



Universidad Gerardo Barrios

Facultad de Ciencia y Tecnología

Fecha:
12/02/18

Introducción a la programación con PHP Programación Computacional IV

Objetivos:

- Conocer las características básicas, operadores, delimitadores, y tipos de datos en PHP
- Entender la sintaxis básica de PHP
- Crear scripts utilizando PHP

Introducción Teórica.

Utilización de PHP

PHP se puede utilizar para:

- ✓ Desarrollar aplicaciones web del lado del servidor. Este es el campo de uso más tradicional de PHP y el que le ha significado una infinidad de seguidores y adeptos.
- ✓ Realizar scripts que se ejecuten desde la línea de comandos. Estos scripts se pueden ejecutar sin la necesidad de un servidor web ni de un navegador.
- ✓ Escribir aplicaciones de interfaz gráfica. Este es el campo más nuevo en el que PHP ha hecho incursión, para utilizarlo es necesario incluir la extensión PHP-GTK que no viene incluida en la distribución principal.

Tipo de aplicaciones que se pueden realizar con PHP

PHP se puede utilizar para crear aplicaciones de:

- ✓ Comercio electrónico.
- ✓ Educación a distancia.
- ✓ Foros de discusión.
- ✓ Sistemas de Gestión de Contenidos.
- ✓ Blogs.
- ✓ Exámenes en línea.
- ✓ Aplicaciones de correo electrónico.

Requerimientos para desarrollar aplicaciones con PHP

- ✓ Para poder realizar scripts de PHP en el lado del servidor es necesario tener instalado:
- ✓ El intérprete de PHP (CGI o módulo).
- ✓ Un servidor web (Apache Web Server es ideal para trabajar con PHP; sin embargo, hoy en día se puede instalar fácilmente en Internet Information Server también).

- ✓ Un navegador web (Internet Explorer, Chrome, Firefox, Safari y Opera son los más difundidos).
- ✓ Un gestor de bases de datos (MySQL es la mejor opción de base de datos para trabajar con PHP).
- ✓ Un editor de texto, de preferencia especializado en sintaxis de PHP.

Sintaxis básica.

El lenguaje PHP es bastante sencillo en cuanto a su sintaxis. Alguien con experiencia en programación con lenguaje C o Perl, no debería tener ningún problema de adaptación. Sin embargo, hay que decir que es más complicado que simplemente escribir código HTML.

Delimitadores de bloque de código PHP.

Existen cuatro diferentes tipos de delimitadores de código PHP. Estos son:

- a) Delimitadores estilo XML:

```
<?php
    //instrucciones php
?>
```
- b) Delimitador estilo script:

```
<script language="php">
    //instrucciones php
</script>
```
- c) Delimitador corto:

```
<?
    //instrucciones php
?>
```
- d) Delimitador estilo ASP:

```
<%
    //instrucciones php
%>
```

Hay que mencionar que solamente los primeros dos tipos de delimitadores están disponibles de forma automática, sin necesidad de realizar configuración alguna en el archivo php.ini. Los últimos dos delimitadores deben ser habilitados en dicho archivo de configuración. Este archivo está en la carpeta de instalación de PHP y deberá modificarlo si necesita utilizar etiquetas cortas o las de estilo ASP y reiniciar los servicios para que funcionen.

```
220 ; this short cut has been a feature for such a long time, it's currently still
221 ; supported for backwards compatibility, but we recommend you don't use them.
222 ; Default Value: On
223 ; Development Value: Off
224 ; Production Value: Off
225 ; http://php.net/short-open-tag
226 short_open_tag = On
227
228 ; Allow ASP-style <% %> tags. Cambiar el valor por defecto Off a On
229 ; http://php.net/asp-tags
230 asp_tags = On
231
```

Delimitador de sentencias.

El terminador o delimitador de sentencias en PHP es el punto y coma (;), el mismo que se utiliza en C/C++.

Por lo tanto, cuando desee terminar una sentencia para iniciar otra debe utilizar el punto y coma. Existen dos casos en los que se puede omitir el punto y coma. Uno es cuando sólo existe una instrucción PHP en el script y el otro es cuando la instrucción sea la última línea del bloque de código PHP. Se sugiere que siempre utilice el punto y coma al final de cualquier instrucción, incluso en los casos antes mencionados para evitar cometer errores por tratar de recordar estos casos especiales. Es más fácil recordar que siempre debe utilizarse que recordar cuando puede omitirse.

Comentarios.

