

	<p style="text-align: center;">UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN</p>
<p style="text-align: center;">CICLO 01-2022</p>	<p style="text-align: center;">GUIA DE LABORATORIO N° 2</p>
	<p>Nombre de la práctica: Introducción a la Programación web con PHP Lugar de ejecución: Laboratorio de Informática Tiempo estimado: 2 horas Materia: Lenguajes Interpretados en el Servidor</p>

I. OBJETIVOS

En esta guía de práctica se pretende:

1. Conseguir que los estudiantes tengan un primer contacto con la programación y sintaxis de PHP.
2. Desarrollar las habilidades necesarias para crear secuencias de comando con PHP.
3. Lograr el dominio de los tipos de datos del lenguaje PHP.
4. Dominar la forma en que se crearán scripts PHP y del proceso para visualizarlos a través de un navegador.

II. INTRODUCCIÓN TEÓRICA

Definición de PHP

PHP es, hoy en día, un lenguaje de programación diseñado para desarrollar páginas web dinámicas que son ejecutadas en un servidor web y, luego, devueltas en formato HTML al navegador del usuario que las solicita. No obstante, en sus principios PHP fue más bien un lenguaje de secuencias de comando, guiones o scripts del lado del servidor desarrollado por Rasmus Lerdorf en 1995 con el propósito de crear un conjunto de scripts que le permitieran contabilizar el número de visitantes que accedían a su hoja de vida que había puesto en línea para conseguir empleo o contratos de trabajo. Debido a que PHP estaba diseñado para trabajar en un ambiente web y que este se podía insertar directamente dentro del código (X)HTML propició que se volviera muy popular para procesar datos ingresados a través de formularios. El significado actual de PHP es Hypertext PreProcessor y a partir de la versión 4.0 es considerado todo un lenguaje de programación y una plataforma sólida para el desarrollo de aplicaciones web del lado del servidor.

Evolución de PHP

Después de alcanzar su objetivo Rasmus Lerdorf libera su código, que fue desarrollado en lenguaje Perl primero, y luego, en lenguaje C. Para ese entonces se hacía alusión al lenguaje como PHP/FI y su significado era Personal Home Page/Form Interpreter, siendo la versión PHP/FI 2.0 liberada en 1997. En este año el proyecto de PHP pasó de ser un proyecto personal de Rasmus Lerdorf a ser un proyecto desarrollado en conjunto. En ese entonces PHP era usado en unos 50,000 dominios en Internet, lo que representaba solamente el 1% del total de dominios en Internet.

La versión PHP 3.0 fue liberada en 1998 y desarrollada principalmente por dos programadores israelíes del Instituto Tecnológico de Israel (Technion) de nombres Andi Gutmans y Zeev Surasky. Una de las mejores características de esta versión era su fácil extensibilidad reflejada en la incorporación de nuevos módulos y soporte para trabajar con una gran cantidad de bases de datos. En su apogeo la versión 3.0 de PHP era utilizada en un 10% de servidores en Internet.

El siguiente paso en la evolución de PHP fue la creación de un nuevo núcleo, escrito desde cero para mejorar su rendimiento y convertirse por primera vez en un lenguaje orientado a objetos. El nuevo motor fue denominado Zend (acrónimo de los nombres de sus nuevos desarrolladores **Zeev** y **Andy**). Esta nueva versión conocida como PHP 4.0 fue liberada en mayo del 2000. Las nuevas prestaciones de esta versión fueron el soporte para la mayoría de servidores web, el manejo de sesiones HTTP de forma nativa, facilidades para programar utilizando orientación a objetos (aunque muchos insisten en que esta versión no es totalmente orientado a objetos), compatibilidad con las expresiones regulares de Perl, encriptación, manejo de formas más seguras para el control de las entradas de usuario.

El 13 de julio del 2007 se anunció la suspensión del soporte para la versión 4 de PHP, no obstante, se siguen liberando mejoras para solventar fallos críticos a nivel de seguridad, es así que se lanzaron posterior a esa fecha las versiones 4.4.8 el 13 de enero del 2008 y posteriormente, la versión 4.4.9 el 7 de agosto del mismo año.

La versión 5 de PHP fue lanzada el 13 de julio del 2004, utilizando el motor Zend Engine 2.0 que entre otras cosas incluyo:

- Mejor soporte para la programación orientada a objetos.
- Mejoras en el rendimiento.
- Mejor soporte para MySQL.
- Mejor soporte a XML.
- Soporte nativo para SQLite.
- Soporte integrado para SOAP.
- Iteradores de datos.
- Manejo de excepciones.
- Mejoras para la implementación con Oracle.

NOTA: Puede consultar sobre la evolución de PHP en forma detallada en el sitio oficial de PHP en <http://www.php.net> o en el sitio de wikipedia <http://www.wikipedia.org>.

Mejoras en la sintaxis de PHP 7.2

El nuevo PHP 7.2 es una importante actualización de este lenguaje de programación que va a permitir a los programadores escribir mejor código y crear mejores aplicaciones web gracias al gran número de cambios y mejoras, tanto en rendimiento como en seguridad, que se han implementado con esta actualización.

