se imprimirá por pantalla el valor de esa variable. Si insertamos una variable dentro de una cadena con comillas simples, se imprimirá el nombre de la variable. Es decir, con comillas dobles se evalúan las variables, y con comillas simples no.

Ejemplo:

```
<?php
$a = 2.5;
$b = 'Hola';
echo "<br/> Ejemplo de echo: a= $a, $b"; //imprime Ejemplo de echo a=$a, b=$b
echo "<br/> Ejemplo de print: a= $a b= $b"; //imprime Ejemplo de echo a=2.5, b=Hola
?>
```

## 4. Operadores aritméticos y lógicos

### 4.1. Operadores de comparación

==	igual	>	mayor	>=	mayor o igual
!=	distinto	<	menor	<=	menor o igual
===	idéntico			!==	no idéntico

Idéntico devuelve true si el contenido de las dos variables es el mismo y además son el mismo tipo de datos.

No idéntico devuelve true si el contenido es diferente o si son de distinto tipo.

### 4.2. Operadores lógicos

&&	Operación AND	and	Operación AND
	Operación OR	or	Operación OR
!	Operación NOT	xor	Operación Or eXclusiva

AND	OR	XOR
1 and 1 = 1	1 or 1 = 1	1 xor 1 = 0
1 and 0 = 0	1 or 0 = 1	1 xor 0 = 1
0 and 1 = 0	0 or 1 = 1	0 xor 1 = 1
0 and 0 = 0	0 or 0 = 0	1 xor 1 = 0
Verdadero si ambos son verdaderos	Verdadero si alguno es verdadero	Verdadero si sólo uno de los dos es verdadero

### 4.3. Operadores aritméticos

Los operadores aritméticos de PHP se pueden aplicar a cualquier variable o constante numérica:

+	suma	\$a=3; \$b=1; \$c = \$a + \$b; // c vale 4
-	resta	\$a=3; \$b=1; \$c = \$a - \$b; // c vale 2
*	producto	\$a=3; \$b=2; \$c = \$a * \$b; // c vale 6

COLEXIO <b>VIVAS</b> S.L.	RAMA:	Informática	CICLO:	DAM	
	MÓDULO	Sistemas de gestión empresarial			CURSO: 2
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:
	UNIDAD	COMPETENCIA	PHP		

/	división	\$a=6; \$b=3; \$c = \$a / \$b; // c vale 2
%	resto	\$a=3; \$b=2; \$c = \$a % \$b; // c vale 1, es el resto de dividir 3/2
++	incremento en 1	\$a=5; \$a++; // a vale 6
--	decremento en 1	\$a=5; \$a--; // a vale 4
+=	sumamos valor	\$a=2; \$a+=3; // \$a = \$a + 3 → \$a vale 5
-=	restamos valor	\$a=2; \$a-=3; // \$a = \$a - 3 → \$a vale -1
*=	multiplicamos por valor	\$a=2; \$a*=3; // \$a = \$a * 3 → \$a vale 6
/=	dividimos por valor	\$a=6; \$a/=3; // \$a = \$a / 3 → \$a vale 2
%=	resto	\$a=6; \$a%=3; // \$a = \$a % 3 → \$a vale 0

En PHP existe una librería completa (Math) con multitud de funciones para distintas operaciones matemáticas  
<http://www.php.net/manual/en/ref.math.php>

## 5. Sentencias condicionales

### 5.1. Sentencias if-else

La sentencia if-else permite ejecutar un bloque de instrucciones si la condición es Verdadera y otro bloque de instrucciones si ésta es Falsa. Es importante tener en cuenta que la condición que evaluemos ha de estar encerrada entre paréntesis (esto es aplicable a todas las sentencias condicionales).


