Introducción al Lenguaje PHP

**Introducción**

PHP (acrónimo recursivo de PHP: *Hypertext Preprocessor*) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser insertado en HTML.

* Lenguaje de **alto nivel** cuyo código se incorpora a páginas web HTML
* Se ejecuta en el **servidor** o en un ordenador configurado como servidor
* Una vez **interpretado** por el servidor, genera una salida HTML
* Parecido en su sintaxis con lenguajes de programación estructurada, como C
* Diseñado para la creación de páginas **web dinámicas**
* Es un lenguaje de **código abierto**
* Lo que distingue a PHP de JavaScript es que el código es ejecutado en el servidor, generando HTML
* Muchas aplicaciones web están construidas con PHP. Ejs.: Wordpress, Joomla, Drupal, osCommerce, Prestashop, phpBB, SMF, Moodle, ...

**¿Qué necesito?**Un servidor con soporte activado para PHP

* Ficheros con la extensión .php, que han de ser tratados por PHP
* Es habitual tener instalada una base de datos como MySQL o análoga

**Esquema funcionamiento**  


**Historia**

* Creado en 1995. Denominado “Zend Engine”, (Zeev Suraski y Andi Gutmans)
* 1998, versión 4. Programación orientada a objetos. Aumento velocidad
* La versión 6 no pasó del borrador
* **PHP 7.4** es la versión actual.

En el sitio TutorialPHP.net (<http://tutorialphp.net/>) se pueden ver las novedades de esta versión.  
**Primera página con PHP**  
Escribir un fichero llamado **hola.php** y llevarlo al directorio raíz del servidor web

<!doctype html>

<html>

<head>

    <meta charset="utf-8" />

    <title>Título de la web</title>

</head>

<body>

    <?php echo '<p>Hola desde el Instituto Ribera </p>'; ?>

</body>

</html>

Para ejecutarlo llamar desde el navegador a hola.php en el dominio correspondiente  
Si se ha instalado en local, cargar con: http://localhost/hola.php o <http://127.0.0.1/hola.php>

El archivo es analizado por el intérprete PHP  
Código que devuelve:

<!doctype html>

<html>

<head>

    <meta charset="utf-8" />

    <title>Título de la web</title>

</head>

<body>

    <p>Hola desde el Instituto Ribera </p>'

</body>

</html>

Formato de las etiquetas especiales de PHP: **<?php** para indicar el inicio de una etiqueta de PHP, etiqueta de cierre **?>**  
Lista de editores PHP, en Wikipedia: <https://en.wikipedia.org/wiki/List_of_PHP_editors>

**Algo útil**  
  
**¿Qué tipo de navegador está utilizando el usuario visitante?**  
Identificar el *string* del agente de usuario que el navegador envía como parte de la petición HTTP  
**$\_SERVER** es una variable especial reservada por PHP que contiene toda la información del servidor web

<?php

echo $\_SERVER['HTTP\_USER\_AGENT'];

?>

Resultando, por ejemplo:  
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)

<?php

if (strpos($\_SERVER['HTTP\_USER\_AGENT'], 'MSIE') !== FALSE) {

    echo 'Estás usando el peor navegador, Internet Explorer.<br />';

}

Resultando para un usuario de Internet Explorer:  
Estás usando el peor navegador, Internet Explorer.

<?php

if (

    strpos($\_SERVER['HTTP\_USER\_AGENT'], 'MSIE') !== FALSE ||

    strpos($\_SERVER['HTTP\_USER\_AGENT'], 'Edge') !== FALSE

) {

?>

    <h3>strpos() debe haber devuelto no falso</h3>

    <p>Estás usando el peor navegador, Internet Explorer</p>

<?php

} else {

?>

    <h3>strpos() debe haber devuelto falso</h3>

    <p>Enhorabuena, no usas un producto Microsoft</p>

<?php

}

?>

Este ejemplo mezcla los modos HTML y PHP. En vez de usar una sentencia echo de PHP para mostrar algo, se sale del modo PHP y se envía solamente HTML

**Comentarios en PHP**

Como en todos los lenguajes de programación y de marcado existen los comentarios

Sintaxis:

* Doble barra **//**
* Barra y asterisco para apertura y asterisco y barra para cierre **/\* \*/**

**Variables en PHP**

Las variables se representan con un **signo de dólar** ($) seguido por el nombre, que es **sensible a minúsculas y mayúsculas**  
Un nombre de variable válido tiene que empezar con una letra o con un carácter de subrayado, seguido de letras, números y caracteres de subrayado, sin espacios.

PHP no requiere (ni soporta) la definición explicita de tipos en la declaración de variables. El **tipo de una variable normalmente no lo indica el programador**; en su lugar, lo decide PHP en tiempo de ejecución dependiendo del contexto en el que se utilice esa variable.  
 El operador de **asignación** es: **=**  
 El operador comparador de **igualdad** es: **==**

Ejemplo de variables:

<!doctype html>

<html>

<head>

    <meta charset="utf-8" />

    <title>Tipos de variables en PHP</title>

</head>

<body>

    <?php

    $a = 137;

    $b = "9.99";

    $c = "ejemplo variables PHP";

    // impresión

    echo $a . "<br>" . $b . "<br>" . $c;

    ?>

</body>

</html>

**Variables reservadas** son las que PHP define con valores por defecto, **están disponibles sin necesidad de requerir librerías** y no se puede cambiar su contenido:

