CAP. 3
OPERATORI

Operatorii sunt folosiți pentru efectuarea diferitelor operații dintre valori (sau variabile), cunoscute și sub numele de operanzi. Rezultatul unei operații este o valoare ce poate fi asignată în anumite cazuri unei variabile. De multe ori, operatorii convertesc tipul operanzilor din operația respectivă.

Observație: Operatorii au prioritate diferită, deci trebuie avut mereu grijă privind ordinea în care se fac operațiile dintr-o expresie complexă. Dacă vrem să forțăm ordinea în care se fac operațiile, putem folosi paranteze.

```
$a = 2 + 4 * 5; // * are o prioritate mai mare decât +
$a = (2 + 4) * 5;
```

3.1 Operatori aritmetici

Operatorii aritmetici realizează operațiile matematice de baza. Lista operatorilor de acest tip disponibili în PHP este prezentată în Tabel 3.1.

Operatorii aritmetici din PHP lucrează toți în același mod cu exemplele matematice învățate în școală, având totuși mici diferențe. De exemplu, simbolul ’’ * ’’ (asterix) este folosit pentru operația de înmulțire, iar simbolul ’’ / ’’(slash) este folosit pentru operația de împărțire.

Operatorul ’’ % ’’ (modulo) convertește numerele către un întreg, folosindu-se de separatorul zecimal înainte de procesare și returnând rezultatul rămas după ce diviziunea a avut loc, după exemplele care urmează:

```
$a = 5 % 2.5;     // rezultatul este  1, nu 0
$b = 10 % 2;      // rezultatul este  0
```

Observație: Operatorul ’’ % ’’ este folositor pentru a identifica dacă un număr ce este par sau impar. $numar % 2 va returna întotdeauna 0 sau 1 . Dacă rezultatul este 0, numărul este par, iar dacă rezultatul este 1 numărul este impar.

Observație: Operatorii aritmetici convertesc operanzii în valori numerice (**int** sau **float**)!

Operator	Operație	Exemplu
+	Adunare	$x + $y
-	Scădere	$x - $y
*	Înmulțire	$x * $y
/	Împărțire. Dacă împărțirea nu este exactă, rezultatul este de tip **float**	$x / $y
%	Modulo. Rezultatul împărțirii primului operand la cel de-al doilea	$x % $y
**	Ridicare la putere. Primul operand este baza, iar cel de-al doilea este puterea. Operație disponibilă începând cu PHP 5.6	$x ** $y

Exemple de utilizare a operatorilor aritmetici:

```
$x = 10;
$y = 6;

echo $x + $y;  // adunare
echo $x - $y;  // scădere
echo $x * $y;  // înmulțire
echo $x / $y;  // împărțire
echo $x % $y;  // modulo (restul împărțirii)
echo $x ** $y; // ridicare la putere
```

Exemple de conversie a tipurilor de date în cazul operațiilor aritmetice:

```
$a = true;
$b = 3;
$c = 2.3;
$d = "3 lei";
$e = "Ion";

var_dump($a + $b); // int(4) - valoarea boolean true se convertește
în număr întreg și devine 1
var_dump($a + $c); // float (3.3) - idem ca mai sus
var_dump($b + $e); // int(3) - $e este string, și convertit în întreg
este 0. Expresia devine 3 + 0 = 3
var_dump($a + $d); // int(4) - true este convertit în 1, stringul "3
lei" este convertit în 3
```

3.2 Ordinea efectuării operațiilor matematice

Calculele în PHP urmează aceleași reguli și precedențe ca standardele algebrice. În tabelul următor sunt prezentați operatorii în ordinea în care sunt interpretați într-o expresie matematică.

Simbol	Denumire	Descriere
()	Paranteze	Operațiile încadrate de paranteze sunt evaluate primele
**	Ridicare la putere	
++ --	Incrementare/ Decrementare	
* / %	Înmulțire/ Împărțire	Dacă o expresie conține două sau mai multe astfel de operații, acestea sunt evaluate de la stânga către dreapta
+ -	Adunare/ Scădere	Dacă o expresie conține două sau mai multe astfel de operații, acestea sunt evaluate de la stânga către dreapta

3.3 Operatori de atribuire

Operatorul de atribuire este " = ". **$a = 4** atribuie valoarea din dreapta semnului, "**4**", variabilei din stânga, "**$a**".

