2 ELEMENTE FUNDAMENTALE ALE LIMBAJULUI PHP (I)

2.1 INTRODUCERE

Ceea ce face calculatorul mai bine decat noi este sa proceseze date, iar pentru aceasta programatorul are nevoie de urmatoarele elemente fundamentale ale unui limbaj de programare:

- o modalitate de a specifica locatia datelor ce se doresc a fi prelucrate
- un mod de a preciza sau impune felul datelor ce fac subiectul prelucrarii (numere, text etc)
- modalitati de a efectua operatii cu datele

Aceste 3 necesitati se materializeaza in cazul PHP (si nu numai) in urmatoarele notiuni:

- **variabile** – nume date unor locatii de memorie, care permit programatorului sa indice datele pe care le prelucreaza programul folosind un simplu nume (o eticheta text) si nu o adresa de memorie
- **tipuri de date** – variabilele pot avea valori de diverse tipuri: numere intregi, numere cu zecimale, valori de adevar, siruri de caractere etc. De tipul de date al variabilei depind spatiul ocupat in memorie, structura zonei de memorie corespunzatoare si operatiile care se pot aplica variabilei
- **operatori** – elemente de sintaxa ale limbajului care aplica o operatie uneia sau mai multor valori, producand un rezultat

2.2 INSTRUCTIUNI PHP SI SEPARAREA LOR

Un program PHP este format dintr-un ansamblu de instructiuni, care pot fi de doua feluri:

- instructiuni simple – in cele mai dese cazuri sunt instructiuni de o linie, desi pot fi impartite in mai multe linii daca programatorul o doreste. Ele se incheie intotdeauna cu ; cu exceptia cazului in care instructiunea in cauza este ultima dinaintea delimitatorului de final al sectiunii de cod PHP (vezi exemplul de mai jos)
- instructiuni compuse (bloc) – reprezinta un ansamblu de instructiuni delimitate prin {}. Acolada de inchidere tine loc de ';'

Exemplu – instructiuni simple:

```
<?php
echo "Acest text va fi afisat pe ecran ";
// putem omite delimitatorul de final ';' inainte de inchiderea sectiunii de cod
echo "sau in browser"
?>
```

Exemplu – instructiuni compuse (exemplul foloseste instructiuni decizionale, care vor fi invatate intr-o lectie ulterioara):

```
if($a == 4){      // instructiune decizionala simpla; daca valoarea lui $a este 4,
      $a = 0;     // se executa instructiunile dintre acolade
      echo "Variabila resetata";
}                 // lipseste delimitatorul de final ;
```

2.3 COMENTARII

Comentariile sunt portiuni de text inserate in codul sursa al unui program cu scop pur informativ (documentarea codului) si care sunt ignorate de catre interpretorul PHP. Pentru a fi tratate ca atare, ele trebuie marcate folosind una dintre modalitatile urmatoare:

- comentarii de o linie, varianta 1 (pe stil C) – incep cu //. Interpretorul va ignora portiunea de linie incepand cu //, fie pana la sfarsitul liniei, fie pana la intalnirea tagului de inchidere a sectiunii PHP (exceptie: cand sunt folosite tagurile <script>…</script>, tagul de inchidere nu incheie comentariul)
- comentarii de o linie, varianta 2 (pe stil Perl sau scripturi de shell Unix) – incep cu #. Acelasi comportament ca cele de mai sus

```
<html><body>Continut:<br />
<?php
echo "Text"; // restul acestei linii (incepand cu //) este ignorat de catre interpretor
echo "More text"; # acelasi efect, dar incepand cu caracterul #
echo "Iata-ne la "; // in ciuda comentariului, textul "sfarsit" se va afisa! ?>sfarsit
</body></html>
```

- comentarii ce se extind pe mai multe linii – sunt cuprinse intre /* si */. Nu sunt permise comentariile de acest fel imbricate (cuprinse unul in altul)

```
/* Portiune de cod comentata integral:
      echo 'Executia a ajuns in acest punct';
      // mesajul din echo NU va fi afisat
*/
```

2.4 VARIABILE

2.4.1 Ce este o variabila

Variabilele in programare sunt zone de memorie in care se stocheaza date si la care programatorul se refera folosind un nume (o eticheta text). Cand programul ruleaza, datele sunt incarcate in memorie la o anumita adresa (de obicei alta de fiecare data), pe care programatorul nu o poate cunoaste in momentul scrierii programului. Variabilele ii dau posibilitatea programatorului de a se referi la zonele de memorie in care sunt stocate datele programului folosind nume, independent de locatia efectiva in care vor fi ele memorate la rularea programului.

Spre deosebire de variabilele din matematica, cele din programare primesc de-a lungul programului diferite valori, in functie de operatiile in care sunt implicate. In zona de memorie corespunzatoare variabilei poate fi memorata initial valoarea 5, apoi 13, apoi 45 etc. O particularitate a variabilelor PHP este aceea ca o variabila poate primi succesiv valori de tipuri diferite (ex: initial valoarea sa este 45, apoi sirul de caractere "PHP", apoi valoarea de adevar TRUE (adevarat) etc).

2.4.2 Numele variabilelor in PHP

Numele de variabile PHP incep intotdeauna cu $, urmat de o combinatie de litere, cifre si underscore (_) . Primul caracter de dupa $ este obligatoriu litera sau underscore.