La forma de utilización es la siguiente:

```
if (condición) {
    instrucciones que se ejecutan si la condición es VERDADERA;
} else {
    Instrucciones que se ejecutan si la condición es FALSA;
}
```

Existe una forma sencilla de usar la sentencia IF cuando no tenemos que usar el ELSE y solo tenemos que ejecutar una línea de código.

```
if ($a > 4) echo "$a es mayor que 4";
```

Las instrucciones a ejecutar después del if y del else tienen que ir entre llaves {}. En el caso de tener únicamente una instrucción se pueden poner o no.

	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Sistemas de gestión empresarial				CURSO: 2
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	
	UNIDAD	COMPETENCIA	PHP			

## 5.2. Sentencias if-elseif-else

La sentencia IF...ELSEIF...ELSE permite ejecutar varias condiciones en cascada.

Ejemplo:

```
<?php
    if ($nombre == "") {
        echo "Tú no tienes nombre";
    } elseif (($nombre == "eva") OR ($nombre == "Eva")) {
        echo "Tu nombre es EVA";
    } else {
        echo "Tu nombre es " . $nombre;
    }
?>
```

## 5.3. Sentencia switch

Cuando se necesitan muchos if-elseif-else anidados, resulta más elegante incluir una sentencia switch. La sintaxis de esta función es la siguiente:

```
switch ($variable){
    case valor1:
        instrucciones;
        break;
    case valor2:
        instrucciones;
        break;
    ...
    default:
        instrucción predeterminada.
}
```

La opción default es opcional

Ejemplos de utilización:

<pre>&lt;html&gt; &lt;body&gt; &lt;?php \$dia="Sábado"; switch (\$dia) {     case "Lunes":         echo "Hoy es Lunes";         break;     case "Martes":         echo "Hoy es Martes";         break;     case "Miercoles":         echo "Hoy es Miercoles";</pre>	<pre>&lt;html&gt; &lt;body&gt; &lt;?php \$dia="Sábado"; switch (\$dia) {     case "Lunes":         echo "Hoy es Lunes";         break;     case "Martes":         echo "Hoy es Martes";         break;     case "Miercoles":         echo "Hoy es Miercoles";</pre>
---	---

COLEXIO <b>VIVAS</b> S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Sistemas de gestión empresarial			CURSO:	2
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	
	UNIDAD	COMPETENCIA	PHP			

<pre> break; case "Jueves":     echo "Hoy es Jueves";     break; case "Viernes":     echo "Hoy es Viernes";     break; case "Sábado":     echo "Hoy es Sábado";     break; case "Domingo":     echo "Hoy es Fin de semana";     break; default:     echo "Esa cadena no corresponde a ningún día de la semana"; } ?&gt; &lt;/body&gt; &lt;/html&gt; </pre>	<pre> break; case "Jueves":     echo "Hoy es Jueves";     break; case "Viernes":     echo "Hoy es Viernes";     break; case "Sábado": case "Domingo":     echo "Hoy es Fin de semana";     break; default:     echo "Esa cadena no corresponde a ningún día de la semana"; } ?&gt; &lt;/body&gt; &lt;/html&gt; </pre>
--	---

Página en HTML (index.php)

Página en HTML (index.php)

Dado un caso concreto para un valor, el switch ejecuta las instrucciones hasta que se encuentra con un break o con una llave de fin de switch.

## 6. Bucles

### 6.1. Bucles numéricos for

La instrucción for se utiliza cuando necesitamos repetir un conjunto de instrucciones un número concreto de veces.

La estructura tiene 3 argumentos:

- Variable de inicialización
- Condición para seguir en el bucle
- Variable que incrementa o decrementa la variable de inicialización Su estructura es la siguiente:

```

for (<Inicialización>; <Condición>; <Incremento o Decremento>){
    <Instrucciones>;
}

```

Ejemplo:

```

<?php
for ($contador= 0; $contador< 5; $contador++){
    echo "<br/> Línea nº $contador";
}
?>

```

COLEXIO <b>VIVAS</b> S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Sistemas de gestión empresarial			CURSO:	2
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	
	UNIDAD	COMPETENCIA	PHP			

Después de un for() nunca pondremos un ; porque esto supondría que el for acabó y no queremos ejecutar ninguna instrucción más.