PHP utiliza los estilos de comentarios del lenguaje C/C++ y del shell de la interfaz de comandos de Unix y Linux. Estos son el comentario de una sola línea `//` y el comentario de bloque `/* ... */`, también utilizados en el lenguaje C. Además, se puede utilizar el comentario de una sola línea utilizado en el shell de Unix y Linux, `#`

Veamos algunos ejemplos:

1) Comentario de una sola línea `//` estilo C++:

```
<?php
    $valor = 5.2;
    //Este es un comentario de una sola línea estilo C
    echo "El valor es: " . $valor;
?>
```

2) Comentario de una sola línea `#` estilo *shell*:

```
<?php
    $nombre = "Julio";
    #Este es un comentario de una sola línea estilo shell
    echo "Su nombre es: " . $nombre;
?>
```

3) Comentario de bloque (varias líneas) `/* ... */`:

```
<?php
    $precio = 25;
    $total = $precio * $POST['descuento'];
    /* En este caso estamos utilizando un comentario
       de bloque, esto quiere decir que todo este texto
       que está leyendo es comentario y que por tanto será
       ignorado por el intérprete de PHP
    */
    echo "El total es " . $total;
?>
```

Salida a pantalla.

En PHP existen dos formas principales de mandar a imprimir texto en la ventana de un navegador. La primera es utilizando la instrucción `echo` y la segunda es utilizar la función `printf()`.

echo

La sentencia `echo` es fácil de utilizar, su sintaxis es la siguiente:

echo cadena_de_texto;

Donde, `cadena_de_texto` puede ser un literal de cadena delimitado por comillas que pueden ser simples (') o dobles ("). La diferencia más importante es que entre comillas dobles se interpretan

las variables y ciertos caracteres especiales, incluyendo etiquetas HTML. En cambio, con comillas simples sólo se interpretan la comilla simple y la barra invertida, por tanto, para evitar que sean interpretados estos caracteres deberá hacer uso de secuencias de escape.

printf()

La función printf() es mucho más versátil que la instrucción echo. Con esta función se pueden mandar a imprimir varios tipos de variables a la vez, utilizando códigos de formato, que indican cómo debe ser formateada la variable que se desea mostrar a la salida. La sintaxis es la siguiente:

printf("cadena_de_texto [%s %d %f %c]", \$cadena, \$entero, \$flotante, \$caracter);

Donde, cadena_de_texto es una cadena delimitada por comillas dobles que puede incluir ciertos códigos de formato opcionales. Si se desea imprimir el contenido de variables deben especificarse códigos de formato para formatear la salida adecuadamente. Los códigos de formato más importantes son:

Elemento	Tipo de variable
%s	Cadena de caracteres.
%d	Número sin decimales.
%f	Número con decimales.
%c	Carácter ASCII.
Aunque existen otros tipos, estos son los más importantes.	

Variables

Las variables en PHP se definen anteponiendo el símbolo dólar (\$) al nombre de la variable. A diferencia de otros lenguajes, PHP posee una gran flexibilidad a la hora de operar con variables. En efecto, cuando definimos una variable asignándole un valor, PHP le atribuye un tipo. Si por ejemplo definimos una variable entre comillas, la variable será considerada de tipo cadena:

\$variable = "5"; //esto es una cadena

Ahora bien, si en el script se realiza una operación matemática con esta variable, no se lanzará ningún mensaje de error sino que la variable cadena será convertida automáticamente en numérica al incluirla en una expresión que involucre un operador matemático:

```
<?php
$cadena = "5"; //esto es una cadena
$entero = 3; //esto es un entero
echo $cadena + $entero
?>
```

Este script dará como resultado "8". La variable cadena con valor de "5", ha sido asimilada como entero (aunque su tipo sigue siendo cadena) para poder realizar la operación matemática. Del mismo modo, podemos operar entre variables tipo entero y real. No debemos preocuparnos de nada, PHP se encarga durante la ejecución de interpretar el tipo de variable necesario para el buen funcionamiento del programa.

Sin embargo, si hay que tener cuidado en no cambiar mayúsculas por minúsculas en el identificador de la variable, ya que, en este sentido, PHP es sensible. Conviene por lo tanto, trabajar ya sea siempre en mayúsculas, o siempre en minúsculas para evitar este tipo de malentendidos a veces muy difíciles de localizar. Durante las prácticas de laboratorio se convendrá que los nombres de variables se digitarán siempre en minúsculas y las constantes en mayúsculas.