Atentie! *Numele de variabile PHP sunt case-sensitive, in vreme ce numele de functii si clase nu sunt!*

Exemplu – nume de variabile valide:

```
$a = 3; // se creaza variabila cu numele $a careia i se asigneaza valoarea 3
$A = 5; //se creaza o alta variabila numita $A careia i se asigneaza valoarea 5.
// $a si $A sunt variabile distincte!
$_x="text"; // crearea variabilei cu numele $_x care primeste o valoare de tip string
```

Exemplu – nume de variabile invalide:

```
$6din49 = 56;           // invalid, deoarece dupa $ urmeaza cifra
$alba-neagra = "gri";   // invalid, deoarece contine caracterul -
```

2.4.3 Initializarea si folosirea unei variabile

In PHP, variabilele nu se declara inaintea folosirii! In alte limbaje, o declaratie de tip **int x;** specifica de la bun inceput tipul de date al variabilei si rezerva un identificator (nume), astfel incat compilatorul/interpretorul putea verifica daca o

variabila a fost definita inaintea folosirii. In PHP, neexistand declarare, orice variabila poate fi folosita direct, cu riscul ca ea sa nu fi fost definita anterior.

```
// initializarea unei variabile cu o valoare
$a = 5; // este creata variabila $a; in locatia sa de memorie se stocheaza valoarea 5
$a = 7; // valoarea precedenta din memorie (5) este suprascrisa cu valoarea 7
echo $a; // afisarea valorii lui $a - se afiseaza 7
echo $b; // $b nu exista, se va afisa sirul vid "" (pe ecran neaparand astfel nimic) si
            // eventual o eroare de tip notice, in functie de configurarea PHP
```

O variabila poate fi stearsa folosind functia predefinita unset:

```
$a = 3; unset($a);
```

Nota: *O variabila nedefinita sau stearsa cu unset($var) va avea valoarea NULL (case-insensitive), care in functie de context se poate converti automat la sirul vid "", 0, false etc. Putem verifica daca o variabila a fost sau nu definita folosind functia* ***isset().***

2.4.4 Tipul de date al unei variabile

Variabilele PHP pot fi folosite pentru a memora date de diverse feluri – iata cateva exemple:

- date numerice – prin intermediul tipului de date *int* (numere intregi) sau *float* (numere cu zecimale)
- valori de adevar – tipul de date *boolean*
- siruri de caractere – tipul de date *string*

(Pentru o discutie detaliata a listei complete de tipuri de date consultati sectiunea din material dedicata lor)

Tipul de date al unei variabile PHP se poate schimba de-a lungul existentei variabilei, in functie de valoarea asignata acesteia. Este posibil ca la un moment dat o variabila sa aiba valoarea numerica 23, iar mai apoi sa capete ca valoare sirul de caractere "text" sau valoarea de adevar FALSE. Aceasta spre deosebire de alte limbaje (ex: C,Java), in care tipul de date al unei variabile era stabilit la declararea acesteia si nu se schimba de-a lungul programului.

```
$a = 45; echo gettype($a);                    // integer (numar intreg)
$a = "patruzecisicinci"; echo gettype($a);    // string (sir de caractere)
```

2.4.5 Afisarea valorii variabilelor

PHP ofera un set bogat de functii si constructii ale limbajului[1] care permit afisarea valorii variabilelor oricare ar fi tipul de date al acestora.

Atentie! *Unele dintre functiile folosite pentru afisarea valorii variabilelor convertesc argumentul primit la string (sir de caractere), rezultatele nefiind intotdeauna cele scontate:*

```
$a = true;
echo $a; // afiseaza 1, deoarece valoarea true este convertita la tipul de date string
```

Printre functiile si constructiile disponibile se numara:

- **echo(**string $arg1, …) - nu este functie, ci constructie a limbajului. Poate primi unul sau mai multe argumente separate prin virgula, pe care le converteste la string (daca e cazul) inainte de a le afisa valoarea. Se poate apela cu sau fara paranteze:

Exemple:

[1] constructii ale limbajului = elemente de limbaj care seamana cu functiile prin modul de apelare insa nu au toate caracteristicile functiilor, diferind deseori fata de acestea prin sintaxa, valoare returnata (care poate lipsi) etc

```
$a = 5; $b = 6; // se creeaza doua variabile de tip int, initializate cu valorile  5 si 6
echo ($a);  // afiseaza valoarea lui $a – numarul intreg 5
echo $b;    // folosire fara paranteze; afiseaza valoarea lui $b – numarul intreg 6
echo $a,$b; // afiseaza 56 (valoarea lui $a urmata imediat de valoarea lui $b)
```

```
$a = 5; $b = false; // de data aceasta $b are tipul de date boolean
echo $a,$b; // afiseaza 5 (valoarea lui $a); valoarea lui $b este convertita la string
rezultand sirul vid "", care la afisarea pe ecran nu produce nici un efect
```

Exemplul urmator foloseste operatorul . care realizeaza concatenarea string-urilor ce il bordeaza:

```
$a = 4; // este creata variabila $a de tip int initializata cu valoarea 4
echo " \$a este :".print($a); // afiseaza 4 $a este 1
```

In acest ultim exemplu, justificarea output-ului este ca intai se evalueaza argumentul lui echo, care este format prin concatenarea intre stringul "\$a este" si rezultatul intors de constructia print, si abia apoi se executa echo primind ca argument valoarea evaluata. A fost necesara precedarea lui $ cu \ pentru a preintampina tratarea lui $a ca un nume de variabila si substituirea lui cu valoarea variabilei (vezi detalii la tipuri de date→string).

- **print(**string $arg**)** – asemanator cu echo, dar primeste un singur argument si returneaza intotdeauna 1

```
$a = 5; $b = 6; //crearea a doua variabile de tip int, initializate cu valorile 5 si 6
echo ($a); // afisarea valorii lui $a – numarul intreg 5
print($b); // afiseaza valoarea lui $b – numarul intreg 6
echo $a,$b; // afiseaza valorile ambelor variabile
print $a,$b; // eroare, print nu accepta mai multe argumente
echo (print($a)); // 51 – intai se executa print, apoi este afisat rezultatul returnat de
                  // print, adica 1
print(echo($a));  // eroare, fiindca echo nu este functie si nu returneaza nimic, asadar
                  // nu poate fi folosit ca argument pentru print
```