La primera de las novedades que van a llegar con esta nueva versión de PHP se va a centrar en el argumento de declaración de tipo (type hints), que ahora va a poder ser considerado con un objeto de tipo de dato, lo cual nos permitirá declarar un objeto genérico como argumento o método, mejorando el paso de información entre ellos.

Otra de las novedades de esta nueva versión es que ahora las funciones correctamente declaradas nos indicarán el tipo de variable esperada en caso de no haberlo declarado en la nueva. Además, a partir de ahora, los desarrolladores podrán omitir un tipo de una subclase sin romper el código.

La seguridad también forma parte de PHP 7.2

Esta nueva versión de PHP se ha centrado, además, en la seguridad, siendo el primer lenguaje de programación en incluir librerías criptográficas modernas por defecto sin tener que requerir de APIs externas.

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 del Wamp para que funcionen.

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
?>
```

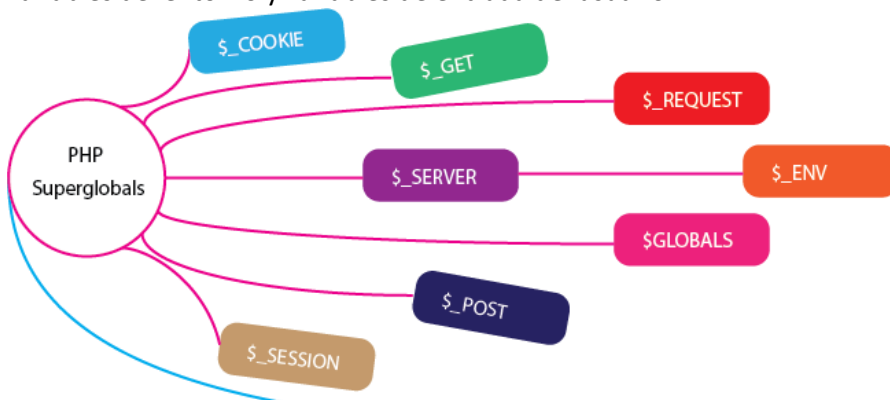
Este *script* dará como resultado "8". La variable cadena con valor de "5", ha sido asimilada como entero (aunque su tipo sigue siendo cadena) para poder realizar la operación matemática. Del mismo modo, podemos operar entre variables tipo entero y real. No debemos preocuparnos de nada, PHP se encarga 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&aacute;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 sólo 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 = "Lenguajes";
$cadena2 = " Interpretados en el Servidor";
$cadena3 = "Materia: $cadena1 $cadena2";
echo $cadena3    //El resultado es: Lenguajes Interpretados en el Servidor
?>
```

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:

```
<?
$a=22;
$mensaje = "Tengo $a años";
echo $mensaje //El resultado es: "Tengo 22 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:

