Limbajul PHP (Hypertext Preprocessor, 1994)

Exemplu de aplicatie cu script PHP

Fisier HTML (**pag2.html**):

```
<html>
<head></head>
<body>
<form action="http://localhost:8888/apl2.php">
<input type="submit" value="Primul exemplu de apel PHP">
</form>
</body>
</html>
```

Fisier PHP (**apl2.php**):

```
<?php
echo ("Ai apasat, am raspuns");
?>
```

C4

Functii in PHP

In PHP se pot crea functii fara prea mari diferente fata de cele din limbajele C/C++.

Exemplu
Script-ul de mai jos calculeaza n! prin utilizarea unei functii recursive:

```
<?php
function fact($n)
{
$p=1;
for($i=1; $i<=$n; $i++)
        $p=$p*$i;
return  $p;
}
echo fact(4);
?>
```

Dupa executia scriptului se va afisa 24.

Functii "matematice"

-abs(numar)-returneaza modulul numarului, abs(-7)=7;
-sin(x),cos(x),tan(x) - sinusul,cosinusul si tangenta unui unghi(argumentul x este in radiani)
-exp(x)-returneaza e^x
-pow(x,y)-returneaza x^y

-log10(x),log(x)-returneaza $\log_{10}(x)$ respectiv $\log_2(x)$
-max(x1,x2,…,xn)-returneaza maximul, minimul dintr-un sir de
min(x1,x2,…,xn) argumente numerice
-ceil(x)-returneaza cel mai mic intreg mai mare sau egal cu x, ceil(4.3)=5;
-round(x)-returneaza intregul rezultat prin rotunjirea lui x, round(4.3)=4;
-floor(x)-returneaza cel mai mare intreg mai mic sau egal cu x, floor(4.3)=4;
-rand(min,max)-returneaza o valoare intreaga aleatoare intre valorile min si max (inclusiv), rand(1,10) poate returna orice valoare intreaga din[1,10]
-pi()=returneaza numarul π
-sqrt(x)-returneaza radacina patrata a lui x, sqrt(4)=2;

Afisarea datelor – echo si print

Pentru a afisa datele pe browser se folosesc **echo** si **print**. Ele sunt constructii special PHP, ambele afisand siruri de caractere.

A) **echo**- poate fi utilizata in multe feluri:
Ex. 5.36: afisam un sir de caractere folosind parantezele rotunde
echo(“Un mesaj”) ;
Ex. 5.37: afisam un sir de caractere fara sa utilizam parantezele rotunde
echo “Un mesaj”;
Ex. 5.39: putem afisa si continutul unor variabile:

```
<?php
$x=6;
Echo (“am $x mere”); // sau echo “Am $x mere”;
?>
```

B) print- se utilizeaza la fel ca echo, dar, daca folosim paranteze, se poate folosi valoarea intoarsa: **true**, daca sirul a fost expediat sau **false**, in caz contrar.
Ex. 5.40:

```
<?php
$x=6;
$y=8;
print (“ana are $x mere si <BR> $y pere”);
?>
```

Functii pentru prelucrarea sirurilor de caractere

In PHP exista un set puternic de functii care lucreaza cu siruri de caractere. Sirurile se memoreaza ca o succesiune de caractere ASCII.

Ex 1: in secventa urmatoare, se afiseaza caracterul “U”;

```
$sir=”Un exemplu”;
```

echo ($sir[0]);

Ex 2: operatorul de concatenare a sirurilor este punctul. Se afiseaza “Ana are 9 mere!”:

```
$x=9;
$sir=”Ana  “. “are ”. $x. “ mere!”;
echo $sir;
```

Ex 3: functia **strlen(sir)** returneaza lungimea sirului, in secventa de mai jos se afiseaza 10:

```
$sir=”Un exemplu”;
echo strlen($sir);
```

functia **strpos(sir1, sir2, [poz_start])** cauta daca sir2 este subsir al lui sir1.

Ex 4 pentru secventa urmatoare se va afisa 5:

```
$sir=”Un exemplu”;
echo  (strops($sir, “em”));
```

functia **strstr(sir1, sir2)** returneaza sir1 din pozitia in care a fost gasit sir2, daca sir2 este subsir pentru sir1 sau false, in caz contrar.

Functia **strcmp(sir1,sir2)** compara lexicographic sir1 cu sir2. Valoarea returnata este:
<0, daca sir1<sir2;
>0 ,daca sir1>sir2;
0, daca sir1=sir2;

Functia **substr(sir,ind, [lung])** returneaza subsirul sirului sir, care incepe in pozitia **ind** si are lungimea lung.

