Funcţii în PHP

```
function nume_functie(param1, param2,…,paramN){ instrucţiuni;    }

$var_returnată=nume_funcţie(param1,param2,..,paramN);
```

O funcţie poate fi definită oriunde în cadrul script-ului, iar în interiorul unei funcţii poate să apară orice secvenţă validă de cod (poate cuprinde definirea altor funcţii, clase etc.). Pentru ca funcţia să returneze un rezultat se foloseşte construcţia return urmată de un parametru ce reprezintă valoarea funcţiei.

Funcţii de afişare

PHP include doua functii utile pentru generarea datelor de iesire formatate.

printf() si **sprintf()**.

Functia **printf()** afiseaza datele sale de iesire, iar **sprintf()** returneaza datele sale de iesire sub forma unei valori sir.

string **sprintf** (string $format [, mixed $args [, mixed $...]])

Returns a string produced according to the formatting string *format*.

(a) Parametri

format

The format string is composed of zero or more directives: ordinary characters (excluding %) that are copied directly to the result, and *conversion specifications*, each of which results in fetching its own parameter. This applies to both **sprintf()** and printf().

Each conversion specification consists of a percent sign (%), followed by one or more of these elements, in order:

1. An optional *sign specifier* that forces a sign (- or +) to be used on a number. By default, only the - sign is used on a number if it's negative. This specifier forces positive numbers to have the + sign attached as well, and was added in PHP 4.3.0.

2. An optional *padding specifier* that says what character will be used for padding the results to the right string size. This may be a space character or a *0* (zero character). The default is to pad with spaces. An alternate padding character can be specified by prefixing it with a single quote ('). See the examples below.
3. An optional *alignment specifier* that says if the result should be left-justified or right-justified. The default is right-justified; a - character here will make it left-justified.
4. An optional number, a *width specifier* that says how many characters (minimum) this conversion should result in.
5. An optional *precision specifier* in the form of a period (`.') followed by an optional decimal digit string that says how many decimal digits should be displayed for floating-point numbers. When using this specifier on a string, it acts as a cutoff point, setting a maximum character limit to the string.
6. A *type specifier* that says what type the argument data should be treated as. Possible types:
    - *%* - a literal percent character. No argument is required.
    - *b* - the argument is treated as an integer, and presented as a binary number.
    - *c* - the argument is treated as an integer, and presented as the character with that ASCII value.
    - *d* - the argument is treated as an integer, and presented as a (signed) decimal number.
    - *e* - the argument is treated as scientific notation (e.g. 1.2e+2). The precision specifier stands for the number of digits after the decimal point since PHP 5.2.1. In earlier versions, it was taken as number of significant digits (one less).
    - *E* - like *%e* but uses uppercase letter (e.g. 1.2E+2).
    - *u* - the argument is treated as an integer, and presented as an unsigned decimal number.
    - *f* - the argument is treated as a float, and presented as a floating-point number (locale aware).
    - *F* - the argument is treated as a float, and presented as a floating-point number (non-locale aware). Available since PHP 4.3.10 and PHP 5.0.3.
    - *g* - shorter of *%e* and *%f*.
    - *G* - shorter of *%E* and *%f*.
    - *o* - the argument is treated as an integer, and presented as an octal number.
    - *s* - the argument is treated as and presented as a string.
    - *x* - the argument is treated as an integer and presented as a hexadecimal number (with lowercase letters).
    - *X* - the argument is treated as an integer and presented as a hexadecimal number (with uppercase letters).