- **print_r**(mixed $expression [, bool $return]) – printeaza sau returneaza valoarea expresiei (in functie de valoarea lui $return). NU printeaza tipul de date al expresiei, spre deosebire de var_dump(). Este in general util la afisarea tablourilor, deoarece afiseaza in mod recursiv fiecare element al tabloului (spre deosebire de echo si print care ar afisa doar "Array")

```
$tablou = array(1, 2, 3);
echo $tablou; // Array
print_r($tablou); /* Afiseaza:
Array
(
    [0] => 1
    [1] => 2
    [2] => 3
)*/
```

- **var_dump(**mixed $expr1,…**)** – cea mai completa functie; afiseaza atat tipul de date cat si valoarea expresiei primite ca argument; pentru tablouri afiseaza fiecare element si tipul sau de date, in mod recursiv:

```
$tablou = array(1, 2, 3);
var_dump($tablou); /* Afiseaza:
array(3) {
  [0]=>  int(1)
  [1]=>  int(2)
  [2]=>  int(3)
}
*/
```

ZCE hint: aveti grija sa cunoasteti bine diferentele intre diferitele modalitati de afisare de variabile!

2.4.6 Variabile predefinite

In cadrul unui program PHP sunt disponibile doua tipuri de variabile:

- variabile definite de catre programator (conform regulilor expuse mai sus)
- variabile predefinite, care au urmatoarele caracteristici esentiale:
  - contin date provenite de la clientul/utilizatorul web si informatii importante despre serverul web, client, conexiune
  - sunt vizibile automat in orice context al programului (inauntrul sau in afara functiilor, structurilor de control etc) fara ca programatorul sa trebuiasca sa ia vreo masura in acest sens. Din acest motive mai sunt numite si "superglobals"

Variabila	Descriere
$_SERVER	informatii oferite de serverul web: headere HTTP primite de la client, calea catre script, informatii despre server
$_GET	input provenit de la utilizator: variabile pasate programului PHP prin intermediul comenzii HTTP GET (ex: http://www.example.com/index.php?**id=132&page=4**)
$_POST	variabile pasate programului PHP prin intermediul comenzii HTTP POST (de ex, cand utilizatorul completeaza un formular si apoi apasa Submit)
$_COOKIE	informatii stocate anterior de catre server pe hard-disk-ul clientului sub forma de cookies si retrimise serverului la urmatoarea cerere HTTP efectuata de client
$_SESSION	variabile de sesiune (variabile ale calor valori se pastreaza intre doua cereri succesive ale clientului HTTP)

Aceste variabile sunt indispensabile lucrului cu formulare HTML si input de la utilizator. Detalii despre continutul si utilizarea lor vor fi incluse intr-o lectie viitoare.

2.5 CONSTANTE

Constantele reprezinta zone de memorie referite prin intermediul unui nume al caror continut, odata initializat, nu se mai schimba. In PHP ele au urmatoarele caracteristici:

- se creeaza folosind constructia **define**, ce primeste urmatorii doi parametri, in aceasta ordine:
  - numele constantei, sub forma de string
  - valoarea constantei, specificata conform cu tipul de date dorit (vezi tipuri de date)
- numele lor este case-sensitive si de obicei este format din majuscule
- au vizibilitate automata in intreg scriptul PHP ("global scope")
- valoarea unei constante
  - poate fi doar scalar (integer, float, boolean, string)
  - se citeste (se afiseaza, se foloseste in expresii etc) fara prefixarea numelui constantei cu $
- lista de constante definite se poate obtine cu functia **get_defined_constants()**

```
define('CURS', 'PHP');          // numele constantei este specificat folosind apostroafe
define("NUME", "Mihai");        // numele constantei este specificat folosind ghilimele
define("VARSTA", 20);           // valoarea este de tip integer, nu-i mai punem ghilimele

echo "Ma cheama " . NUME;                       // fara $ in fata numelui constantei!
echo  ", am " . VARSTA . "ani";                 // Folosirea constantei in expresii care
echo ", si sunt la cursul de " . CURS;          // utilizeaza operatorul de concatenare .

echo "Am facut " . (VARSTA+10). " de ani";      // constanta ca operand in adunare

CURS = "Java";    // eroare, constantele nu pot fi modificate
```

2.4 TIPURI DE DATE IN PHP

2.4.1 Descriere generala

Tipul de date pe care o variabila PHP il are la un moment dat poate fi determinat de doua operatii:

- ultima valoare asignata variabilei. De exemplu, o variabila poate fi creata asignandu-i-se un numar intreg, pentru ca mai apoi sa capete valoarea true (de tip boolean = valoare de adevar).
- ultima conversie explicita aplicata variabilei. Tipul de date al unei variabile poate fi schimbat fara a-i asigna o valoare de alt tip, folosind functia predefinita *settype()*

Pentru ca o valoare prezenta ca atare in sursa programului (asignata unei variabile, sau parte a unei expresii) sa fie considerata de catre interpretor ca fiind de tip integer, boolean etc este necesar ca valoarea sa fie reprezentata corespunzator. Spre exemplu, *true* si *false* sunt cele doua valori posibile pentru tipul de date boolean, iar numerele intregi sunt considerate automat de tip integer:

```
$a = true; // este creata variabila cu numele $a de tip boolean, cu valoarea true
$a = 4; // in locatia referita de $a este acum memorata valoarea 4, cu tip de date int
settype($a, 'string'); // se schimba explicit tipul de date al lui $a in sir de caractere
var_dump($a); // string(1) "4"
```

Tipurile de date PHP pot fi impartite in urmatoarele categorii:

- **tipuri de date scalare** - variabilele de acest tip pot memora o singura valoare la un moment dat
  - numerice – **int** si **float**
  - valori de adevar – **boolean**
  - siruri de caractere - **string**
- **tipuri de date compuse** - valoarea unei variabile ce are astfel de tip de date este formata din informatii multiple, fiecare putand fi de alt tip de date
  - **array** – succesiune de perechi cheie→valoare, unde cheia poate fi integer sau string iar valoarea poate avea orice tip de date
  - **object** – entitate care inglobeaza atat o colectie de date de diverse tipuri cat si functiile necesare prelucrarii lor
- **tipuri de date speciale**
  - **resource** – tip de date pentru variabile ce memoreaza informatii externe PHP, obtinute prin functii speciale (ex: rezultatul interogarii unei baze de date, referinta catre un fisier deschis etc)
  - **NULL** – folosit pentru a indica faptul ca o variabila nu are valoare (fie din cauza ca nu a fost definita, fie in urma stergerii explicite)

Vor fi prezentate in continuare tipurile de date scalare, urmand ca cele compuse si speciale sa fie detaliate in lectii ulterioare, dedicate (tablouri, clase si obiecte, lucrul cu fisiere si stream-uri, lucrul cu baze de date etc).