## 6.2. Bucles lógicos while

La instrucción while se utiliza cuando queremos repetir una serie de instrucciones en función de una variable lógica (no de un contador numérico).

Su estructura es la siguiente:

```
while (condicion){
    <Instrucciones>;
}
```

El bucle se ejecutará siempre que la condición entre paréntesis sea verdadera (true). Esto significa que, a priori, no se tiene que saber cuántas veces se va a ejecutar el bucle (como en el for)

Ejemplo

```
<?php
$contador = 0;
while ($contador < 5) {
    echo "<br/> Línea nº $contador";
    $contador++;
}
?>
```

## 6.3. Bucles lógicos do...while

Existen dos bucles que se ejecutan en función de una condición lógica. El bucle while, que evalúa la condición y si es verdadera ejecuta las instrucciones y el bucle do...while, que primero ejecuta las instrucciones y después comprueba la condición.

Como mínimo el conjunto de instrucciones de un bucle do...while siempre se ejecuta una vez.

Ejemplo:

```
<?php
$contador = 0;
do {
    echo "<br/> Línea nº $contador";
    $contador++;
} while ($contador < 5)
?>
```

Importante. Podemos detener la ejecución de un bucle cualquiera o de un switch mediante la instrucción break situada en cualquier punto del código.

## 7. Cadenas de caracteres

PHP es un lenguaje orientado a cadenas de caracteres y por tanto integra infinidad de funciones para trabajar con ellas. En este tema se recopilan algunas de las más comunes. Se pueden consultar las funciones en <http://php.net/manual/en/book.strings.php>

COLEXIO <b>VIVAS</b> S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Sistemas de gestión empresarial			CURSO:	2
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	
	UNIDAD	COMPETENCIA	PHP			

Para entender cómo funcionan las cadenas de caracteres en PHP hay que tener en cuenta que el primer carácter de la cadena está situado en la posición 0.

- **strlen(cadena)**: Indica la longitud de la cadena de caracteres utilizada como argumento, es decir el número de caracteres que la componen.

Ejemplo <?php echo strlen(1234567); ?> devuelve 7 que es la longitud de la cadena.

- **strcmp(cadena1, cadena2)**: Compara las cadenas en modo binario. Devuelve:
  - < 0 si la cadena 1 es menor que la cadena 2
  - > 0 si la cadena 1 es mayor que la cadena 2
  - 0 si las dos cadenas son iguales

**Esta comparación es sensible a mayúsculas y minúsculas.**

Ejemplo:

```
<?php
$var1 = "Hola";
$var2 = "hola";
$comparacion = strcmp($var1, $var2); echo $comparacion;
?>
```

Da como resultado \$comparacion=-1 porque al distinguir entre mayúsculas y minúsculas la cadena mayor es \$var2

- **strcasecmp(cadena1, cadena2)**: Compara cadenas en modo binario pero no es sensible a mayúsculas y minúsculas. Devuelve:
  - < 0 si la cadena es menor que la cadena 2
  - > 0 si la cadena 1 es mayor que la cadena 2
  - 0 si las dos cadenas son iguales

Ejemplo:

```
<?php
$var1 = "Hola";
$var2 = "hola";
$comparacion = strcasecmp($var1, $var2);
echo $comparacion;
?>
```

Da como resultado \$comparacion=0 porque al no distinguir entre mayúsculas y minúsculas, las dos cadenas son iguales.

- **preg\_split(separador, cadena)**: Divide una cadena en varias usando un carácter separador. Devuelve un array.



COLEXIO <b>VIVAS</b> S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Sistemas de gestión empresarial			CURSO:	2
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	
	UNIDAD	COMPETENCIA	PHP			

Ejemplo:

```
<?php
$linea = "María Pérez Rodríguez";
list($uno, $dos, $tres) = preg_split("/ /", $linea, 3);
echo "<br> 1: $uno <br> 2: $dos <br> 3: $tres";
?>
```

Devuelve en \$uno la cadena “María” en \$dos la cadena “Pérez” y en \$tres la cadena “Rodríguez”. Con la instrucción list lo que hacemos es asignar de una sola vez las tres variables a las tres cadenas.