```
<?
    $a=22;
    $mensaje = "Tengo \$a años";
    echo $mensaje //El resultado es: "Tengo 22 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.

```
<?
    $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 espacios o tabuladores 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
\"	Comilla doble
\\$	Carácter dólar (\$)

\###	Carácter ASCII octal (#: 0-7)
\x##	Carácter ASCII hexadecimal (#: 0-F)

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.
*	Multipliación	\$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 and \$b	TRUE si tanto \$a como \$b son TRUE .
or	O	\$a or \$b	TRUE si cualquiera de \$a o \$b es TRUE .
xor	O exclusivo (Xor)	\$a 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	\$var1 = "Hola "; \$var2 = "mundo"; \$saludo = \$var1 . \$var2;	Se imprimirá en pantalla: Hola mundo
.=	Concatenación y asignación	\$var1 = "Hola mundo "; \$var1.="cruel y perverso";	Se imprimirá en pantalla: Hola mundo cruel y perverso.

NOTA: Observe que no debe olvidar colocar espacios en blanco al terminar o al comenzar la subcadena para que no queden palabras unidas.

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	\$a == \$b	TRUE si \$a es igual a \$b.
===	Idéntico	\$a === \$b	TRUE si \$a es igual a \$b, y son del mismo tipo. (PHP 4 o superior)
!=	Diferente	\$a != \$b	TRUE si \$a no es igual a \$b.
<>	Diferente	\$a <> \$b	TRUE si \$a no es igual a \$b.
!==	No idénticos	\$a !== \$b	TRUE si \$a no es igual a \$b, o si no son del mismo tipo. (PHP 4 o superior)
<	Menor que	\$a < \$b	TRUE si \$a es estrictamente menor que \$b.
>	Mayor que	\$a > \$b	TRUE si \$a es estrictamente mayor que \$b.
<=	Menor o igual que	\$a <= \$b	TRUE si \$a es menor o igual que \$b.
>=	Mayor o igual que	\$a >= \$b	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	\$a = “Hola”
.=	Concatenación y asignación	\$a .= “ mundo”
+=	Adición y asignación	\$a += \$b
-=	Sustracción y asignación	\$a -= \$b
*=	Multipliación y asignación	\$a *= \$b
/=	División y asignación	\$a /= \$b
%=	Módulo y asignación	\$a %= \$b

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

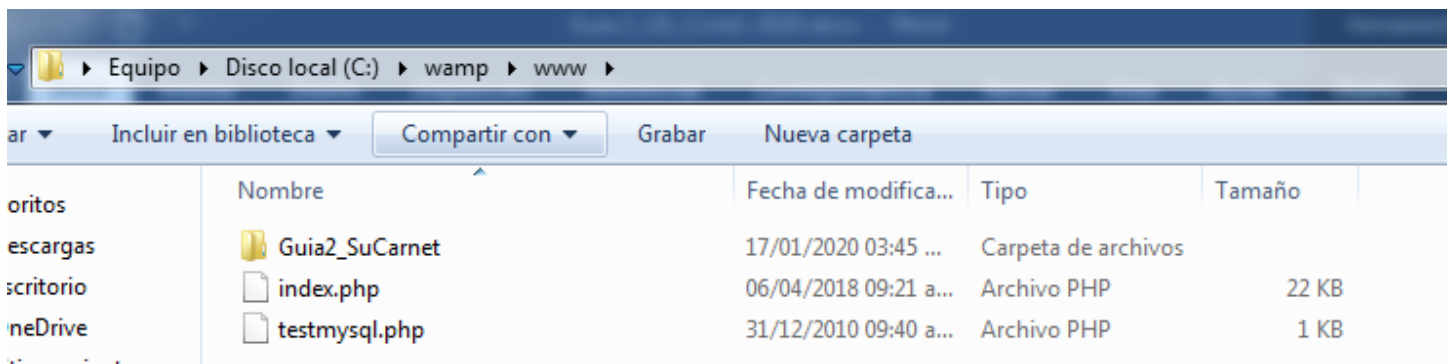
III. MATERIALES Y EQUIPO

Para la realización de la guía de práctica se requerirá lo siguiente:

No.	Requerimiento	Cantidad
1	Guía de práctica #1: Introducción a la programación web con PHP	1
2	Computadora con WampServer y Sublime Text 3 o superior instalado	1
3	Memoria USB o disco flexible	1

IV. PROCEDIMIENTO

1. Para realizar la primera guía de práctica debe estar consciente que para poder ver las páginas con código PHP, primero debe echar a andar el programa que se utilizará como servidor web. Para ello ejecute el WampServer, que incluye el servicio necesario para poner en funcionamiento el servidor web Apache.
2. Los archivos creados con el editor Sublime Text o PHP Designer 2007 deben guardarse en una ubicación especial, conocida como carpeta web, que es donde se almacenan todas las páginas que serán gestionadas por el servidor web. Esta carpeta en la instalación por defecto del WampServer es www. Búsquela en la carpeta wamp, dentro de ella estará ubicada la carpeta www.



3. Por cuestión de orden deberá crear una carpeta con el nombre Guia2_SuCarnet dentro de esta ira creando una carpeta para cada ejercicio y almacenar dentro de ella cada recurso, el cual debe descargar de la página de la Universidad.

Ejercicio #1:

1. Utilice SublimeText 3 o el PHP Designer 2007 para digitar el siguiente código que mostrará cómo se inserta código PHP dentro de un documento web realizado con lenguaje HTML:

Primer script: partehtmlphp.php

```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4     <title>Scripts con (X)HTML y PHP</title>
5     <meta charset="utf-8" />
6     <meta name="viewport" content="width=device-width,user-scalable=no,initial-
7 scale=1.0,maximum-scale=1.0,minimum-scale=1.0" />
8     <!-- Definiendo una hoja de estilo local -->
9     <link rel="stylesheet" media="screen" href="css/htmlphp.css">
10    <script src="js/modernizr.custom.lis.js"></script>
11    <script src="js/prefixfree.min.js"></script>
12 </head>
```

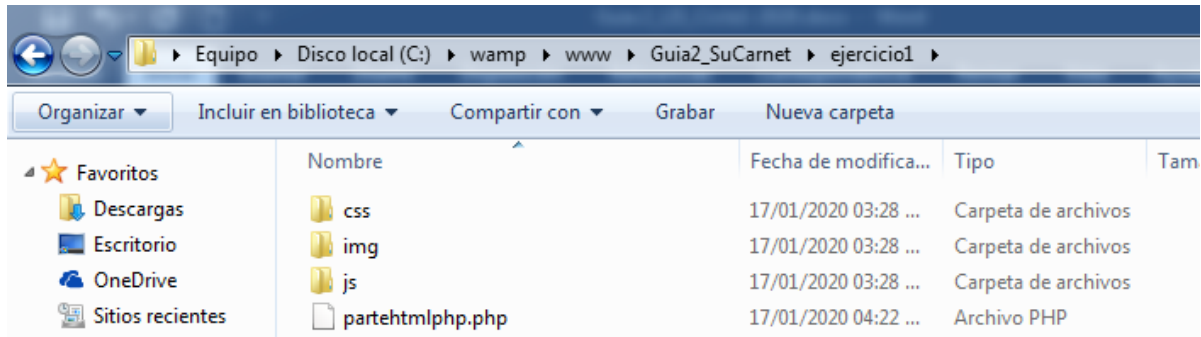
```
13 <body>
14 <div class="wrap">
15 <header>
16     <h1>Demostración de HTML y PHP</h1>
17 </header>
18 <!-- Parte de la página web generada con HTML -->
19 <section class="main">
20     <article id="html">
21         <div class="text">
22             <p>
23                 &lt;!DOCTYPE html&gt;<br />
24                 &lt;html&gt;<br />
25                 ...
26             </p>
27             <p>Parte de HTML normal.</p>
28             <p>
29                 &lt;/html&gt;
30             </p>
31         </div>
32         <div class="img">
33             
34         </div>
35     </article>
36 <?php
37     //Parte de la página generada con lenguaje PHP
38     echo "<article id=\"php\">\n";
39     echo "<div class=\"text\">";
40     echo "<p>&lt;?php<br />";
41     echo "...<br />";
42     echo "Parte con PHP.</p>";
43     echo "<p>?&gt;</p>";
44     echo "</div>";
45     echo "<div class=\"img\">";
46     echo "<img src=\"img/php.png\" alt=\"Logo de HTML5\" />";
47     echo "</div>";
48     echo "\n</article>\n";
49 ?>
```