2.4.2 bool sau boolean

2.4.2.1 Descriere si modalitati de specificare

Folosit pentru a memora valori de adevar (adevarat sau fals). Valorile posibile pentru acest tip de date sunt *true* sau *false*. Cele doua valori sunt case-insensitive si NU trebuie incluse intre (case-insensitive,)

```
$a = true;  //se creeaza variabila $a de tip boolean si i se asigneaza valoarea “true”
$b = FALSE; //se creeaza variabila $b de tip boolean si i se asigneaza valoarea “false”
```

```
$b = 8;
$c= (5>4);          // se creeaza variabila c de tip boolean si i se atribuie rezultatul
                    // evaluarii expresiei 5>4, adica true (vezi operatori de comparare)
$c = ($b > 7);      // idem, dar in comparatie poate interveni o variabila
```

```
$nume=”dan”; // se creeaza variabila $nume de tip string, initializata cu stringul dan
$d= ($nume==”ion”);// se creeaza variabila de tip boolean $d si i se asigneaza false
                   // (rezultatul evaluarii expresiei ($nume==”ion”) )
```

Atentie! *Daca includeti cele doua valori posibile (true si false) intre ghilimele sau apostroafe, ele vor fi interpretate ca string!*

```
$a = "true";      // se creeaza variabila $a de tip string
```

2.4.2.2 Utilitatea tipului de date boolean

Un program (fie el scris in PHP sau in alt limbaj) presupune de obicei luarea de decizii bazate pe valori ale variabilelor, rezultate in urma prelucrarilor efectuate de program. Pentru luarea deciziilor se fac comparatii, rezultatul acestora fiind un verdict de tip "adevarat" sau "fals" (vezi operatori de comparare)

Exemplu: un utilizator completeaza un formular web in care isi introduce, printre altele, varsta si apasa Submit. Scriptul PHP care primeste datele introduse va verifica daca valoarea introdusa pentru varsta este mai mica decat varsta umana maxima posibila (fie ea 120) si va obtine un rezultat de tip "adevarat" sau "fals"; in cazul "fals", se va executa o portiune de cod ce determina afisarea unei erori catre utilizator, iar in cazul "adevarat" input-ul de la utilizator va fi acceptat si prelucrat.

```
if($varsta > 120){
      echo "Ati introdus o varsta invalida!";
}else{
 // alte instructiuni, pentru prelucrarea datelor
 …
}
```

Nota: *Exemplul de mai sus foloseste operatorul de comparare > si instructiunea decizionala if, care vor fi detaliate intr-o lectie viitoare.*

2.4.3 int sau integer

Folosit pentru memorare de numere intregi, **cu semn**. Desi in general este memorat pe 32 de biti, **dimensiunea lui depinde de platforma**; anvergura tipului de date int este memorata in constanta PHP_INT_SIZE, iar valoarea sa maxima in PHP_INT_MAX.

Daca, in urma unei operatii, valoarea unei variabile de tip int depaseste valoarea maxima a acestui tip de date, ea trece automat in float. De asemenea, daca impartirea a doi intregi nu este exacta, rezultatul va fi un float (nu mai exista impartire intreaga, ca in alte limbaje).

Pentru a fi recunoscuta ca atare, valoarea unui integer poate fi specificata in codul sursa mai multe feluri:

```
$x = 4;           // numar intreg, zecimal, pozitiv
$y = -15758;      // numar intreg, zecimal, negativ
$z = 0x5a;        // specificare in hexazecimal; echivalent cu $z = 90;
$t = 067;         // specificare in octal; echivalent cu $t = 55;
```

Observatie: *un numar in hexazecimal este recunoscut dupa prefixul 0x ce precede succesiunea de digiti, iar un numar octal are ca simbol distinctiv cifra 0 din fata.*

Ori de cate ori interpretorul va intalni in codul sursa un numar specificat intr-unul dintre aceste 4 feluri, il va considera automat de tip int. De aceea tipul de date al unei variabile care primeste o astfel de valoare devine int:

```
$a = true;  // tipul de date al lui $a este boolean, pt interpretorul
            // recunoaste true ca fiind o valoare de tip boolean
```

```
$a = 15;     // tipul de date al lui $a devine int, deoarece 15 este una dintre
             // reprezentarile corecte pentru o valoare de tip int
```

2.4.4 float

Folosit pentru memorarea numerelor cu zecimale. Numarul de biti folositi pentru stocarea sa in memorie este, ca si in cazul int, dependent de platforma.

Modalitati de a specifica valoarea unui float:

```
$f1 = 5.14;
$f2 = 514E-2;      // 514 x 10-2
$f3 = 0.0514e2;    // 0.0514 x 102
```

Nu toate numerele cu zecimale pot fi reprezentate cu acuratete folosind acest tip de date. De aceea nu este recomandabila verificarea egalitatii a doua variabile de tip float:

```
echo (int)((0.1+0.7)*10); // rezultatul este 7!!
printf("%.15f", (19.6*100)); // afisare cu 15 zecimale; rezulta 1960.000000000000227 !!

// lucrul cu numere foarte mari; float are si aici precizie limitata
$a1 = 12345678901234568;
$a2 = 12345678901234567;       // $a1 -1
var_dump($a1 == $a2);          // true!!
```

2.4.5 string

2.4.5.1 Descriere generala

Acest tip de date este folosit in general pentru a memora siruri de caractere. Multe informatii intalnite in programare consta din succesiuni de caractere: denumirile produselor dintr-un magazin virtual, numele si adresele persoanelor aflate intr-o baza de date etc.