- **sprintf(cadena de formato, var1, var2...)**. Formatea una cadena de texto del mismo modo que printf, pero el resultado se devuelve también como una cadena. Ejemplo:

```
<?php
$entero = 6;
$decimal = 5.5;
$cadena = "hola mundo";
$resultado = sprintf("%d %f %s", $entero, $decimal, $cadena ); echo "<br> sprintf: $resultado";
?>
```


Devuelve en \$resultado la cadena: “6 5.500000 hola mundo”;

- **substr(cadena, inicio, len)**: Devuelve una subcadena comprendida dentro de otra, empezando por “inicio” y de longitud “len”. Si “inicio” es positivo o 0, empieza desde inicio hacia adelante, y si es negativo, la posición inicial se cuenta desde el final de la cadena, contando también hacia adelante tantos caracteres como marque “len”

Ejemplos:

```
<?php
echo substr('abcdef', 1); // Devuelve bcdef
echo substr('abcdef', 1, 3); // Devuelve bcd
echo substr('abcdef', 0, 4); // Devuelve abcd
echo substr('abcdef', -1, 1); // Devuelve f
echo substr("abcdef", -2); // Devuelve "ef"
echo substr("abcdef", -3, 1); // Devuelve "d" echo substr("abcdef", -3, 3); // Devuelve "def"
?>
```

- **strpos(cadena1, cadena2)**: Busca la cadena2 dentro de cadena1 indicando la posición en la que se encuentra. Esta función devuelve false (o algo que se pueda interpretar como tal, un o una cadena vacía), en caso de que no se encuentren coincidencias.

	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Sistemas de gestión empresarial				CURSO: 2
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	
	UNIDAD	COMPETENCIA	PHP			

Ejemplo:

```
<?php
$mi_cadena = 'abc';
$character = 'b';
$posicion = strpos($mi_cadena, $character); if ($posicion === false) {
echo "No se encontró '$character' en la cadena '$mi_cadena';
} else {
echo "Se encontró '$character' en la cadena '$mi_cadena'; echo " en la posición $posicion";
}
?>
```

- **str\_replace(cadena1, cadena2, texto):** Substituye la cadena 1 por la cadena 2 en el texto.


Ejemplo:

```
<?php
$texto = 'Cucaracho';
$cadena1 = 'ara';
$cadena2 = 'uru';
$resultado = str_replace($cadena1, $cadena2, $texto); echo "<br/> Cadena original: $texto";
echo "<br/> Cadena final: $resultado";
?>
```

Devuelve como resultado la cadena “Cucurucho”.

- **strtolower(cadena):** Convierte a cadena en minúsculas.
- **strtoupper(cadena):** Convierte la cadena en mayúsculas.

```
<?php
$cadena = 'Hola Mundo';
echo strtolower($cadena); // imprime 'hola mundo' echo strtoupper($cadena); // imprime 'HOLA
MUNDO'
?>
```

	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Sistemas de gestión empresarial				CURSO: 2
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	
	UNIDAD	COMPETENCIA	PHP			

## 8. Matrices o arrays de datos

### 8.1. Matrices numéricas y asociativas

Una matriz en PHP es una variable capaz de almacenar más de un dato. A la posición donde se almacena el dato se le llama clave, y al dato almacenado se le llama valor. Dependiendo del tipo de clave, existen dos tipos de matrices:

- Numéricas. La clave es un número (empezando por 0).
- Asociativas. La clave es un nombre de campo.

Ejemplos:

```
<?php
// matriz numérica:
$peliculas[0] = '8 apellidos vascos';
$peliculas[1] = 'Lo imposible';
// matriz asociativa:
$nota['Pedro'] = 8;
$nota['Carmen'] = 10;
?>
```

Al ser PHP un lenguaje no tipado, las matrices pueden almacenar cualquier tipo de datos en sus posiciones (en una posición un entero, en otra una cadena de caracteres...)