Exemple:

```
<?php
$num = 5.7;
echo $num;
$num = sprintf("%05.2f", $num);
echo '<br>'. $num;
?>      Rezultatul afisat va fi:  5.7  05.70
```

Cu ajutorul expresiei "%05.2f", sprintf() formateaza numarul din $num astfel incat acesta sa fie format din 5 caractere dintre care unul punct (.) si 2 zecimale.
exemplu cu **printf()**:

```
<?php
$n = 8;
printf("Valoarea lui n este: %d", $n);
?>  Va afisa "Valoarea lui n este: 8".
```

Funcţii pentru şiruri

- *str_repeat($şir, $n)* – repetă şirul $şir de un număr de n ori.
- *strrchr($şir, $caracter)* –returnează parte a unui $şir, începând cu ultima apariţie a caracterului $caracter în şirul $şir.
- *trim($şir)* – elimină spaţiile din stânga şi din dreapta unui şir.
- *explode($separator, $şir)* – "rupe" valorile dintr-un şir în care ele sunt delimitate de un separator, şi le plasează într-un vector
- *implode($şir, $vector)* - preia valorile dintr-un vector şi le reuneşte într-un şir
- *number_format($număr)* – afişează valoarea numerică folosind separatorul de mii.
- *strpos($sir_princip,$sir_căutat)* - returnează poziţia în care se regăseşte şirul căutat în şirul principal.
- *substr($şir,$start,$end)* – extrage parte dintr-un şir începând din poziţia $start şi până în poziţia $end.
- *int strlen(string str)* –returnează lungimea unui şir de caractere;
- *string strstr(sirul de baza, sirul cautat)* – returnează subşirul din şirul de bază care începe cu şirul căutat (exemplu*: $email = 'abc@utm.com'; $domain = strstr($email, '@'); print $domain; // tipareste @utm.com.*).
- *string strtolower(string str)* – converteşte un şir la litere mici.
- *string strtoupper(string str)* – converteşte un şir la litere mari.
- *string ucwords(string str)* – converteşte un şir astfel încat va avea fiecare iniţiala a fiecarui cuvant scrisă cu majusculă. Restul literelor rămân neschimbate.
- *string ucfirst(string str)* – converteşte un şir astfel încât va fi scris cu iniţiala majuscula. Restul literelor ramân neschimbate.

- *int strcmp(string str1, string str2)* – compară şirul str1 cu şirul str2 din punct de vedere al codului ASCII, şi returnează urmatoarele valori întregi:<0 dacă *str1* este mai mic decat *str2*, > 0 dacă *str1* este mai mare decat *str2* şi 0 dacă sirurile sunt egale.
- *trim()* - funcţie care elimina spatiile goale de al inceputul şi sfarsitul unui sir de caractere specificat ca parametru (asemanator funcţie standard în C);

Unele funcţii PHP au argumente opţionale, care pot fi specificate sau omise, în conformitate cu intenţiile programatorului.

Când se produce o eroare în timpul execuţiei unei funcţii, PHP generează mesaje de eroare. Uneori, asemenea mesaje de eroare sunt nedorite. În acest caz, puteti suprima generarea mesajelor de eroare prin prefixarea numelui funcţie invocate cu ajutorul caracterului @. De exemplu, pentru a suprima mesajele de eroare care pot aparea în timpul execuţiei funcţie f( ), invocati aceasta funcţie dupa cum urmează: Y = @f(x);

Definirea argumentelor prestabilite

PHP va permite să definiţi funcţii cu argumente prestabilite. Dacă invocati o funcţie care are un argument prestabilit, dar nu furnizaţi nici o valoare pentru argumentul respectiv, argumentul ia o valoare prestabilită specificată. Exemplu:

```
function calcul($val , $rata = 0.01)
{
    echo "<BR>cant=$val";
    echo "<BR>rata=$rata";
    return $val * $rata;
}
$cumparaturi = 123.45;
echo "<BR>cumparaturi = $cumparaturi";
$impozit = calcul($cumparaturi,0.1);
echo "<BR>impozit = $impozit";
$cumparaturi = 123.45;
echo "<BR>cumparaturi = $cumparaturi";
$impozit = calcul($cumparaturi);
echo "<BR>impozit = $impozit";
```

Valoarea lui n este: 8
cumparaturi = 123.45
cant=123.45
rata=0.1
impozit = 12.345
cumparaturi = 123.45
cant=123.45
rata=0.01
impozit = 1.2345

