VARIABLES Y OPERADORES

ÍNDICE

MOTIVACIÓN

Los lenguajes de programación se basan en el trabajo con la información y PHP no es diferente. Debemos tener en cuenta qué tipos de datos vamos a poder utilizar, cómo crear variables que nos ayuden a contener estos datos y aprender a trabajar con ellos.

Vamos a ver cómo operar con datos y variables en PHP.

PROPÓSITOS

Al finalizar esta sección serás capaz de:

- Conocer los tipos de datos existentes en el lenguaje.
- Saber cómo crear y trabajar con variables y constantes.
- Conocer los operadores de los que disponemos.

PREPARACIÓN PARA LA SECCIÓN

En esta sección aprenderás a usar variables, constantes y operadores. Para realizar los ejemplos será necesario que tengas instalado VS con PHP Server (plugin) o algún programa para editar los PHP y un servidor web para poder probarlos.

1. VARIABLES

Lo primero que hay que entender es que en un lenguaje de programación el código es leído por el ordenador como en castellano. Es decir, se lee de arriba abajo y, dentro de la misma línea, de izquierda a derecha.

A continuación, podemos decir que las instrucciones son todas aquellas líneas de código que indican que algo tiene un valor. Por ejemplo, una instrucción puede ser la siguiente (asignación):

```
$var = 5;
```

En este ejemplo asignamos a una variable el valor 5. Podemos definir a una **variable** como una pequeña zona de memoria en la que guardamos un valor. Este valor se reemplaza cuando le volvemos a asignar otro valor. Por ejemplo, si tenemos un estuche de gafas, solamente podemos guardar una a la vez. Por lo tanto, si ya tenemos una en el estuche, debemos sacarla y añadir la nueva.

Si queremos asignar el valor 5 a dos variables, no es necesario crear dos líneas de asignación, sino que, sabiendo que el código siempre se ejecuta de izquierda a derecha, podemos hacer una doble asignación:

```
$var = $var2 = 5;
```

Como hemos ido viendo en los ejemplos anteriores, en PHP las variables se representan con el signo del dólar **($)** delante del nombre de la variable, los nombres de variables son **"case sensitive",** es decir, distinguen entre mayúsculas y minúsculas.

Las variables deben empezar con una letra o el carácter subrayado **(_)** seguido de tantas letras y números como deseemos. Es ***recomendable*** usar minúsculas y, si es palabra compuesta, usar la primera letra en mayúscula como carácter separador. Más adelante veremos los tipos de datos en PHP. En el siguiente ejemplo, vemos dos datos numéricos y texto (entre comillas).

```
$_miVariable = 23;
$variable = "Alice in borderland";
```

Nulo (null)

Es un tipo que indica una variable sin valor. Una variable es considerada de tipo *null* si se le asigna la constante **NULL.** Si no se le ha asignado ningún valor o se ha usado la función **'unset()'.** Por ejemplo:

```
$variable = null;
```

La variable **'$this'** es especial del navegador y no puede ser asignada. Se verá su importante uso más adelante.

Como todos los lenguajes de programación, PHP tiene una serie de variables predefinidas (**superglobales**). Estas dan información y valores a cualquier *script* que se ejecute. Las variables predefinidas más comunes son:

- $GLOBALS, referencia a todas las variables de ámbito global.
- $_GET, variables del http *get.*
- $_POST, variables del http post.
- $_SERVER, información del entorno.
- $_FILES, carga de archivos http.
- $_REQUEST, variables http *request.*
- $_SESSION, variables de sesión.
- $_ENV, variables de entorno.
- $_COOKIE, *cookies* http.
- $http_response_header, encabezados http.
- $argc, número de argumentos pasados a un *script.*
- $argv, *array* de los argumentos pasados a un *script.*

2. ENTRECOMILLADO

Las comillas sirven para envolver los contenidos que hay que mostrar y se diferencia entre comillas simples y comillas dobles. Cada una de ellas tiene su función.

```
echo "01- nombre: $nombre1 <br>";
echo '02- nombre: ';
```