$a = 4 reprezintă o expresie de atribuire.

Exemple de atribuire:

```
$x = true;
$y = $x;

var_dump($x); // 2
var_dump($y); // 2
```

Operatorul de atribuire poate fi combinat cu operatorii aritmetici sau cu operatorul de concatenare string-uri pentru a minimiza expresiile în care aceeași variabilă apare în stânga și în dreapta operatorului.

Atribuire	Echivalent	Descriere
x = y	**x = y**	Operandul din stânga primește aceeași valoare ca expresia din dreapta
x += y	**x = x + y**	Adunare și atribuire
x -= y	**x = x - y**	Scădere și atribuire
x *= y	**x = x * y**	Înmulțire și atribuire
x /= y	**x = x / y**	Împărțire și atribuire
x %= y	**x = x % y**	Modulo și atribuire
x .= y	**x = x . y**	Concatenare de string-uri și atribuire

Exemple de atribuire:

```
$a = 1;

$a += 2;            // echivalent $a = $a + 2;
var_dump($a);       // 3

$a -= 1;            // echivalent $a = $a - 1;
var_dump($a);       // 2

$a *= 5;            // echivalent $a = $a * 5;
var_dump($a);       // 10

$a /= 2;            // echivalent $a = $a / 2;
var_dump($a);       // 5

$a %= 3;            // echivalent $a = $a % 3;
var_dump($a);       // 2

$a .= 'text';       // echivalent $a = $a . 'text';
var_dump($a);       // 2text
```

3.4 Operatori de comparare

Operatorii de comparare stabilesc dacă două expresii (valori) sunt egale (sau diferite), sau dacă una din ele este mai mare (mai mică), decât cealaltă. Rezultatul operației de comparare este de tip boolean (**true** sau **false**). Lista completă a operatorilor de comparare este prezentată în tabelul următor.

Operator	Exemplu	Descriere
==	$x == $y	Egalitate. Returnează **true** dacă ambii operanzi au aceeași valoare (chiar dacă nu au același tip)
===	$x === $y	Identitate. Returnează **true** dacă ambii operanzi au aceeași valoare și același tip de date
!=	$x != $y	Valori diferite. Returnează **true** daca operanzii au valori diferite
<>	$x <> $y	Idem **!=**
!==	$x !== $y	Returnează **true** daca au valori diferite sau tip de date diferit
>	$x > $y	Mai mare. Returnează **true** dacă operandul din stânga este mai mare decât cel din dreapta
<	$x < $y	Mai mic. Returnează **true** dacă operandul din stânga este mai mic decât cel din dreapta
>=	$x >= $y	Mai mare sau egal. Returnează **true** dacă operandul din stânga este mai mare sau egal decât cel din dreapta
<=	$x <= $y	Mai mic sau egal. Returnează **true** dacă operandul din stânga este mai mic sau egal decât cel din dreapta
<=>	$x <=> $y	Operatorul “navă spațială” (spaceship) este folosit pentru a compara două expresii. Returnează -1, 0 sau 1 când $x este, respectiv, mai mic decât, egal sau mai mare decât $y. Introdus în versiunea 7 de PHP

Exemple de utilizare a operatorilor de comparație:

```
$x = 100;
$y = "100";

var_dump($x == $y);   // true deoarece cele două valori sunt egale
var_dump($x === $y); // false deoarece cele tipurile de date sunt
diferite
var_dump($x != $y);   // false deoarece cele două valori sunt egale
var_dump($x <> $y);   // false deoarece cele două valori sunt egale
var_dump($x !== $y); // true deoarece cele tipurile de date sunt
diferite

$x = 100;
$y = 50;

var_dump($x > $y);    // true deoarece $x este mai mare decât $y

$x = 10;
$y = 50;
```

```php
var_dump($x < $y);    // true deoarece $x este mai mic decât $y

$x = 50;
$y = 50;

var_dump($x >= $y);   // true deoarece $x este mai mare sau egal cu $y
var_dump($x <= $y);   // true deoarece $x este mai mic sau egal cu $y

$x = 5;
$y = 10;

echo ($x <=> $y);     // -1 deoarece $x este mai mic decât $y

$x = 10;
$y = 10;

echo ($x <=> $y);     // 0 deoarece valorile sunt egale

$x = 15;
$y = 10;

echo ($x <=> $y);     // +1 deoarece $x este mai mare decât $y
```

