Programare Web

Curs 5

Capitolul 4. Limbajul PHP

Obiective

- Obiectivul acestui curs un este o prezentare completa a limbajului PHP ci o introducere (pentru cei care un au utilizat inca acest limbaj).
- Limbajul este simplu si usor de invatat, semanand cu alte limbaje de programare deja studiate.
- Exista o multitudine de carti si documentatii accesibile online din care se pot aprofunda diversele aspecte ale limbajului si folosirii acestuia.
- In cursurile de PHP vom prezenta atat interfata sa cu SGBD-ul MySQL cat si scurte exemple de lucru cu Oracle si ODBC.

Introducere

- Așa cum am văzut în capitolul precedent, în cazul scripturilor scrise în limbajul C acestea trebuie să execute o serie de operații de "bucătărie internă" care pot încurca pe un utilizator care nu este un bun cunoscător al limbajului și pot genera erori:
  - detectarea metodei (*GET* sau *POST*) și implicit a locului de unde se citesc datele (variabila de mediu *QUERY_STRING* sau intrarea standard *stdin*)
  - descompunerea șirului primit de la browser în perechi *simbol-valoare*
  - transmiterea unui preambul al rezultatului în format *MIME*.

Introducere

- Aceste inconveniente pot fi evitate prin folosirea unui limbaj evoluat care să preia toate aceste operații și care să permită o gestiune simplă atât a simbolilor primiți de la browser (datele completate în formă) cât și a altor variabile cum sunt cele de mediu.
- Există mai multe soluții în această direcție:

1. Folosirea unor limbaje specializate în scrierea de scripturi cum este *perl*.
2. Inserarea în cadrul fișierelor HTML a unor programe care se execută la nivelul browserului (cum este *Java*) și care pot conține inclusiv proceduri de conectare și exploatare a unei baze de date.

Introducere

3. Inserarea în cadrul fișierelor HTML a unor programe care se execută la nivelul serverului de web înainte ca acesta să trimită rezultatul către browser.

- Un exemplu de astfel de limbaj este *PHP* descris în capitolul de față.
- Deci PHP este un limbaj pentru inserturi în fișierele HTML, inserturi care sunt înlocuite la evaluarea fișierului de către serverul de web cu rezultatele execuției lor.

Introducere

- PHP a reprezentat initial o abreviere pentru "*Personal Home Pages* tools" și a fost creat în 1994 de Rasmus Lerdorf.
- În primele versiuni (PHP/FI) el conținea suport doar pentru pachetul de baze de date mSQL (*mini SQL*).
- Actualmente PHP inseamna 'PHP: Hypertext Processor' putând fi folosit în conjuncție cu o largă listă de SGBD-uri, atât comerciale cât și necomerciale (*free software*) incluzand MySQL, dBase, Oracle, DB2, PostgreSQL, Sybase, InterBase, SQLServer, ODBC, etc.

Introducere

- Acest pachet poate fi instalat atât ca un interpretor de sine stătător (deci un CGI separat) cât și ca un modul pentru serverul de web (Apache).
- Pachetul poate fi instalat de asemenea sub Windows IIS/PWS şi Apache.
- Informații complete despre instalare și configurare în diverse variante pot fi găsite în manualul care îl însoțește.

Inserare cod

- Introducerea de inserturi scrise în PHP în fișierele HTML poate fi făcută în mai multe moduri (funcția *echo* are același efect cu *printf* din limbajul C):

1. <?php echo "Varianta 1 tip XML\n"; ?>
2. <? echo "Varianta 2, cea mai simpla\n"; ?>
3. <script language="php">
   echo "Varianta 3, tip limbaj pentru scripturi";
   </script>;
4. <% echo "varianta 4, tip ASP"; %>

Inserare cod

- Pentru folosirea unora dintre aceste variante interpretorul de PHP trebuie configurat corespunzator (variantele 2 si 4)
- Optiunile respective se gasesc in fisierul de configurare php.ini

Descriere

- Insertul poate conține o singură instrucțiune, ca în exemplele de mai sus, sau mai multe instrucțiuni.
- Formatul instrucțiunilor este liber, putându-se continua pe mai multe linii.
- Ca și în C fiecare instrucțiune este terminată cu punct și virgulă.
- Directiva de inchidere bloc php tine si loc de ;
- Daca dupa ultimul bloc nu exista ; nu este eroare (e chiar folositor uneori cand se fac incluziuni de fisiere)