3. OPERADOR DE CADENAS, CONCATENACIÓN

Existen dos operadores especiales de cadena que nos valen para concatenar varias cadenas: el primero es el símbolo del punto **(.),** el cual devuelve la unión de las cadenas de derecha e izquierda del punto; el segundo es el punto igual **(.=),** que añade al lado izquierda el lado derecha del símbolo.

```
echo "01- nombre: " . $nombre1 . '<br>';
echo '02- nombre: ' . $nombre2 . '<br>';
$nombreCompleto = $nombre . “ ” . $apellido;
```

Otro ejemplo:

```
<?php
$uno = "hola ";
$dos = $uno."mundo"; // $dos contiene "hola mundo"
$uno = "hola ";
$uno .= "mundo"; // $uno contiene "hola mundo"
?>
```

4. TIPOS DE DATOS

En PHP se utilizan varios tipos de datos primitivos. Podemos definir como "primitivos" aquellos datos que conoce el lenguaje de programación (PHP). Más adelante veremos la diferencia con los tipos de datos referenciados.

Booleanos *(boolean)*

Este tipo expresa un valor de verdadero o falso. La sintaxis de uso es la siguiente:

```
<?php $variables = true; ?> /*el valor puede ser true o false 1 o 0 */
```

Enteros *(integer)*

Es un número negativo o positivo sin decimales, por ejemplo:

```
<?php $variable = 234; ?> /*positivo*/
<?php $variable = -23; ?> /*negativo*/
```

Este tipo de datos también acepta valores en hexadecimal y octal:

```
<?php $variable = 0123; ?> /*octal*/
<?php $varible = 0x1B; ?> /*hexadecimal*/
```

Cualquier valor que manejemos que tras una operación nos devuelva un número fuera del rango de los enteros se convertirá automáticamente en un ***float***, datos que serán devueltos con ese tipo.

Número de punto flotante *(float)*

Son números con posiciones decimales, por ejemplo:

```
<?php $variable = 3.168; ?>
```

Cadenas *(string)*

Es un conjunto de caracteres sin límite establecido; pueden crearse con comillas simples o con comillas dobles:

```
<?php $variable = 'primera variables cadena'; ?>
<?php $variable = "primera variables cadena"; ?>
```

Cuando queramos poner una comilla simple o una doble dentro de una cadena como valor y usemos ese mismo carácter como delimitador, tenemos que ***escapar* la comilla,** es decir, indicarle de algún modo que queremos poner una comilla y no cerrar la cadena. Para ello usamos la barra invertida **(\).** En caso de querer poner dos barras también hay que escaparlas, para lo cual usaremos doble barra **(\\),** por ejemplo:

```
<?php $variable = 'Citando a D\'hartañan "todos para uno…"'; ?>
```

Existen dos métodos menos utilizados para crear cadenas que evitan tener que escapar las comillas: Heredoc y Nowdoc. En ambos se usan tres símbolos: menor que (<<<) con un identificador justo después de ellos, después el contenido de la cadena (en una nueva línea) y, por último, el identificador que le asignáramos para cerrar la definición seguido de punto y coma para cerrar comando (;). Este último identificador debe ser lo único, junto al punto y coma, que aparezca en la línea.

```
<?php
   $variables = <<<PRUEBA
   Esto es una prueba de heredoc en la
   Cual no tengo escapar las ‘ y puedo escribir
   El texto tal cual, muy útil para poner consultas SQL
   PRUEBA;
?>
```

5. CARACTERES DE ESCAPE *(STRING)*

Si la cadena la encerramos entre comillas dobles tenemos una serie de caracteres escapados para crear efectos en el texto:

\n	Nueva línea
\t	Tabulación horizontal
\r	Retorno de carro (salto de línea “Enter”)
\f	Avance de página
\v	Tabulador vertical
****	Barra invertida
\”	Comillas dobles
\$	Símbolo del dólar