```

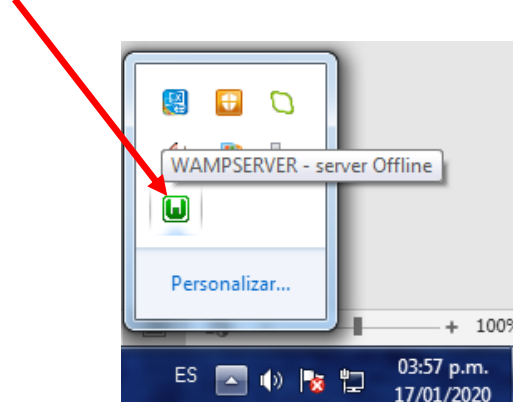
50 </section>
51 </body>
52 <!-- Fin del div class="wrap" -->
53 </div>
54 </html>

```

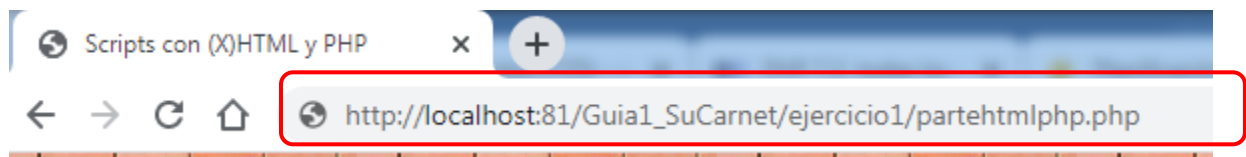
- Guarde el primer archivo con el nombre **partehtmlphp.php** en la carpeta predeterminada para los documentos web de su servidor web Apache. En una instalación predeterminada del **WampServer** debería ser: C:\wamp\www. Si usted modificó esta carpeta debe asegurarse de que sea en la carpeta correcta de los documentos web.



- Para visualizar el resultado en el navegador verifique que esté iniciado el servidor web Apache. Si lo está deberá observar en el área de estado de la barra de tareas de Windows (donde aparece la hora) el icono del **WampServer** en funcionamiento:



- Por último, para ver el resultado de los scripts php, diríjense con el navegador a <http://localhost/tuarchivo.php> (donde «tuarchivo.php» es el archivo php que queremos correr.), en el caso de ejemplo de la guía sería de la siguiente forma:



Nota: De forma predeterminada, al ejecutar la aplicación WAMP en un servidor Apache, el puerto host local se establece en el **puerto 80**. Sin embargo, puede ajustar esta configuración en su servidor mediante la edición del archivo "**http.conf**" y colocar un nuevo número de puerto. Al cambiar el puerto local de **WAMP**, estará instruyéndole a su servidor Apache para que envíe comunicaciones desde una ubicación diferente, así como se muestra en la imagen del ejemplo.

- Observe la página PHP que acaba de cargarse en el navegador y verifique que el resultado es el esperado. Es recomendable que verifique en todos los navegadores disponibles: Chrome, Firefox, Internet Explorer, Safari y Opera



NOTA: De ahora en adelante solamente se le proporcionará el código HTML combinado con PHP que debe digitar, para volver a cargar una página en el navegador ya sabe cuáles deben ser los pasos a seguir. Lo mismo será con cualquier otro archivo adicional como hoja de estilo o secuencia de comando de JavaScript, etc.

Ejercicio #2: El siguiente ejemplo muestra cómo opera PHP la declaración de las variables, En este ejemplo hemos definido tres variables, \$a, \$b y \$c y con la instrucción echo hemos impreso el valor que contenían, insertando un salto de línea entre ellas.

Primer script: uso_variable.php