Variables predefinidas de PHP

Estas son variables que están disponibles para cualquier script PHP que se ejecute en un servidor web con el módulo PHP instalado. Algunas de ellas pueden ser muy útiles para obtener información del cliente o del mismo servidor. El comportamiento y disponibilidad de estas variables depende del servidor sobre el que se estén ejecutando, específicamente de su configuración, de la versión de PHP y de otros factores.

A partir de la versión 4.1.0 PHP dispone de un conjunto de matrices predefinidas que contienen variables del servidor web, variables del entorno y variables de entrada del usuario. Estas matrices son automáticamente globales o, también llamadas, superglobales. Entre estas matrices se pueden mencionar:

\$GLOBALS, es una matriz asociativa que contiene una referencia a cada variable disponible en el ámbito de las variables globales del script. La forma de acceder a las variables es utilizando el nombre de las variables globales entre comillas (dobles o simples) como índice de la matriz.

\$_SERVER, es una matriz asociativa que contiene información como cabeceras, rutas y ubicaciones de scripts. Las entradas de esta matriz se crean en el servidor web. No hay garantía alguna de que el servidor vaya a proveer estos valores realmente. Dentro de las entradas que pueden encontrarse en esta matriz se pueden mencionar:

'PHP_SELF': que proporciona el nombre del archivo de script ejecutándose actualmente, relativo a la raíz del documento.

'SERVER_ADDR': que proporciona la dirección IP del servidor web en el que se está ejecutando el script actual.

'SERVER_NAME': que proporciona el nombre del servidor web bajo el que está siendo ejecutado el *script* actual. Si se ejecuta en un host virtual devolverá el nombre definido para tal host.

'SCRIPT_FILENAME': que proporciona la ruta absoluta del nombre del *script* que está siendo ejecutado actualmente.

Existen muchas más entradas para la matriz **\$_SERVER** que pueden consultar en el manual oficial de PHP.

\$_GET, que es una matriz asociativa que contiene variables proporcionadas al script por medio del método HTTP GET. Esto significa que cuando define que las variables de un formulario serán pasadas por el método GET, es en esta matriz donde se almacenarán sus valores de acuerdo al nombre que asignó al control de formulario HTML.

\$_POST, es una matriz asociativa que contiene las variables pasadas al script a través de método HTTP POST. Al igual que **\$_GET**, cuando se define que el método de paso de valores provenientes de un formulario será POST, la matriz contendrá dichos valores y para tener acceso a ellos deberá usar como llave de la matriz el nombre que le dio al control de formulario.

\$_COOKIE, es una matriz asociativa que contiene las variables pasadas al script mediante cookies HTTP.

\$_SESSION, es una matriz asociativa que contiene las variables de sesión disponibles en el script actual.

\$_REQUEST, es una matriz asociativa que contiene cualquiera de los contenidos de las matrices superglobales **\$_GET**, **\$_POST** y **\$_COOKIE**.

Existen otras matrices superglobales que se dejan como investigación al estudiante.

Ejemplo:

```
<?php
    $cad = "El script que est&acute;s ejecutando: " . $_SERVER['PHP_SELF'] . " ";
    $cad .= "En el servidor: " . $_SERVER["SERVER_NAME"] . "<br>";
    echo "<h3>" . $cad . "</h3>";

?>
```

Constantes.

Sintaxis

Se puede definir una constante utilizando la función `define()`. Una vez definida, no se puede modificar ni eliminar.

Sólo se puede definir como constantes valores escalares (boolean, integer, float y string).

Para obtener el valor de una constante únicamente es necesario especificar su nombre. A diferencia de las variables, no se tiene que especificar el prefijo `$`. También se puede utilizar la función `constant()`, para obtener el valor de una constante, en el caso de que queramos expresarla de forma dinámica. Usa la función `get_defined_constants()` para obtener una lista de todas las constantes definidas.

Nota: Las constantes y las variables (globales) se encuentran en un espacio de nombres distinto. Esto implica que por ejemplo `TRUE` y `$TRUE` son diferentes.

Cuando se utiliza una constante que todavía no ha sido definida, PHP asume que se está refiriendo al nombre de la constante en sí. Se lanzará un aviso si esto sucede. Usa la función `define()` para comprobar la existencia de dicha constante.