6. CONSTANTES

Una constante es un identificador al cual se asocia un valor que no puede variar durante la ejecución del *script,* aunque por convención se declaran en mayúsculas hay que tener cuidado, ya que son sensibles a mayúsculas y minúsculas, y su nombre sigue el mismo estándar que para las variables.

```
<?php
Define(“IVA”, 21);
echo IVA;
?>
```

El ejemplo anterior define una constante llamada **IVA** y le asigna el valor **21.**

Solamente pueden ser constantes los valores de tipos *boolean,* entero, flotante, cadena y *arrays* (los últimos a partir de PHP 7). Como vemos en el ejemplo, para hacer referencia a su valor basta con poner el nombre de la constante, nunca se usa el identificador de variables dólar.

Las diferencias entre variable y constante son:

- Las constantes no llevan el signo del dólar **($).**
- Las constantes siempre deben definirse usando **define(),** no basta con asignarles un valor como a las variables.
- Las constantes no se rigen por normas de ámbito y pueden ser definidas y accedidas desde cualquier sitio.
- Las constantes no pueden variar su valor ni ser vaciadas una vez definidas.
- Las constantes solo contienen valores *boolean,* enteros, flotantes, cadenas o *arrays.*

Desde la versión 5.3 de PHP se puede crear variables sin usar la función **define,** con el literal **'const',** quedando la definición:

```
<?php const IVA = 21 ?>
```

En PHP existen también una serie de **constantes predefinidas** en el sistema de programación. Muchas de estas son creadas por diferentes extensiones y solo se podrán usar en el caso de contar con dichas extensiones. Vamos a ver seis constantes de este tipo que cambian dependiendo de dónde son usadas:

- __LINE__, línea actual del fichero.
- __FILE__, ruta absoluta del fichero.
- __DIR__, directorio del fichero.
- __FUNCTION__, nombre de la función.
- __CLASS__, nombre de la clase.
- __METHOD__ nombre del método de la clase.

7. OPERADORES: ASIGNACIÓN, MATEMÁTICOS

Operadores de asignación

El operador básico de asignación es el símbolo igual **(=)** e indica que el operando de la izquierda coge el valor de la derecha, pudiendo hacer una serie de asignaciones complejas. Veamos un ejemplo:

```
<?php $a = ($b = 4) + 5; ?>
```

En el ejemplo anterior, **$a** valdrá 9 y **$b** valdrá 4.

Además de este operador básico de asignación, existen otros que son combinaciones de este y operadores del lenguaje, como, por ejemplo:

```
<?php
$a = 4;
$a += 4; // establece $a en 8, como si: $a = $a + 4;
$b = "Hola ";
$b .= "mundo"; // establece $b en "Hola mundo"
?>
```

Operadores aritméticos

Son aquellos que nos permiten hacer cálculos matemáticos básicos, y son:

- **Negación:** opuesto de un valor dado (-valor).
- **Adición:** suma de dos valores (valor + valor).
- **Sustracción:** resta de dos valores (valor - valor).
- **Multiplicación:** productos de dos valores (valor * valor).
- **División:** cociente de dos valores (valor / valor).
- **Módulo:** resto de una división (valor % valor).

Por ejemplo:

```
$c  =  $a -   $b;     //-2

$c  =  $a *   $b;     //8

$c  =  $b /   $a;     //2

$c  =  $b %   $a;     //0

?>
```

8. INCREMENTEO/DECREMENTO

Estos operadores afectan solamente a tipos numéricos y de cadena, y son los siguientes:

- **Pre-incremento:** incrementa valor en uno y devuelve valor (++valor).
- **Post-incremento:** devuelve valor y luego lo incrementa uno (valor++).
- **Pre-decremento:** decrementa valor en uno y lo devuelve (--valor).
- **Post-decremento:** devuelve valor y luego lo decrementa uno (valor--).