A la hora de crear una matriz numérica, si no se indica la posición donde se desea almacenar el dato, PHP insertará los datos secuencialmente detrás del último elemento asignado. No es necesario insertar los datos en orden y pueden quedar huecos en las posiciones intermedias.

Ejemplo:

```
<?php
// matriz numérica:
$peliculas[] = '8 apellidos vascos'; // si $peliculas está vacía, insertará en la posición 0
$peliculas[] = 'Lo imposible'; // posición 1
$peliculas[3] = 'Kamikaze'; // posición 3, quedando un hueco en la posición 2
?>
```

Se pueden construir arrays mediante el constructor `array()`, al que le pasaremos los elementos como `clave=>valor`. En el caso de arrays numéricos si no se inserta clave, se colocarán secuencialmente tras el último elemento insertado.

Para visualizar el contenido de un array de forma fácil se puede utilizar la función `print_r($NombreDelArray)`. Los datos se visualizan en formato `clave=>valor`.

	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Sistemas de gestión empresarial				CURSO: 2
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	
	UNIDAD	COMPETENCIA	PHP			

Ejemplo:

```
<?php
// matriz numérica:
$peliculas = array(0 => '8 apellidos vascos', 1 => 'Lo imposible', 3 => 'Kamikaze');
$actores = array('Clara Lago', 'Naomi Watts', 'Carmen Machi');
// matriz asociativa:
$notas = array('Pedro' => 8, 'Carmen' => 10);
// impresión por pantalla:
print_r($peliculas); print_r($actores); print_r($notas);
?>
```

## 8.2. Inicializar un array

Para generar arrays de un tamaño concreto y con unos valores concretos, se utiliza la instrucción `array_fill()`. Este tipo de instrucciones se utilizan muy habitualmente para inicializar arrays constantes a cero, pero también se pueden utilizar para insertar X valores constantes en un array. Su sintaxis es:

```
$nombreArray = array_fill(posicionInicial, numeroElementosAInsertar, valor);
```

Donde:

- `posicionInicial`: Primer índice donde se insertará el valor deseado
- `numeroElementosAInsertar`: Número de posiciones a rellenar en el array.
- `valor`: Valor que queremos insertar, puede ser un entero, un decimal, un string...

Ejemplo:

```
<?php
$arrayCeros = array_fill(0, 4, 0);
$arrayString = array_fill(5, 3, 'Mila');
?>
```

En estos ejemplos `$arrayCeros` será un array de 4 ceros (en las posiciones de 0 a 3), y en el segundo `$arrayString` será un array que tiene en las posiciones de 5 a 7 el texto 'Mila'.

## 8.3. Arrays multidimensionales

Se pueden definir arrays de varias dimensiones, tanto numéricas como asociativas e incluso arrays combinados (una dimensión numérica, y otra asociativa).

Ejemplo:

```
<?php
$alumno[0][1] = 'Pedro';
$alumno['alumno']['nombre'] = 'Carmen';
$alumno[0]['nombre'] = 'Pedro';
$matriz = array(array('nombre' => 'Pedro', 'edad' => 20, 'altura' => 1.75), array('nombre' => 'Rocío', 'edad' => 30, 'altura' => 1.60));
```

COLEXIO <b>VIVAS</b> S.L.	RAMA:	Informática	CICLO:	DAM
	MÓDULO	Sistemas de gestión empresarial		
	PROTOCOLO:	Apuntes clases	AVAL:	1
	UNIDAD	COMPETENCIA	PHP	
			DATA:	
			CURSO:	2

```
);
print_r($matriz);
?>
```

En el caso de la variable \$matriz, se está definiendo una matriz de la forma:

	Nombre	Edad	Altura
0	Pedro	20	1.75
1	Rocío	20	1.60

Donde la clave de la primera dimensión es numérica y la clave de la segunda dimensión es asociativa.