De retinut insa ca, in PHP, o variabila de tip string poate contine o succesiune de date binare (nu neaparat caractere printabile – litere, cifre etc) si ca multe dintre functiile PHP predefinite care lucreaza cu string sunt binary-safe[2].

2.4.5.2 Modalitati de definire

Pentru ca o valoare din codul sursa sa fie considerata de tip string, este necesara reprezentarea ei intr-una dintre urmatoarele forme:

2.4.5.2.1 Includerea valorii intre ghilimele

Valoarea delimitata de ghilimele va fi considerata automat de tip string:

```
$s = "carpe diem";// "carpe diem" este recunoscut ca string, in consecinta variabila
                  //  $s va avea acum tipul de date string
```

Din cauza faptului ca ghilimelele sunt folosite ca delimitatori, aparitia lor in cadrul sirului de caractere ar putea cauza o interpretare gresita:

```
$s = "Constructia "heredoc" va fi discutata mai jos";
                  ↳   // aceste ghilimele vor fi interpretate ca delimitator de final
```

[2] functii binary-safe = functii de manipulare de stringuri care trateaza un string ca pe o simpla insiruire de octeti, fara a se astepta ca stringul sa aiba un anumit format sau sa contina exclusiv caractere printabile (litere, cifre etc)

```
                                    // al sirului Constructia, rezultand o eroare la executie
```

De aceea, pentru a reprezenta caracterul " in cadrul unui string, este necesara precedarea acestuia cu caracterul \, care are rol de *escape character* (=elimina semnificatia speciala a caracterului ce-l urmeaza). In acest fel, " nu mai este interpretat ca delimitator de final al string-ului:

```
$s = "Constructia \"heredoc\" va fi discutata mai jos";
```

Exista si alte caractere cu semnificatii speciale in cadrul unui string, care trebuie la randul lor precedate de \ daca se doreste includerea caracterului in cauza in string:

- \ - are dublu rol:
  - de escape character (discutat mai sus)
  - de indicator al unei secvente speciale – ex: \n este reprezentarea pentru newline, \t pt tab etc

Observatie: *\n semnifica trecerea pe linia urmatoare atunci cand output-ul programului se afiseaza in linia de comanda. Atunci cand scriptul PHP genereaza o pagina web, \n cauzeaza doar trecerea pe linia urmatoare in codul sursa HTML, ceea ce in browser nu va avea nici un efect vizual. Pentru a trece pe linia urmatoare in HTML se foloseste tag-ul
 sau
.*

- $ - indica inceputul unui nume de variabila

```
$s = "Caracterul \\ are rol de \"escape character\" "; echo $s;
// Caracterul \ are rol de "escape character"

$s = "Linia 1\nLinia2"; echo $s;
// Linia 1
// Linia 2

$a = 5;
$s = "Variabila \$a are valoarea $a"; echo $s;
// Variabila $a are valoarea 5
```

In cadrul unui string specificat prin includerea intre ghilimele se efectueaza automat substituirea numelor de variabile cu valoarea lor. Atunci cand interpretorul intalneste in cadrul unui string caracterul $, incearca sa formeze un nume de variabila folosind $ si cat mai multe caractere ce-i urmeaza (oprindu-se la primul caracter invalid pentru nume de variabile), si apoi substituie acest nume cu valoarea variabilei in cauza:

```
$s = "curs"; $s1 = "Merg la un $s"; // Merg la un curs
$s2 = "Foaie de par$s";             // Foaie de parcurs
```

Daca imediat dupa numele variabilei urmeaza alte caractere valide, interpretorul va forma un nume mai lung decat am dori, rezultatul fiind un nume de variabila care fie nu a fost definita, fie va fi o variabila decat a intentionat programatorul. De aceea, in astfel de cazuri este necesara delimitarea clara a numelui de variabila, lucru care se realizeaza folosind acolade:

```
// interpretorul incearca sa foloseasca cat mai multe caractere pt numele de variabila
$s = "curs";
$s1 = "Merg la mai multe $suri";    // Merg la mai multe – variabila $suri nu este
                                    // definita si se va substitui cu sirul vid ""
$s2 = "Merg la mai multe ${s}uri"; // Merg la mai multe cursuri – numele variabilei este
                                    // delimitat explicit
$s3 = "S{$s}uri ale societatii";    // un alt exemplu de delimitare explicita
```

2.4.5.2.2 Includerea valorii intre apostroafe

Atunci cand un sir de caractere din codul sursa este delimitat cu apostroafe, el va fi considerat automat de tip string, insa in interiorul sau nu se va mai face substitutia variabilelor:

```
$s = 'Variabila $a are valoarea true'; // nu este nevoie de \$a, ca in cazul ghilimelelor
```

In aceste conditii, $ nu mai este caracter special si nu mai trebuie precedat de \. In schimb, delimitatorul de string (apostroful) este cel care, in cazul aparitiei in string, trebuie marcat corespunzator:

```
// apostroful din mijloc ar fi fost interpretat ca fiind cel de inchidere de string…
$s = 'Editura O\'Reilly';
```

2.4.5.2.3 Includerea valorii intr-o constructie de tip "heredoc"

Constructia heredoc permite specificarea unei valori de tip string care se extinde mai multe linii. Sirul de caractere rezultat va include si caracterele newline corespunzatoare. Constructia heredoc are ca delimitator de inceput simbolul <<< urmat de o eticheta aleasa de programator (ce respecta regulile ce se aplica numelor de variabile) , iar delimitatorul de final este aceeasi eticheta, singura pe linie (fara alineat sau comentarii!) si urmata optional de punct si virgula:

```
$s = <<<GATA                    // eticheta aleasa este GATA
Un text
pe mai multe linii
GATA;
```

