

Operadores lógicos

Mediante operadores lógicos es posible *evaluar* un conjunto de *variables lógicas*, es decir, aquellas cuyos valores sean únicamente: VERDADERO o FALSO (1 ó NUL).

El resultado de esa evaluación será siempre 1 ó NUL.

\$A AND \$B

El operador **AND** devuelve VERDADERO (1) en el caso de que **todas** las variables lógicas comparadas sean verdaderas, y FALSO (NUL) cuando **alguna** de ellas sea *falsa*.

\$A && \$B

El operador **&&** se comporta de forma idéntica al operador **AND**. La única diferencia entre ambos es que *operan con distinta precedencia*.

Más abajo veremos el orden de precedencia de los distintos operadores.

\$A OR \$B

Para que el operador **OR** devuelva VERDADERO (1) es suficiente que **una sola** de las variables lógicas comparadas sea *verdadera*. Únicamente devolverá FALSO (NUL) cuando **todas** ellas sean FALSAS.

\$A || \$B

El operador **||** se comporta de forma idéntica al operador **OR**. Su única diferencia es el orden de precedencia con el que opera.

\$A XOR \$B

El operador **XOR** devuelve VERDADERO (1) sólo en el caso de que sea *cierta una sola* de las variables, y FALSO (NUL) cuando ambas sean ciertas o ambas sean falsas.

! \$A

Este operador NOT (negación) devuelve VERDADERO (1) si la variable lógica \$A es FALSA y devuelve FALSO (NUL) si el valor de esa variable \$A es VERDADERO.

Sintaxis alternativa

Tal como hemos descrito los distintos operadores lógicos sería necesario que \$A y \$B contuvieran valores lógicos, y eso requeriría un paso previo para asignarles valores de ese tipo.

Habría que recurrir a procesos de este tipo:

```
$A = $x>$y;
$B= $x >=$z;
$A && $B;
```

pero se obtendría el mismo resultado escribiendo:

```
$x>$y && $x >=$z;
```

que, aparte de ser la forma habitual de hacerlo, *nos evita dos líneas de instrucciones*.

Aunque el propio ejemplo se auto comenta, digamos que al utilizar operadores lógicos se pueden sustituir las variables lógicas por expresiones que den como

Operadores lógicos

Aquí tienes las *tablas de verdad* de los distintos operadores lógicos.

El operador AND						
Variables	\$a=32; \$b=0; \$c=-7; \$d=4.32; \$f="23 Ptas"					
Ejemplo de sintaxis	\$x=\$a>\$b; \$y=\$a>\$c; \$z=\$a>\$f; echo (\$x AND \$y AND \$z);					
Otra sintaxis	echo (\$a>\$b AND \$a>\$c AND \$a>\$f);					
Condición A	Condición B	Condición C	A	B	C	Resultado
\$a>\$b	\$a>\$c	\$a>\$f	1	1	1	1
\$a>\$b	\$a>\$c	\$a<\$f	1	1		
\$a>\$b	\$a<\$c	\$a>\$f	1		1	
\$a<\$b	\$a>\$c	\$a>\$f		1	1	
\$a<\$b	\$a<\$c	\$a>\$f			1	
\$a<\$b	\$a>\$c	\$a<\$f		1		
\$a>\$b	\$a<\$c	\$a<\$f	1			
\$a<\$b	\$a<\$c	\$a<\$f				

El operador &&						
Variables	\$a=32; \$b=0; \$c=-7; \$d=4.32; \$f="23 Ptas"					
Condición A	Condición B	Condición C	A	B	C	Resultado
\$a>\$b	\$a>\$c	\$a>\$f	1	1	1	1
\$a>\$b	\$a>\$c	\$a<\$f	1	1		
\$a>\$b	\$a<\$c	\$a>\$f	1		1	
\$a<\$b	\$a>\$c	\$a>\$f		1	1	
\$a<\$b	\$a<\$c	\$a>\$f			1	
\$a<\$b	\$a>\$c	\$a<\$f		1		
\$a>\$b	\$a<\$c	\$a<\$f	1			
\$a<\$b	\$a<\$c	\$a<\$f				

El operador OR						
Variables	\$a=32; \$b=0; \$c=-7; \$d=4.32; \$f="23 Ptas"					
Condición A	Condición B	Condición C	A	B	C	Resultado
\$a>\$b	\$a>\$c	\$a>\$f	1	1	1	1
\$a>\$b	\$a>\$c	\$a<\$f	1	1		1
\$a>\$b	\$a<\$c	\$a>\$f	1		1	1
\$a<\$b	\$a>\$c	\$a>\$f		1	1	1
\$a<\$b	\$a<\$c	\$a>\$f			1	1
\$a<\$b	\$a>\$c	\$a<\$f		1		1
\$a>\$b	\$a<\$c	\$a<\$f	1			1
\$a<\$b	\$a<\$c	\$a<\$f				

El operador						
Variables	\$a=32; \$b=0; \$c=-7; \$d=4.32; \$f="23 Ptas"					
Condición A	Condición B	Condición C	A	B	C	Resultado
\$a>\$b	\$a>\$c	\$a>\$f	1	1	1	1
\$a>\$b	\$a>\$c	\$a<\$f	1	1		1
\$a>\$b	\$a<\$c	\$a>\$f	1		1	1
\$a<\$b	\$a>\$c	\$a>\$f		1	1	1
\$a<\$b	\$a<\$c	\$a>\$f			1	1
\$a<\$b	\$a>\$c	\$a<\$f		1		1
\$a>\$b	\$a<\$c	\$a<\$f	1			1
\$a<\$b	\$a<\$c	\$a<\$f				