Exemple

- Obisnuit:

```
<?php
        echo ‘Hello world';
    ?>
```

- Fara ; in final de bloc:

```
<?php echo ‘Hello world' ?>
```

- Fara directiva de inchidere la ultimul bloc (dar cu ; !)

```
 <?php echo ‘Putem omite directiva in final';
```

Comentarii

- Comentariile sunt ca si in C / Unix shell:

```
<?php
    echo `Test'; // Comentariu pe linie
    /* Comentariu pe
       potential mai multe linii */
    echo `Ceva'; # Comentariu pe linie
?>
```

Tipuri si variabile

- PHP este un limbaj interpretat. In consecinta nu este necesara declararea variabilelor
- O variabila imprumuta tipul valorii stocate in ea la momentul respectiv.
- O variabila isi poate schimba tipul pe parcursul executiei scriptului PHP
- O variabila se creaza in momentul in care se depune o valoare in ea.
- Pentru conversia intre tipuri (daca e necesara) se pot folosi construcții de conversie de tip *cast* similare cu cele din C sau funcția *settype*.

Tipuri

- Exista 8 tipuri de date in PHP: 4 tipuri scalare, doua compuse si doua speciale

1. Boolean
2. Integer
3. Float (include double)
4. String
5. Array
6. Object
7. Resource
8. Null

Variabile

- Numele oricarei variabile este prefixat cu simbolul $.
- Acesta e un marcaj de variabila si nu face efectiv parte din nume.
- Pentru aflarea tipului unei variabile se poate folosi functia gettype sau var_dump
- Instrucțiunea de atribuire este identică cu cea din limbajul C.
- Dacă variabila nu este deja definită, ea se crează automat:

```
$nume = "Ion";
$adresa = "Bucuresti";
$sir = "1234";
$numar = (int) $sir; // exemplu de cast
```

Tipul Boolean

- O variabila de tip Boolean poate contine valoarea True sau False.
- In cazul conversiei la Boolean, sunt considerate False (printre altele):
  - Literalul **FALSE**
  - Valoarea intreaga sau reala (float, double) 0 (zero)
  - Un sir vid
  - Sirul "0"
  - Un array cu 0 elemente
  - Tipul special NULL (incluzand variabilele ne-setate = inexistente)
- Orice alta valoare este considerata True (inclusiv orice resursa)

Tipul intreg

- Specificarea se poate face in bazele 10, 8 si 16:

```
<?php
    $a = 1234; // zecimal
    $a = -123; // zecimal, negativ
    $a = 0123; // octal
    $a = 0x1A; // hexazecimal
?>
```

Numere intregi

- Daca o cifra este incorecta restul cifrelor se ignora:

```
<?php
    $a = 123EU4PLECLA5678; // 123
    $a = 012389; // octal 0123
    $a = 0x1ASPARAGUS; // hexa 0x1A
?>
```

- Daca se depaseste capacitatea de reprezentare pentru intregi (~2^31), valoarea devine automat float
- La conversia de la float la intreg numarul e rotunjit spre 0. Daca se depaseste capacitatea de reprezentare pentru intregi rezultatul este nedefinit (nu se emite nici o atentionare!)
- La conversia de la string la numar se ia prefixul intreg al numarului (ca mai sus):

```
$a = 1 + "3 iezi cucuieti"; // $a devine 4
```

Tipul real (float, double)

- Se pot scrie in formatul uzual sau exponential:

```
<?php
      $a = 1.234;
      $b = 1.2e34;
      $c = 12E-34;
?>
```

- Valorile limita sunt dependente de platforma dar uzual numerele sunt pana la ~1.8e308 cu o precizie de 14 cifre.
- Conversia de la sir la float se face similar cu cea a intregilor (pana la primul caracter care nu face parte dintr-o reprezentare corecta de numar real)

Tipul ARRAY

- Un tablou PHP este o succesiune de perechi (cheie, valoare).
- Li se mai spune si tablouri asociative.
- Un tablou poate fi exploatat in modul clasic (chei pornind de la 0 ca in C) sau ca tablou asociativ (acces prin cheie, cheile putand sa nu fie succesive si nici numerice.

