Tema 3. Programación basada en lenguajes de marcas con código embebido

DESARROLLO EN ENTORNO SERVIDOR 2º DAW

MARÍA CRIADO DOMÍNGUEZ

Índice

Operadores

- Una expresión es una combinación de operadores, variables, constantes, y funciones que está formada correctamente, es decir, es válida sintácticamente, y que tiene sentido, o loque es igual, es válida semánticamente.
- Toda expresión produce un valor al ser procesada.
- La mayor parte del código que realicemos en PHP van a ser expresiones.

Operadores

Un operador es un elemento, palabra o símbolo, que al aplicarlo sobre otros elementos, los operandos, proporciona un valor.

Según el tipo de operación que realizan, los operadores más utilizados se clasifican en:

- Aritméticos: son los operadores empleados en las operaciones aritméticas: suma, resta,...
- Asignación: se utilizan para almacenar un dato dentro de una variable.
- De bit: permiten evaluar y manipular bits determinados dentro de un entero.
- Comparación: como su nombre indica, comparan dos elementos.
- Lógicos: el resultado obtenido de estos operadores se valora a true o false, Cierto o Falso.

Operadores Aritméticos

Operadores	**Representación**
Suma	$x + $y
Resta	$x - $y
Multiplicación	$x * $y
División	$x / $y
Módulo	$x % $y (resto de la división)
Exponenciación	$x ** $y
Negación	-$x

Operadores Comparación

Ejemplo	Nombre	Resultado
$a == $b	Igual	**TRUE** si *$a* es igual a *$b* después de la manipulación de tipos.
$a === $b	Idéntico	**TRUE** si *$a* es igual a *$b*, y son del mismo tipo.
$a != $b	Diferente	**TRUE** si *$a* no es igual a *$b* después de la manipulación de tipos.
$a <> $b	Diferente	**TRUE** si *$a* no es igual a *$b* después de la manipulación de tipos.
$a !== $b	No idéntico	**TRUE** si *$a* no es igual a *$b*, o si no son del mismo tipo.
$a < $b	Menor que	**TRUE** si *$a* es estrictamente menor que *$b*.
$a > $b	Mayor que	**TRUE** si *$a* es estrictamente mayor que *$b*.
$a <= $b	Menor o igual que	**TRUE** si *$a* es menor o igual que *$b*.
$a >= $b	Mayor o igual que	**TRUE** si *$a* es mayor o igual que *$b*.
$a <=> $b	Nave espacial	Un integer menor que, igual a, o mayor que cero cuando *$a* es respectivamente menor que, igual a, o mayor que *$b*. Disponible a partir de PHP 7.

Operadores Comparación

Si se compara un número con un string o la comparación implica strings numéricos, entonces cada string es convertido en un número y la comparación se realiza numéricamente.

PHP 7 introduce un nuevo tipo de operador, que se puede utilizar para comparar expresiones llamado nave espacial cuyo resultado es:

$a <=> $b evalúa a:

0 si $a == $b

-1 si $a < $b

1 Si $a > $b

Operadores Asignación

Operador	Ejemplo	Equivalencia
=	$a=$b;	$a toma el valor de $b
+=	$a += $b;	$a = $a + $b
-=	$a -= $b	$a = $a - $b
*=	$a *= $b;	$a = $a * $b;
/=	$a /= $b;	$a = $a / $b;
%=	$a %= $b;	$a = $a % $b;
.=	$a .= $b;	$a = $a . $b;

Operadores Asignación

Los operadores de incremento y decremento sólo afectan a números y strings, sin afectar a arrays, objects o resources.

Decrementar un valor NULL no tiene efecto, pero si se incrementa se obtiene 1.
Incrementar o decrementar booleanos no tiene efecto.