Ex. 5: se afiseaza “exemplu”:

```
$sir=”Un exemplu”;
echo (substr($sir,3));
```

Functia **substr_replace(sir1,sir2,ind,[lung])** returneaza sirul rezultat prin inlocuirea in sir1, a subsirului care incepe in pozitia ind si are lungimea lung cu sir2.

Masive in PHP

In PHP exista o “libertate” extraoridinara de lucru cu masive (tablouri unidimensionale, bidimensionale). In primul rand masivele nu se declara. Se utilizeaza implicit.

Exemplu pentru crearea si afisarea unui masiv cu 5 componente:

```
<?php

//crearea unui vector
```

```
for ($i=0;$i<=5;$i++)

$x[$i]=$i;

//afisarea unui vector

for ($i=0;$i<=5;$i++)

echo ($x[$i].” “);

?>
```

Exemplu pentru crearea si afisarea unei matrice cu 6 linii si 6 coloane elementele fiind date de suma liniei si a coloanei pe care se afla elementul respectiv.

```
<?php

//crearea unei matrice

for ($i=0;$i<=5;$i++)

for ($j=0;$j<=5;$j++)

$mat[$i][$j]=$i+$j;

//afisarea unei matrice

for ($i=0;$i<=5;$i++){

        for ($j=0;$j<=5;$j++)

                echo($mat[$i][$j]." ");

        echo(" <BR>");

}

?>
```

In PHP indicii pot fi si siruri de caractere.

Functii pentru masive:

1. count(nume_vector)-returneaza numarul de component ale vectorului.

2. foreach(vector as indice=>variabila)-are rolul de a ne permite o parcurgere usoara a unui vector cu variabila de ciclare indice, in care citirea se face intr-o variabila.

Exemplu:

```
<?php

$x["Ioana"]="0724xxx";

$x["Valentina"]="031xxx";

$x["Cristian"]="0232xxx";

$x["Paul"]="0211xxx";

foreach($x as $i=>$nume)

echo($i." ".$nume."<BR>");

?>
```

3. current(vector)-returneaza valoarea retinuta de elementul din vector asupra caruia se gaseste pointer-ul.

4. key(vector)-returneaza indicele elementului din vector asupra caruia se gaseste pointer-ul.

5. next(vector)-deplaseaza pointer-ul pe elementul uramator din vector si returneaza valoarea retinuta de acesta.

6. prev(vector)-deplaseaza pointer-ul pe elementul anterior al vectorului si returneaza valoarea retinuta de acesta.

Functii de sortare:

1. Functia Asort(vector) sorteaza crescator vectorul dupa valoarea retinuta de fiecare element.

2. Functia Arsort(vector) sorteaza *descrescator* vectorul dupa valoarea retinuta de fiecare element.

3. Functia Ksort(vector) sorteaza crescator vectorul dupa indici.

4. Functia Krsort(vector) sorteaza *descrescator* vectorul dupa indici.

5. Functiile sort() si rsort() sorteaza crescator respectiv descrescator un masiv de indici 0,1,…,n. In cazul in care avem un masiv asociativ,vechile valori ale indicilor se pierd.

Variabile cookie

Atunci cand utilizarea Internet-ului a luat amploare s-a pus urmatoarea problema: cum trebuie procedat, pentru ca server-ul sa utilizeze anumite optiuni ale utilizatorului, astfel incat atunci cand acesta intra pe site sa nu mai piarda timpul selecatandu-le din nou?

Mecanismul care sa la baza acestei probleme se bazeaza pe memorarea pe calculatorul vizitatorului unui anumit site, a unor informatii sub forma unor mici fisiere text. Operatia poate fi comandata de pe server si tot de pe server se poate comanda citirea, actualizarea sau stergerea acestor mici fisire, numite usual, prin abuz de limbaj, variabile cookie.

Pentru creare folosim functia setcookie(nume_variabile, valoare, data_expirare)

Exemplu crearea unei variabile cookie.

```
<?php

        setcookie("copac","verde",time()+3600);

?>
```

Pentru afisare folosim intructiunea $HTTP_COOKIE_VARS.