Observație: Când se folosește un operator de comparare pentru a compara un string cu un număr, ambii operanzi sunt convertiți în numere (int sau float). Același lucru se întâmplă când ambii operanzi sunt string-uri numerice.

```php
$a = 2;
$b = '3';

var_dump($a < $b); // true - '3' este convertit în int(3)

$a = 'text';
$b = 0;

/*
Când se compară un string cu un număr, se convertesc ambii operanzi
în numere.
În acest caz 'text' este convertit în 0
*/
var_dump($a == $b); // true

$a = '3text';
$b = '2';

var_dump($a == $b); // false - '3text' este convertit în 3
```

3.5 Operatorii de incrementare/decrementare

Operatorul '**++**' se numește operator de incrementare și are ca rezultat creșterea valorii unei variabile cu o unitate. Operatorul '**--**' se numește operator de decrementare și are ca rezultat scăderea valorii unei variabile cu o unitate. Operatorii de incrementare și decrementare sunt prezentați în tabelul următor.

Operator	Denumire	Descriere
++$x	Pre-incrementare	Incrementează valoarea lui $x cu o unitate după care returnează $x
$x++	Post-incrementare	Returnează $x după care incrementează valoarea lui $x cu o unitate
--$x	Pre-decrementare	Decrementează valoarea lui $x cu o unitate după care returnează $x
$x--	Post-decrementare	Returnează $x după care decrementează valoarea lui $x cu o unitate

Poziția operatorilor este importantă deoarece efectul asupra calculului poate fi schimbat. Când operatorii sunt înaintea variabilei, 1 este adăugat sau scăzut înaintea realizării calculului, după exemplul care urmează:

```
$x = 5;
$y = 6;
print(--$x * ++$y); // rezultatul este 28 (4 * 7)
print($x-- * $y++); // rezultatul este 30 (5 * 6)
```

3.6 Operatori logici

Operatorii logici sunt folosiți pentru a opera cu valori de adevăr (boolene) iar rezultatul acestor operatori este tot o valoare booleana. Aceștia sunt prezentați în tabelul următor.

Operator	Denumire	Descriere
and	Și logic	Rezultatul este **true** dacă ambii operanzi sunt evaluați ca fiind **true**
or	Sau logic	Rezultatul este **true** dacă cel puțin unul dintre cei doi operanzi este evaluat ca fiind **true**
xor	Sau exclusiv	Rezultatul este **true** dacă operanzii sunt diferiți și **false** dacă sunt egali
&&	Și logic	Rezultatul este **true** dacă ambii operanzi sunt evaluați ca fiind **true**
||	Sau logic	Rezultatul este **true** dacă cel puțin unul dintre cei doi operanzi este evaluat ca fiind **true**
!	Negare	Inversează valoarea de adevăr a unei epresii

Exemple de utilizare a operatorilor logici:

```
$a = true;
$b = !$a;                       // $b devine false
var_dump($b);                   // bool(false)

var_dump(false && false);    // false
var_dump(false && true);     // false
var_dump(true && false);     // false
var_dump(true && true);      // true


var_dump(false || false);    // false
var_dump(false || true);     // true
var_dump(true || false);     // true
var_dump(true || true);      // true


var_dump(false xor false);   // false
var_dump(false xor true);    // true
var_dump(true xor false);    // true
var_dump(true xor true);     // false
```

Observație: Se pot folosi paranteze pentru a dicta ordinea operațiilor.

```
$a = false && false || true;

var_dump($a); // true

$a = false && (false || true);

var_dump($a); // false
```