Exemple

```
<?php
     $a = array(“pw" => “examen",
                4 => ”an terminal”,
                “succes” =>true);
     echo $a[“pw”], $a[4]
// $b = array cu 2 dimensiuni
     $b = array(“medii" => array(1 => 9.45, 2 => 9.5, 3
     => 8.12, 4 => 9.90, “stat”=>10));

     echo $b[“medii"][2];          // 9.5
     echo $b[“medii"][“stat"];     // 10

// $c e identic cu $b
     $c = array(“medii" => array(1 => 9.45, 9.5, 8.12,
     9.90, “stat”=>10));

     echo $c[“medii"][2];          // 9.5
     echo $c[“medii"][“stat"];     // 10
?>
```

Tipul Array

- Cheia trebuie sa fie scalara (un alt array sau obiect)
- Adaugarea inca unui element cu cheie maxima negativa adauga o pereche cu cheia 0 (incepand cu v4.3.0)
- Cheia TRUE devine 1
- Cheia FALSE devine 0
- Cheia NULL devine sirul vid

Exemplu

```
$regiune = array(-12 => “Oltenia”);
 // o variabila de tip array cu cheia
 // maxima -12
 // Adaugam noi elemente si vom crea noi
 // perechi cu chei incepand cu 0:
$regiune[] = "Muntenia"; // elementul 0
$regiune[] = "Moldova";  // elementul 1
```

- In lipsa, cheile pleaca de la 0:

```
$orase = array(“Bucuresti”, “Ploiesti”,
    “Campina”)  // chei 0, 1, 2
```

Conversii

- La conversia din tipurile intreg, real, string, boolean si resursa in tipul array se creaza un tablou cu un singur element cu cheia 0 si valoarea respectiva.
- Daca se converteste un obiect la array, obtinem un array avand ca elemente proprietatile obiectului. Mai multe amanunte in documentatia PHP.
- Conversia unei valori nule la array duce la un array vid (Atentie: vid nu inseamna nul!)

Comparatii

- 2 tablouri se pot compara astfel:
  - Egalitate: $a == $b adevarat daca au aceleasi perechi (cheie, valoare)
  - Identitate: $a === $b adevarat daca au aceleasi perechi (cheie, valoare) in aceeasi ordine si cu aceleasi tipuri
  - Inegalitate: $a <> $b sau $a != $b. Inversa egalitatii
  - Nonidentitate: $a !== $b. Inversa identitatii/

Comparatii

- Se pot afla diferentele dintre 2 tablouri folosind functia array_dif care returneaza valorile dintr-un array care nu se gasesc in al doilea:

```
<?php
    $array1 = array("ion", "vasile", "ion", "elen
    a");
    $array2 = array("vasile", "ion", "mia");
    $rezultat = array_diff($array1, $array2);
    print_r($rezultat);
                // rezultat: [0]=>"elena"

?>
```

Reuniune

- Doua tablouri se pot reuni folosint operatorul +:

$c = $a + $b

- Rezultatul contine perechile primului array la care se adauga perechile din al doilea array cu o cheie care nu exista in primul.
- Exemplu:

```
<?php
    $array1 = array("ion", "vasile", "ion", "elen
    a");
    $array2 = array("vasile", "ion", "mia");
    $rezultat = $array2 + $array1;
    print_r($rezultat);
?>
```

- Vom obtine un tablou cu 4 elemente: elementele din array2 (chei 0, 1, 2) si ultimul element din array 1 (cheia 3):

```
(
    "vasile", "ion", "mia", "elena")
```

Tipul Obiect

- A fost descris in detaliu la orele de laborator.
- Daca o valoare de alt tip este convertita la tipul obiect, obtinem o instanta a clasei *stdClass*.
- Daca se converteste la obiect o valoare nula, noua instanta va fi vida.
- Daca un tablou se converteste la obiect cheile devin proprietati.

Tipul Resursa

- Tipul resursa este un tip special, variabilele de acest tip contin o referinta catre o resursa externa.
- Felul resursei se poate obtine cu functia get_resource_type (ex: mysql link, file, domxml document, etc)
- Conversia la tipul resursa nu are sens (din definitia tipului)
- O resursa care nu mai este referita este detectata automat de 'garbage collector' si eliberata (deci nu e necesara eliberarea manuala).