Veamos un ejemplo de cómo preincrementar un número y un carácter.

```
<?php
$num=34;
echo "pre-incremento de $num ";
echo "ahora la variable vale ".++$num."<br/><br/>";
$num2='F';
echo "pre-incremento de $num2 ";
echo "ahora la variable vale ".++$num2."<br/><br/>";
?>
```

Otro ejemplo:

```
<?php
$b = $a = 5;
/* asignar el valor cinco a la variable $a y $b */
$c = $a++;
/* post-incremento, asignar el valor original de $a (5) a $c */
$e = $d = ++$b;
/* pre-incremento, asignar valor incrementado de $b (6) a $d y $e
*/
?>
```

9. VALOR Y REFERENCIA

Para acceder a los datos guardados en la memoria, como los valores de las variables y otros, podemos seguir varios métodos. Es muy importante conocer cómo se comportan los distintos elementos con los que trabajos, ya sean variables, *arrays* u objetos, por ejemplo. Es de gran ayuda determinar cómo se pasan esos datos, si por valor o por referencia.

- **Valor:** cada variable será independiente y tendrá su propio valor.
- **Referencia:** cada variable referenciada apunta una misma dirección de memoria, que es donde se encuentra el valor. Es como darle a una misma variable otro nombre, pudiendo acceder u operar desde cualquiera de ellos.

```
<?php
$a = 3;
$b = &$a; // $b es una referencia para $a
?>
```

10. OPCIONES DE CONVERSIÓN

Para los tipos de datos que hemos visto existen una serie de **opciones de conversión.** Vamos a ver una por una:

Booleanos

Para convertir un valor a *boolean* usaremos **(bool),** aunque en este caso no suele hacer falta, ya que un valor es convertido automáticamente a **'true'** o **'false'** cuando se establece una relación de este tipo (por ejemplo, en un **'if'**). Los siguientes valores son considerados falsos al convertir a *boolean* un valor:

- Entero con valor 0.
- Punto flotante con valor 0.0.
- Cadena vacía.
- Vector con 0 elementos.
- Objeto sin variable miembro.
- El tipo especial NULL.
- Objetos 'SimpleXML' creados desde etiquetas vacías

El resto devolverán un verdadero. Aun así, podemos usar la expresión **(bool)** para forzar la conversión. Por ejemplo:

```
var_dump((bool) "");          // bool(false)
var_dump((bool) 1);           // bool(true)
var_dump((bool) array(12));   // bool(true)
var_dump((bool) array());     // bool(false)
```

La función **var_dump** nos da información sobre las variables, muestra lo indicado en los comentarios detrás de cada expresión.

Enteros

Para convertir un valor a entero usaremos **(int)** o **(integer).** También podemos usar la función **intval().** Veamos unos ejemplos:

```
$var = ((int) “12” ); //convierte el texto “12” a numérico
$var = ((integer) “12” ); //convierte el texto “12” a numérico
$var = intval('42'); // convierte el texto “42” a numérico
```

Números de punto flotante

Para todos los tipos de valores (menos las cadenas) la conversión se realiza del mismo modo que si el valor se hubiera convertido a entero.

En el caso de las cadenas, si la cadena no contiene los caracteres **‘.’, ‘e’** ni **‘E’** y el valor entra entre los límites de los enteros, la cadena será evaluada como un entero; en el resto de casos se tomará como flotante. Veamos unos ejemplos para aclararlo:

```
$var = 1 + "10.5";            // $var es float (11.5)
$var = 1 + "-1.3e3";          // $var es float (-1299)
$var = 1 + "cas-1.3e3";       // $var es integer (1)
$var = 1 + "cas3";            // $var es integer (1)
$var = 1 + "10 pájaros";      // $var es integer (11)
$var = 4 + "10.2 pulpos";     // $var es float (14.2)
$var = "10.0 pulpos " + 1;    // $var es float (11)
$var = "10.0 pulpos " + 1.0;  // $var es float (11)
```

