Entorns de Desenvolupament

4. Operadors lògics i de comparació
5. Operadors aritmètics

Unió Europea
Fons social europeu
L'FSE inverteix en el teu futur

IES DR. LLUÍS SIMARRO

4. Operadors lògics i de comparació

Els operadors de comparació permeten **comparar** dos valors del mateix tipus (o assimilable), tal com el seu nom indica que donarà com a resultat un valor **booleà** (`true`, `false`). En general, això ens servirà per **prendre decisions**, per exemple per utilitzar-les en una estructura condicional **if .. else**. Per a això en PHP disposem dels operadors que s'indiquen en la següent taula.

Nom	Exemple	Resultat
Més gran que	**$a > $b**	• **true** si $a és major que $b • **false** en cas contrari
Més xicotet que	**$a < $b**	• **true** si $a és menor que $b • **false** en cas contrari
Major o igual que	**$a >= $b**	• **true** si $a és major o igual que $b • **false** en cas contrari
Menor o igual que	**$a <= $b**	• **true** si $a és menor o igual que $b • **false** en cas contrari
Diferent	**$a <> $b o $a != $b**	• **true** si $a és diferent a $b • **false** en cas contrari
Idèntic o estrictament igual	**$a === $b**	• **true** si $a és igual a $b i són del mateix tipus • **false** en cas contrari
No idèntic o estrictament diferent	**$a !== $b**	• **true** si $a no és igual a $b o no són del mateix tipus • **false** en cas contrari
Igual	**$a == $b**	• **true** si $a és igual a $b • **false** en cas contrari

La sintaxi coincideix amb l'empleada en altres llenguatges de programació.

A més dels operadors habituals hi ha els operadors **===** que s'interpreta com "és estrictament igual" i **!==** Que s'interpreta com "no és estrictament igual".

De moment cal tenir en compte que si una variable conté **$text1 = "1"** i fem la comparació **$text1 === 1**, obtindrem **false**, és a dir, que no és igual (perquè un text no és igual a un nombre). No obstant això una comparació com **$text1 == 1** retornarà **true** ja que aquesta comparació no és estricta i **tracta de realitzar automàticament conversions** per comprovar si es pot establir una equivalència entre els dos valors. En aquest cas es busca l'equivalent numèric del text i després es fa la comparació, motiu pel qual s'obté **true**.

És a dir, si es **compara** un **número** amb una **cadena** de caràcters o la comparació implica cadenes de caràcters numèriques, cada **cadena** de caràcters és **convertida** en un **nombre** i la comparació es realitzarà numèricament excepte quan l'operador utilitzat és **===** o **!==** perquè, en aquests casos, també comparem el tipus.

Operadors Lògics

Els operadors lògics, ens permeten crear condicions per a les diferents estructures en PHP, tant en estructures condicionals com en estructures repetitives. Els operadors lògics més importants són **and** i **or**. Els operands són valors **booleans** (`true`, `false`).

Nom	exemple	Resultat
and o &&	**$a and $b** **$a && $b**	• **true** si $a és **true** i $b és **true** • **false** en cas contrari
or o ||	**$a or $b** **$a || $b**	• **true** si $a o $b és **true**, o tots dos • **false** en cas contrari
xor	**$a xor $b**	• **true** si $a o $b és **true**, però no ambdós • **false** en cas contrari
not o !	**!$a** **not $a**	• **true** si $a no és **true** • **false** en cas contrari

Una condició pot ser tan llarga com es vulga o necessite i usar tants operadors lògics com es vulga o necessiten. L'ús dels parèntesis atorgarà la prioritat de l'execució

d'unes operacions i altres (com en matemàtiques).

Les **expressions** on s'utilitzen operadors lògics i relacionals tornen un **valor booleà**, és a dir, veritable (**true**) o fals (**false**). Per exemple:

- Si **$a = true** i **$b = false** l'expressió **$a && $b** retorna **false** (és falsa perquè no es compleix que **$a** i **$b** siguen veritables).
- Si **$a = true** i **$b = false** l'expressió **$a || $b** retorna **true** perquè un dels dos operands és veritable.
- Si **$a = true** l'expressió **!$a** retorna **false** (l'oposat o contrari).