Estas son las diferencias entre constantes y variables:

- ✓ Las constantes no son precedidas por un símbolo de dólar (`$`)
- ✓ Las constantes solo pueden ser definidas usando la función `define()`, nunca por simple asignación
- ✓ Las constantes pueden ser definidas y accedidas sin tener en cuenta las reglas de alcance del ámbito.
- ✓ Las constantes no pueden ser redefinidas o eliminadas después de establecerse; y
- ✓ Las constantes solo pueden albergar valores escalares

Ejemplo. Definiendo constantes

```
<?php
    define("CONSTANT", "LIS.");
    echo CONSTANT; // muestra el mensaje "LIS."
    echo "<br>", Constant; // muestra "Constant".

?>
```

Tratamiento de cadenas

Existen tres formas de asignar cadenas a una variable en PHP, que son: delimitándolas entre comillas dobles, entre comillas simples y usando delimitadores tipo Perl (HereDoc):

Para asignar a una variable una cadena usando comillas dobles debe hacer una declaración de este tipo:

```
$cadena = "Esta es la información de mi variable";
```

Para mostrar el valor de una variable pueden usarse la instrucción echo o la función print():

```
echo $cadena //obtendríamos: Esta es la información de mi variable;
```

```
echo "Esta es la información de mi variable"; //daría el mismo resultado
```

Algo importante con respecto a los delimitadores de comillas dobles es que interpretan variables si son colocadas dentro de comillas dobles. En terminología de programación, se dice que son interpoladas. Por ejemplo:

```
<?php  
$cadena1 = "Aplicaciones";  
$cadena2 = " Prácticas de Software II";  
$cadena3 = "Materia: $cadena1 $cadena2";  
echo $cadena3 //El resultado es: Aplicaciones Prácticas de Software II  
?>
```

También podemos introducir variables dentro de nuestra cadena lo cual nos puede ayudar mucho en el desarrollo de nuestros scripts. Lo que veremos no es el nombre, sino el valor que almacena la variable:

```
<?php  
$a=55;  
$mensaje = "Tengo $a años";  
echo $mensaje //El resultado es: "Tengo 55 años"  
?>
```

Puede ser que en lugar de imprimir el valor de la variable, lo que se desee es imprimir el nombre mismo de la variable. Al colocarlo entre comillas dobles, como se hizo en el ejemplo anterior, no sería posible. La única solución sería encerrarlo entre comillas simples o utilizar código o secuencias de escape en la cadena delimitada por comillas dobles. Como se muestra a continuación:

```
<?php  
$a=55;  
$mensaje = "Tengo \$a años";  
echo $mensaje //El resultado es: "Tengo 55 años"  
?>
```

Si se usan comillas simples debe realizar una instrucción como la siguiente:

```
$cadena = 'Coloque acá su cadena';
```

Para poder mostrar una comilla simple dentro de una cadena delimitada por comillas simples debe utilizarse una secuencia de escape colocando el símbolo de barra invertida antes de ella. Así:

```
$cadena='Todos lo llamaban \'el mesías\'';
```

Los únicos caracteres que deben escaparse cuando se encierran entre comillas simples son la comilla simple y la barra invertida.

Otra forma de delimitar cadenas es mediante el uso de la sintaxis heredoc ("<<<"). Debe indicarse un identificador después de la secuencia <<<, luego la cadena, y luego el mismo identificador para cerrar la cita.

```
<?php
$frase = <<<ANILLOS
"Hay muchos vivos que merecen la muerte y hay
muchos muertos que merecen la vida,
¿quién eres tú para impartir ese derecho?" - Gandalf.
ANILLOS;
echo $frase;
?>
```

El identificador de cierre debe comenzar en la primera columna de la línea. Asimismo, el identificador usado debe seguir las mismas reglas que cualquier otra etiqueta en PHP: debe contener solo caracteres alfanuméricos y de subrayado, y debe iniciar con un caracter no-dígito o de subrayado.

Unos aspectos importantes a considerar acerca de esta sintaxis son:

- ✓ La línea con el identificador de cierre no puede contener otros caracteres, excepto quizás por un punto-y coma (;). Esto quiere decir en especial que el identificador no debe usar sangría, y no debe haber espaciostabuladores antes o después del punto-y-coma.
- ✓ Es importante también notar que el primer caracter antes del identificador de cierre debe ser un salto de línea, tal y como lo defina su sistema operativo. Esto quiere decir \r en Macintosh, por ejemplo.