* **$\_SERVER**: Información del entorno del servidor y de ejecución
* **$\_GET**: Variables en el encabezado HTTP GET
* **$\_POST**: Variables recibidas en el encabezado HTTP GET
* **$\_COOKIE**: Variable con la cual podemos crear, acceder, editar o destruir *cookies*
* **$\_FILES**: Variables que llegan al servidor con archivos mediante carga
* **$\_REQUEST**: Es una variable de array asociativo que por defecto contiene el contenido de $\_GET, $\_POST y $\_COOKIE
* **$\_SESSION**: Variables de sesión

**Tipos de datos**

En PHP **NO** es obligatorio **indicar el tipo de datos** al que pertenece una variable. Existen los siguientes tipos:

***Tipos escalares:***

* **boolean** para valores lógicos
* **integer** para números enteros
* **float** (número de coma flotante, también conocido como double)
* **string** para cadenas de caracteres

***Tipos compuestos:***

* **array** para mapas, matrices y vectores
* **object** para instancias de clases

***Tipos especiales:***

* **resource** para referencia a datos externos
* **NULL** para representar datos sin valor

**Booleanos**

Este es el tipo más simple. Un boolean expresa un valor lógico. Puede ser true (verdadero) o false (falso).

**Sintaxis**

Para especificar un literal boolean, se usan alguna de las palabras clave **TRUE** o **FALSE**. Ambas son **insensibles** a **mayúsculas y minúsculas**.

<?php

$Correcto = true; // Asigna true a la variable $Correcto

Un valor es considerado como FALSE cuando:

* el boolean **FALSE** mismo
* el integer 0 (cero)
* el float 0.0 (cero)
* el valor string vacío, y el string "0"
* un array con cero elementos
* el tipo especial **NULL** (incluidas variables no establecidas)

Cualquier otro valor será considerado como TRUE.

**Enteros(integer)**

Los integer pueden ser especificados mediante notación decimal (base 10), hexadecimal (base 16), octal (base 8) o binaria (base 2), opcionalmente precedidos por un signo (- o +).

Para usar la notación octal, se antepone al número un 0 (cero). Para usar la notación hexadecimal, se antepone al número un 0x. Para usar la notación binaria, se antepone al número un 0b.

Los enteros se pueden especificar usando una de las siguientes sintaxis:

<?php

$a = 1234; // número decimal

$a = -123; // un número negativo

$a = 0123; // número octal (equivalente a 83 decimal)

$a = 0x1A; // número hexadecimal (equivalente a 26 decimal)

?

El tamaño de un integer depende de la plataforma en la que se utilice, aunque usualmente se suele tomar un valor máximo de 32bits con signo.

**Nota**: Debido a que PHP asigna los tipos en función del contexto en el que se utilizan, la combinación de un tipo entero con un tipo string puede dar como resultado otro entero, si el string se evalúa en un contexto numérico.

<?php

$a = '1234'; // $a es una variable de  tipo string

$b = 1+$a;   // $a se asume de tipo integer ya que se  evalúa en un contexto numérico

echo $b;     // Da como resultado  1235

?>

**Desbordamiento de enteros**

Si PHP encuentra un número fuera de los límites de un integer, se interpretará como un float en su lugar. También, una operación cuyo resultado es un número fuera de los límites de un integer devolverá en su lugar un float.

**Nº de punto flotante (float)**

Los números de punto flotante (también conocidos como "flotantes", "dobles" o "números reales") pueden ser especificados usando cualquiera de las siguientes sintaxis:

<?php

$a = 1.234;

$b = 1.2e3;

$c = 7E-10;

El tamaño de un flotante depende de la plataforma, aunque un valor común consiste en un máximo de ~1.8e308 con una precisión de aproximadamente 14 dígitos decimales (lo que es un valor de 64 bits en formato IEEE).

Cadenas de caracteres (string)

Un string es una serie de caracteres donde un carácter es lo mismo que un byte. Esto significa que PHP solo soporta el conjunto de 256 caracteres y por lo tanto no tiene soporte nativo Unicode.

Un string puede llegar a almacenar hasta **2Gb de tamaño**.

Un valor de tipo string (literal) puede ser especificado de cuatro formas diferentes:

* *comillas simples*'...'
* *comillas dobles*"..."
* *sintaxis heredoc (comillas dobles en bloque)*

**Comillas simples**

Es la **forma más sencilla de representar un string** y la que consume menos ciclos de CPU para su representación, por lo que es la recomendada cuando sólo vamos a manipular cadenas de caracteres puros sin variables ni códigos de escape.

Para especificar una comilla simple dentro de un string encerrado entre comillas simples, tendremos que escribirlo anteponiendo el carácter de barra invertida (\). Para especificar una barra invertida literal, se duplica (\\).

<?php

// Cadena de caracteres

$cadena='Cadena de caracteres';

// Me llamo 'Luis Roca' y mi web es minubeinformatica.com

$cadena='Me llamo \'Luis Roca\' y mi web es minubeinformatica.com';

// El archivo se encuentra en c:\wamp64\www

$cadena='El archivo se encuentra en c:\\wamp64\\www';

?>

Esta sintaxis no se puede utilizar para representar caracteres de escape como avance de línea, salto de carro, tabuladores, etc…

**Comillas dobles**

La característica más importante de entrecomillar un string mediante comillas dobles es el hecho que los nombres de las variables son expandidas. Esto quiere decir que se mostrará como resultado el contenido de la variable en lugar del nombre. En consecuencia, esta sintaxis necesita más ciclos de CPU para poder interpretar el contenido.