```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <title>Ejemplo de PHP</title>
5   <style>
6     body {
7       font-family:"Arial Black";
8     }
9   </style>
10 </head>
11 <body>
12 <?php
13 //declaracion de variables
14   $a = 1;
15   $b = 3.34;
16   $c = "Hola Mundo";
17   echo $a,"<br>",$b,"<br>",$c;
18 ?>
19 </body>
20 </html>
```

El resultado en el navegador sería el siguiente:

1
3.34
Hola Mundo

Ejercicio #3: El siguiente ejemplo muestra cómo opera PHP los distintos tipos de operadores aritméticos, Los operadores de PHP son muy parecidos a los de C y JavaScript, si usted conoce estos lenguajes le resultaran familiares y fáciles de reconocer.

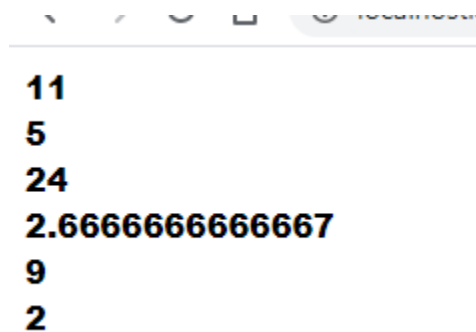
Primer script: opera_aritmeticos.php

```

1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4 <title>Ejemplo de PHP - Operadores aritmeticos</title>
5 <style>
6   body {
7     font-family:"Arial Black";
8   }
9 </style>
10 </head>
11 <body>
12 <?php
13   $a = 8;
14   $b = 3;
15   echo $a + $b, "<br>";
16   echo $a - $b, "<br>";
17   echo $a * $b, "<br>";
18   echo $a / $b, "<br>";
19   $a++;
20   echo $a, "<br>";
21   $b--;
22   echo $b, "<br>";
23 ?>
24 </body>
25 </html>

```

El resultado en el navegador sería el siguiente:



11
5
24
2.6666666666667
9
2

Ejercicio #4: El siguiente ejemplo muestra cómo opera PHP los distintos tipos de operadores de comparación, Los operadores de comparación son usados para comparar valores y así poder tomar decisiones.

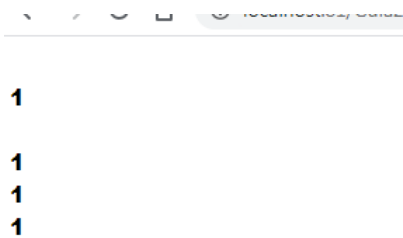
Primer script: opera_comparacion.php


```

1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4 <title>Ejemplo de PHP - Operadores de comparacion</title>
5 <style>
6   body {
7     font-family:"Arial Black";
8   }
9 </style>
10 </head>
11 <body>
12 <?php
13   $a = 8;
14   $b = 3;
15   $c = 3;
16   echo $a == $b, "<br>";
17   echo $a != $b, "<br>";
18   echo $a < $b, "<br>";
19   echo $a > $b, "<br>";
20   echo $a >= $c, "<br>";
21   echo $b <= $c, "<br>";
22 ?>
23 </body>
24 </html>

```

El resultado en el navegador sería el siguiente:



```

1 
1 
1 
1 
1 

```

Ejercicio #5: El siguiente ejemplo muestra cómo opera PHP los distintos tipos de operadores lógicos, Los operadores lógicos son usados para evaluar varias comparaciones, combinando los posibles valores de estas.

Primer script: opera_logicos.php