Códigos o secuencias de escape

En PHP, al igual que en otros lenguajes existen ciertos caracteres que generan problemas cuando se encuentran dentro de cadenas, debido a que dentro del lenguaje se interpretan de alguna forma especial. Para poder visualizar correctamente dichos caracteres en una operación de salida se utilizan secuencias de escape que involucran dichos caracteres especiales. La siguiente tabla muestra varios de estos caracteres especiales con su correspondiente secuencia de escape:

Secuencia de escape	Significado
\b	Espacio hacia atrás (<i>backspace</i>)
\f	Cambio de página (<i>form feed</i>)
\n	Cambio de línea (<i>line feed</i>)
\r	Retorno de carro (<i>carriage return</i>)
\t	Tabulación horizontal
\\	Barra inversa (<i>backslash</i>)
\'	Comilla simple

Operadores

Los operadores son símbolos especiales que se utilizan en los lenguajes de programación para poder realizar operaciones con las expresiones. Como lo que se obtiene en dichas operaciones es un valor, el resultado también será una expresión.

Los operadores se pueden clasificar como: unarios (que operan sobre un único valor o expresión), binarios (que operan sobre dos valores o expresiones) y ternarios (que consta de tres valores o expresiones)

En PHP existen diversos tipos de operadores y se pueden clasificar de la siguiente manera: aritméticos, lógicos, de cadena, de ejecución, de comparación, de asignación, de incremento/decremento, etc.

Operadores aritméticos

Son los operadores que permiten realizar operaciones numéricas sobre las variables y expresiones. Se muestran en la siguiente tabla:

Símbolo	Nombre	Ejemplo	Resultado
+	Adición	$\$a + \b	Suma de $\$a$ y $\$b$.
-	Substracción	$\$a - \b	Diferencia entre $\$a$ y $\$b$.
*	Multiplicación	$\$a * \b	Producto de $\$a$ y $\$b$.
/	División	$\$a / \b	Cociente de $\$a$ y $\$b$.
%	Módulo	$\$a \% \b	Resto de $\$a$ dividido por $\$b$.

Debe tener en cuenta que el operador de división “/” devuelve un número con punto flotante en todos los casos, incluso cuando los dos operandos son enteros.

Operadores lógicos

Los operadores lógicos se utilizan para realizar comparaciones entre expresiones. Pueden combinarse para formar expresiones de comparación más complejas. Los operadores lógicos de PHP se muestran en la siguiente tabla:

Símbolo	Nombre	Ejemplo	Resultado
and	Y	$\$a \text{ and } \b	TRUE si tanto $\$a$ como $\$b$ son TRUE.
or	O	$\$a \text{ or } \b	TRUE si cualquiera de $\$a$ o $\$b$ es TRUE.
xor	O exclusivo (Xor)	$\$a \text{ xor } \b	TRUE si $\$a$ o $\$b$ es TRUE, pero no ambos.
!	No	$! \$a$	TRUE si $\$a$ no es TRUE.
&&	Y	$\$a \&\& \b	TRUE si tanto $\$a$ como $\$b$ son TRUE.
	O	$\$a \b	TRUE si cualquiera de $\$a$ o $\$b$ es TRUE.

Operadores de cadena

Estos operadores se utilizan en combinación con variables o expresiones de cadena. Los dos operadores válidos en PHP para operar con cadenas son el operador de concatenación que se representa con un símbolo de punto (.) y el operador de asignación sobre concatenación, representado por un punto seguido de un símbolo de igual que (.=). Vea la siguiente tabla:

Símbolo	Nombre	Ejemplo	Resultado
.	Concatenación	<code>\$var1="Hola "; \$var2="mundo"; \$saludo=\$var1 . \$var2;</code>	Se imprimirá en pantalla: Hola mundo

Símbolo	Nombre	Ejemplo	Resultado
.=	Concatenación y asignación	<code>\$var1="Hola mundo "; \$var1.="cruel y perverso";</code>	Se imprimirá en pantalla: Hola mundo cruel y perverso

Operadores de comparación

Se utilizan para verificación de condiciones en ciertas expresiones, sobre todo en expresiones condicionales.