Utilizarea MySQL

MySQL este un server de baze de date disponibil gratuit, cu sursa deschisă (open-source) care oferă fiabilitate şi avantaje reale. A fost dezvoltat de firma suedeză MySQL AB. Administrarea bazei de date se realizează folosind utilitare care lucrează în linia de comandă. Cel mai important utilitar este mysql, un shell interactiv pentru controlul şi interogarea bazei de date. Utilitarele rulează cel mai bine pe sistemul Linux, platformă pe care MySQL a fost dezvoltat iniţial. Alte două utilitare cu sursă deschisă, oferite pe platformă Windows, care oferă un set de comenzi de administrare sunt *MySqlManager*, un utilitar de interogare în mod grafic similar cu *mysql* şi *WinMySQL* admin, o consolă pentru administrarea detaliilor configurării lui MySQL. Recent cea mai utilizată metodă pentru serverele care au instalat panoul de comandă CPANEL este **PHPMyAdmin**, care oferă o interfată grafică pentru manipularea datelor din MySQL.

MySQL operează în bază unui model client/server. Orice maşină care doreşte sa proceseze interogări asupra unei baze de date MySQL trebuie să ruleze MySQL server(mysqld), care este responsabil de tot traficul de tip intrări/ieşiri (incoming/outgoing) cu bază de date. Modelul de securitate folosit de MySQL se bazează pe nume de utilizator, parolă, nume server (hostname) sau adresă de IP şi privilegii, fiind similar celui generic folosit de sistemele Unix. Prin privilegii se înţeleg în cazul MySQL operaţiunile ce vor fi permise asupra bazei/bazelor de date, tabelelor sau indecşilor, cum sunt de exemplu SELECT, INSERT, UPDATE, DELETE, CREATE, DROP.

Datele sunt obiectul celor mai multe operaţii de prelucrare, iar sistemele de gestiune a bazelor de date furnizează cele mai complexe şi puternice facilităţi pentru lucrul cu datele. PHP include o bibliotecă de funcţii care furnizează o interfaţă cu sistemul MySQL de gestiune a bazelor de date. Folosind aceste funcţii, un program PHP poate obţine accesul la datele rezidenţe într-o bază de date MySQL şi le poate modifica.

O baza de date (în cazul nostru MySQL) este un program ce poate stoca o cantitate foarte mare de informaţii şi o poate organiza într-un format accesibil în mod direct sau de către un alt program (in cazul nostru PHP).

Într-o bază de date, informaţia este organizată sub formă tabelară, în care coloanele se numesc câmpuri iar liniile se numesc înregistrări. Capul de tabel determină structura bazei de date. Un sistem de gestionare a bazelor de date (SGBD) este un program care permite utilizatorilor interacţiunea cu baza de date. Un SGBD asigură:

- crearea bazei de date
- introducerea informaţiilor în baza de date
- actualizarea informaţiilor
- extragerea datelor
- controlul accesului la date

Obiectivul esenţial, al unui SGBD este furnizarea unui mediu eficient, adaptat utilizatorilor care doresc să consulte sau să actualizeze informaţiile conţinute în baza de date.O baza de date poate conţine mai multe tabele, ce pot fi legate intre ele.

Un câmp se caracterizează prin:

- numele câmpului (reprezintă un nume simbolic prin care câmpul se poate identifica),
- tipul câmpului (pentru identificarea tipului de date care pot fi stocate în câmpul respectiv),
- lungimea câmpului (numărul maxim de caractere care pot fi stocate în câmpul respectiv).

Caracteristicile MySQL-ului sunt:

- este o platformă deosebit de stabilă;
- este independent de sistemul de operare pe care ruleaza (Windows, Linux, Unix, etc);
- este gratuit în anumite condiţii de licenţiere (Open Source Software) .

Afişarea interogării în execuţie şi rularea ei pe baza de date se face cu ajutorul unor aplicaţii separate. Cele mai bune două instrumente sunt:

- Monitorul MySQL – un instrument cu linie de comandă pentru interactionarea cu serverul MySQL;
- phpMyAdmin, o interfaţă MySQL bazată pe PHP.