```

1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4 <title>Ejemplo de PHP - Operadores logicos</title>
5 <style>
6   body {
7     font-family:"Arial Black";
8   }
9 </style>
10 </head>
11 <body>
12 <?php
13   $a = 8;
14   $b = 3;
15   $c = 3;
16   echo ($a == $b) && ($c > $b), "<br>";
17   echo ($a == $b) || ($b == $c), "<br>";
18   echo !($b <= $c), "<br>";
19 ?>
20 </body>
21 </html>

```

Ejercicio #6: El siguiente ejemplo muestra cómo opera PHP los distintos tipos de datos que maneja (enteros, flotantes, cadenas y booleanos) y el uso de constantes ocupando variables locales definidas directamente en el código.

Primer archivo: tiposdatos.php

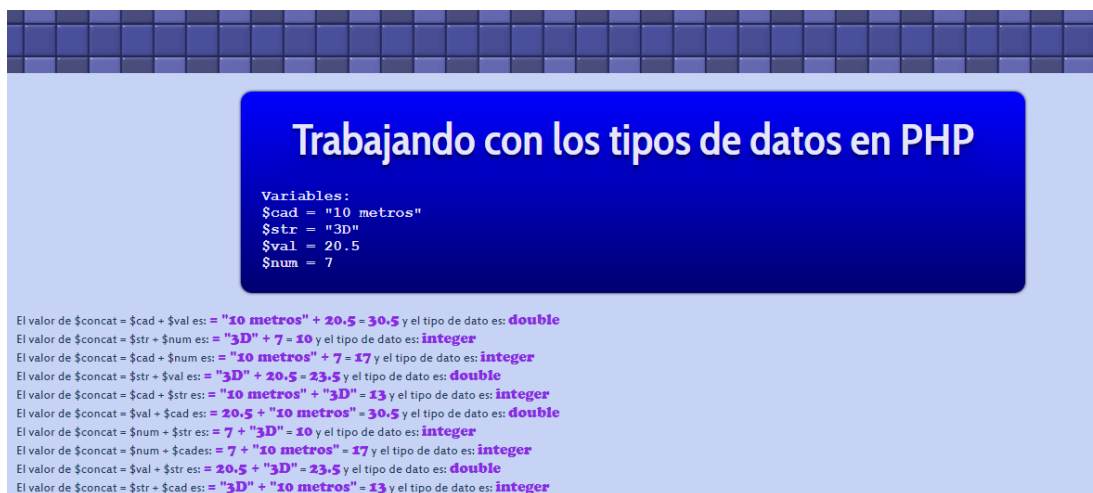
```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4     <title>Conversión de cadenas</title>
5     <meta charset="utf-8" />
6     <link rel="stylesheet"
7 href="http://fonts.googleapis.com/css?family=Cabin+Condensed:600" />
8     <link rel="stylesheet" media="screen" href="css/conversion.css" />
9     <!--[if lt IE 9]>
10         <script src="//html5shiv.googlecode.com/svn/trunk/html5.js"></script>
11     <![endif]-->
12 </head>
13 <body>
14 <?php
15     //Definiendo los valores a utilizar para los cálculos
16     define("SALTO", "\n");
17     $cad = "10 metros";
18     $str = "3D";
19     $val = 20.5;
20     $num = 7;
21
22     //Construyendo el header
23     echo "<header>\n";
24     echo "\t<h1>Trabajando con los tipos de datos en PHP</h1>\n";
25     echo "\t<p id=\"datos\">\n";
26     echo "\t\tVariables:<br />\n";
27     echo "\t\t\$cad = \"\$cad\"<br />\n";
28     echo "\t\t\$str = \"\$str\"<br />\n";
29     echo "\t\t\$val = \$val <br />\n";
30     echo "\t\t\$num = $num <br />\n";
31     echo "\t</p>\n";
32     echo "</header>\n";
33
34     //Primera operación
35     $concat = $cad + $val;
36     echo '<section>';
37     echo "\n";
38     echo '<article>';
39     echo '<p>El valor de $concat = $cad + $val es: <span> ' . $cad . ' ' + ' ' . $val . '</span> = <span>';
40     echo $concat . '</span> y el tipo de dato es: <span>' . gettype($concat) . "</span></p>" . SALTO;
41     echo '</article>';
42     echo "\n";
43
44     //Segunda operación
45     $concat = $str + $num;
46     echo '<article>';
47     echo '<p>El valor de $concat = $str + $num es: <span> ' . $str . ' ' + ' ' . $num . '</span> = <span>';
48     echo $concat . '</span> y el tipo de dato es: <span>' . gettype($concat) . "</span></p>" . SALTO;
49     echo '</article>';
50     echo "\n";
51
52     //Tercera operación
53     $concat = $cad + $num;
54     echo '<article>';
55     echo '<p>El valor de $concat = $cad + $num es: <span> ' . $cad . ' ' + ' ' . $num . '</span> = <span>';
56     echo $concat . '</span> y el tipo de dato es: <span>' . gettype($concat) . "</span></p>" . SALTO;
57     echo '</article>';
58     echo "\n";
59
60     //Cuarta operación
61     $concat = $str + $val;
62     echo '<article>';
63     echo '<p>El valor de $concat = $str + $val es: <span> ' . $str . ' ' + ' ' . $val . '</span> = <span>';
64     echo $concat . '</span> y el tipo de dato es: <span>' . gettype($concat) . "</span></p>" . SALTO;
65     echo '</article>';
66     echo "\n";
67
68     //Quinta operación
69     $concat = $cad + $str;
70     echo '<article>';
```