En la siguiente tabla se muestra la lista completa de ellos:

Símbolo	Nombre	Ejemplo	Resultado
==	Igual	<code>\$a == \$b</code>	TRUE si \$a es igual a \$b.
===	Idéntico	<code>\$a === \$b</code>	TRUE si \$a es igual a \$b, y son del mismo tipo. (PHP 4 o superior)
!=	Diferente	<code>\$a != \$b</code>	TRUE si \$a no es igual a \$b.
<>	Diferente	<code>\$a <> \$b</code>	TRUE si \$a no es igual a \$b.
!==	No idénticos	<code>\$a !== \$b</code>	TRUE si \$a no es igual a \$b, o si no son del mismo tipo. (PHP 4 o superior)
<	Menor que	<code>\$a < \$b</code>	TRUE si \$a es estrictamente menor que \$b.
>	Mayor que	<code>\$a > \$b</code>	TRUE si \$a es estrictamente mayor que \$b.
<=	Menor o igual que	<code>\$a <= \$b</code>	TRUE si \$a es menor o igual que \$b.
>=	Mayor o igual que	<code>\$a >= \$b</code>	TRUE si \$a es mayor o igual que \$b.

Operadores de asignación.

Existe un solo operador básico de asignación, que se lee “se asigna a” y no “es igual a”, como podría parecer.

Además de este operador de asignación existen operadores combinados con operador de asignación que también se mostrarán en esta parte. Vea la siguiente tabla:

Símbolo	Nombre	Ejemplo
=	Se asigna a	<code>\$a = "Hola"</code>
.=	Concatenación y asignación	<code>\$a .= " mundo"</code>
+=	Adición y asignación	<code>\$a += \$b</code>
-=	Sustracción y asignación	<code>\$a -= \$b</code>
*=	Multiplicación y asignación	<code>\$a *= \$b</code>
/=	División y asignación	<code>\$a /= \$b</code>
%=	Módulo y asignación	<code>\$a %= \$b</code>

Operadores de ejecución.

PHP soporta un operador de ejecución: las comillas invertidas (`). ¡Note que no se trata de comillas sencillas!

PHP intentará ejecutar el contenido entre las comillas como si se tratara de un comando del intérprete de comandos; su salida será devuelta (es decir, no será simplemente volcada como salida; puede ser asignada a una variable).

Ejemplo:

```
<?php
    $salida = `ls - al`;
    echo "<pre>$salida</pre>";
?>
```

Operadores de incremento/decremento

Estos operadores son, en realidad, una mejora a los operadores aritméticos de adición y sustracción, para el caso muy particular en que uno de los operandos sea la unidad. Existen variantes para este operador dependiendo si primero se hace la asignación y luego el incremento/decremento o viceversa. Veamos la siguiente tabla:

Símbolo	Nombre	Finalidad
++\$var	Pre-incremento	Incrementa el valor de \$var en 1 y luego retorna el nuevo valor de \$var
\$var++	Post-incremento	Retorna primero el valor actual de \$var y después incrementa este valor en 1
--\$var	Pre-decremento	Decrementa el valor de \$var en 1 y luego retorna el nuevo valor de \$var
\$var--	Post-decremento	Retorna primero el valor actual de \$var y después decrementa este valor en 1

Ejercicios:

Indicaciones:

- ✓ Cree una carpeta con el nombre practica4 dentro de htdocs.
- ✓ Guarde los ejercicios realizados en la carpeta creada en el punto uno, al finalizar la clase deberá subir el proyecto al repositorio de gitLab creado en las practicas anteriores, asegúrese de agregar al docente a su proyecto para poder realizar la revisión.

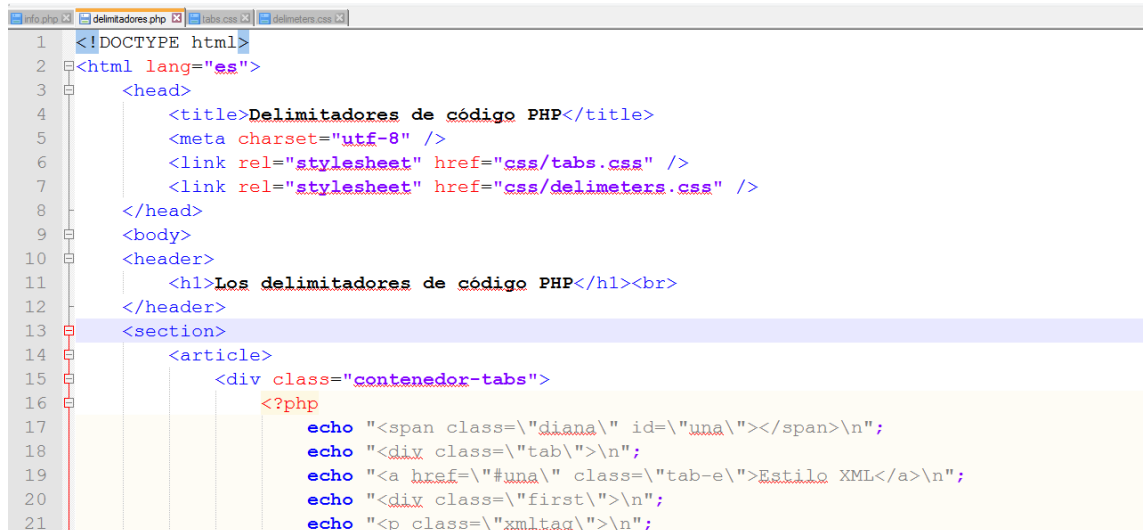
1. Escriba el siguiente código y ejecute en el navegador.



```
1 <html>
2   <head>
3     <title>PHP-Info</title>
4   </head>
5   <body>
6     <?php
7       echo phpinfo();
8     ?>
9   </body>
10  </html>
```