Cadenas

Cualquier valor puede convertir a cadena mediante la expresión **(String)** o la función **strval(),** teniendo en cuenta que el valor **true** de los tipos *boolean* es convertido a **“1”** y el valor **false** a **“0”.**

```
$var = ((String) 23);
$var = strval(23);
```

11. FUNCIONES ÚTILES

Algunas funciones útiles:

- empty: determina si una variable está vacía.
- gettype: para obtener el tipo de una variable.
- is_array: comprueba si una variable es un *array.*
- is_bool: comprueba si una variable es de tipo booleano.
- is_double: alias de **is_float.**
- is_float: comprueba si el tipo de una variable es **float.**
- is_int: comprueba si el tipo de una variable es **integer.**
- is_integer: alias de **is_int.**
- is_null: comprueba si una variable es **NULL.**
- is_numeric: comprueba si una variable es un número o un *string* numérico.
- is_object: comprueba si una variable es un objeto.
- is_string: comprueba si una variable es de tipo *string.*
- isset: determina si una variable está definida y no es **NULL.**
- print_r: imprime información legible para humanos sobre una variable.
- unset: destruye una variable especificada.
- var_dump: muestra información sobre una variable.

12. HEADER Y EXIT

La instrucción **header** permite redireccionar y **exit** terminar la ejecución del *script.*

```
header("Refresh:3; url=https://www.google.es");
exit("fin del script");
```

13. CADENAS Y MÉTODOS

En PHP, una cadena es un vector de caracteres, es decir, podemos tratarlo como si fuera un vector y, por ejemplo, acceder a alguno de los caracteres por el **id** de posición.

Llamamos vector o *array* a una colección de datos donde cada dato tiene una entidad propia. Esos datos pueden ser de cualquier tipo. Aunque el *array* y las cadenas son de la misma naturaleza, las verás detalladamente en otra unidad didáctica.

Con los corchetes **[2]** se accede a la clave donde se encuentra el carácter, que en este caso coincide con la posición 2, siendo 0 la primera posición, y se obtiene el carácter **'s'.**

```
<?php
   $varible = "esto es una prueba";
   echo $variable[2]; /* mostrará 's'*/
?>
```

Las cadenas tienen un tratamiento especial y existen bastantes métodos para realizar multitud de operaciones con ellas.


```
<?php
   $varible = "esto es una prueba";
   echo $variable[2]; /* mostrará 's'*/
?>
```

14. AMPLIACIÓN VARIABLES

Ámbito de las variables

Una de las cosas más importante que debemos tener en cuenta a la hora de usar las variables es el ámbito en el que se definen. Las variables por defecto creadas de modo normal tienen un ámbito local, es decir, pueden ser usadas en el contexto donde se crearon, pero no dentro de funciones, ya que al definir una función se crea un ámbito local nuevo.

Veamos un ejemplo para ver cómo funcionan las variables y sus diferentes ámbitos:

```
<html>
<body>
<p>
<?php
 $var = 2;
 $var_2 = 5;
 echo "valor sumado: " .($var + $var_2)."<br/>";
 function prueba(){
 echo "valor sumado en funcion: " .($var + $var_2);
 }
 prueba();
?>
</p>
</body>
</html>
```

Este ejemplo nos devolverá 7 en el primer **'echo'** fuera de la función y 0 en el segundo, dentro de la función, ya que en este segundo caso las variables que intenta coger son de ámbito local a la función y no están definidas.

Para evitar el caso anterior podemos hacer uso de la palabra reservada **'global',** a la cual le pasamos por parámetro las variables que queremos tomar como globales, quedando el ejemplo anterior así:

```
<?php
$var = 2;
$var_2 = 5;
echo "valor sumado: " .($var + $var_2)."<br/>";
function prueba()
{
global $var, $var2; //se usan las locales al programa
echo "valor sumado en funcion: " .($var + $var_2);
 }
prueba();
?>
```