Operador	Efecto
++$x	Incrementa $x en 1 y devuelve $x
$x++	Retorna $x y luego incrementa $x en 1
--$x	Decrementa $x en 1 y devuelve $x
$x--	Retorna $x y luego decrementa $x en 1

Operadores Lógicos

Ejemplo	Nombre	Resultado
$a and $b	And (y)	**TRUE** si tanto *$a* como *$b* son **TRUE**.
$a or $b	Or (o inclusivo)	**TRUE** si cualquiera de *$a* o *$b* es **TRUE**.
$a xor $b	Xor (o exclusivo)	**TRUE** si *$a* o *$b* es **TRUE**, pero no ambos.
! $a	Not (no)	**TRUE** si *$a* no es **TRUE**.
$a && $b	And (y)	**TRUE** si tanto *$a* como *$b* son **TRUE**.
$a || $b	Or (o inclusivo)	**TRUE** si cualquiera de *$a* o *$b* es **TRUE**.

Operadores Bit

Ejemplo	Nombre	Resultado
$a & $b	And (y)	Los bits que están activos en ambos *$a* y *$b* son activados.
$a | $b	Or (o inclusivo)	Los bits que están activos ya sea en *$a* o en *$b* son activados.
$a ^ $b	Xor (o exclusivo)	Los bits que están activos en *$a* o en *$b*, pero no en ambos, son activados.
~ $a	Not (no)	Los bits que están activos en *$a* son desactivados, y viceversa.
$a << $b	Shift left(desplazamiento a izquierda)	Desplaza los bits de *$a*, *$b* pasos a la izquierda (cada paso quiere decir "multiplicar por dos").
$a >> $b	Shift right (desplazamiento a derecha)	Desplaza los bits de *$a*, *$b* pasos a la derecha (cada paso quiere decir "dividir por dos").

Tomas de decisión

- En PHP los guiones se construyen en base a sentencias. Utilizando llaves, puedes agrupar las sentencias en conjuntos, que se comportan como si fueran una única sentencia.
- Para definir el flujo de un programa en PHP, al igual que en la mayoría de lenguajes de programación, hay sentencias para dos tipos de estructuras de control:
  - sentencias condicionales, que permiten definir las condiciones bajo las que debe ejecutarse una sentencia o un bloque de sentencias;
  - sentencias de bucle, con las que puedes definir si una sentencia o conjunto de sentencias se repite o no, y bajo qué condiciones.

Tomas de decisión

- En PHP los guiones se construyen en base a sentencias. Utilizando llaves, puedes agrupar las sentencias en conjuntos, que se comportan como si fueran una única sentencia.

- Para definir el flujo de un programa en PHP, al igual que en la mayoría de lenguajes de programación, hay sentencias para dos tipos de estructuras de control:
  - sentencias condicionales, que permiten definir las condiciones bajo las que debe ejecutarse una sentencia o un bloque de sentencias.
  - sentencias de bucle, con las que puedes definir si una sentencia o conjunto de sentencias se repite o no, y bajo qué condiciones.
  - Además, en PHP puedes usar también (aunque no es recomendable) la sentencia goto, que te permite saltar directamente a otro punto del programa que indiques mediante una etiqueta.

Tomas de decisión Condicionales

- La sentencia if permite definir una expresión para ejecutar o no la sentencia o conjunto de sentencias siguiente. Si la expresión se evalúa a true (verdadero), la sentencia se ejecuta. Si se evalúa a false (falso), no se ejecutará.

```php
<?php
    if ($a < $b)
        print "a es menor que b";
    elseif ($a > $b)
        print "a es mayor que b";
    else
        print "a es igual a b";
?>
```

Tomas de decisión
Switch

- La sentencia switch es similar a enlazar varias sentencias if comparando una misma variable con diferentes valores. Cada valor va en una sentencia case. Cuando se encuentra una coincidencia, comienzan a ejecutarse las sentencias siguientes hasta que acaba el bloque switch, o hasta que se encuentra una sentencia break. Si no existe coincidencia con el valor de ningún case, se ejecutan las sentencias del bloque default, en caso de que exista.

```
<?php
   switch ($a) {
      case 0:
         print "a vale 0";
         break;
      case 1:
         print "a vale 1";
         break;
      default:
         print "a no vale 0 ni 1";
   }
?>
```