¿Qué es lo que hace la funcion phpinfo()?

2. Delimitadores.



```
1 <!DOCTYPE html>
2 <html lang="es">
3   <head>
4     <title>Delimitadores de código PHP</title>
5     <meta charset="utf-8" />
6     <link rel="stylesheet" href="css/tabs.css" />
7     <link rel="stylesheet" href="css/delimiters.css" />
8   </head>
9   <body>
10    <header>
11      <h1>Los delimitadores de código PHP</h1><br>
12    </header>
13    <section>
14      <article>
15        <div class="contenedor-tabs">
16          <?php
17            echo "<span class='diana' id='una'></span>\n";
18            echo "<div class='tab'>\n";
19            echo "<a href='#una' class='tab-e'>Estilo XML</a>\n";
20            echo "<div class='first'>\n";
21            echo "<p class='xmltag'>\n";
```

```

22     echo "Este texto está escrito en PHP, utilizando las etiquetas más ";
23     echo "usuales y recomendadas para delimitar el código PHP, que son: ";
24     echo "<?php ... ?>.<br>\n";
25     echo "</p>\n";
26     echo "</div>\n";
27     echo "</div>\n";
28     ?>
29
30     <script language="php">
31         echo "<span class=\"diana\" id=\"dos\"></span>\n";
32         echo "<div class=\"tab\">\n";
33         echo "<a href=\"#dos\" class=\"tab-e\">Script</a>\n";
34         echo "<div>\n";
35         echo "<p class=\"htmltag\">\n";
36         echo "A pesar de que estas líneas están escritas dentro de un script PHP,
37         echo "Están enmarcadas dentro de etiquetas HTML: ";
38         echo "<?script&gt; ... <?/script&gt;";
39         echo "</p>\n";
40         echo "</div>\n";
41         echo "</div>\n";
42     </script>
43
44     echo "<span class=\"diana\" id=\"tres\"></span>\n";
45     echo "<div class=\"tab\">\n";
46     echo "<a href=\"#tres\" class=\"tab-e\">Etiquetas cortas</a>\n";
47     echo "<div>\n";
48     echo "<p class=\"shorttag\">\n";
49     echo "Este texto también está escrito con PHP, utilizando las etiquetas ";
50     echo "cortas, <br>\n que son: <?> ... ?>";
51     echo "</p>\n";
52     echo "</div>\n";
53     echo "</div>\n";
54     ?>
55     <?
56     echo "<span class=\"diana\" id=\"cuatro\"></span>\n";
57     echo "<div class=\"tab\">\n";
58     echo "<a href=\"#cuatro\" class=\"tab-e\">Estilo ASP</a>\n";
59     echo "<div>\n";
60     echo "<p class=\"asptag\">\n";
61     echo "Este texto está escrito en PHP, como los dos ejemplos anteriores. ";
62     echo "Sin embargo, se ha delimitado con etiquetas estilo ASP: ";
63     echo "<?;% ... ?>.<br>\n";
64     echo "</p>\n";
65     echo "</div>\n";
66     echo "</div>\n";
67     </div>
68     </article>
69     </section>
70     </body>
71 </html>

```

NOTA:

Si al ejecutar el script en el navegador visualiza lo siguiente, significa que el archivo de configuración del PHP (php.ini) no tiene habilitadas las etiquetas cortas ni las etiquetas estilo ASP. Debe habilitarlas según se indica en la introducción teórica de esta guía de práctica.