Tipuri de câmpuri dată/oră

Există în MySQL cinci tipuri de câmpuri folosite pentru stocarea datei calendaristice şi a orei care sunt:

- Date
- Datetime
- Timestamp
- Time
- Year

Câmpul de tip **date** stochează valori în format AAAA-LL-ZZ şi permite introducerea valorilor cuprinse între 1000-01-01 şi 9999-12-31.
Câmpul de tip **datetime** stochează valori în format AAAA-LL-ZZ HH:MM:SS, cuprinse între 1000-01-01 00:00:00 şi 9999-12-31 23:59:59.
Câmpul de tip **timestamp** stochează automat timpul atunci când se modifică valoarea unei înregistrări (printr-o operaţie de introducere sau actualizare).
Câmpul de tip **time** stochează timpul în format HH:MM:SS.
Câmpul de tip **year** poate stoca date cuprinse între 1901 şi 2155.
Câmpuri de tip şir sunt:

- Char
- Varchar
- Tinytext
- Text
- Mediumtext
- Longtext
- Enum

Câmpul de tip **char** are lungimea maximă de 255 caractere. Este de lungime fixă (atunci când introducem o valoare cu lungimea mai mică decât lungimea maximă a câmpului, câmpul va fi completat în partea dreaptă cu spaţii).
Câmpul de tip **varchar** are lungimea maximă de 255 caractere, dar este de lungime variabilă (câmpurile nu vor mai fi completate cu spaţii ca la tipul char).
Câmpurile de tip **blob** şi text pot stoca o cantitate variabilă de date.
Câmpurile de tip **enum** permit stocarea unei valori dintr-o mulţime de valori specificată.

Principalele comenzi MySQL

Cele mai uzuale operaţii cu bazele de date sunt:

Comanda	Semnificatie
CREATE	crează o baza de date sau un tabel
DROP	sterge o baza de date sau un tabel
INSERT	adauga inregistrari intr-un tabel
DELETE	sterge inregistrari dintr-un tabel
UPDATE	actualizeazăninregistrarile dintr-un tabel
SELECT	selectează un tabel
ALTER	modifică structura unui tabel
SHOW	Afişare baze de date, tabele
USE	Deschide o bază de date

1. Crearea unei baze de date se face cu comanda:
CREATE DATABASE nume_bază;

De exemplu, crearea bazei de date, numită student se realizează cu comanda
CREATE DATABASE student;
Caracterul **;** este obligatoriu la sfârşitul oricărei comenzi..

2. Afişarea bazelor de date existente pe server se face cu comanda:
SHOW DATABASES;

3. Accesarea (deschiderea) unei baze de date pentru a putea fi folosită se face cu comanda:
USE nume_bază;

4. Crearea unei tabele într-o bază de date presupune mai întâi deschiderea bazei de date şi apoi crearea propriu-zisă a tabelei:
USE biblioteca;

```
CREATE TABLE carti (
  codc int(4) NOT NULL auto_increment,
  numecarte varchar(40) default NULL,
  autor varchar(30) default NULL,
  data date default NULL,
  pret int(3) NOT NULL default '0',
  stoc int(5) default NULL,
  valoare int(5) default NULL,
  PRIMARY KEY  (codc)
) TYPE=MyISAM;
```

Explicaţii:

- AUTO_INCREMENT funcţionează cu orice tip intreg. La fiecare rând nou adăugat în baza de date, numarul asociat va fi incrementat;
- NULL înseamnă fără valoare (diferit de spaţiu sau zero);
- NOT NULL înseamnă că orice înregistrare completată cu ceva; PRIMARY KEY reprezintă elementul de referinţă pentru fiecare linie.

5. Afişarea tabelelor conţinute de o bază de date presupune deschiderea bazei de date şi apoi folosirea comenzii
SHOW TABLES;

6. Afişarea structurii unei tabele se face cu comanda
DESC nume_tabelă;
În acest caz, vor fi afişate numele câmpurilor, tipul şi lungimea lor.