Bucles

while

Usando while puedes definir un bucle que se ejecuta mientras se cumpla una expresión. La expresión se evalúa antes de comenzar cada ejecución del bucle

```php
<?php
  $a = 1;
  while ($a < 8)
     $a += 3;
  print $a; // el valor obtenido es 10
?>
```

Bucles

Do/while

Es un bucle similar al anterior, pero la expresión se evalúa al final, con lo cual se asegura que la sentencia o conjunto de sentencias del bucle se ejecutan al menos una vez.

```
<?php
  $a = 5;
  do
     $a -= 3;
  while ($a > 10);
  print $a; // el bucle se ejecuta una sola vez, con lo que el valor obtenido es 2
?>
```

Bucles

for

Son los bucles más complejos de PHP. Al igual que los del lenguaje C, se componen de tres expresiones

```
<?php
      for ($a = 5; $a<10; $a+=3) {
            print $a; // Se muestran los valores 5 y 8
            print "<br />";
      }
?>
```

Bucles

Break/Continue

- Se utilizan para controlar la ejecución dentro de las sentencias en las que se encuentra el programa.
- Para salir de una estructura de control condicional o iterativa puede usarse BREAK o CONTINUE.
- Ambas se emplean de una forma análoga, con la diferencia de que mientras BREAK finaliza totalmente la ejecución del bucle; CONTINUE hace que se salte a la siguiente iteración.

Arrays

Un array es un tipo de datos que nos permite almacenar varios valores. Cada miembro del array se almacena en una posición a la que se hace referencia utilizando un valor clave. Las claves pueden ser numéricas o asociativas.

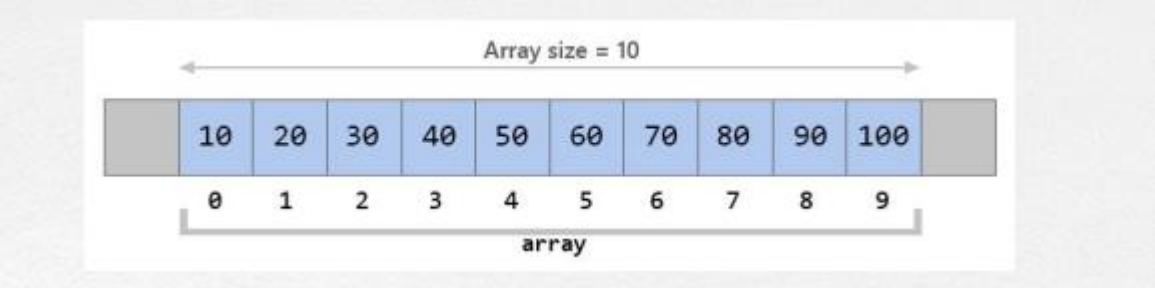
En PHP, hay tres tipos de arrays:

- Arrays numéricos: arrays con un índice numérico
- Arrays asociativos: arrays con claves con nombre
- Arrays multidimensionales: los arrays de varias dimensiones son arrays que contienen uno o más arrays en sus elementos. La dimensión de un array indica la cantidad de índices que necesita para seleccionar un elemento.

Arrays

Arrays numéricos

- Creación
  - Constructor array()
  - Sintaxis corta con corchete [ ]




```
<?php
  $coches = array("Volvo", "BMW", "Toyota");
  echo "Las marcas de coches son:" .$coches[0]. ", "
  .$coches[1]. " y " .$coches[2]. ".";
?>
```

Arrays

Arrays numéricos

Recorrer

```
for ($i=0; $i < count($dias); $i++) {
        echo "<p>".$dias[$i]."</p>";
    }
foreach($dias as $d)
        echo "<p>".$d."</p>";
```