Tipul NULL

- Tipul NULL are o singura valoare, NULL.
- O variabila nula nu contine nici o valoare.
- O variabila este considerata nula daca:
  1. I-a fost asignata valoarea NULL
  2. Nu i-a fost asignata inca nici o valoare (deci eventual ea nu exista).
  3. A fost dealocata cu functia unset($variabila)
- O variabila se poate testa daca e nula sau nu cu functia is_null($variabila).
- O variabila se poate testa daca exista sau nu (cazurile 2 si 3 de mai sus) cu functia isset($variabila, ...)
- O variabila se poate testa daca este goala cu empty($variabila). Sunt considerate goale variabilele care contin echivalentul lui FALSE (inclusiv care contin valoarea NULL)

Variabile

- Asa cum am mai spus, numele variabilelor este prefixat cu $ (marcaj de variabila).
- Numele este case-senzitiv (literele mari sunt considerate diferite de cele mici).
- Un nume corect PHP incepe cu litera sau underscore si continua cu litere, cifre si underscore.
- Variabilele pot contine referinte vatre alte variabile (adresa se preleveaza cu &, ca in C). Exemplu:

Variabile

```
<?php
    $unu = 'Ceva';
    $doi = &$unu; // $doi e o referinta la $unu.
    $doi = "Altceva";   // Modificam $doi
    echo $unu;          // Ambele contin acelasi
    echo $doi;          // sir
?>
```

- Nu se poate preleva cu & adresa unei expresii (doar a unei variabile).
- PHP initializeaza variabilele cu valori implicite dar nu este bine sa ne bazam pe aceste valori (0 pentru numere, False pentru boolean, etc).

Variabile predefinite

- Exista un numar mare de variabile predefinite (de sistem) pe care scriptul le poate folosi.
- Cele mai utilizate sunt:
  - $_GET, $_POST, $_COOKIES, $_REQUEST contin valorile transmise scriptului cu metodele GET, POST, cookie sau reuniunea lor
  - $_SESSION contine variabile care se pot folosi de o succesiune de executii de scripturi care formeaza o sesiune de lucru
  - $GLOBALS contine toate variabilele globale ale scriptului
- O descriere a acestor variabile si a altora din aceeasi categorie se gaseste in documentatia PHP

Domeniul (scope)

- Variabilele sunt cunoscute în contextul în care au fost create.
- Cele definite în afara oricărei funcţii sunt similare variabilelor globale din C.
- Cele definite în funcţii (prin asignare) sunt locale acelei funcţii.
- Spre deosebire de limbajul C o variabilă globală nu este cunoscută în interiorul unei funcţii decât dacă este declarată cu *global* în acea funcţie.

Domeniul (scope)

- *Exemplul 1:*

```
$a = 1; /* variabila globala */
function Ecou()
{ $a = 2;  /* se asigneaza o valoare variabilei locale */
  echo $a;  /* tiparire variabila locala */
}
Ecou();
echo $a;  /* tiparire variabila globala */
```

- Rezultat 21

Domeniul (scope)

- *Exemplul 2:*

```
$a = 1;  /* variabila globala */
function Ecou()
{ global $a
   $a = 2;/* se asigneaza o valoare variabilei globale */
   echo $a;/* tiparire variabila globala */
}
Ecou();
echo $a;  /* tiparire variabila globala */
```

- Rezultat 22

Domeniul (scope)

- Un alt mod de a defini varibile globale cunoscute şi în interiorul funcţiilor este folosirea tabloului asociativ predefinit $GLOBALS, având ca indici numele variabilelor globale:
- *Exemplu:*

```
$a = 1;  /* variabila globala */
function Ecou()
{  $GLOBALS["a"] = 2;  /* se asigneaza o valoare
                          variabilei globale */
   echo $GLOBALS["a"];/* tiparire variabila globala */
}
Ecou();
echo $a;  /* tiparire variabila globala */
```

- Rezultat 22

Variabile statice

- Ca şi în C se pot defini variabile locale funcţiilor dar care îşi păstrează valoarea de la un apel la altul.
- Aceste variabile se numesc *statice*.
- Variabilele statice pot fi iniţializate cu o valoare care apoi se modifică şi este păstrată pentru apelurile viitoare:

```
function Increment()
{ static $a = 0;
  echo $a;
  $a++;
}
```

- Rezultatul apelului repetat al acestei funcţii va fi afişarea numerelor 0, 1, 2, . . .

Macrosubstitutie

- Numele unei variabile se poate găsi în altă variabilă.
- Acest procedeu, numit macrosubstituţie, este întâlnit şi în alte limbaje, cum este Xbase (dBase, Fox, Clipper).
- Instrucţiunile:

```
$a = "Limbajul";
$$a = " PHP";
```

definesc două variabile: $a cu valoarea “Limbajul” şi $Limbajul cu valoarea “ PHP”.