7. Pentru a modifica structura unei tabele se foloseşte comanda ALTER TABLE.

De exemplu, pentru a modifica lungimea câmpului pret de la int(3) la int(4) se foloseşte comanda
ALTER TABLE carti MODIFY pret int(4);

Pentru a adăuga un nou câmp, numit observatii, comanda este:
ALTER TABLE `cursuri`.`carti` ADD `observatii` VARCHAR(40) NOT NULL;

Pentru a schimba denumirea câmpului observatii în obs, comanda este:
ALTER TABLE `cursuri`.`carti` CHANGE `observatii` `obs` VARCHAR(40) NOT NULL;

8. Ştergerea unei tabele se face cu comanda
DROP TABLE;
De exemplu, pentru ştergerea tabelei numită „diverse”, vom folosi comanda
DROP TABLE diverse;
Comanda DROP TABLE trebuie folosită cu mare grijă, întrucât, în urma executării ei, atât structura cât şi datele conţinute în tabele sunt şterse.

9. Comanda INSERT introduce înregistrări într-o tabelă existentă.
Forma generală a comenzii este:
INSERT INTO nume_tabelă [(câmp1,camp2,…,câmp n)] VALUES (valoare1,valoare2,…, valoare n);

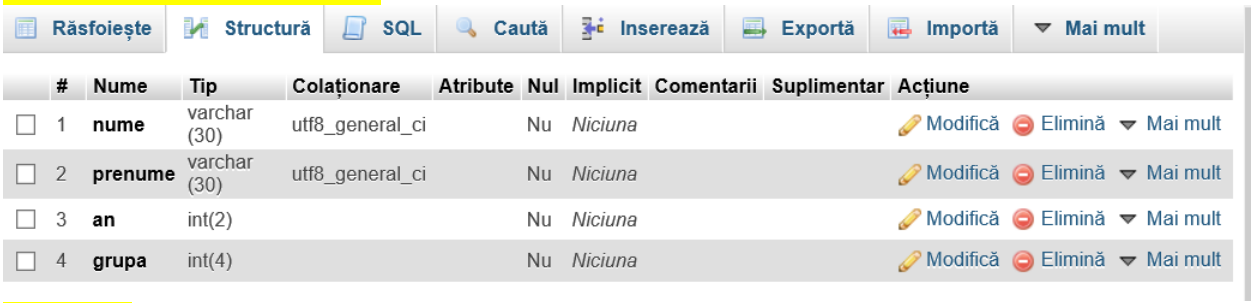
10. Comanda SELECT este utilizată pentru a extrage înregistrările din una sau mai multe tabele. Sintaxa generală este:

SELECT [DISTINCT] câmp1, câmp2,…, câmp n
FROM nume_tabelă
WHERE condiţie
GROUP BY nume_câmp
ORDER BY nume_câmp [ASC | DESC]
LIMIT [numărul_primei_înregistrări_dorite, numărul_de_înregistrări_returnat]

PHP permite lucrul cu un număr mare de funcţii MySQL. În PHP exista funcţii pentru toate operatiile executate asupra bazelor de date MySQL.
Aplicatia 1

nume1: prenume:
anul de studii: grupa: Trimite Anuleaza

Baza de date “test” Table “utm”



	#	Nume	Tip	Colaționare	Atribute	Nul	Implicit	Comentarii	Suplimentar	Acțiune
☐	1	nume	varchar (30)	utf8_general_ci		Nu	*Niciuna*			Modifică Elimină Mai mult
☐	2	prenume	varchar (30)	utf8_general_ci		Nu	*Niciuna*			Modifică Elimină Mai mult
☐	3	an	int(2)			Nu	*Niciuna*			Modifică Elimină Mai mult
☐	4	grupa	int(4)			Nu	*Niciuna*			Modifică Elimină Mai mult