```

71 echo '<p>El valor de $concat = $cad + $str es: <span> = "' . $cad . ' + "' . $str . '"/>= <span>';
72 echo $concat . '</span> y el tipo de dato es: <span>' . gettype($concat) . "</span></p>" . SALTO;
73 echo '</article>';
74 echo "\n";
75
76 //Sexta operación
77 $concat = $val + $cad;
78 echo '<article>';
79 echo '<p>El valor de $concat = $val + $cad es: <span> = ' . $val . ' + "' . $cad . '"/>= <span>';
80 echo $concat . '</span> y el tipo de dato es: <span>' . gettype($concat) . "</span></p>" . SALTO;
81 echo '</article>';
82 echo "\n";
83
84 //Séptima operación
85 $concat = $num + $str;
86 echo '<article>';
87 echo '<p>El valor de $concat = $num + $str es: <span> = ' . $num . ' + "' . $str . '"/>= <span>';
88 echo $concat . '</span> y el tipo de dato es: <span>' . gettype($concat) . "</span></p>" . SALTO;
89 echo '</article>';
90 echo "\n";
91
92 //Octava operación
93 $concat = $num + $cad;
94 echo '<article>';
95 echo '<p>El valor de $concat = $num + $cades: <span> = ' . $num . ' + "' . $cad . '"/>= <span>';
96 echo $concat . '</span> y el tipo de dato es: <span>' . gettype($concat) . "</span></p>" . SALTO;
97 echo '</article>';
98 echo "\n";
99
100 //Novena operación
101 $concat = $val + $str;
102 echo '<article>';
103 echo '<p>El valor de $concat = $val + $str es: <span> = ' . $val . ' + "' . $str . '"/>= <span>';
104 echo $concat . '</span> y el tipo de dato es: <span>' . gettype($concat) . "</span></p>" . SALTO;
105 echo '</article>';
106 echo "\n";
107
108 //Décima operación
109 $concat = $str + $cad;
110 echo '<article>';
111 echo '<p>El valor de $concat = $str + $cad es: <span> = "' . $str . ' + "' . $cad . '"/>= <span>';
112 echo $concat . '</span> y el tipo de dato es: <span>' . gettype($concat) . "</span></p>" . SALTO;
113 echo '</article>';
114 echo "\n";
115 echo '</section>';
116 echo "\n";
117 ?>
118 </body>
119 </html>

```

El resultado en el navegador sería el siguiente:



Trabajando con los tipos de datos en PHP

Variables:
\$cad = "10 metros"
\$str = "3D"
\$val = 20.5
\$num = 7

El valor de \$concat = \$cad + \$val es: = "10 metros" + 20.5 = 30.5 y el tipo de dato es: **double**
El valor de \$concat = \$str + \$num es: = "3D" + 7 = 10 y el tipo de dato es: **integer**
El valor de \$concat = \$cad + \$num es: = "10 metros" + 7 = 17 y el tipo de dato es: **integer**
El valor de \$concat = \$str + \$val es: = "3D" + 20.5 = 23.5 y el tipo de dato es: **double**
El valor de \$concat = \$cad + \$str es: = "10 metros" + "3D" = 13 y el tipo de dato es: **integer**
El valor de \$concat = \$val + \$cad es: = 20.5 + "10 metros" = 30.5 y el tipo de dato es: **double**
El valor de \$concat = \$num + \$str es: = 7 + "3D" = 10 y el tipo de dato es: **integer**
El valor de \$concat = \$num + \$cades: = 7 + "10 metros" = 17 y el tipo de dato es: **integer**
El valor de \$concat = \$val + \$str es: = 20.5 + "3D" = 23.5 y el tipo de dato es: **double**
El valor de \$concat = \$str + \$cad es: = "3D" + "10 metros" = 13 y el tipo de dato es: **integer**

Ejercicio #7: Este último ejemplo ilustrará cómo obtener los datos ingresados en un formulario y procesarlos con un script PHP sencillo que, de momento, no utilizará ninguna validación del lado del servidor, solamente validación del lado

del cliente con JavaScript que ya conoce. En posteriores prácticas abordaremos el tema de la validación en el lado del servidor.

Primer script: ingresodatos.html

```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="utf-8" />
5   <meta name="viewport" content="width=device-width,user-scalable=no,initial-
6 scale=1.0,maximum-scale=1.0,minimum-scale=1.0" />
7   <title>Formulario de ingreso de datos</title>
8   <link rel="stylesheet" href="css/forms.css" />
9   <script src="js/validar.js"></script>
10  <script src="js/modernizr.custom.lis.js"></script>
11 </head>
12 <body>
13 <header>
14   <h1>Ingreso de datos</h1>
15 </header>
16 <section>
17   <article>
18     <form action="procesar.php" method="POST">
19       <label>Cliente:</label>
20       <input type="text" id="client" name="client" />
21       <label>Producto:</label>
22       <input type="text" id="product" name="product" />
23       <label>Precio:</label>
24       <input type="text" id="price" name="price" />
25       <label>Cantidad:</label>
26       <input type="text" id="quantity" name="quantity" />
27       <span id="numbersOnly">Ingrese números enteros</span>
28       <input type="submit" id="enviar" name="submit" value="Enviar"
29 class="submit" />
30     </form>
31   </article>
32 </section>
33 </body>
34 </html>
```

Segundo script: procesar.php

```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="utf-8" />
5   <meta name="viewport" content="width=device-width,user-scalable=no,initial-scale=1.0,
6 maximum-scale=1.0,minimum-scale=1.0" />
7   <title>Información recibida</title>
8   <link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Nobile" />
9   <link rel="stylesheet" href="css/tables.css" />
10  <script src="js/modernizr.custom.lis.js"></script>
11 </head>
12 <body>
13 <section>
14 <article>
15 <div id="info">
16 <table id="hor-zebra" summary="Datos recibidos del formulario">
17 <caption>Información de formulario</caption>
18 <thead>
```