- În acest caz instrucţiunea:

```
echo "$a ${$a}";
```

va afisa Limbajul PHP

Variabile externe

- În această categorie intră variabilele corespunzătoare simbolilor primiti de la un formular şi variabilele de mediu setate de serverul de web, în acelaşi mod ca în cazul scripturilor scrise în limbajul C.
- Să presupunem că avem următoarea formă:

```
<form action="actiune.php" method="post">
    Nume: <input type="text" name="nume"><br>
    Localitate: <input type="text" name="adresa[localitate]"><br>
    Strada: <input type="text" name="adresa[strada]"><br>
    Numar: <input type="text" name="adresa[numar]"><br>
    Optiuni: <br>
    <select multiple name="so[]">
        <option value="Windows 95">Windows 95
        <option value="Windows XP">Windows XP
        <option value="Windows Vista">Windows Vista
        <option value="Linux">Linux
        </select>
    <input type="submit">
</form>
```

Variabile externe

- Scriptul *actiune.php* care tratează această formă poate primi variabilele:
  - *$nume*, variabilă simplă
  - *$adresa*, un tablou asociativ cu trei elemente
  - *$so*, un tablou având atâtea elemente câte selecţii s-au făcut în meniul vertical din formă.
- Acest lucru se intampla insa doar daca optiunea de configurare a PHP register_globals e setata pe On (implicit ea e insa Off, fiind potential o bresa de securitate.
- In mod normal valorile celor 3 variabile se gasesc in $_POST si $_REQUEST.

Constante

- Constantele se definesc similar cu limbajul C, cu define:

```
<?php

    // Constante valide
    define("MATERIE",      "Programare Web");
    define("_EVAL_UARE",     "Examen");
    define("NOTA10",   "10");

    // Nume invalid
    define("2PAC",     "Cantaret");

    // Asa arata constantele PHP,
    // e bine sa nu avem si noi la fel
    define("__NOTA__", "10");
?>
```

Constante

- Spre deosebire de variabile:
  - Constantele nu au un nume care incepe cu $
  - Constantele pot fi definite doar cu define() nu prin atribuire
  - Constantele nu au domeniu de valabilitate ca variabilelee (se pot folosi si in functii de exemplu).
  - Nu pot sa-si schimbe valoarea si nu pot fi dealocate (unset)
  - Constantele pot contine doar valori scalare (boolean, intreg, real sau sir)

Constante predefinite

- PHP-ul pune la dispozitie si o serie de constante predefinite.
- Acestea au forma __Nume__
- Printre ele sunt:
  - __LINE__ numarul liniei curente in sursa PHP
  - __FILE__ calea si numele complet al fisierului sursa PHP
  - __DIR__ directorul acelui fisier
  - __FUNCTION__ numele functiei curente (doar cu litere mici in PHP4)
  - __CLASS__ numele clasei (doar cu litere mici in PHP4)
  - __METHOD__ numele metodei din clasa (doar PHP5)

Expresii

- Expresiile in PHP sunt similare celor din limbajul C
- Se pot folosi constructii de tipurile:
  - $a++, ++$a
  - $a--, --$a
  - $a += 3; (echivalenta cu $a = $a + 3;), in loc de + putand fi orice operator valid pentru operatia respectiva
  - Atribuiri multiple, ca de exemplu:

    $a = $b = ++$c; sau

    $a = $b += 10;

Atribuirile intorc o valoare

- Ca si in limbajul C atribuirile intorc o valoare:

  if ($con = mysql_connect(...)) ...
- Ca si in limbajul C o expresie logica e evaluata doar pana in momentul in care valoarea sa este certa:

  mysql_connect(...) or die('Conexiune esuata');

Operatori

- Aritmetici: +, -, *, /, % (modul)
- Logici: ==, ===, !=, !==, <, >, <=, >=
- Conectori logici: and, &&, or, ||, ! (negare), xor (sau exclusiv)
- Operatori pe siruri: . (concatenare)
- Operatori pe biti: &, |, ~ (inversare biti)

Structuri de control

- Exista o serie de structuri de control care sunt similare celor din limbajul C.
- Vom avea ca si acolo decizii, cicluri, alegere
- Se pot defini ca si in C functii (nu exista decat functii, nu si proceduri ca in Pascal).

Decizia

- În PHP aceste instrucţiuni sunt asemănătoare ca sintaxă cu cele similare din limbajul C.
- Sintaxa:

```
if (conditie_1)
  { instructiuni_1 }
elseif (conditie_2)
  { instructiuni_2 }
. . . . . .
else { instructiuni_N }
```

- *elseif* şi *else* sunt opţionale (similar cu limbajul C).