In cadrul constructiei heredoc, ca si in cazul ghilimelelor, se substituie automat variabilele intalnite cu valorile lor. In aceste conditii, $ este caracter special, insa ghilimelele nu mai trebuie precedate de \ daca se doreste folosirea lor in cadrul stringului:

```
$1 = "Bond";
$s2 = <<<DELIM
      My name is "$s1".
      "James $s1".
DELIM;
// My name is "Bond".
// "James Bond".
```

2.4.5.2.4 Numere memorate ca string

Exista si cazuri in care intr-un string este memorata o succesiune de cifre; cand asignam o astfel de valoare unei variabile, **acest lucru nu determina insa schimbarea tipului de date al variabilei in cauza la int sau float**! (vezi codul de mai jos). Exemplu: numerele de mobil ale persoanelor este mai convenabil sa fie stocate ca string (desi sunt formate numai din cifre) deoarece 1) daca il stocam ca numar, pierdem 0-ul de inceput, si 2) ca string avem posibilitatea de a accesa facil caracterele componente, putand astfel determina usor operatorul de telefonie (in functie de cifra care urmeaza dupa 07) si lua decizii in functie de el.

```
$s1 = 1234;       // variabila $s1 este de tip integer
$s1 = "1234";     // $s1 este acum de tip string!
```

2.4.5.3 Accesarea caracterelor componente ale stringului

Caracterele componente ale unui string pot fi accesate folosind constructia $sir[index], cu indexul fiind de tip int si incepand de la 0:

```
$s = "InfoAcademy";// creaza variabila $s de tip string si ii asigneaza sirul InfoAcademy
echo $s[4]; // afiseaza A (caracterul aflat pe pozitia 4 daca numaram de la 0, sau
pozitia 5 daca numaram de la 1)
```

2.4.5.4 Concatenarea string-urilor

Doua string-uri pot fi concatenate cu ajutorul operatorului . (punct):

```
$s1 = "Info"; $s2 = "Academy"; $s3 = $s1.$s2; echo $s3; // InfoAcademy
```

Atentie! *String-urile NU mai pot fi concatenate cu +, ca in alte limbaje! In PHP, operatorul + face conversia la numar a celor doi operanzi.*

2.5 CONVERSII DE TIP DE DATE

Valoarea unei variabile sau expresii poate fi convertita la un alt tip de date in doua situatii:

- conversia explicita a valorii unei variabile sau expresii, de catre programator, folosind operatorul de *cast* ():

```
$a = "3 turme de miei cu 3 ciobanei"; // $a are acum tipul de date string
var_dump((int)$a);                    // int (3) , rezultat din conversia stringului la int
```

- conversie implicita (efectuata automat), atunci cand contextul in care se afla expresia sau variabila cere un anumit tip de date. Multi operatori, functii, structuri de control etc. convertesc automat expresiile primite ca operanzi/argumente. De exemplu, echo converteste argumentele sale la string:

```
$a = true;  // crearea variabilei $a cu tip de date boolean, initializata cu true
echo $a;    // 1, rezultat din conversia booleanului la string(vezi tabelul de conversie)
$a = false; // $a ramane boolean dar isi schimba valoarea in false
echo $a;    // nu afiseaza nimic, pt ca conversia lui false la string da sirul vid ""
```

IMPORTANT! Tabelul de conversie atasat acestui material cuprinde regulile de conversie intre toate tipurile de date PHP. Chiar daca pe moment nu aveti cunostintele necesare pentru a intelege intregul tabel, el reprezinta o resursa indispensabila la care veti reveni pe masura ce invatati operatori, structuri de control, functii, tablouri, clase si obiecte etc.

Zend Certified Engineer **ZCE hint:** asigurati-va ca stapaniti bine continutul acestui tabel, in special conversiile intre tipurile de date scalare!

Daca se doreste aflarea sau verificarea tipului de date al unei variabile, PHP pune la dispozitie functii predefinite precum: **gettype(), var_dump(), is_float(), is_int(), is_string(), is_bool(), is_array(), is_object(), is_resource(), is_null()**.

2.6 OPERATORI

2.6.1 Descriere generala

Operatorii sunt elemente de limbaj care efectueaza o operatie asupra uneia sau mai multor valori (numite operanzi) si care produc o valoare finala (rezultatul operatiei). Vor fi prezentate in continuare principalele categorii de operatori prezenti in limbajul PHP.

2.6.2 Operatori aritmetici

Realizeaza operatiile matematice de baza. Tipul de date al rezultatului poate diferi de al operanzilor (vezi cazul impartirii).

+	adunare numerica, daca operanzii sunt de tip numeric. Daca operanzii sunt de tip array, realizeaza compunere!
-	Scadere
*	Inmultire
/	impartire. Daca primul operand nu se imparte exact la cel de-al doilea, rezultatul este float
%	modulo (restul impartirii primului operand la al doilea)

Atentie! *Operatorii aritmetici convertesc operanzii la valori numerice! (vezi tabelul de conversie)*

```
$a = 4; $b = true; $c = “text”; $d = “2 ani”;   // variabile de 3 tipuri de date diferite
echo $a + $b;     // 5 (valoarea lui $b e convertita la int, rezultand 1)
echo $a + $c;     // 4 (valoarea lui $c e convertita la int, rezultand 0)
echo $a + $d;     // 6 (valoarea lui $d e convertita la int, rezultand 2)
echo $b + $c;     // 1 ($b convertit la int da 1, iar $c convertit la int da 0)
```

2.6.3 Operatori de atribuire

Operatorul de atribuire = poate fi folosit de sine statator sau compus cu unul aritmetic. Rezultatul acestui operator este chiar valoarea atribuita:

```
$a = (5+ ($b = 4)); echo $a." ".$b  // 9 4
$b+= 7; echo $b;                    // 11 - echivalent cu $b = $b + 7
$a *= $b; echo $a;                  // 99 - echivalent cu $a = $a * $b
```