Exemplu afisare:

```
<?

//afiseaza verde

echo $HTTP_COOKIE_VARS["copac"];

?>
```

==========

Functii in PHP (completare)

In PHP 4 functiile pot fi definite de catre utilizator folosind urmatoarea sintaxa:

```
function numef ($param1, $param2, ..., $paramN) {
// instructiuni
}
```

In PHP4 o functie poate fi definita oriunde in cadrul script-ului si in interiorul unei functii poate sa apara orice secventa valida de cod care include definirea de alte functii si definitii de clase. Argumentele unei functii trebuie separate prin virgula, si, implicit, acestea sunt transmise prin valoare. Pentru ca functia sa returneze un rezultat se foloseste constructia **return** care primeste ca parametru o expresie care

reprezinta valoarea functiei. In momentul in care este intalnita constructia **return**, executia functiei se incheie. In exemplul urmator se calculeaza cu ajutorul unei functii PHP, patratul unui numar.

Exemplu	Rezultat

<?php
function patrat ($n) {
return $n * $n;
}
echo "4^2=".patrat(4)."";
?>
	4^2=**16**

Transmiterea parametrilor prin referinta

Pentru a transmite parametri unei functii prin referinta, fapt care implica modificarea valorii parametrilor si pastrarea noii valori dupa ce executia functiei s-a incheiat, se foloseste operatorul "&" inaintea numelui parametrului formal, in momentul definirii functiei. Urmatorul exemplu indica modul in care se modifica valoarea unei variabile in interiorul unie functii si modul in care aceasta modificare este resimtita in exteriorul acesteia:

Exemplu	Rezultat

<?php
function modificInt ($s) {
$s .= " prima functie.";
echo "In modificInt: ".$s."
";
}
function modificExt (&$s) {
$s .= " a doua functie.";
echo "In modificExt: ".$s."
";
}
$s="Iesire din ";
echo "In script: ".$s."
";
modificInt ($s);
echo "In script: ".$s."
";
modificExt ($s);
echo "In script: ".$s."
";
?>
	In script: Iesire din
	In modificInt: Iesire din prima functie.
	In script: Iesire din
	In modificExt: Iesire din a doua functie.
	In script: Iesire din a doua functie.

Nu exista posibilitatea de supraincarcare a unei functii, de redefinire a ei dupa ce aceasta a fost definita in cadrul scriptului respectiv si nu exista nici posibilitatea de anulare a unei functii. O functie poate fi apelata inainte de definirea ei.

Parametri cu valori implicite

In PHP parametrii formali pot avea valori implicite, si, in cazul in care parametrul actual lipseste, atunci se va considera ca are valoarea implicita. Urmatorul exemplu ilustreaza modul de folosire al functiilor cand acestea au parametri formali cu valori implicite:

Exemplu	Rezultat

<?php
function comanda ($s="cafea") {
return "Ati comandat ".$s.".";
}
echo comanda();
echo "
";
echo comanda ("suc");
?>
	Ati comandat cafea.
	Ati comandat suc.

In cazul in care se folosesc parametri cu valori implicite este necesar ca orice parametru care are o valoare implicita sa se afle in partea dreapta a tuturor parametrilor pentru care nu se folosesc valori implicite, in caz contrar interpretorul PHP nu poate sa decida carui parametru sa-i atribuie valoarea de pe o anumita pozitie din lista de parametri. De exemplu, daca avem o functie a carei declaratie este ***function*** *transform ($baza=10, $nr)* si care returneaza rezultatul transformarii lui *$nr* din baza 16 in baza *$baza*, a carei valoare implicita este 10, daca se apeleaza *transform (50)*, interpretorul nu atribuie valoarea 50 parametrului *$nr*, ci parametrului *$baza* si genereaza o eroare deoarece lipseste valoarea parametrului *$nr*.

Functii cu numar variabil de parametri

O alta facilitate a limbajului PHP este aceea ca ofera programatorului posibilitatea de a utiliza functii care

au un numar nedeterminat de parametri. Functiile care folosesc un numar variabil de parametri nu au nici o particularitate in ceea ce priveste definirea lor. Aceste functii se definesc la fel ca cele prezentate anterior, dar pentru a putea accesa parametri se vor folosi urmatoarele functii predefinite:
- *func_num_args ( )* - aceasta functie returneaza numarul parametrilor functiei care a apelat-o; daca aceasta functie este apelata din exteriorul unei functii definite de utilizator se va genera un mesaj de avertizare;
- *func_get_arg (arg_num)* - returneaza valoarea parametrului care se afla pe pozitia arg_num in lista de parametri; primul parametru are numarul de ordine 0; daca este apelata din exteriorul unei functii definite de utilizator se va genera un mesaj de avertizare;
- *func_get_args ( )* - returneaza un tablou unidimensional care contine valorile parametrilor pe care functia apelanta i-a primit; daca aceasta functie este apelata din exteriorul unei functii definite de utilizator se va genera un mesaj de avertizare. In continuare aveti 2 exemple de utilizare a acestor functii. Primul exemplu afiseaza lista parametrilor functiei folosind
functia *func_num_args* si *func_get_arg*, iar al doilea exemplu afiseaza aceeasi lista folosind numai functia *func_get_args*.