Si un string está encerrado entre comillas dobles ("), PHP interpretará más sentencias de escape como caracteres especiales.

Códigos de escape

|  |  |
| --- | --- |
| **Código de escape** | **Significado** |
| \n | avance de línea |
| \r | retorno de carro |
| \t | Tabulador horizontal |
| \v | Tabulador vertical |
| \e | Escape |
| \f | Avance de página |
| \\ | Barra invertida |
| \$ | Signo del dólar |
| \” | Dobles comillas |

<?php

echo "Esto es una cadena sencilla";

// Resultado:  El nombre de mi web es: “MiNubeInformatica”

echo  "El  nombre de mi web es: \"MiNubeInformatica\"";

$var1 = 10;

$var2 = 5;

$suma = $var1+$var2;

//  Resultado: Las variables $var1 y $var2 contienen 10 y 5. Sumadas dan como  resultado: 15

echo "Las variables \$var1 y \$var2 contienen $var1 y $var2. Sumadas dan  como resultado: $suma";

**Sintaxis heredoc**

Una tercera forma de delimitar un string es mediante la sintaxis **heredoc**: **<<<** *(tres signos <)*. Después de este operador, se deberá de escribir un nombre o identificador y después una nueva línea. A continuación, va el propio string, y para cerrar la notación se pone el mismo identificador.

El identificador de cierre debe empezar en la primera columna de la nueva línea, es decir al principio de una nueva línea sin espacios a la izquierda.

<?php

$CadenaHeredoc = <<< HDOC

<p>Esto es un texto  largo encerrado entre los identificadores ’HDOC’</p><p>Su contenido  es equivalente a la utilización de las dobles comillas.</p>

HDOC;

echo $CadenaHeredoc;

$Nombre = "Luis";

$Comentario = <<< COMENT

<p>$Nombre: Para poder enviar este formulario es necesario  rellenar todos los campos marcados con un asterisco. </p>

COMENT;

echo $Comentario;

/\* Resultado:

Esto es un texto largo encerrado entre los  identificadores ’HDOC’

Su contenido es equivalente a la  utilización de las dobles comillas.

Luis: Para poder enviar este formulario es  necesario rellenar todos los campos marcados con un asterisco.

\*/

?>

Resulta práctico a la hora de asignar o mostrar grandes cantidades de texto en las que necesitamos representar códigos de escape o contenidos de variables.

**Obtener el tipo de dato de una variable**

Cuando deseemos conocer el tipo de dato de una variable utilizaremos la función ***gettype***:

string gettype($variable)

Devuelve un literal con el nombre del tipo al que pertenece la variable.

<?php

    echo gettype(100).'<BR/ >';

    echo gettype(100.56).'<BR/ >';

    echo gettype(TRUE).'<BR/  >';

    echo gettype("HOLA").'<BR/ >';

    $var=100;

    $var2='Hola Mundo';

    echo gettype($var).'<BR/ >';

    echo gettype($var2).'<BR/ >';

/\* Resultado:

integer

double

boolean

string

integer

string

\*/

?>

### **Forzado de tipos**

Es la técnica por medio de la cual forzamos, obligamos que el contenido de una variable sea o se convierta a un tipo de datos determinado.

Generalmente el forzado de tipos no será necesario ya que el propio PHP identifica y asigna los tipos según el contexto donde se utilice el valor.

El forzado de tipos en PHP funciona de la misma manera que en C:, donde el nombre del tipo deseado se escribe entre paréntesis antes de la variable que se quiera forzar.

Están permitidos los siguientes forzados de tipos:

* (***int****), (****integer****) -*forzado a integer
* *(****bool****), (****boolean****) -*forzado a boolean
* *(****float****), (****double****), (****real****) -*forzado a float
* *(****string****) -*forzado a string
* *(****array****) -*forzado a array
* *(****object****) -*forzado a object

<?php

    // $numero es un integer según el contexto  de la asignación

    $numero = 47;

    // $var es un boolean al forzar el tipo de  la variable $numero y se le asigna el valor true

    $var = (boolean) $numero;

    // $num es un string según el contexto de  la asignación

    $num = '100';

    echo '<p>'.$num.'<br/>Es de tipo: '.gettype($num).'</p>';

    // $num es un integer al forzar el valor

    $num = (int) '100';

    echo '<p>'.$num.'<br/>Es de tipo: '.gettype($num).'</p>';

/\* Resultado:

  100

  Es de tipo: string

  100

  Es de tipo: integer

\*/

?>

Ejemplos



**Constantes en PHP**

Una constante es un identificador para expresar un valor simple. Como el nombre sugiere, este valor no puede variar durante la ejecución del script.

Una constante es sensible a mayúsculas por defecto. Por convención, los identificadores es decir; los nombres de constantes suelen declararse en mayúsculas.

Un nombre de constante válido empieza con una letra o un carácter de subrayado, seguido por cualquier número de letras, números, o subrayados.

El alcance de una constante es global, Es decir, es posible acceder a ellas sin preocuparse por el ámbito de alcance.

**Sintaxis**

Las constantes se definen usando la función ***define*()** o con la palabra reservada *const* fuera de la definición de una clase a partir PHP 5.3.0.

define('DTO',0.10); // El identificador tiene que ir encerrado entre comillas

const DTO=0.10;     // El identificador NO tiene que ir encerrado entre comillas

* Una vez definida la constante, no puede ser modificada ni eliminada.
* Solo se puede definir como constantes valores escalares (boolean, integer, float y string).
* Para obtener el valor de una constante solo es necesario especificar su nombre. A diferencia de las variables, no se tiene que especificar el prefijo $.