Insert.html

```
<html>
<head>
</head>
<body>

<form name="f" method="post" action="insert.php">
nume1:<input type="text" name="n" />
prenume:<input type="text" name="p" /><br />
anul de studii:<input type="text" name="an" />
grupa:<input type="text" name="grupa" />
<input type="submit" value="Trimite" />
<input type="reset" value="Anuleaza" />
</form>

</body>
</html>
```

insert.php

```
<?php
//include "conexiune.php";
$con=mysqli_connect("localhost","root","","test");
if (mysqli_connect_errno())
  {
  echo "Failed to connect to MySQL : " . mysqli_connect_error();
  }
mysqli_select_db($con,"test");
//preiau datele din formular
$nume=$_POST['n'];
$prenume=$_POST['p'];
$an=$_POST['an'];
$grupa=$_POST['grupa'];
//lcfirst() - Make a string's first character lowercase
//strtolower() - Make a string lowercase
```

```
//strtoupper() - Make a string uppercase
//ucwords() - Uppercase the first character of each word in a string
$nume=ucwords($nume);
$prenume=ucwords($prenume);

$a="INSERT INTO utm (nume, prenume, an,grupa) VALUES ('$nume','$prenume',$an,$grupa);";
//echo $a;
if (mysqli_query($con,$a)) {echo "datele au fost introduse";

} else {
 echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

echo $nume."   ".$prenume;
echo "
<form name='a' method='post' action ='raspuns.php'>
continuati:<input type='text' name='r' size='1' value='d' /><b>da/nu</b>
<input type='submit' value='trimite' />
</form>
";
?>
```

raspuns.php

```
<?php
$rasp=$_POST['r'];
//echo $rasp;
if((strtoupper($rasp))=='D')
{include "insert.html";}
else
        {include "meniu.html";}
?>
```

conexiune.php

```
<?php
$con=mysqli_connect("localhost","root","","test");
if (mysqli_connect_errno())
  {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
  }
mysqli_select_db($con,"test");

?>
```

Aplicatia 2

```
<html>
<head>
<title>Afisare</title>
</head>
<body>
<form name="f1" method="post" action="afisare.php">
```

```
alegeti anul
<input type="text" name="anul" />
<input type="submit" value="Afiseaza lista" class="c3">
</form>
</body>
</html>
```

afisare.php

```
<?php

$con=mysqli_connect("localhost","root","","test");
if (mysqli_connect_errno())
  {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
  }
mysqli_select_db($con,"test");
$anul=$_POST['anul'];


$x="select * from utm where an=$anul order by an desc,grupa,nume asc";

$y=mysqli_query($con,$x);

          echo "<table border='1' bgcolor='B0E0E6'>
                    <tr>

         <td><b>Nume</b></td>
         <td><b>Prenume</b></td><td><b>Anul</b></td><td><b>Grupa</b></td>
        </tr>";

//lista tabel

$t=0;

while($z = mysqli_fetch_assoc($y))
{
          $t++;
          if($t%2==0)
                    echo "<tr bgcolor='#C0C0C0'>

                              <td>".$z["nume"]."</td>
                        <td>".$z["prenume"]."</td>
                              <td>".$z["an"]."</td>
                              <td>".$z["grupa"]."</td>
                        ";
          else
                    echo "<tr bgcolor='#DCDCDC'>

                              <td>".$z["nume"]."</td>
                        <td>".$z["prenume"]."</td>
```

```
                                <td>".$z["an"]."</td>
                                <td>".$z["grupa"]."</td>
                        ";
}
                echo "</table>";
echo "
<form name='a' method='post' action ='raspuns1.php'>
continuati:<input type='text' name='r' size='1' value='d' /><b>da/nu</b>
<input type='submit' value='trimite' />
</form>
";

?>
```

'raspuns1.php'

```
<?php
$rasp=$_POST['r'];
//echo $rasp;
if((strtoupper($rasp))=='D')
{include "afisare.html";}
else
{include "meniu.html";}

?>
```

meniu.html

```
<html>
<head>
<title>Meniu Aplicatie</title>
</head>
<body>
<a href="insert.html"> <b>Inserare date in BD</b></a><br>
<a href="afisare.html"> <b>Afisare date in BD</b></a><br>
</body>
</html>
```