Exemplu	Rezultat

<?php
function lista_parametri ( ) {
for ($i=0; $i<func_num_args ( ); $i++) {
print_r (func_get_arg ($i));
echo "
";
}
}
echo lista_parametri ("Comanda:", 1, "calculator", 2, "procesoare", "configuratie", array ("local", 2, 3));
?>
	Comanda:
	1
	calculator
	2
	procesoare
	configuratie
	Array ([0] => local [1] => 2 [2] => 3)

<?php
function lista_parametrii ( ) {
foreach (func_get_args () as $i) {
print_r ($i);
echo "
";
}
}
echo lista_parametrii ("Comanda:", 1, "calculator", 2, "procesoare", "configuratie", array ("local", 2, 3));
?>
	Comanda:
	1
	calculator
	2
	procesoare
	configuratie
	Array ([0] => local [1] => 2 [2] => 3)

Valorile returnate de functii
Rezultatul obtinut dupa apelarea unei functii poate avea orice tip. O functie poate sa returneze chiar si liste sau obiecte. In PHP exista un caz special de rezultat numit referinta. Pentru ca o functie sa poata returna o referinta, aceasta trebuie declarata folosindu-se operatorul '&' inaintea numelui functiei. Acest operator trebuie sa apara inaintea numelui functiei si in momentul cand o variabila primeste ca valoare referinta rezultata din apelul functiei. In exemplul urmator se defineste o functie al carui rezultat il constituie o referinta:

Exemplu	Rezultat

<?php
function &refer ( ) {
global $s;
return $s;
}
$s = "Acesta este continutul variabilei referite cu ajutorul functiei.";
$z = &refer ( );
echo $z;
?>
	Acesta este continutul variabilei referite cu ajutorul functiei.

Spre deosebire de majoritatea limbajelor de programare moderne, o functie PHP poate sa returneze o referinta la o variabila care a fost declarata in interiorul functiei, insa acest lucru nu este indicat deoarece, in anumite cazuri, poate duce la efecte neasteptate ale executarii unui script PHP. In alte limbaje de programare efectele devin uneori fatale.

Variabilele de tip functie
O alta facilitate a limbajului PHP in ceea ce priveste functiile este aceea ca suporta variabile de tip

functie. Acest lucru este util atunci cand se folosesc liste de functii pentru prelucrarea anumitor tipuri de date. Pentru a atribui un nume de functie unei variabile in PHP se foloseste aceeasi constructie ca in cazul atribuirii unui sir de caractere, si anume, o variabila va primi ca valoare numele functiei scris intre ghilimele simple sau duble. In cazul in care interpretorul PHP gaseste un nume de variabila urmata de o lista de parametri, acesta cauta functia pe care variabila o refera si in cazul in care exista, o executa. Variabilele de tip functie nu functioneaza cu constructii ale limbajului ca echo, unset, isset, empty, include etc. Mai jos aveti un exemplu care ilustreaza modul de lucru cu variabilele de tip functie.