<?php

    $USUARIO = 'Carmen';

    $precio = 120;

    define('USUARIO','Luis');

    define('DTO',0.10);

    const  ABIERTO=false;

    echo $USUARIO; # Carmen variable

    echo USUARIO;  # Luis constante

    echo ABIERTO;  # false

    $importe=$precio\*DTO; # realiza el cálculo  con un integer (variable) y un float (constante)

    // El  descuento aplicable es de: 12 €

    echo  "El descuento  aplicable es de: $importe €";

?>

**Las primeras instrucciones en PHP**

**Instrucción echo.** (**más usado que print)**  
Inserta texto dentro del documento HTML subyacente al código PHP

<?php

echo "salida en pantalla. ";

$a = 9;

echo $a; echo " El valor de a es $a.";

?>

Mostraría: salida en pantalla. 9 El valor de a es 9

Echo puede presentar de forma simultánea varias cadenas de caracteres y/o constantes y variables

* Basta con ponerlas una a continuación de otra utilizando una coma o punto como separador entre cada una de ellas.
* Cadenas entre comillas.
* Nombres de constantes nunca van entre comillas.

echo "Con echo los números no necesitan ir entre comillas: " ,3,"<br>";

En pruebas de velocidad/respuesta, echo es **más rápido** que print

**Instrucción print**  
Al igual que echo visualiza información en una página HTML  
print también permiten **concatenar** cadenas en una misma instrucción.

* sólo es posible usar el punto como elemento de unión
* Cadenas entre comillas.
* Nombres de constantes deben ir entre comillas.

print "En el caso de print si son necesarias: " . "7" . "<br>";

print ("incluso entre paréntesis necesitan las comillas:

"."45"."<br>");

print "Solo hay una excepción en el caso de  print. ";

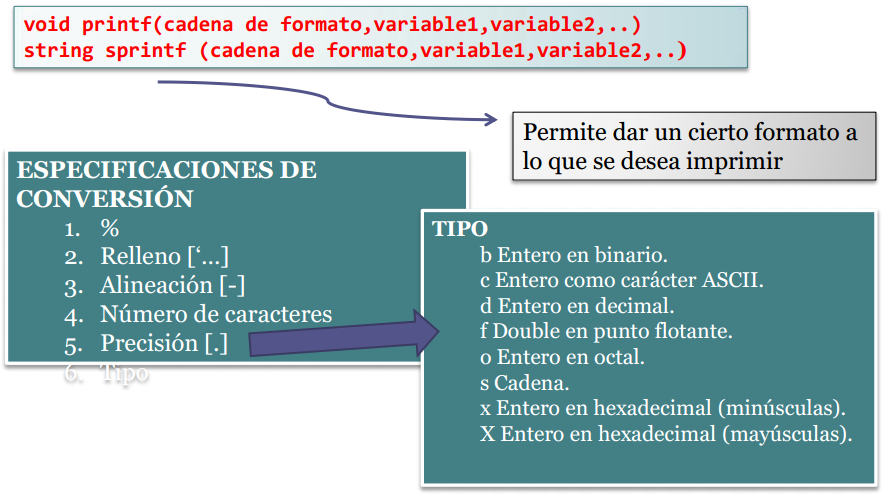
print "Si los números van en un print independiente no necesitan

comillas ";

print 23;

**Formatos de Presentación en PHP**

**Formatos en cadenas**



Ejemplo

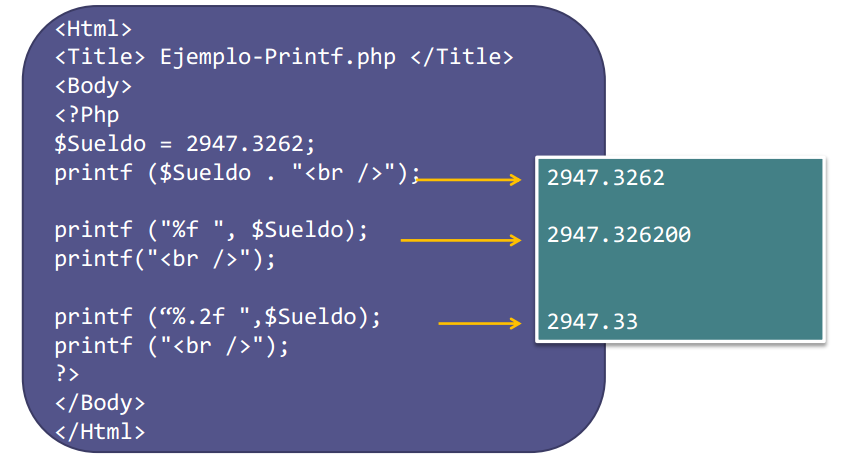
printf ("%02d/%02d/%04d", $dia, $mes, $año);

//Imprimiría dos dígitos para el día y el mes y 4 para el año)

printf("%05d",32);//presentaria 00032

printf("%'\*5d",32);//presentaria \*\*\*32

**Formatos en números**



**Operadores en PHP**

En PHP como en otros lenguajes disponemos de una serie de operadores comunes:

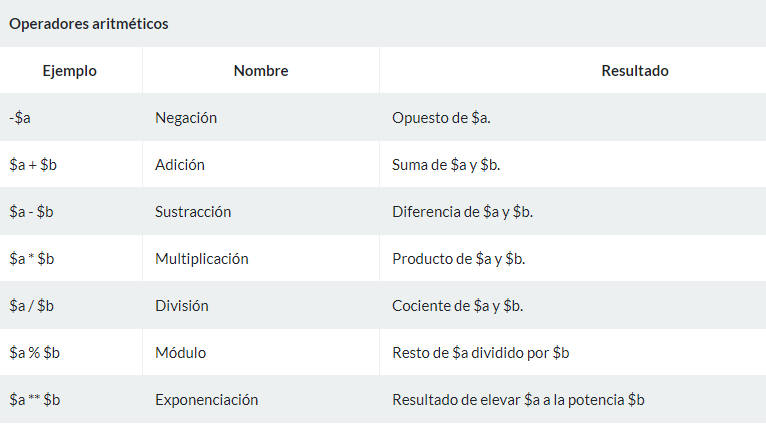
* **Aritméticos**
* **Asignación**
* **Comparación**
* **Lógicos**

Así como una serie de operadores especiales para:

* **Incrementos y Decrementos**
* Trabajar con Cadenas
* Trabaja a nivel de Bits
* Trabajar con Arrays *(los veremos el módulo dedicado al trabajo con arrays)*

**Operadores aritméticos**

Los operadores aritméticos permiten la realización de operaciones matemáticas con los valores (variables y constantes). Pueden ser utilizados con tipos de datos enteros (integer) o de coma flotante (float). Si ambos son enteros, el resultado es entero; si alguno de ellos es float, el resultado será float.



**Operadores de comparación o relacionales**

Se utilizan para establecer una relación entre dos valores. PHP compara estos valores entre si y esta comparación produce un resultado lógico (true o false).

Los operadores relacionales comparan valores del mismo tipo (numéricos o cadenas), entre ellos tienen el mismo nivel de prioridad en su evaluación, pero tienen menor prioridad que los aritméticos.



Otro operador condicional es el operador "*?:*" (o ternario), que funciona como en C y otros muchos lenguajes.

(expr1)**?** (expr2)**:**(expr3);

La expresión toma el valor expr2 si expr1 se evalúa a cierto, y expr3 si expr1 se evalúa a falso.

Equivale a una sentencia condicional simple if else expresada en una sola línea.

 // Teniendo las siguientes variables definidas:

  $a = 10;

  $c = 5;

  $Resultado=($a>$c)?  'Es Mayor':'Es Menor';

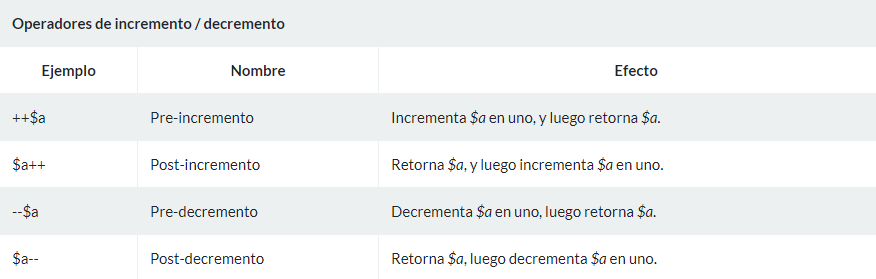
  // Dara como resultado Es Mayor porque $a es  mayor que $b

  echo $Resultado;

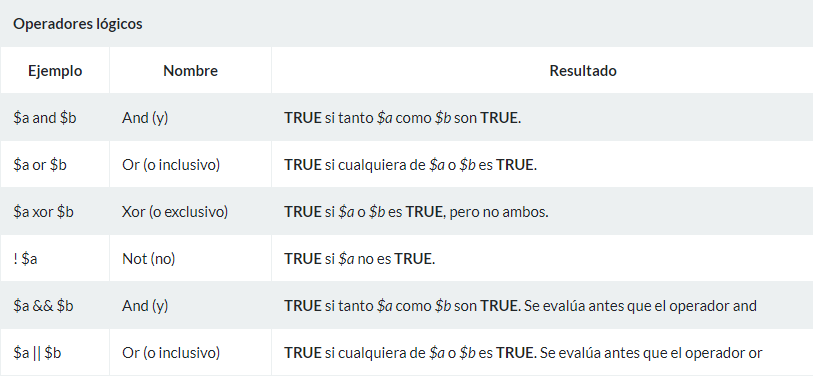
**Operadores de incremento y decremento**

PHP soporta los operadores de pre- y post- decremento e incremento al estilo de otros lenguajes como C.