En este caso, se mostrará 7 en los dos **'echo'** que aparecen en el código. También podemos usar el vector **$GLOBALS** reservado, que contiene el nombre y valor de las variables globales. Veamos el ejemplo:

```
<?php
   $var = 2;
   $var_2 = 5;
   echo "valor sumado: " .($var + $var_2)."<br/>";
   function prueba(){
echo "sumado en funcion: " .($GLOBALS['var'] + $GLOBALS['var_2']);
}
prueba();
?>
```

Recoger variables con GET y POST

En programación web, lo más común es que los datos de las variables sean re- llenados por el usuario mediante un formulario. Estos formularios HTML pueden mandar los datos de dos maneras diferentes: con el método POST o con el método GET. En la página en la cual recibamos el envío de un formulario podemos acceder a los datos de este formulario usando las variables predefinidas
$_GET; $_POST o **$_REQUEST.** Las dos primeras se usarán en envíos GET y POST respectivamente, mientras que la tercera contiene ambos envíos.

En este ejemplo crearemos una página que contenga un formulario con varios campos enviados con método POST y que llame a sí misma, y mostraremos dichos datos al principio de la web.

Aunque hay que conocerlo, este método es inseguro para recoger datos en la web. Aunque si se usan **$_GET** y **$_POST,** se integran en los llamados filtros, que aportarán un extra de seguridad contra las inyecciones de código malicioso a través de los formularios.

Veremos estos dos métodos en profundidad más adelante.

```
<html>
<body>
<p>
 <?php
 echo "Nombre: ".$_POST['nombre']."<br/>";
 echo "Email: ".$_POST['email'];
 ?>
 <form action="ejemplo6.php" method="post">
 Nombre: <input type="text" name="nombre" /><br />
 Email: <input type="text" name="email" /><br />
 <input type="submit" name="submit" value="enviar" />
 </form>
</p>
</body>
</html>
```

CONCLUSIONES

Los operadores en PHP son las herramientas que nos van a permitir crear programas más complejos y de mayor utilidad. Esto, unido a los tipos, variables y constantes, nos darán una mayor potencia al uso de las funcionalidades de PHP.

RECAPITULACIÓN

Ahora que has superado la unidad didáctica, ya sabes:

- Los tipos de datos que existen.
- Qué son las variables y cómo definirlas.
- El ámbito de las variables.
- Qué son las constantes y cómo definirlas.
- Cuáles son los principales operadores de PHP y cómo usarlos.

AUTOCOMPROBACIÓN

1. **Es un tipo primitivo:**

   a) Los verdaderos y falsos.

   b) El texto.

   c) Los números decimales.

   d) Todas las respuestas son correctas.

2. **¿Cuál de los siguientes no es un tipo de datos en PHP?**

   a) Booleano.

   b) Cadena.

   c) Vectores.

   d) Bit.

3. **¿Cuál es el carácter escapado de las cadenas que nos permite saltar de línea en un texto?**

   a) \f.

   b) \v.

   c) \n.

   d) \t.

4. **¿Cuál es el método que nos permite escribir un texto sin tener que escapar ningún carácter, ya que se muestra sin comprobar si contiene código?**

   a) Text.

   b) Nowdoc.

   c) NoCode.

   d) Todas las respuestas son incorrectas.

5. **¿Cuáles son las dos maneras de convertir un valor en cadena?**

   a) “toStr” y “String”.

   b) Solamente “String”.

   c) Solamente “toString”.

   d) “String” y “strval”.

6. ¿Cuál es el signo usado para representar las variables?

a) $.

b) %.

c) ¿.

d) &.

PROPUESTAS DE AMPLIACIÓN

Para saber cuáles son todos los tipos de recursos disponibles en PHP, visita la siguiente página web:

- http://www.php.net/manual/es/resource.php.

BIBLIOGRAFÍA

- CABEZAS, L. M. *Php 5.* Madrid: Anaya Multimedia, 2010.
- Manual de la página web oficial de PHP: http://www.php.net/manual/es/install.php.
- Material de teoría y ejercicios aportado por los profesores de Master.D.