#### 8.4. Recorrer un array numérico


Los arrays numéricos se pueden recorrer con un bucle for. Para saber el número de elementos que tiene el array, se utiliza la función count(\$variable), que proporciona el número de elementos en el caso de arrays unidimensionales, y el número de filas en el caso de arrays multidimensionales.

##### Ejemplo con un array unidimensional

<pre>&lt;?php \$numeros = array(10, 20, 30, 40, 50); for (\$i = 0; \$i &lt; count(\$numeros);     \$i++) {     echo "\\$numeros[\$i] = \\$numeros[\$i] &lt;br/&gt;"; } ?&gt;</pre>	<pre>\$numeros[0] = 10 \$numeros[1] = 20 \$numeros[2] = 30 \$numeros[3] = 40 \$numeros[4] = 50</pre>
Código PHP	Impresión por pantalla

##### Ejemplo con un array bidimensional

<pre>&lt;?php \$numeros = array(array(10, 11, 12, 13), array(20, 21, 22, 23)); for (\$i = 0; \$i &lt; count(\$numeros); \$i++) {     for (\$j = 0; \$j &lt; count(\$numeros[\$i]);         \$j++) {         echo "\\$numeros[\$i][\$j] = " .             \$numeros[\$i][\$j] . "&lt;br/&gt;";     } } ?&gt;</pre>	<pre>\$numeros[0][0]=10 \$numeros[0][1]=11 \$numeros[0][2]=12 \$numeros[0][3]=13 \$numeros[1][0]=20 \$numeros[1][1]=21 \$numeros[1][2]=22 \$numeros[1][3]=23</pre>
Código PHP	Impresión por pantalla

	RAMA:	Informática	CICLO:	DAM
	MÓDULO	Sistemas de gestión empresarial		
	PROTOCOLO:	Apuntes clases	AVAL:	1
	UNIDAD	COMPETENCIA	PHP	
			DATA:	
			CURSO:	2

## 8.5. Recorrer un array asociativo

En el caso de los arrays asociativos se utiliza el bucle foreach. La sintaxis de este bucle es:

foreach ( \$array as \$valor) { instrucciones; }	foreach ( \$array as \$clave => \$valor) { instrucciones; }
Opción 1	Opción 2

La instrucción foreach para cada elemento de la variable \$array, coge la clave, la mete en la variable \$clave, y coge el valor y lo mete en la variable \$valor. Recorre el array hasta que llega al final por lo que no es necesario insertar la condición de fin de bucle como en el caso del for.

### Ejemplo con un array unidimensional:

<pre>&lt;?php     \$datos = array('nombre' =&gt; 'Pedro',     'edad' =&gt; 20, 'altura' =&gt; 1.75);     foreach (\$datos as \$clave =&gt; \$valor) {         echo "\$clave : \$valor&lt;br/&gt;";     } ?&gt;</pre>	<pre>nombre :Pedro edad : 20 altura : 1.75</pre>
Código PHP	Impresión por pantalla

### Ejemplo con un array bidimensional:

En el caso de los arrays bidimensionales tenemos que anidar dos bucles foreach:

<pre>&lt;?php     \$datos = array('dato1' =&gt; array('nombre' =&gt;     'Pedro', 'edad' =&gt; 20, 'altura' =&gt; 1.75),     'dato2' =&gt; array('nombre' =&gt; 'Rocío', 'edad' =&gt;     20, 'altura' =&gt; 1.60)     );     foreach (\$datos as \$clavefila =&gt;     \$valorfila) { echo "\$clavefila:&lt;br/&gt;";     foreach (\$valorfila as \$clavecol =&gt;     \$valorcol) { echo "\$clavecol : \$valorcol     &lt;br/&gt;";     }     } }</pre>	<pre>dato1: nombre : Pedro edad : 20 altura : 1.75 dato2: nombre : Rocío edad : 20 altura : 1.6</pre>
Código PHP	Impresión por pantalla

Se puede utilizar un bucle while haciendo uso de las funciones list() e each() que funciona exactamente igual que el bucle foreach.