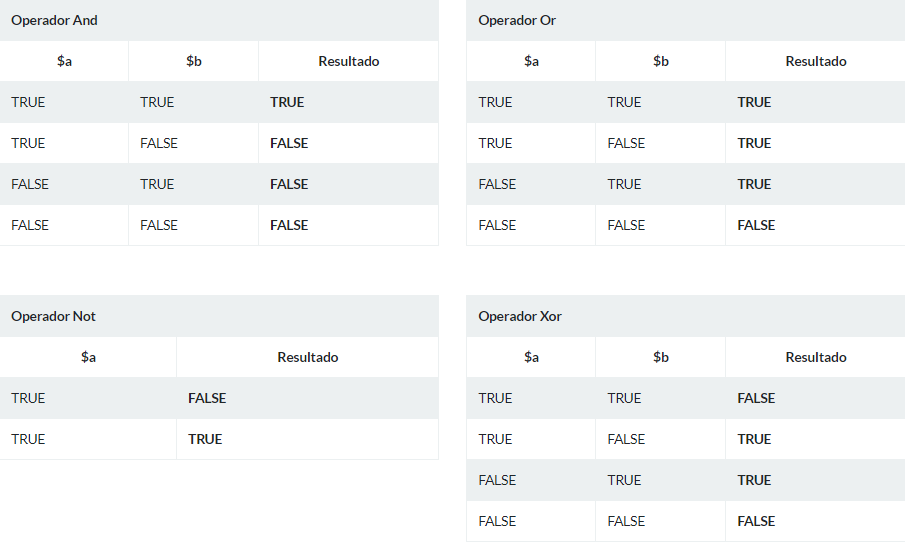
Los Operadores de incremento y decremento son operadores unarios que agregan o sustraen uno de sus operandos, respectivamente. El operador de incremento aumenta el valor de su operando en 1. El operando debe ser un tipo de dato numérico. Similarmente, el operador de decremento disminuye el valor de su operando en 1.



**Operadores lógicos**  
También llamados operadores booleanos, se utilizan para crear condiciones compuestas en una fórmula. Al igual que los operadores condicionales devuelven un valor lógico de verdadero o falso y determinar si la expresión completa se cumple en función de las tablas de verdad.



Las tablas de verdad de los operadores And, Or, Not y Xor se muestran en las tablas siguientes:



**Operador de concatenación**  
  
Se usa para unir diferentes valores (cadenas o números) con cadenas de caracteres, se representa por un punto (.)  
  
Ejemplo:

<?php

echo 'Mi color preferido es el ' . 'rojo';

// el resultado es: Mi color preferido es el rojo

$bebida = "leche";

// se puede usar para unir variables con textos

echo 'Mi bebida preferida es ' . $bebida;

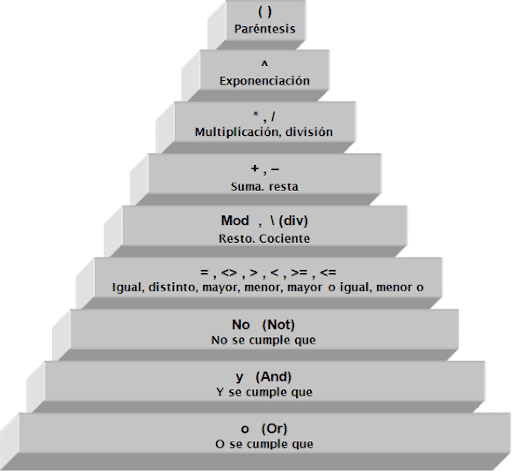
// el resultado es: Mi bebida preferida es leche

$numero = 137;

echo "Me gusta el numero " . $numero;

// el resultado es -> Me gusta el numero 137

?>

**Orden de precedencia de operadores**  
  


**Estructuras de control en PHP**

En PHP disponemos de estructuras condicionales e iterativas para el control del flujo del programa:

* **Condicionales**
  + Simples: if
  + Dobles: if else
  + Múltiples if elseif else, switch
* **Iterativas**
  + Mientras: while, do .. while
  + Para: for, foreach (lo veremos el módulo dedicado al trabajo con arrays)

**Condicional if**

La construcción *if* es una de las más importantes características de muchos lenguajes, incluido PHP. Permite la ejecución condicional de fragmentos de código. PHP caracteriza una estructura if que es similar a la de C,Java,…:

#### **Simples: if**

**if**(Expresión)

Sentencia;

**if**(Expresión) {

Sentencia;

Sentencia;

…

}

**Sintaxis de corchetes {}**

**if**(Expresión):

Sentencia;

Sentencia;

…

**endif**;

**Sintaxis alternativa de dos puntos:**

Expresión se evaluará a su valor booleano, si se evalúa a TRUE se ejecutará el bloque de sentencias.

Si queremos indicar más de una sentencia, tendremos que encerrarlas entre signos de {} o utilizar la sintaxis alternativa donde el corchete de apertura ({) se sustituye por dos puntos (**:**) y el corchete de cierre (}) por (**endif;**).

Esta sintaxis alternativa es admitida también por otras estructuras condicionales e iterativas.

<?php

   $a = 10;

   $b = 30;

   echo '<H2>CONDICIONAL SIMPLE</H2>',

        '<p><strong>VARIABLES:</strong><br/>',

        '$a = ',$a,'<br/>',

        '$b = ',$b,'</p>',

        '<p><strong>RESULTADO</strong><br/>';

// Se evaluarán las tres condicionales por  separado

   if($a > $b) echo '$a es mayor que $b</p>';

   if($b > $a) echo '$b es mayor que $a</p>';

   if($b === $a) echo '$a y $b son iguales</p>';

/\* RESULTADO

CONDICIONAL SIMPLE

VARIABLES:

$a = 10

$b = 30

RESULTADO

$b es mayor que $a\*/

?>

#### **Dobles: if else**

La sentencia *else*, extiende una sentencia if para ejecutar una sentencia diferente en caso que la expresión if original se evalúe como FALSE.

Ejecutará el bloque de sentencias del if si expresión es evaluada a TRUE y extenderá la ejecución de sentencias al else si la expresión se evalúa a FALSE.