2.6.4 Operatori de incrementare si decrementare

Sunt folositi pentru incrementarea sau decrementarea variabilelor (=cresterea sau descresterea valorii cu o unitate). Pot fi folositi in doua variante:

- varianta prefix – variabila se incrementeaza; rezultatul operatorului este valoarea incrementata
- varianta sufix – variabila se incrementeaza; rezultatul operatorului este valoarea NEINCREMENTATA

```
$a = 5;
echo $a++; // 5, dar $a devine 6
echo ++$a; // 7
```

2.6.5 Operatori de comparare

Operatorii de comparare produc un rezultat de tip boolean ce indica rezultatul comparatiei. Ei sunt: >, <, >= (mai mare sau egal), <= (mai mic sau egal), == (egal, non-strict), === (egal in sens strict), != (diferit, non-strict), !== (diferit, in sens strict).

```
$a = (5>4);       // se evalueaza intai operandul drept al asignarii; $a devine true
$b = ($a == false); // $a era true, asadar rezultatul lui ==, memorat in $b, este false
```

Se observa ca operatorii de testare a egalitatii si non-egalitatii sunt de doua feluri:

- **non-strict**: operatorii == si !=, care fac automat conversii ale operanzilor (vezi tabelul de conversie) si compara valorile rezultate. Regulile dupa care se fac conversiile sunt:
    - daca ambii operanzi sunt string:
        - daca amandoi contin numere valide, se face comparare numerica a celor doua valori
        - in caz contrar, compararea este alfabetica
    - daca un operand este string si unul numeric, stringul este convertit la numar
    - daca un operand este boolean, celalalt este convertit la boolean
- **strict**: operatorii === si !==. Cei doi operanzi sunt egali daca valorile lor sunt egale SI sunt de acelasi tip de date, fara nici un fel de conversie

```
$a = "text";
var_dump($a == 0);  // bool (true)!! un string nevid si non-numeric convertit la int da 0
var_dump($a === 0); // bool (false) - tipurile de date ale operanzilor sunt diferite
```

In tabelul de conversie atasat materialului sunt sintetizate modalitatile in care se face conversia intre diversele tipuri de date prezente in PHP.

2.6.6 Operatori logici

Sunt folositi pentru crearea unor expresii complexe de tip boolean prin combinarea de valori elementare. Sunt utilizati cu precadere pentru specificarea conditiilor instructiunilor decizionale, prin agregarea rezultatelor mai multor operatori de comparare (exemplu: daca username-ul introdus de un utilizator web este prezent in baza de date cu utilizatori SI parola introdusa este cea corecta pentru username-ul in cauza, SAU daca utilizatorul era deja autentificat, i se afiseaza una dintre paginile protejate ale site-ului; in caz contrar se va afisa pagina de login).

Operatorii logici sunt urmatorii:

- **&&** – SI logic. Rezultatul sau este true numai daca ambii operanzi se evalueaza ca true. Daca macar unul dintre operanzi este false, rezultatul va fi false
- || – SAU logic. Rezultatul sau este true daca oricare dintre operanzi se evalueaza true. Rezultatul este false numai atunci cand ambii operanzi sunt false
- **!** – negare (inversarea valorii de adevar)
- **and** – SI logic, dar cu precedenta mai mica decat && si =
- **or** – SAU logic, dar cu precedenta mai mica decat || si =
- **xor** – SAU exclusiv, cu precedenta mai mica decat &&, || si =. Daca operanzii au valori egale rezultatul este false, cand au valori diferite este true.

Operand 1	Operand 2	&&,and	||, or	xor
false	false	false	**false**	false
false	true	false	true	true
true	false	false	true	true
true	true	**true**	true	false

Exemple:

```
$a = true; $b = false;
$c = $a && !$b;                 // $a: true, !$b: true, true && true: true
$d = ($a || $b) && ($a xor $b);// $a || $b: true, $a xor $b: true, true && true: true
```

Pot fi folosite ca operanzi orice expresii a caror valoare este de tip boolean:

```
$a = 5; $b = 7;
$c = ($a > $b) && ($a == ($b-2) );
// $a > $b: false, $a == ($b-2): true; false && true: false
```

Operatorii &&/and si ||/or functioneaza cu "scurtcircuitare" – al doilea operand nu se mai evalueaza daca primul este suficient pentru a decide valoarea expresiei. De exemplu, in expresia false && f($a), functia f($a) nu se mai apeleaza deoarece, primul operand fiind false, este clar de la bun inceput ca rezultatul expresiei va fi false (vezi tabelul).

Atentie! *Operanzii sunt convertiti automat la boolean! (vezi tabelul de conversie)*

Exemplu:

```
$a = (true || print 's-a executat print; ');
var_dump($a);     // bool (true); print returneaza 1 care convertit la boolean da true
$a = (false || print 's-a executat print; ');
var_dump($a);     // s-a executat print; bool(true) (primul operand nu decide rezultatul)
```

Precedenta diferita a diversilor operatori duce la evaluarea diferita a unei expresii in cazul in care nu sunt folosite paranteze pentru prioritizare:

```
$a = false || print 'a fost apelat print';
// || are precedenta mai mare decat =, asadar se evalueaza intai || si rezultatul se
// asigneaza lui $a. Este ca si cum am scrie $a = (false || print 'a fost apelat print');
$a = false or print 'a fost apelat print';
// or are precedenta mai mica decat =; se evalueaza =, iar rezultatul sau este considerat
ca fiind primul operand al lui or.
// Este ca si cum am scrie ($a = false) or (print 'a fost apelat print')
```

Iata un exemplu real care se foloseste de precedenta diferita a operatorilor - conectarea la o baza de date (detaliile exacte ale lucrului cu baze de date vor fi incluse intr-o lectie viitoare):

```
$conn = mysql_connect_db($nume_db,$user,$pass) or die ("Eroare la conectarea cu MySQL!");
```