El operador XOR					
Variables	\$a=32; \$b=0; \$c=-7; \$d=4.32; \$f="23 Ptas"				
Condición A	Condición B		A	B	Resultado
\$a>\$b	\$a>\$c		1	1	
\$a>\$b	\$a<\$c		1		1
\$a<\$b	\$a>\$c			1	1
\$a<\$b	\$a<\$c				

Ejemplo de uso de los operadores lógicos

```
<?
# asignemos valores a cuatro variables
$a=3;
$b=6;
```

resultado ese tipo de valores.

Orden de precedencia

Cuando se usan los operadores lógicos se plantean situaciones similares a lo que ocurre con las operaciones aritméticas.

Dado que permiten trabajar con secuencias de operaciones sería posible, por ejemplo, una operación de este tipo:

\$a<\$b OR \$c<\$b && \$a<3

Surgirían preguntas con estas:

- ¿qué comparación se haría primero OR ó &&?
- ¿se harían las comparaciones en el orden natural?
- ¿alguna de los operadores tiene prioridad sobre el otro?

Igual que en las matemáticas, también aquí, hay un orden de prioridad que es el siguiente:

NOT, &&, ||, AND, XOR y, por último, **OR**.

De esta forma la operación && se realizaría antes que ||, mientras que si pusiéramos AND en vez de && sería la operación || la que se haría antes y, por lo tanto, los resultados podrían variar de un caso a otro.

Aquí también es posible, de la misma manera que en la aritmética, utilizar paréntesis para priorizar una operación frente a otra.

Es muy importante prestar atención a la *construcción correcta* de estas estructuras. Un descuido en la atención a las prioridades puede ser origen –lo es frecuentemente– de resultados incoherentes que suelen ser detectados bajo una apariencia *aleatoria*.

Ten muy en cuenta que al depurar programas no siempre se ven fácilmente este tipo de errores de programación.

Puede que para determinados valores los resultados sean los esperados y que, sin embargo, al utilizar otros distintos pueda manifestarse la incoherencia.

Si te ocurriera esto no pienses que es el ordenador el que está *haciendo cosas raras*, procura revisar los paréntesis y los criterios de prioridad de los distintos operadores contenidos en la estructura lógica.

```
$c=9;
$d=17;
# utilizemos operadores de comparación
# y recojamos sus resultados en nuevas variables
$x= $a<$b;
$y= $a<$b;
$z=$c>$b;
# apliquemos un operador lógico (por ejemplo &&)
# e imprimamos el resultado

print("Resultado FALSO si no sale nada: ".$y && $z)."<br>";
# hagamos la misma comparación sin utilizar la variables $y y $z
# que deberá darnos el mismo resultado
print("<br>Resultado FALSO si no sale nada: ".$a<$b && $c>$b)."<br>";
/* veamos ahora que ocurre al ampliar la estructura
¿qué ocurriría si escribiéramos
  $a<$b OR $c>$b && $d<$a ?
El operador && tiene preferencia ante OR,
luego haria primero la comparación $c>$b && $d<$a
9 > 6 es cierto, 17 < 3 es falso, luego como &&
solo devuelve cierto cuando ambos sean ciertos
el resultado de esa opción es FALSO.
Veamos que ocurre al comparar $a<$b OR falso (resultado anterior)
como 3 < 6 es cierto OR operará con cierto y falso
que dará como resultado CIERTO, ya que basta que se
cumpla una de las condiciones */
/* vamos a comprobarlo mediante este operador condicional
no conocemos aun su sintaxis pero adelantemosla un poco...
si el el contenido del paréntesis que va detrás del if es cierto
imprimirá cierto y en caso contrario imprimirá falso
Aquí debería imprimirmos cierto */
if($a<$b OR $c>$b && $d<$a) {
    print "cierto<br>";
}else{
    print "falso<br>";
}

# Cambiemos la estructura anterior por $a<$b || $c>$b AND $d<$a
# ahora se operará primero || que como antes dará cierto
# pero ese resultado operado mediante AND con falso dará FALSO
# AL CAMBIAR LOS OPERADORES POR SUS SIMILARES el resultado el DISTINTO

if($a<$b || $c>$b AND $d<$a) {
    print "cierto<br>";
}else{
    print "falso<br>";
}

# un paréntesis nos devolverá a la situación anterior
# escribamos $a<$b || ($c>$b AND $d<$a)
# veremos que el resultado es CIERTO como ocurría en el primer caso

if($a<$b || ($c>$b AND $d<$a)) {
    print "cierto<br>";
}else{
    print "falso<br>";
}

?>
```

ejemplo23.php

Ejercicio nº 18

Define tres variables: *langosta*, *angula* y *caviar*, y asígnale valores numéricos. Luego, escribe una estructura lógica que devuelva **cierto** en el caso de que **dos de ellas superen** los precios respectivos de: **50**, **70** y **500**, y **falso** en cualquier otro caso. Compruébalo modificando los valores de las variables y viendo los resultados y guárdalo como **ejercicio18.php**.

Anterior Índice Siguiente