Decizia

- Exemplu:

```
if ($a > $b)
{ print "a este mai mare ca b";}
elseif ($a == $b)
{ print "a este egal cu b";}
else
  { print "a este mai mic decat b";}
```

Ciclul WHILE

- Sintaxa:

```
while ( conditie )
{ instructiuni }
```

- Exemplu:

```
$i = 10;
while ($i >= 0)
{ print $i--; }
```

Ciclul DO

- Sintaxa:

```
do
{ instructiuni }
while ( conditie );
```

- Exemplu:

```
$i = 10;
do
{ print $i--;}
while ($i>0);
```

Ciclul FOR

- Sintaxa:

```
FOR (expr1; expr2; expr3)
  instructiune
```

- Execuţia unui astfel de ciclu se face astfel:
  - Se evaluează expresia expr1
  - Cât timp expresia expr2 are valoarea adevărat se repetă operaţiile:
    - Se execută instrucţiunea (*instructiune*)
    - Se evaluează expresia expr3
- Exemplu:

```
for ($i = 1; $i <=10; $i++)
  { print $i;}
```

- Efectul va fi afişarea valorilor de la 1 la 10.

FOREACH

- Sintaxa (2 variante):

```
foreach (expr_array as $valoare)
      statement
foreach (expr_array as $cheie =>
    $valoare)
      statement
```

- Se foloseste pentru parcurgerea unui tablou (ciclu dupa elementele unui tablou)

Exemplu

```
<?php
// tiparirea unui tablou
$arr = array("one", "two", "three");

foreach ($arr as $val) {
    echo "Value: $val<br />\n";
    }
foreach ($arr as $k => $val) {
echo "Cheie: $k; Val: $val<br />\n";
    }
    ?>
```

Break si Continue

- Aceste instrucţiuni se folosesc pentru a ieşi dintr-un ciclu, respectiv pentru a se trece necondiţionat la un nou pas al ciclului chiar dacă pasul curent nu s-a terminat.
- Exemplu:

Tipărirea numerelor impare dintre 1 şi 10

```
for ($i = 1;;$i++)
{    if ($i > 10) { break; }
if ($i % 2)  { continue; }
print $i;
}
```

Alegerea (Switch)

- Sintaxa:

```
switch (expr)
{
case val1:
  instructiuni
case val2:
  instructiuni
. . . . . .
default:
  instructiuni
}
```

Alegerea (Switch)

- Efectul este următorul:
  - Se evaluează expresia expr
  - Se parcurg etichetele case (*val1, val2, ...)* una după alta. În cazul în care se găseşte o egalitate, se execută instrucţiunile de la acea etichetă până la prima instrucţiune *break* sau până se sfârseşte întregul *switch*.
  - Dacă nu există nici o egalitate se execută instrucţiunile de la default

Alegerea (Switch)

- Exemplu:

```
switch ($i)
{
case 0:
print "i egal cu 0";
break;
case 1:
print "i egal cu 1";
break;
case 2:
print "i egal cu 2";
break;
default:
print "i nu este egal cu 0, 1 sau 2";
}
```

- De remarcat că dacă instrucţiunile *break* ar lipsi, în cazul în care $i este egal cu 0 se tipăresc toate cele patru mesaje iar în cazul în care este egal cu 1 doar ultimele trei.

Functii

- Programele PHP pot conţine funcţii definite de utilizator, inclusiv funcţii recursive.
- Sintaxa definiţiei unei funcţii este următoarea:

```
function nume_functie
  (lista_parametri)
{
instructiuni
}
```

- Dacă se doreşte ca funcţia să întoarca o valoare, se foloseşte instrucţiunea:

  return *expresie*;

Functii

- Exemplu:

```
function la_patrat ($numar)
{
return $numar * $numar;
}
echo la_patrat(10);
```

Functii

- Folosirea unei funcţii se poate face doar după definiţia acesteia.
- Parametri sunt transmişi prin valoare.
- Dacă se doreşte transmiterea prin referinţă a unui argument, se poate folosi construcţie &variabila:

```
function la_patrat (&$numar)
{
$numar = $numar * $numar;
}
$a = 10;
la_patrat($a);
echo $a;  // tipareste 100
```