**if**(Expresión)

Sentencia;

**else**

Sentencia;

**Sintaxis de corchetes {}**

**if**(Expresión) {

Sentencia;

Sentencia;

…

} **else** {

Sentencia;

Sentencia;

…

}

**Sintaxis alternativa de dos puntos:**

**if**(Expresión):

Sentencia;

Sentencia;

…

**else**:

Sentencia;

Sentencia;

…

**endif**;

La sintaxis alternativa también se aplica a else utilizando los dos puntos (:).

<?php

  $a = 100;

  $b = 100;

  echo '<H2>CONDICIONAL DOBLE</H2>',

       '<p><strong>VARIABLES:</strong><br/>',

       '$a = ',$a,'<br/>',

       '$b = ',$b,'</p>',

       '<p><strong>RESULTADO</strong><br/>';

//  Se evalúa la condicional doble y la  condicional simple por separado

  if($a > $b):

     echo '$a es mayor que $b<br />';

     echo 'La diferencia de $a - $b es: '.($a - $b).'</p>';

  else:

     echo '$b es mayor o igual que $a<br />';

     echo 'La diferencia de $b - $a es: '.($b - $a).'</p>';

  endif;

  if($b === $a)

     echo '$a y $b son iguales';

/\* RESULTADO

CONDICIONAL DOBLE

VARIABLES:

$a = 100

$b = 100

RESULTADO

$b es mayor o igual que $a

La diferencia de $b - $a es: 0

$a y $b son iguales\*/

?>

#### **Múltiples: if elseif else**

La sentencia *elseif* es una combinación de if y else. Pueden haber varios elseif dentro de la misma sentencia if. La sentencia elseif es ejecutada solamente si la expresión if precedente y cualquiera de las expresiones elseif precedentes son evaluadas como FALSE, y la expresión elseif actual se evalúa como TRUE.

**if**(Expresión)

Sentencia;

**elseif**(Expresión)

Sentencia;

**else**

Sentencia;

**Sintaxis de corchetes {}**

**if**(Expresión) {

Sentencia;

Sentencia;

…

} **elseif**(Expresión) {

Sentencia;

Sentencia;

…

} **else** {

Sentencia;

Sentencia;

…

}

**Sintaxis alternativa de dos puntos :**

**if**(Expresión):

Sentencia;

Sentencia;

…

**elseif**(Expresión):

Sentencia;

Sentencia;

…

**else**:

Sentencia;

Sentencia;

…

**endif**;

La sintaxis alternativa también se aplica a elseif utilizando los dos puntos (:).

<?php

  $a = 100;

  $b = 100;

  echo '<H2>CONDICIONAL MÚLTIPLE</H2>',

       '<p><strong>VARIABLES:</strong><br/>',

       '$a = ',$a,'<br/>',

       '$b = ',$b,'</p>',

       '<p><strong>RESULTADO</strong><br/>';

//  Se evalúa la condicional múltiple

  if($a > $b):

     echo '$a es mayor que $b<br />';

     echo 'La diferencia de $a - $b es: '.($a - $b).'</p>';

  elseif($b > $a):

     echo '$b es mayor que $a<br />';

     echo 'La diferencia de $b - $a es: '.($b - $a).'</p>';

  else:

     echo '$a y $b son iguales</p>';

  endif;

/\* RESULTADO

CONDICIONAL DOBLE

VARIABLES:

$a = 100

$b = 100

RESULTADO

$a y $b son iguales\*/

?>

**Condicional múltiple switch**

La sentencia ***switch*** es similar a una serie de sentencias IF en la misma expresión. En muchas ocasiones, se quiere comparar la misma variable (o expresión) con muchos valores diferentes, y ejecutar una parte de código distinta dependiendo de a qué valor es igual.

if(Expresión) {

Sentencia;

Sentencia;

…

} elseif(Expresión) {

Sentencia;

Sentencia;

…

} else {

Sentencia;

Sentencia;

…

}

**switch**(Expresión) {

**case** valor1:

Sentencia;

Sentencia;

…

**break**; Finaliza la ejecución de las sentencias del bloque case

**case** valor2:

Sentencia;

Sentencia;

…

**break**;

**default**:

Sentencia;

Sentencia;

…

}

<?php

  $forma\_pago=2;

  switch($forma\_pago) {

    case 0: $pago='Contado';

    break;

    case 1: $pago='Transferencia bancaria';

    break;

    case 2: $pago='Contra reembolso';

    break;

    default: $pago='No definida';

  }

  echo '<H2>CONDICIONAL MÚLTIPLE  switch</H2>',

       '<p><strong>VARIABLES:</strong><br />',

       '$forma\_pago = ',$forma\_pago,'</p>',

       '<p><strong>RESULTADO</strong><br />';

  echo "La forma de pago seleccionada es: $pago</p>";

/\* RESULTADO

CONDICIONAL MÚLTIPLE switch

VARIABLES:

$forma\_pago = 2

RESULTADO

La forma de pago seleccionada es: Contra reembolso\*/

?>

La sentencia switch ejecuta línea por línea (realmente, sentencia a sentencia). Al comienzo, no se ejecuta código. Sólo cuando se encuentra una sentencia case con un valor que coincide con el valor de la expresión switch PHP comienza a ejecutar las sentencias. PHP continúa ejecutando las sentencias hasta el final del bloque switch, o la primera vez que vea una sentencia break.

La condición que va dentro del switch(condición) puede ser también un **string**.