Arrays

Arrays asociativos o indexados

- Arrays cuyos keys son Strings personalizados
- Los Strings son CaseSensitive

```
<?php
        $edad = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
        echo "Peter tiene " . $edad['Peter'] . " años.";
?>
```

Arrays

Arrays asociativos o indexados

Recorrer

```
foreach($edad as $d)
    echo "<p>".$d."</p>";
foreach ($edad as $variable => $valor) {
        print "<p>".$variable." ->";
        print $valor."</p>";
        }
```

Arrays

Arrays multidimensional

```
<?php

$ciclos = array(
      "DAW" => array ("PR" => "Programación", "BD" => "Bases de
  datos", ..., "DWES" => "Desarrollo web en entorno servidor"),
      "DAM" => array ("PR" => "Programación", "BD" => "Bases de
  datos",  ...,  "PMDM"  =>  "Programación  multimedia  y  de
  dispositivos móviles")

 );

?>
```

Arrays

Arrays multidimensional

Recorrer

```
foreach ($ciclos as $ciclo => $array) {
        print "<p>".$ciclo." </p>";
        foreach ($array as $inicial => $nombre) {
            print "<p>".$inicial." ->";
            print $nombre."</p>";
        }
    }
```

Arrays

Arrays multidimensional

Recorrer

```
foreach ($ciclos as $ciclo => $array) {
        print "<p>".$ciclo." </p>";
        foreach ($array as $inicial => $nombre) {
            print "<p>".$inicial." ->";
            print $nombre."</p>";
        }
    }
```

Arrays
Recorrer

Pero en PHP también hay otra forma de recorrer los valores de un array. Cada array mantiene un puntero interno, que se puede utilizar con este fin. Utilizando funciones específicas, podemos avanzar, retroceder o inicializar el puntero, así como recuperar los valores del elemento (o de la pareja clave / elemento) al que apunta el puntero en cada momento.

Función	Resultado
reset	Sitúa el puntero interno al comienzo del array.
next	Avanza el puntero interno una posición.
prev	Mueve el puntero interno una posición hacia atrás.
end	Sitúa el puntero interno al final del array.
current	Devuelve el elemento de la posición actual.
key	Devuelve la clave de la posición actual.
each	Devuelve un array con la clave y el elemento de la posición actual. Además, avanza el puntero interno una posición.

Arrays
Recorrer

- Las funciones reset, next, prev y end, además de mover el puntero interno devuelven, al igual que current, el valor del nuevo elemento en que se posiciona. Si al mover el puntero te sales de los límites del array (por ejemplo, si ya estás en el último elemento y haces un next), cualquiera de ellas devuelve false. Sin embargo, al comprobar este valor devuelto no serás capaz de distinguir si te has salido de los límites del array, o si estás en una posición válida del array que contiene el valor "false".
- La función key devuelve null si el puntero interno está fuera de los límites del array.
- La función each devuelve un array con cuatro elementos.
  - Los elementos 0 y 'key' almacenan el valor de la clave en la posición actual del puntero interno.
  - Los elementos 1 y 'value' devuelven el valor almacenado.

Arrays
Funciones

- Una vez definido un array puedes añadir nuevos elementos (no definiendo el índice, o utilizando un índice nuevo) y modificar los ya existentes (utilizando el índice del elemento a modificar).
- También se pueden eliminar elementos de un array utilizando la función unset
- La función array_values recibe un array como parámetro, y devuelve un nuevo array con los mismos elementos y con índices numéricos consecutivos con base 0.
- Para comprobar si una variable es de tipo array, utiliza la función is_array.
- Para obtener el número de elementos que contiene un array, tienes la función count.
- Si quieres buscar un elemento concreto dentro de un array, puedes utilizar la función in_array. Devuelve true si encontró el elemento o false en caso contrario.
- La función array_search, que recibe los mismos parámetros pero devuelve la clave correspondiente al elemento, o false si no lo encuentra.
- Y si lo que quieres buscar es un clave en un array, tienes la función array_key_exists, que devuelve true o false.

https://www.php.net/manual/es/ref.array.php