Functii

- Se poate transmite prin valoare adresa sa:

```
function la_patrat ($numar)
{
$numar = $numar * $numar;
}
$a = 10;
la_patrat($a);
echo $a;  // tipareste 10
la_patrat(&$a);
echo $a;  // tipareste 100
```

Functii

- La definirea unei funcţii se pot asigna şi valori implicite pentru argumente.
- În cazul în care acestea lipsesc la apel sunt luate implicit valorile din definiţie:

```
function la_patrat ($numar = 4)
{
return $numar * $numar;
}
echo la_patrat(10); // tipareste 100
echo la_patrat();   // tipareste 16
```

Functii

- Astfel de argumente trebuie să fie ultimele din listă.
- De exemplu secvenţa:

```
function inmultire ($numar1 = 4,
  $numar2)
{
return $numar1 * $numar2;
}
echo inmultire(10);
```

va semnala o eroare deoarece automat valoarea 10 va fi asignată primului argument.

Functii PHP

- Pachetul PHP pune la dispoziţie un număr foarte mare de funcţii, atât de uz general cât şi funcţii specifice accesului la diverse sisteme de gestiune a bazelor de date.
- În continuare sunt prezentate doar o parte dintre acestea, incluzând funcţiile uzuale de acces la baze de date MySQL si Oracle

Functii ARRAY

`array array(`*`lista valori`*`);`

- Crează un *array* conţinând valorile din listă. Pentru un *array bidimensional* se poate folosi operatorul => pentru asocierea celor doi indici.
- Exemplu:

```
$note = array(1, 2, 3, 4, 5, 6, 7,
  8, 9, 10);
$calificative("S"=>"Satisfacator",
  "B"=>"Bine","F"=>"Foarte bine");
```

- În primul caz s-a definit un *array* cu o singură dimensiune iar în cel de-al doilea unul cu două dimensiuni.

Functii ARRAY

void list(*lista de valori*);

- Asignează o listă de variabile ca şi când ar fi un *array*.
- Exemplu:

```
$rezultat = mysql_query("select
  nume_s, grupa from studenti");
while (list($n, $g) =
  mysql_fetch_row($rezultat))
  { print( "Nume: $n<br>Grupa:
  $g<br>\n"); }
```

Functii ARRAY

```
int count(variabila);
```

- Întoarce numărul de elemente din variabilă.
- Dacă este un *array* acest numar poate fi mai mare ca 1. Dacă variabila nu este setată întoarce 0.

```
array each(array array);
```

- întoarce următoarea pereche (index, valoare) dintr-un array sub forma unui tablou având indicii 0 şi 1 sau *key* şi *value*.

Functii ARRAY

- De notat că fiecare variabilă de tip *array* are asociat un pointer intern care arată către unul dintre elementele sale iar funcţia *each()* se poate folosi pentru parcurgerea sa.
- Exemplu:

```
$calificative("S"=>"Satisfacator",
  "B"=>"Bine","F"=>"Foarte bine");
list($c, $d) = each($calificative);
```

- Atunci:
  - $c are valoarea "S"
  - $d are valoarea "Satisfacator"

Functii ARRAY

```
mixed next(array tablou);
```

- Întoarce următorul element al unui *array* sau *false* dacă nu mai sunt elemente, avansând deci pointerul intern asociat tabloului.
- De notat că dacă un tablou are elemente nule şi pentru acestea valoarea întoarsă va fi *false*. De aceea, pentru parcurgerea unui tablou se recomandă folosirea lui *each()*.

Functii ARRAY

```
mixed prev(array tablou);
```

- Întoarce precenentul element al unui *array* sau *false* dacă nu mai sunt elemente, decrementând deci pointerul intern asociat tabloului.
- Aceeasi observatie ca la next (folosire each)

```
mixed reset(array tablou);
```

- Setează pointerul intern asociat tabloului la primul element al acestuia.
- Întoarce valoarea acestui element.

```
int sizeof(array tablou);
```

- Întoarce numărul de elemente ale unui tablou. Este analog cu *count()*.

Functii de informare/setare

```
int error_reporting(int [level]);
```

- Setează nivelul de erori care sunt raportate de PHP, conform tabelului de mai jos. Valorile respective pot fi cumulate prin adunare în cazul în care se doresc setate simultan mai multe tipuri de raportări..

```
string getenv(string variabila);
```

- Întoarce valoarea unei variabile de mediu sau *false* în caz de eroare.