## 8.6. Eliminación de datos de un array

Para eliminar elementos de una matriz se utiliza el comando

```
unset($variable[clave]).
```

COLEXIO <b>VIVAS</b> S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Sistemas de gestión empresarial			CURSO:	2
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	
	UNIDAD	COMPETENCIA	PHP			

Ejemplo:

```
<?php
$matriz = array(array('nombre' => 'Pedro', 'edad' => 20, 'altura' => 1.75), array('nombre' => 'Rocío',
'edad' => 20, 'altura' => 1.60)
);
unset($matriz[1]); // elimina toda la segunda fila de la variable $matriz print_r($matriz);
unset($matriz[0]['altura']); print_r($matriz);
?>
```

En el primer unset(), se eliminaría toda la segunda fila de la variable matriz.

En el segundo unset(), se elimina un dato concreto de un campo.

## 8.7. Ordenar un array

Cuando se accede a una base de datos para hacer una búsqueda, los resultados deben enviarse al usuario de forma ordenada. PHP dispone de multitud de funciones que permiten ordenar matrices, algunos ejemplos son:

- sort(\$matriz): Ordena alfanuméricamente los valores de los elementos de una matriz. No ordena por claves, sino por el contenido de los campos.
- rsort(\$matriz): Igual que sort pero ordena de forma inversa.
- ksort(\$matriz): Ordena alfanuméricamente el array por claves, e mantiene la relación entre índices y valores asociados.


Todas estas funciones devuelven true cuando la ordenación se llevó a cabo correctamente y false en caso de error.

Ejemplo:

```
<?php
$fruta = array('d' => 'limon', 'a' => 'mandarina', 'c' => 'uvas', 'b' => 'kiwi'); echo 'Matriz normal: ';
print_r($fruta);
echo '<br/> Ordenada con ksort: '; ksort($fruta);
print_r($fruta);
echo '<br/> Ordenada con sort: '; sort($fruta);
print_r($fruta); echo '<br/> Ordenada con rsort: '; rsort($fruta);
print_r($fruta);
?>
```

La salida por pantalla será:

```
Matriz normal: Array ( [d] => limon [a] => mandarina [c] => uvas [b] => kiwi )
Ordenada con ksort: Array ( [a] => mandarina [b] => kiwi [c] => uvas [d] => limon )
Ordenada con sort: Array ( [0] => kiwi [1] => limon [2] => mandarina [3] => uvas )
Ordenada con rsort: Array ( [0] => uvas [1] => mandarina [2] => limon [3] => kiwi )
```

	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Sistemas de gestión empresarial				CURSO: 2
	PROTOCOLO:	Apuntes clases	AVAL:	1	DATA:	
	UNIDAD	COMPETENCIA	PHP			

## 8.8. Otras funciones de array

- `isset($matriz[$clave])`: devuelve true se el campo \$clave da matriz asociativa \$matriz fue creado. Devuelve false en otro caso. Si existe pero está vacía, dado que o espacio en memoria existe, devuelve true.
- `current($matriz)` e `pos($matriz)`: devuelve el valor de la posición actual del puntero interno de la matriz, e devuelve false cuando está al final da matriz o cuando la matriz está vacía.
- `key($matriz)`: devuelve el índice de la posición actual (no el valor). Si es un array escalar, devuelve el número, y si no la cadena de caracteres.
- `next($matriz)`: devuelve el valor del siguiente elemento y avanza el puntero interno una posición. Devuelve false si no existen más.
- `prev($matriz)`: igual que next pero con el anterior, retrocediendo el puntero interno y devolviendo false si el elemento actual es el primero.
- `end($matriz)`: coloca el puntero interno en el último elemento de un array escalar.
- `reset($matriz)`: rebobina el puntero interno de la matriz al su primer elemento, devolviendo o bien el valor de ese elemento, o false si la matriz está vacía.