Aplicatie cos cumparaturi fara baze de date

```
<?php
        session_start();
        $i=0;
        $produse[$i]["denumire"] = "produs 1"; $produse[$i++]["pret"] = "5";
        $produse[$i]["denumire"] = "produs 2"; $produse[$i++]["pret"] = "10";
        $produse[$i]["denumire"] = "produs 3"; $produse[$i++]["pret"] = "15";
        $produse[$i]["denumire"] = "produs 4"; $produse[$i++]["pret"] = "20";
        $produse[$i]["denumire"] = "produs 5"; $produse[$i++]["pret"] = "25";
```

```
	$produse[$i]["denumire"] = "produs 6"; $produse[$i++]["pret"] = "30";
	$produse[$i]["denumire"] = "produs 7"; $produse[$i++]["pret"] = "35";
	$produse[$i]["denumire"] = "produs 8"; $produse[$i++]["pret"] = "40";
	$produse[$i]["denumire"] = "produs 9"; $produse[$i++]["pret"] = "45";
	$produse[$i]["denumire"] = "produs 10";		$produse[$i++]["pret"] = "50";
		if(isset($_POST["adauga"])) {
		$cant = $_POST["cant"];
		if(is_numeric($cant)) {
			$pr["id"] = $_POST["prod"];
			$pr["cant"] = $cant;
			$gasit = false;
			if(isset($_SESSION["cos"]) && is_array($_SESSION["cos"]))
			foreach($_SESSION["cos"] as $i=>$p) {
				if($p["id"]==$pr["id"]) {
					$_SESSION["cos"][$i]["cant"] = $pr["cant"];
					if($_SESSION["cos"][$i]["cant"]<=0)
						unset($_SESSION["cos"][$i]);
					$gasit = true;
				}
			}
			if(!$gasit)
				$_SESSION["cos"][] = $pr;
		}
	}
?>
<html>
	<head><title>Exemplu Cos</title>
	<script>
	function sterge(x)
	{
		x.value="";
	}
	</script>
	</head>
	<body>
		<form action="TEST.php" method="post">
			<label for="prod">
			Lista Produse:
			</label>
			<select id="prod" name="prod">
				<?php
					foreach($produse as $k=>$p):
				?>
				<option value="<?php echo $k; ?>"><?php echo $p["denumire"]." -
".$p["pret"]." RON"?></option>
				<?php endforeach; ?>
			</select>
			<label for="cant">
			Cantitate:
			</label>
```

```
                    <input type="text" name="cant" id="cant" onclick="sterge(cant)" />

                    <input type="submit" value="Afisare in pagina" name="adauga" />
                    <input type="reset" value="anuleaza" />
                </form>
                <br />
                <?php   if(isset($_SESSION["cos"]) && is_array($_SESSION["cos"]) &&
count($_SESSION["cos"])):               ?>
                <table border="1" align="center">
                    <thead>
                        <tr>
                            <th>Denumire Produs</th>
                            <th>Pret unitar</th>
                            <th>Cantitate</th>
                            <th>Valoare [in lei]</th>
                            </tr>
                    </thead>
                    <tbody>
                        <?php
                            foreach($_SESSION["cos"] as $p):
                        ?>
                        <tr><td><?php echo $produse[$p["id"]]["denumire"]; ?></td>
                        <td align="center"><?php echo
sprintf("%01.2f",$produse[$p["id"]]["pret"]); ?></td>
                        <td align="center"><?php echo $p["cant"]; ?></td>
                        <td align="center"><?php echo
sprintf("%01.2f",($produse[$p["id"]]["pret"]*$p["cant"])); ?> </td></tr>
                        <?php   endforeach; ?>
                    </tbody>
                    </table>
                <a href="descarca_cos.php"><i>Goleste cosul</i></a>
                        <?php                   endif;          ?>
        </body>
</html>
```

```
<?php
        session_start();
        unset($_SESSION["cos"]);
        header("Location: TEST.php");
?>
```

Programarea orientată pe obiecte (POO) în PHP

În programarea orientată-obiect un sistem informatic este privit ca un model fizic de simulare a comportamentului unei părţi din lumea reală sau conceptuală. Acest model fizic este definit prin intermediul unui limbaj de programare şi el se concretizează într-o aplicaţie ce poate fi executată pe un sistem de calcul.