$ip = getenv("REMOTE_ADDR"); // prelevare adresa IP a clientului

Functii de informare/setare

```
void putenv(string setare);
```

- Crează o noua variabilă de mediu. Exemplu:

putenv("USER = $user");

```
int phpinfo(void);
```

- Întoarce o suită de informaţii privind PHP: opţiuni de compilare, versiune, informaţii despre serverul de web, variabile de mediu, versiunea sistemului de operare, etc.

```
string phpversion(void);
```

- Întoarce sub forma unui şir numărul versiunii PHP folosite

Alte functii

`void eval(string sir);`

- Şirul de caractere (care trebuie să conţină expresii valide PHP) este evaluat.
- Exemplu:

```
$nume = 'Ion';
$str = 'Ma numesc $nume<br>';
echo $str;
eval( "\$str = \"$str\";" );
echo $str;
```

- Va avea ca efect tipărirea mesajelor:

  Ma numesc $nume

  Ma numesc Ion

Alte functii

```
void die(string mesaj);
```

- Afişeaza un mesaj şi termină execuţia scriptului.
- Exemplu:

```
$nume_fisier = '/usr/local/date.txt';
$f = fopen($nume_fisier, 'r')
  or die "Nu se poate deschide fisierul
  $filename";
```

```
void exit(void);
```

- Termină imediat execuţia scriptului.

Alte functii

```
void sleep(int secunde);
```

- Întârzie execuţia un număr de secunde.

```
echo(string arg1, string
        [argn]...);
```

- sau

```
echo string arg1, string
        [argn]...;
```

- Evaluează şi afişează parametrii primiţi ca argument.

Alte functii

```
print(string arg);
```

- Afişează argumentul

```
int printf(string format, mixed
    [argumente]...);
```

- Afişează argumentele formatate conform şirului *format* (asemănător cu limbajul C).

```
int strcmp(string str1, string str2);
```

- Compară două şiruri. Întoarce o valoare < 0, egală cu 0 sau >0 după cum *str1* este mai mic, egal sau mai mare decât *str2*. Literele mari şi mici sunt considerate diferite.

```
int strlen(string sir);
```

- Întoarce lungimea unui şir de caractere.

Alte functii

```
string substr(string sir, int start, int
                                [lung]);
```

- Întoarce un subşir conţinând caracterele începând cu cel dat de *start* şi având lunginea *lung*. Dacă *start* este negativ, numărătoarea pentru stabilirea caracterului iniţial este făcută de la sfârşitul şirului. Dacă *lung* este negativ ultimul caracter al subşirului este cel aflat la distanţa *lung* de sfârşitul şirului. Întoarce întotdeauna cel puţin un caracter, cel dat de *start*.
- Exemple:

```
$rest = substr("abcdef", 1); // întoarce "bcdef"
$rest = substr("abcdef", 1, 3); // întoarce "bcd"
$rest = substr("abcdef", -1); // întoarce "f"
$rest = substr("abcdef", -2); // întoarce "ef"
$rest = substr("abcdef", -3, 1); // întoarce "d"
$rest = substr("abcdef", -1, -1); // întoarce "bcde"
```

Alte functii

```
int is_array(mixed var);
```

Întoarce *true* dacă variabila este un *array* şi *false* altfel.

```
int is_double(mixed var);
int is_float(mixed var);
int is_real(mixed var);
```

Întorc *true* dacă variabila este un număr real şi *false* altfel.

```
int is_int(mixed var);
int is_integer(mixed var);
int is_long(mixed var);
```

Întorc *true* dacă variabila este un număr întreg şi *false* altfel.

Alte functii

```
int is_object(mixed var);
```

Întoarce *true* dacă variabila este un obiect şi *false* altfel.

```
int is_string(mixed var);
```

Întoarce *true* dacă variabila este un şir de caractere şi *false* altfel.

```
int isset(mixed var);
```

Întoarce *true* dacă variabila există şi *false* altfel.

```
int unset(mixed var);
```

Elimină o variabilă.

```
string strval(mixed var);
```

Converteşte la şir de caractere o variabilă scalară

Bibliografie

- Documentatia PHP

http://www.php.net/docs.php

Cateva carti disponibile online:

- Sams - Teach Yourself PHP in 10 Minutes(2005)

http://www.net130.com/CMS/Pub/book/book_web/book_web_php/2005_10_19_70383.htm

- Object Oriented PHP Concepts Techniques and Code (si altele), la adresa:

http://cid-846ffdcf0d3320d8.skydrive.live.com/browse.aspx/eBook