L'operador **||** s'obté en la majoria dels teclats prement **ALT GR + 1,** és a dir, la tecla **ALT GR** i el número 1 simultàniament.

Els operadors **&&** i **||** es diuen **operadors en curtcircuit** perquè si no es compleix la condició d'un terme no s'avalua la resta de l'operació. Per exemple:

- L'expressió (**$a == $b && $c != $d && $h >= $k**) té tres avaluacions: la primera comprova si la variable **$a** és igual a **$b**. Si no es compleix aquesta condició, el resultat de l'expressió és `false` i no s'avaluen les altres dues condicions posteriors.
- En un cas com (**$a < $b || $c != $d || $h <= $k**), primer s'avalua si **$a** és menor que **$b**. Si es compleix aquesta condició el resultat de l'expressió és `true` i no s'avaluen les altres dues condicions posteriors.

Operador de negació aplicat sobre nombres o text

Si **$a = true** la seua negació **!$a** retorna **false**. **Però què passa si $a és un nombre o un text?**

Si **$a** és un **nombre** es considera que equival a **false** si el seu valor numèric és 0, o que equival a **true** si el seu valor numèric és diferent de zero. Seguidament s'aplica la negació.

Per exemple si **$a = 7**:

- **$a** es considera equivalent a **true.**

- **!$a** és **false**.

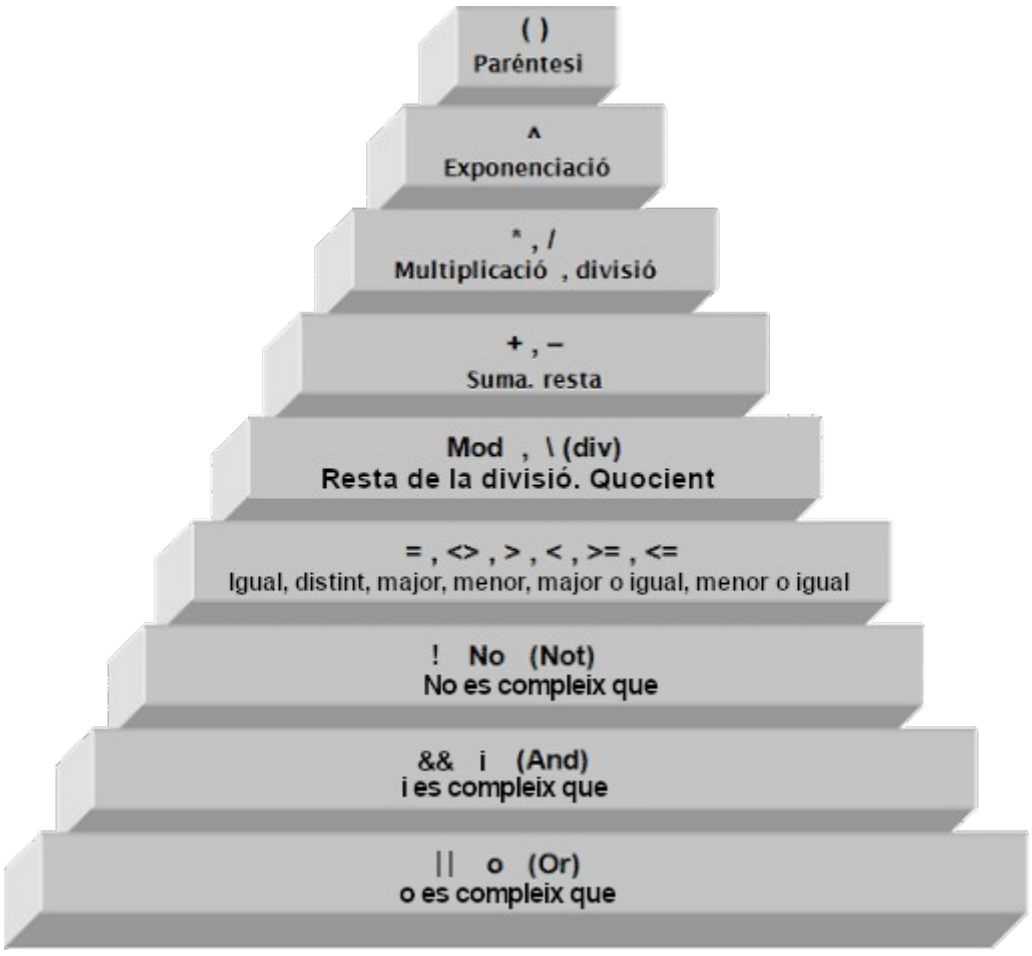
  Si **$a=0**:
- **$a** es considera equivalent a **false.**
- **!$a** és **true**.