Los delimitadores de código PHP

Estilo XML **Script**

\n"; echo "

Este texto está escrito en PHP, utilizando las etiquetas más usuales y recomendadas para delimitar el código PHP, que son: <?php ... ?>.

\n"; echo "\n"; echo "\n"; ?> <% echo "n"; echo "\n"; echo "Estilo ASP"; echo "\n"; echo "\n"; echo "Este texto está escrito en PHP, como los dos ejemplos anteriores. "; echo "Sin embargo, se ha delimitado con etiquetas estilo ASP: "; echo "<% ... %>.\n"; echo "\n"; echo "\n"; echo "\n"; %>

Al realizar la modificación, reiniciar los servicios y recargar la página debería ver la página web de la siguiente forma:

Los delimitadores de código PHP

Estilo XML **Script** **Etiquetas** **Estilo ASP**

Este texto está escrito en PHP, utilizando las etiquetas más usuales y recomendadas para delimitar el código PHP, que son: <?php ... ?>.

3. Si se desea conocer el tipo y valor de una expresión, se puede usar la función `var_dump()`. Para obtener una representación legible para humanos del tipo de una variable para propósitos de depuración, se puede usar la función `gettype()`. Para comprobar si una variable es de un cierto tipo, no se debe usar `gettype()`, si no las funciones `is_tipo`.

```
1 <?php
2     $un_bool = TRUE;           //valor booleano
3     $un_str = "Programacion";  //una cadena
4     $un_str2 = 'Programacion'; //una cadena
5     $un_int = 12;              //un entero
6
7
8     echo gettype($un_bool);    //imprime:booleano
9     echo gettype($un_str);     //imprime: string
10
11     //si este valor es un entero, incrementarlo en cuatro
12     if(is_int($un_int)){
13         $un_int +=4;
14     }
15
16     //si $bool es una cadena, imprimirla
17     // (no imprime nada)
18     if(is_string($un_bool)){
19         echo "Cadena: $un_bool";
20     }
21 ?>
```

4. Operaciones con php

```
1 <html>
2 <head>
3 <title>Ejemplo de operaciones</title>
4 </head>
5 <body>
6 <h1>Ejemplo de operaciones aritmeticas en PHP</h1>
7 <?php
8     $a = 8;
9     $b = 3;
10    echo $a + $b, "<br>";
11    echo $a - $b, "<br>";
12    echo $a * $b, "<br>";
13    echo $a / $b, "<br>";
14    $a++ ;
15    echo $a, "<br>";
16    $b--;
17    echo $b, "<br>";
18    ?>
19 </body>
20 </html>
```

5. Comparación

```
1 <html>
2 <head>
3 <title>Ejemplo de operadores de Comparacion</title>
4 </head>
5 <body>
6 <h1>Ejemplo de operaciones comparacion en PHP</h1>
7 <?php
8     $a = 8;
9     $b = 3;
10    $c = 3;
11    echo $a == $b, "<br>";
12    echo $a != $b, "<br>";
13    echo $a < $b, "<br>";
14    echo $a > $b, "<br>";
15    echo $a >= $c, "<br>";
16    echo $a <= $c, "<br>";
17    ?>
18 </body>
19 </html>
20
```

Anote el significado de las operaciones de comparación:

PREGUNTA	RESPUESTA
==	
!=	
<	
>	
>=	
<=	

6. Operadores Lógicos

```
1 <html>
2   <head>
3     <title>Ejemplo de operadores Logicos</title>
4   </head>
5   <body>
6     <h1>Ejemplo de operaciones logicas en PHP</h1>
7     <?php
8         $a = 8;
9         $b = 3;
10        $c = 3;
11        echo ($a == $b) && ($c > $b), "<br>";
12        echo ($a == $b) || ($b == $c), "<br>";
13        echo !($b <= $c)b, "<br>";
14    ?>
15 </body>
16 </html>
```

Anote el significado de las operaciones de comparación

PREGUNTA	RESPUESTA
&&	

7. Resuelva las expresiones presentadas en el siguiente cuadro, escriba la respuesta (True or False):

\$i = 9;
\$f = 33.5;
\$c = 'X';

EXPRESIÓN	RESULTADO
(\$i >= 6) && (\$c == 'X')	
(\$i >= 6) (\$c == 12)	
(\$f < 11) && (\$i > 100)	
(\$c != 'P') ((\$i + \$f) <= 10)	
\$i + \$f <= 10	
\$i >= 6 && \$c == 'X'	
\$c != 'P' \$i + \$f <= 10	