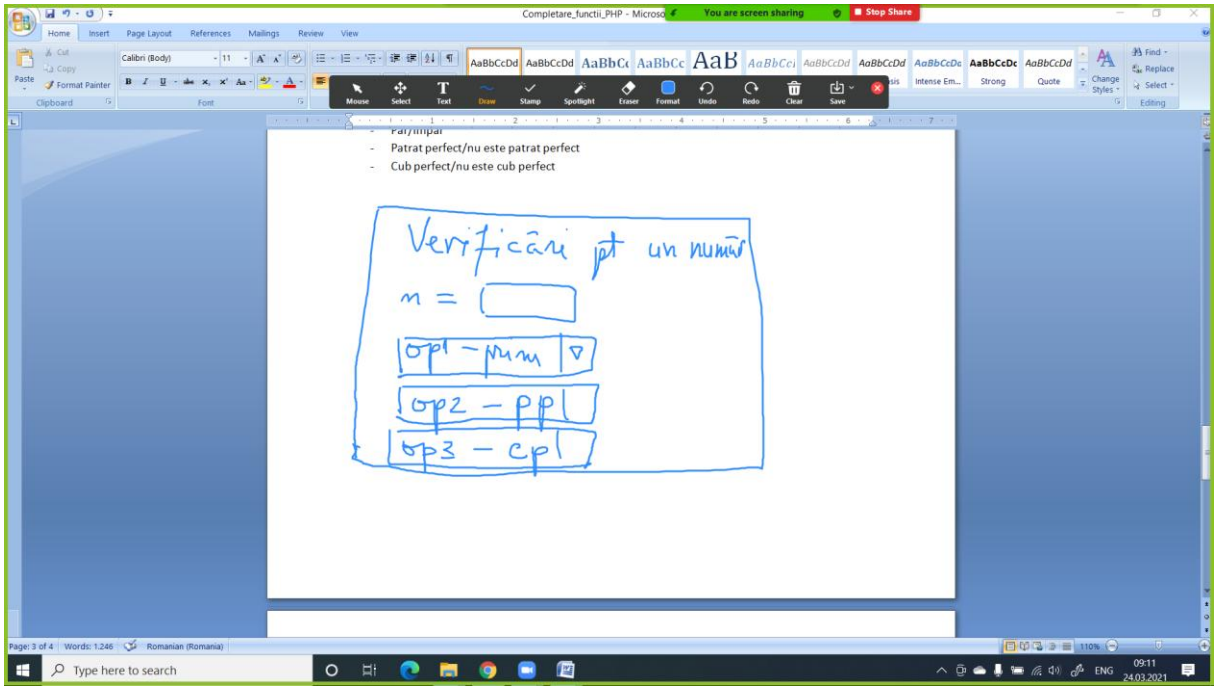
Exemplu	Rezultat

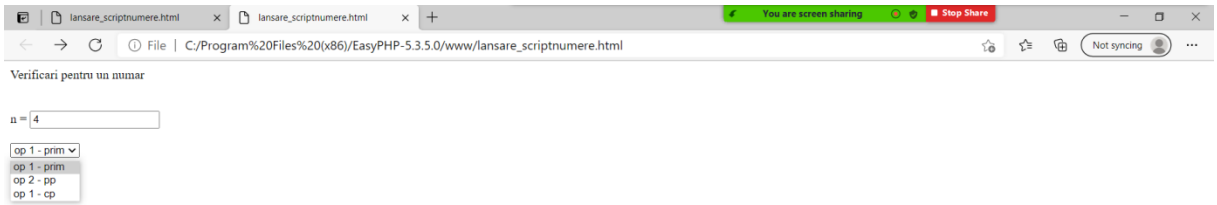
<?php
function produs ($a, $b) {
return $a * $b ;
}
function suma ($a, $b) {
return $a + $b;
}
$operatie = 'produs';
$rez = $operatie (4, 5);
echo "4 * 5 = ".$rez."
";
$operatie = 'suma';
$rez = $operatie (4, 5);
echo "4 + 5 = ".$rez."
";
?>
	4 * 5 = **20**
	4 + 5 = **9**

Aplicatie

Creati o aplicatie web, care sa citeasca un numar natural n si sa afiseze prin intermediul unei liste cu trei optiuni:

- Par/impar
- Patrat perfect/nu este patrat perfect
- Cub perfect/nu este cub perfect





<html>

<head>

</head>

```
<body>
	<form action="http://localhost:8888/numere.php" method="post">
		Verificari pentru un numar <br><br><br>
		n = <input type="text" name="n"> <br><br>
		<select name = "lista">
			<option value = "op 1 - prim">   op 1 - prim </option>
			<option value = "op 2 - pp">      op 2 - pp </option>
			<option value = "op 3 - cp">      op 1 - cp </option>
		</select> <br><br>
		<input type="submit" value="Trimite">
	</form>
</body>
</html>
```

```
<?php
function prim($n){
if($n < 2)
	return 0;
for($i=2;$i*$i<=$n;$i++)
	if($n % $i == 0)
		return 0;
return 1;
}

function pp($n){
```

```
	$i = 1;
	while($i * $i < $n)
		$i++;
	if($i*$i == $n)
		return 1;
	return 0;
}

function cp($n){
	$i = 1;
	while($i * $i * $i < $n)
		$i++;
	if($i*$i*$i == $n)
		return 1;
	return 0;
}

$n = $_POST['n'];
$op = $_POST['lista'];

if($op == "op 1 - prim")
	if(prim($n) == 1)
		echo "numarul ".$n." este prim";
			else
		echo "numarul ".$n." nu este prim";
```

```
if($op == "op 2 - pp")
        if(pp($n) == 1)
                echo "numarul ".$n." este patrat perfect";
                        else
                echo "numarul ".$n." nu este patrat perfect";


if($op == "op 3 - cp")
        if(cp($n) == 1)
                echo "numarul ".$n." este cub perfect";
                        else
                echo "numarul ".$n." nu este cub perfect";

?>
```