Per **cadenes de text**, la cadena buida es considera que equival a **false** i qualsevol altra cadena es considera que equival a **true**.

Per exemple, si **$text1 = ""** (la cadena buida), aleshores **$text1** equival al valor **false** i **!$text1** a **true**.

Ordre de prioritat, prelació o precedència

Els operadors lògics i matemàtics tenen un **ordre de prioritat o precedència**. Aquest és un esquema general que indica l'ordre en què s'han d'avaluar en la majoria dels llenguatges de programació:



Sent **$A =3** i **$B=5** una expressió com:

```
$A+$B == 8 && $A-$B == 1
```

- S'avalua primer **$A+$B** que val 8.
- Després s'avalua **$A-$B** que val -2.
- Després s'avalua si es compleix que la primera operació `$A+$B== 8`. (`true`)
- Després si les compleix la segona operació `$A-$B == 1`, resultant que no (`false`).
- Finalment l'expressió completa s'avalua a `false`.

Exemple

Codi del fitxer exemple1.php:

```
<?php
	$a = 3;
	$b = 9;
	$resultat = ($a <= 3 and $b <> 9);
	if ($resultat == true) {
		echo "Es compleix la condició";
	} else {
		echo "No es compleix la condició";
	}
?>
```

Sent el resultat final en pantalla:

No es compleix la condició

Comparació de cadenes de text amb operadors relacionals

Quan es comparen dues **cadenes** de **text** es **comparen lletra a lletra** pel valor de **l'equivalent numèric** (taula ASCII) de cada **lletra**. Cada lletra té un nombre associat, per exemple, la a és el número 97, la b el 98, etc.

Si comparem **"avellana" < "meló"** obtenim **true**.

Si comparem **"cotxe" > "vehicle"** obtenim **false**.

No obstant això, els codis numèrics poden generar **resultats no previstos**. Per exemple, quin codi numèric és menor, el de la **a** o el de la **A**?

En la taula ASCII, tots els codis numèrics de majúscules són menors que els de minúscules, amb la qual cosa comprovem que **'Zulu' < 'avellaner'** retorna **true** (cosa que a priori ens resultarà estranya, perquè esperem que alfabèticament la z siga major que la a).

Per **comparar** cadenes en base a un ordre **alfabètic** necessitarem usar **altres tècniques** que comentarem més endavant.

5. Operadors aritmètics

En PHP disposem dels **operadors habituals** en els diferents llenguatges de programació. Aquests operadors ens permeten realitzar **operacions aritmètiques**: **suma**, **resta**, **multiplicació**, **divisió**, etc. així com obtenir el **mòdul** o **resta** d'una **divisió** entre dos enters.

Nom	Exemple	Resultat	Exemple (amb $a=8 i $b=4)
Suma	**$a + $b**	El resultat de la suma.	12
Resta	**$a - $b**	El resultat de la resta.	4
Multiplicació	**$a * $b**	El resultat de la multiplicació.	32
Divisió	**$a / $b**	El resultat de la divisió.	2
Resta o mòdul	**$a % $b**	La resta de la divisió de $a entre $b	0

Nota: Els números es **converteixen** a **enters abans d'efectuar l'operació**. És a dir, 9 % 4.5 dóna com a resultat 1 i no 0 perquè calcula la resta de 9 entre 4, no de 9 entre 4.5

L'operador **resta de la divisió entera** o **mòdul** és un operador útil en alguns processos repetitius en programació. Fixa't en els valors que pren quan van progressant els valors que pren una variable. En l'exemple que mostrem a continuació serveix per comptar fins a dos i començar de nou repetitivament.

$a	$a % 3
1	1
2	2
3	0
4	1
5	2
6	0
7	1
8	2

Cal destacar que l'operador % és d'**ús exclusiu entre nombres enters**.