Observație: Operatorii logici evaluează operanzii ca valori booleene.

```
$a = 1;
$b = 0;

var_dump($a || $b); // bool(true)
/*
1 este intreg. Convertit la boolean devine true
0 este intreg. Convertit la boolean devine false
$a și $b au fost convertite doar în cadrul operației logice.
Ele își păstrează tipul de date inițial
*/
var_dump($a);         // int(1)
var_dump($b);      // int(0)

var_dump("" || 0); // bool(false) - șirul "" este evaluat ca false,
la fel și 0.
```

Observație: Operatorii "**&&**", "**and**", "**||**", "or", funcționează cu "scurtcircuitare", adică al doilea operand nu este evaluat dacă se poate stabili rezultatul sigur al operației logice doar în funcție de primul operand. Acest lucru este important atunci când al doilea operand reprezintă apelarea unei funcții, deoarece funcția poate fi (sau nu) apelată în funcție de acest lucru.

```
function scrie($ceva){
    echo $ceva;
    return false;
}

$a = true || scrie('ceva cu ||'); // funcția scrie() nu este apelată
deoarce primul operand al lui || (sau logic) este true, deci
rezultatul operației va fi true indiferent de al doilea operand

$a = false || scrie('altceva cu ||'); // în acest caz nu se poate
stabili rezultatul operației doar în funcție de primul operand, deci
se va executa și funcția
```

3.7 Operatorul ternar

Operatorul ternar seamănă cu un "**if...else...**" simplu. Așa cum îi spune și numele, este un operator ce folosește trei operanzi. Structura operatorului ternar este următoarea:

(conditie) ? valoare_daca_true : valoare_daca_false

Se evaluează valoarea de adevăr a condiției dintre paranteze. Dacă această condiție este **true**, rezultatul expresiei este valoarea de după "**?**". Daca este **false**, rezultatul expresiei este valoarea de după ":".

```
$a = (2 < 3) ? 'text1' : 'text2';

echo $a; // text1

$a = (2 > 3) ? 'x' : 'y';

echo $a; // y
```

3.8 Operatorul Null coalescing

Operatorul "**??**" sau "Null colescing" a fost introdus în versiunea 7 de PHP și verifică dacă o variabilă/expresie are valoarea NULL.

Structura operatorului null coalescing este următoarea:

$x = *expr1* **??** *expr2*

Dacă **expr1** nu este NULL **$x** va avea valoarea returnată de **expr1**, dar dacă **expr1** este NULL, atunci **$x** va avea valoarea returnată de **expr2**.

```php
$a = null;
$b = 3;

$x = $a ?? 'text1';
$y = $b ?? 'text2';

echo $x; // text1

echo $y; // 3
```

Teme

1. Creați o funcție care primește 2 parametrii și afișează în pagină rezultatul următoarelor operații matematice (+, -, *, /, %, **) cu cei doi parametri pe post de operanzi. La final se va afișa ce operație produce cel mai mare rezultat.
2. Creați o funcție care primește un parametru. Aceasta trebuie să verifice dacă acel parametru se află în intervalul 0 - 10 (capete incluse). În pagină se va afișa tabla de înmulțire de la 0 la 10 pentru acel parametru (ex: dacă parametrul este 4 va se va afișa în pagină pe câte o linie distincta: 4 x 0 = 0; 4 x 1 = 4; ... 4 x 10 = 40;). Puteți construi o funcție ajutătoare.
3. Creați o funcție care primește doi parametri și afișează mesaje de genul: "2 este mai mic decât 3; 2 este mai mic sau egal cu 3; .. etc." în funcție de valorile de adevăr pentru următoarele comparații (<, <=, ==, >=, >). Se vor afișa cate 5 mesaje pentru fiecare apelare a funcției.
4. Creați o funcție care primește trei parametri. Dacă parametrul 2, respectiv 3 sunt 0 sau mai mici decât 0, atunci aceștia vor primi în interiorul funcției valoarea 1, respectiv 2 (folosind operatorul ternar). Dacă diferența dintre primii 2 parametrii este mai mare decât 5 și al treilea parametru este număr par, atunci afișați rezultatul operației următoare: (param1 împărțit la param2) ridicat la puterea param2.