Printre avantajele programarii pe obiecte sunt:

- cod mai structurat şi mai lizibil,
- lucrul organizat,
- depanarea mai usoara a programelor,
- refolosirea codului,

Dezavantajele principale sunt:

- rulează mai încet,
- timpii de dezvoltare sunt mai mari.

În PHP 4.x nu sunt implementate toate facilităţile POO. Pentru programarea pe obiecte în PHP, este recomandata utilizarea PHP5.

O clasa este o colectie de variabile şi funcţii care operează asupra variabilelor respective. Implementarea unei clase conţine atât variabile, cât şi funcţii, ea reprezentând un şablon (template) cu ajutorul căruia pot fi create instanţe specifice. Sintaxa folosită pentru declararea unei clase în PHP este:

```
class nume_clasa {
// date membre
var nume_variabilă_1
...
var nume_variabilă_m
// metode
function nume_funcţie_1 (parametri) {
... // definitia funcţie
}...
function nume_funcţie_n (parametri) {
... // definirea funcţie
}
}
```

Pentru numele unei clase poate fi utilizat orice identificator permis în PHP cu o singura exceptie: sdtclass; acest identificator este folosit de PHP în scopuri interne. Pentru a initializa variabilele cu valori care nu sunt constante trebuie folosit un constructor.

Obiecte

Un obiect reprezinta o variabilă de tipul clasei. Fiecare obiect are o serie de caracteristici, sau ***proprietăţi*** cât şi anumite funcţii predefinite numite ***metode***. Aceste proprietăţi şi metode ale unui obiect corespund variabilelor şi funcţiilor din definiţia clasei.

Operaţia de initializare a unui obiect se numeşte instanţiere. Clasele pot fi folosite pentru a genera instanţe multiple. Instanţierea (trecerea de la clasa la obiect) reprezintă,

atribuirea unor proprietăţi specifice clasei, astfel încât aceasta să indice un obiect anume, care se diferenţiază de toate celelalte obiecte din clasa printr-o serie de atribute.

Proprietăţi

În corpul unei clase, se pot declara variabile speciale, numite proprietăţi. În PHP 4.x acestea se declară cu cuvantul cheie *var în* fata.

```
class Dictionar {
  var $traduceri = array();
  var $tip = "Ro";
}
```

Aceasta sintaxa este în continuare acceptata (în PHP 5), dar doar pentru a asigura compatibilitate cu versiuni anterioare de PHP.

In PHP 5, codul arata astfel:

```
/**
 * PHP versiunea 5
 */
 class Dictionar {
  public $traduceri = array();
  public $tip = "Ro";}
```

Proprietăţile obiectelor se pot accesa folosind operatorul "->".

Metode

Metodele sunt funcţii cu ajutorul carora se poate opera asupra variabilelor clasei. Într-o formă simplă, metodele sunt funcţii declarate în interiorul unei clase.

$this este o pseudo variabilă ce reţine adresa obiectului curent (referinţa către obiectul curent). Exemplu:

```
class operatori {
var $x = 5;
var $y = 4;
function Suma ( ) {
return $this -> x + $this -> y;
}
function Produs ( ) {
return $this -> x * $this -> y;
}
}
```

Pentru a crea un obiect de tipul **operatori** vom utiliza o instructiune *$oper = new operatori; M*etodele clasei se pot apela

```
echo "suma este ".$operatori -> Suma ( )."<br>";
echo "produsul este ".$operatori -> Produs ( );
```

Constructori

Un constructor este o metoda (funcţie) a unei clase care este apelata automat în momentul în care este creata o noua instanta a clasei (cu ajutorul operatorului **new**). In PHP este considerata ca fiind un constructor orice funcţie care are acelasi nume cu clasa în interiorul careia este definita. Constructorii pot fi folosiţi pentru initializarea datelor membre cu valori care nu sunt constante. Ei pot avea argumente, iar acestea pot fi optionale. Pentru a putea utiliza clasa fara a specifică nici un parametru în momentul crearii unui obiect, se recomanda stabilirea unor valori implicite pentru toate argumentele constructorului. În cazul în care nu este definit un constructor pentru o anumita clasa, se utilizează constructorul clasei de bază, dacă aceasta există.