<?php

  $tamanyo=2;

  $unidad='G'; // Megabytes

  echo '<H2>CONDICIONAL MÚLTIPLE switch (sin rupturas)</H2>',

       '<p><strong>VARIABLES:</strong><br/>',

       '$tamanyo = ',$tamanyo,'<br />',

       '$unidad = ',$unidad,'<br />',

       '<p><strong>RESULTADO</strong><br/>';

// Pasar Gb, Mb o Kb a bytes

  switch($unidad) {

    case 'G':    $bytes=$tamanyo\*(2\*\*30);break;// pow(2, 30)

    case 'M':    $bytes=$tamanyo\*(2\*\*20);break;// pow(2, 20)

    case 'K':    $bytes=$tamanyo\*(2\*\*10);break;//pow(2, 10)

  }

  echo "$tamanyo $unidad son ".number\_format($bytes,0,',','.')." Bytes</p>";

/\* RESULTADO

CONDICIONAL MÚLTIPLE switch (sin  rupturas)

VARIABLES:

$tamanyo = 2

$unidad = M

$bytes = 2

RESULTADO

2 M son 2.097.152 Bytes\*/

En el script anterior he utilizado la función *number\_format* de PHP para mostrar los bytes en formato de millares con punto para los miles, coma para los decimales y con 0 decimales.

// Devuelve un string con la representación de un número con los millares agrupados.

**number\_format**(Numero,Decimales,SeparadorDecimal,SeparadorMiles)

La **lista de sentencias** de un case puede **también estar vacía**, lo cual simplemente pasa el control a la lista de sentencias del siguiente case.

**switch**(Expresión) {

**case** valor1:

**case** valor2:

**case** valor3:

Sentencia;

Sentencia;

**break**;

**case** valor4:

Sentencia;

Sentencia;

**break**;

**default**:

Sentencia;

}

**El primer bloque de sentencias se ejecutará con que se cumplan alguna de las tres primeras sentencias case (valor1 o valor2 o valor3).**

**Iterativas mientras: Bucle while**

Como en la mayoría de los lenguajes de programación la estructura iterativa que se ejecuta mientras se cumpla la expresión de condición es el bucle *while*, es decir que la expresión se evalúe a TRUE.

**while**(Expresión) {

Sentencia;

Sentencia;

**[break;]** Provoca la ruptura incondicional del bucle

Sentencia;

}

<?php

  $Contador=0;

  echo '<p>';

  while($Contador<=6) {

    echo "Contador con valor: $Contador</br>";

    $Contador++;

  }

  echo '</p>';

 /\* RESULTADO

Contador con valor: 0

Contador con valor: 1

Contador con valor: 2

Contador con valor: 3

Contador con valor: 4

Contador con valor: 5\*/

?>

**Iterativas mientras: Bucle do .. while**

Similar al bucle **while**, tenemos el bucle ***do..while*** donde la única diferencia es que la expresión de repetición se evalúa al final, por lo que como mínimo su bucle de instrucciones se ejecutarán una vez.

**do** {

Sentencia;

Sentencia;

**[break;]**

Sentencia;

} **while**(Expresión);

En cualquier parte del bloque de sentencias, podemos incluir la sentencia *break* para obligar a que finalice el bucle y que la ejecución continúe con la siguiente instrucción del código. Esta sentencia resulta útil cuando deseamos crear una ruptura en la ejecución a expensas de la Expresión que condiciona las iteraciones del bucle.

<?php

  $Contador=7;

  echo '<p>';

  do {

    echo "Contador con  valor: $Contador</br>";

  } while(++$Contador<=6);

  echo '</p>';

/\*RESULTADO

Obtenemos el mismo resultado que en el caso  del ejemplo anterior del bucle while\*/

**Iterativas para: Bucle for**

Como ocurre con el resto de lenguajes el bucle idóneo cuando conocemos el nº de iteraciones que se tienen que realizar es el bucle *for*. El bucle for es el más complejo en PHP y se comporta como su homólogo en C, Java…...

**for**(Inicialización; Condición; Incremento) {

Sentencia;

Sentencia;

**[break;]**

.....;

}

* La Inicialización se ejecuta incondicionalmente una vez al principio del bucle.
* Al comienzo de cada iteración, se evalúa Condición. Si se resuelve como TRUE, el bucle continúa y las sentencias anidadas se ejecutan. Si se evalúa como FALSE, la ejecución del bucle finaliza.
* Al final de cada iteración, se ejecuta el Incremento.

Cada una de las expresiones puede estar vacía, pero no todas. Si Condición está vacía el bucle debería iterar indefinidamente, es decir, generaría un bucle infinito que podría dejar el script o la página bloqueada.

Cuando utilizamos esta modalidad de sintaxis es más que recomendable crear una condicional entre las sentencias que se ejecutan, que pueda ejecutar la instrucción break para salir del bucle.

El bucle **for** también **admite la sintaxis alternativa de dos puntos (:)** con la palabra (***endfor;***) como cierre del bucle.

<?php

  $desde=10;

  $hasta=20;

  $resultado=0;

// Obtener la suma de los números del 10 al  20 por medio de un bucle for

  for($i=$desde;$i<=$hasta;$i++) {

    $resultado+=$i;

  }

$linea = <<< PARRAFO

<p>La suma de los  números comprendidos entre el $desde y el $hasta es: $resultado</p>

PARRAFO;

echo $linea;

/\* RESULTADO

La suma de los números comprendidos entre el 10 y el 20 es: 165\*/