- **7 % 3** retorna 1 ja que la resta de dividir 7 entre 3 és 1.
- 8 % 2 retorna 0 ja que la resta de dividir 8 entre 2 és zero.

Al valor obtingut l'anomenem **mòdul** (en altres llenguatges en lloc del símbol %, s'usa la paraula clau **mod**) i a aquest operador de vegades se li denomina "operador mòdul".

Encara que en altres llenguatges hi ha un operador d'exponenciació per calcular potències, en PHP no és així. Per **calcular una potència** podem fer diverses coses:

a) Recórrer a **multiplicar n vegades el terme**. Per exemple a^3 la podem calcular com a*a*a. Òbviament això no és pràctic per a potències d'exponents grans.

b) Fer **servir un bucle** que done lloc a la repetició de l'operació multiplicació n vegades.

c) Utilitzar **eines pròpies del llenguatge** que permeten realitzar aquesta operació. Aquesta opció és la més senzilla. N'hi ha prou amb escriure **pow (base, exponent)** perquè PHP realitze el càlcul de la potència. Per exemple **pow (2, 3)** retorna 2 elevat a 3 que resulta 8.

Les expressions amb operadors segueixen un **ordre de prelació o de precedència** que determinen l'ordre amb què s'executen. Amb els operadors matemàtics la multiplicació i divisió tenen precedència sobre la suma i la resta. Si hi ha expressions amb diversos operadors del mateix nivell, l'operació s'executa **d'esquerra a dreta**. Per evitar resultats no desitjats, en casos on puga existir dubte es **recomana l'ús de parèntesi** per deixar clar amb quin ordre s'han d'executar les operacions. Per exemple, si dubtes si l'expressió **3*a/7+2** s'executarà en l'ordre que tu vols, especifica l'ordre desitjat utilitzant parèntesi: per exemple **3*((a/7)+2)**.

Operadors d'increment i decrement

Nom	Exemple	Resultat
Pre-increment	**++$a**	Incrementa $a en 1 i després retorna $a
Post-increment	**$a++**	Retorna $a i després incrementa $a en 1.
Pre-decrement	**--$a**	Decrementa $a en 1 i després retorna $a
Post-decrement	**$a--**	Retorna $a i després decrementa $a en 1.

++ i **--** són només vàlids per a **variables numèriques** i serveixen per incrementar una unitat el valor de la variable. Depenent d'on es col·loquen (abans o després de la variable) el resultat del càlcul pot diferir a causa del moment en què s'executa l'addició de la unitat.

Cal tenir en compte que **++**, **--**, **+=**, **-=** i ***=** són expressions que sempre s'apliquen sobre variables. Per exemple **no és vàlid escriure 2++** perquè 2 no és una variable. Totes aquestes operacions es poden substituir per una altra equivalent més evident. Molts programadors prefereixen no usar aquests operadors perquè fan menys llegible el codi. A altres programadors els agrada usar-los perquè els estalvia escriure.

Exemples

Codi exemple2.php.

```
<?php
	$a=8 ;
	echo $a++; // mostra 8
	echo "<br/>" ;
	echo $a; // mostra 9
?>
```

Codi exemple3.php.

```
<?php
	$a=8 ;
	echo ++$a; // mostra 9
	echo "<br/>" ;
	echo $a; // mostra 9
?>
```

En els dos exemples anteriors podem observar clarament la diferència entre el pre-increment i el post-increment. El mateix passa amb el pre-decrement i post-decrement.

Operadors d'assignació

Amb l'ús dels operadors d'assignació, podrem simplificar (**escriure abreujadament**) algunes expressions d'assignació.

Nom	Exemple	Resultat
Suma	`$a += $b;`	`$a = $a + $b;`
Resta	`$a -= $b;`	`$a = $a - $b;`
Multiplicació	`$a *= $b;`	`$a = $a * $b;`
Divisió	`$a /= $b;`	`$a = $a / $b;`
Resta de la divisió entera o mòdul	`$a %= $b;`	`$a = $a % $b;`

Els operadors **+=**, **-=** i ***=** són formes **abreujades** d'escriure operacions habituals. Cal tenir en compte que **++**, **-**, **+=**, **-=** i ***=** són expressions que sempre s'apliquen sobre variables.