```

19 <tr class="thead">
20 <th scope="col">Campo</th>
21 <th scope="col">Valor</th>
22 </tr>
23 </thead>
24 <tbody>
25 <?php
26 if(isset($_POST['submit']) && $_POST['submit'] == "Enviar"):
27     echo "\t<tr class=\"odd\">\n";
28     echo "\t\t<td>\nCliente\n</td>\n";
29     //Accediendo a los datos del formulario usando la función extract()
30     extract($_POST);
31     $cliente = !empty($client) ? $client : "<a href=\"ingresodatos.html\">No ingresó el cliente.</a>";
32     echo "\t\t<td>\n" . $cliente . "\n</td>\n";
33     echo "\t</tr>\n";
34     echo "\t<tr>\n";
35     echo "\t\t<td>\nProducto\n</td>\n";
36     $producto = !empty($product) ? $product : "<a href=\"ingresodatos.html\">No ingresó el producto.</a>";
37     echo "\t\t<td>\n" . $producto . "\n</td>\n";
38     echo "\t</tr>\n";
39     echo "\t<tr class=\"odd\">\n";
40     echo "\t\t<td>\nPrecio\n</td>\n";
41     $precio = !empty($price) ? $price : "<a href=\"ingresodatos.html\">No ingresó el precio</a>";
42     echo "\t\t<td>\n" . $precio . "\n</td>\n";
43     echo "\t</tr>\n";
44     echo "\t<tr>\n";
45     echo "\t\t<td>\nCantidad\n</td>\n";
46     $cantidad = !empty($quantity) ? $quantity : "<a href=\"ingresodatos.html\">No ingresó la cantidad</a>";
47     echo "\t\t<td>\n" . $cantidad . "\n</td>\n";
48     echo "\t</tr>\n";
49     echo "\t<tr class=\"odd\">\n";
50     echo "\t\t<td>\nTotal a pagar\n</td>\n";
51     if(isset($cliente) && isset($producto) && floatval($precio)>0 &&
52         floatval($cantidad)>0):
53         echo "\t\t<td>\n$ " . number_format($precio * $cantidad, 2, '.', ',') . "\n</td>\n";
54     else:
55         echo "\t\t<td>\nNada que cobrar\n</td>\n";
56     endif;
57     echo "\t</tr>\n";
58 else:
59     echo "\t<tr class=\"odd\">\n";
60     echo "\t\t<td>\nNo se han ingresado desde el formulario.</td>\n";
61     echo "\t</tr>\n";
62 endif;
63 ?>
64 </tbody>
65 </table>
66 <div id="link">
67 <a href="ingresodatos.html" class="button-link">Ingresar nuevos datos</a>
68 </div>
69 </div>
70 </article>
71 </section>
72 </body>
73 </html>

```

Resultado en el navegador:

Ingreso de datos

CLIENTE:
Fausto González

PRODUCTO:
Camiseta casual

PRECIO:
16.75

CANTIDAD:
2

ENVIAR

Información de formulario

Campo	Valor
Cliente	Fausto González
Producto	Camiseta casual
Precio	16.75
Cantidad	2
Total a pagar	\$ 33.50

Ingresar nuevos datos

V. DISCUSIÓN DE RESULTADOS

- 1) Realice un script PHP que muestre mediante la utilización de variables sus datos personales: nombre completo, lugar de nacimiento (departamento y país, si es extranjero), edad y carnet de la universidad. Muestre estos datos en una tabla.
- 2) Cree un script PHP que utilice un formulario para solicitar una cantidad que debe ser ingresada en dólares y muestre como respuesta a cuánto equivale esa cantidad en euros. Su respuesta debe indicar la cantidad en dólares que se ingresó en una columna de una tabla de resultados y en la columna siguiente su equivalente en euros. Utilice buena presentación de la tabla y validación de la entrada del usuario.

VI. INVESTIGACIÓN COMPLEMENTARIA

- 1) Investigue las constantes de PHP conocidas como constantes predefinidas o "mágicas" y las constantes para el tratamiento de errores que brinda el lenguaje. Muestre en una tabla de dos columnas la constante y su utilidad. Por ejemplo:

E_ALL	Todos los errores y advertencias soportados.
--------------	---

- 2) Investigue todas las funciones que dispone PHP para determinar el tipo de una variable. Haga una tabla con dos columnas, una con la sintaxis de la función y otra con la descripción. Por ejemplo:

boolean is_numeric(\$arg)	Devuelve true si la variable es un número o una cadena numérica o falso en otro caso.
----------------------------------	--

VII. BIBLIOGRAFÍA

1. Cabezas Granado, Luis Miguel. PHP 6 Manual Imprescindible. 1ra. Edición. Editorial Anaya Multimedia. Madrid, España. 2010.
2. Doyle, Matt. FUNDAMENTOS PHP PRÁCTICO. 1ra Edición. Editorial Anaya Multimedia. Madrid, España. 2010.
3. Tim Converse / Steve Suehring. LA BIBLIA DE PHP 6 y MySQL. Editorial Anaya Multimedia. 1a. Edición. Madrid, España. 2009.
4. Welling, Luke / Thomson, Laura. DESARROLLO WEB CON PHP Y MySQL. Traducción de la 3ra Edición en inglés. Editorial Anaya Multimedia. Madrid, España 2005.
5. Gutiérrez, Abraham / Bravo, Ginés. PHP 5 A TRAVÉS DE EJEMPLOS. 1ra. Edición. Editorial Alfaomega. México, junio 2005.
6. Ellie Quigley / Marko Gargenta. PHP y MySQL Práctico para Diseñadores y Programadores Web. Anaya Multimedia / Prentice Hall. 1a. edición. 2007. Madrid, España.
7. Gil Rubio, Francisco Javier / Villaverde, Santiago Alonso. CREACIÓN DE SITIOS WEB CON PHP 5. 1ra. Edición. Editorial McGraw-Hill/Interamericana. Madrid, España 2006.
8. John Coggeshall. LA BIBLIA DE PHP 5. 1ra. Edición. Editorial Anaya Multimedia. Madrid, España 2005.