Operandul stang al lui *or* este rezultatul operatorului de asignare. In functie de valoarea acestuia, expresia poate fi evaluata in doua moduri:

- in cazul unei probleme de conectare, rezultatul asignarii va fi null, care convertit la boolean da false. Cand primul operand este false, operatorul *or* va evalua si cel de-al doilea operand, asadar se va executa instructiunea die() care are ca efect incheierea executiei programului cu afisarea unui mesaj de eroare
- daca conectarea se realizeaza cu succes, rezultatul asignarii va fi de tip resource, care convertit la boolean da true. Primul operand fiind true, *or* nu il va mai evalua pe cel de-al doilea si executia va continua cu instructiunea de pe linia urmatoare

2.6.7 Operatori la nivel de bit

Sunt folositi pentru operatii aplicate reprezentarii in binar a numerelor:

- & - SI la nivel de bit
- | - SAU la nivel de bit
- ^ - SAU exclusiv lsa nivel de bit
- ! – negare pe biti (comutarea valorii bitilor)
- <<, >> - deplasare pe biti la stanga sau la dreapta

Bit 1	Bit 2	&	|	^
0	0	0	**0**	0
0	1	0	1	1
1	0	0	1	1
1	1	**1**	1	0

2.6.8 Operatorul de concatenare pentru string

Concatenarea (alipirea) a doua string-uri se realizeaza in PHP cu ajutorul operatorului . (punct)

```
$a = “varza”; $b = “viezure”; $c = “manz”;
echo “Dacii aveau “ . $a . ‘,’ . $b . ‘,’ . ‘ si ‘ . $c;
// Dacii aveau varza, viezure si manz
// Se concateneaza “Dacii aveau “ cu valoarea lui $a, apoi cu stringul ce contine virgula
// apoi cu $b etc. Rezultatul final al concatenarii este pasat ca argument lui echo
```

Atentie! *Concatenarea de string-uri NU se poate face cu +, ca in alte limbaje!*

2.6.9 Operatorul ternar

Acest operator foloseste 3 operanzi si produce un rezultat care depinde de valoarea de adevar a primului operand:

```
(expresie_boolean) ? valoare_pt_cazul_true : valoare_pt_cazul_false
// Daca expresia se evalueaza true, rezultatul operatorului va fi valoarea operandului 2
// In caz contrar, reuzltatul este valoarea operandului 3
```

Expresiile ce joaca rol de operand 2 si 3 trebuie sa aiba o valoare – nu este permisa folosirea de instructiuni care nu au ca rezultat valori:

```
$a = (true)? echo “true” : echo “false” ; // eroare, echo nu produce valoare
$a = (true)? print “true” : 13 ; // functioneaza; $a devine 1 (=ceea ce returneaza print)
```

Efectul sau ar putea fi obtinut si prin folosirea instructiunii conditionale *if*, insa cu operatorul ternar scrierea este mai condensata.

```
$varsta = 50;
$varsta_prelucrata = ($varsta<40)? “$varsta de ani” : “o frumoasa varsta”;
echo “Astazi sotia mea a implinit “.$varsta_prelucrata;
```

2.7 BIBLIOGRAFIE

- PHP Manual
  - sintaxa de baza: http://ro.php.net/manual/en/language.basic-syntax.php
  - variabile: http://ro.php.net/manual/en/language.variables.php
  - tipuri de date: http://ro.php.net/manual/en/language.types.php
  - constante: http://ro.php.net/manual/en/language.constants.php
  - operatori: http://ro.php.net/manual/en/language.operators.php
- Zend PHP 5 Certification Study Guide – capitolul 1: "PHP Basics", pag 3-25

2.8 ANEXA 1 – tabel de conversie intre tipurile de date PHP

	boolean	int	float	string	array	object	null	resource
mod de conversie	*(bool)expr* *(boolean)expr*	*(int)expr* *(integer)expr* *intval(expr)*	*(float)expr* *(double)expr* *(real)expr* *floatval($var)*	*(string)expr* *strval($var)* *"$var"*	*(array)expr*	*(object)expr*	*unset($var)*	-
boolean	-	False→0 True→1	False→0 True→1	False→"" True→"1"	Array cu un element avand valoarea respectiva	Instanta de stdClass cu un camp numit $scalar ce contine valoarea	-	-
int	0→false Altceva→true	-	Valoarea numerica respectiva	Numarul ca string			-	-
float	0→false Altceva→true	Un intreg obtinut prin rotunjire catre 0 (!)	-				-	-
string	"" si "0"→false Altceva→true	Daca stringul incepe cu o portiune numerica **zecimala** valida (un eventual semn+cifre+un eventual .+cifre+eventual [eE] si exponent) conversia da numar. Daca stringul contine e,E sau . rezultatul va fi float, in caz contrar integer. Daca stringul nu incepe cu nr valid, conversia da 0		-			-	-
array	Array cu 0 elemente →false Altceva→true	Nedefinit (pe moment se comporta ca si cum expresia/variabila ar fi intai convertita la boolean, insa acest comportament se poate schimba in viitoare versiuni)	Ca si cum conversia s-ar face intai la integer si apoi la float	"Array"	-	Instanta de stdClass in care campurile au ca nume cheile din array si ca valori valorile din array	-	-
object	PHP4: obiect fara campuri→false PHP 5: Obiect SimpleXML creat pe baza unui tag gol → false Orice alt obiect→true			PHP4: "Object" PHP5: valoarea intoarsa de metoda __toString() a obiectului	Un array ale carui elemente sunt atributele obiectului. Cheile array-ului sunt numele atributelor, cu mici particularitati: atributele private vor primi ca prefix numele clasei, iar cele protected vor fi prefixate cu *	-	-	-
null	false			""	Array gol (array())	Instanta de stdClass fara campuri	-	-
resource	true			"Resource id #N"	Array cu un element	idem int (obiect cu un camp $scalar)	-	-