Operatorlar

Riyazi operatorlar

+ toplama
- çıxma
* vurma
/ bölmə
% qalıq (məsələn, 5 % 2=1)

Məntiq operatorları

& və (AND)
| və ya (OR)
^ (XOR)
~ inkar (NOT)
>> soldakı ədədi özünün dəhə qədəri mənimsətməklə artırır.
<< soldakı ədədi özünün dəhə qədəri mənimsətməklə azaldır.

Sadə hesab əməliyyatları

Digər proqramlaşdırma dillərində olduğu kimi, burada da hesab əməlləri sadə məntiqlə yerinə yetirilir.
PHP bütün riyazi funksiyaları, mötərizələri, məntiqi funksiyaları, bir vahid artmağa, azalmağa və başqaları.

Məsələn:
$x++; - **x** dəyişəninin bir vahid artması;
$x=$y=4; - **x** və **y** dəyişənlərinin hər ikisi 4-ə bərabərdir;
$b=$a=5;
$c=$a++;

Burada **c** dəyişəni **a** dəyişəninə bərabər olur və bir vahid artır.
$x=2*($a++);

İndi ardıcıl 5 ədədin ekrana yazılması proqramına baxaq:

```
<?php

$i = 1;
echo$i;
```

```
$i++;
echo"".$i;

$i++;
echo"".$i;

$i++;
echo"".$i;

$i++;
echo"".$i;

?>
```

Echo və Print funksiyaları

PHP-də ən çox istifadə olunan funksiyalardan biri də **echo**-dur. **echo**-nun başlıca vəzifəsi daxil edilmiş yazını ekrana çıxarmaqdir.

Koda baxaq:

```
<?php

echo"Salam dünya!";

?>
```

Bu yazını daxil etdikdə, ekrana "Salam dünya!" yazısı çıxacaq.

PHP-də yazını ekrana çıxarmaq üçün başqa funksiyalar da var. Bunlardan biri də, **print** funksiyasıdır:

```
<?php

print"Salam dünya!";

?>
```

Bundan başqa, **echo** və **print**-in başqa cür yazılış şəkilləri də var. Bunlara misal olaraq:

```php
<?php

echo("Salam dünya!");

// və

print("Salam dünya!");

// göstərmək olar.

?>
```

Dəyişənlər

PHP-də məlumatları daşımaq üçün dəyişənlərdən istifadə olunur.

Koda baxaq:

```php
<?php

$a=5;
echo"$a";

?>
```

yazdıqda, ekrana dəyişəndəki dəyər olan 5 dəyərini əks etdirdi.

Dəyişənlər ilk baxışda lazımsız funksiyaya oxşasa da, PHP-də əsas yerini tutur.
Bundan başqa, dəyişənlərin hesabına biz riyazi əməlləri də yerinə yetirə bilərik, məsələn:

```php
<?php

$a=5;
$b=4;
echo$a+$b;

?>
```

Dəyişənlərə xas olan bir xüsusiyyət də, onların ən sonda elan olunan dəyişəni nəzərə almasıdır, məsələn:

```
<?php

$a=5;
$a=4;
echo"$a";

?>
```

Bu kodda proqram sondakı dəyişəni nəzərə alır və nəticədə ekranda 4 dəyərini əks etdirir.

PHP-də məlumatın tipləri. Tiplərin dəyişdirilməsi

Əvvəl deyildiyi kimi, PHP-də dəyişənlərin tipləri ilə bağlı özündə müəyyən elastiklik imkanı verir, proqram daxilində bir dəyişənlə həm yazını, həm də rəqəmi işləmək mümkündür. Lakin buna baxmayaraq, PHP-də əsas məlumat tiplərinin komplekti vardır, hansı ki, dəyişənlərlə bağlı işlərdə açıqcasına göstərilə bilərlər:
integer;
string;
boolean;
double;
array;
object;

Belə bir funksiya var: **gettype**(), hansı ki, PHP dəyişənə təyin etdiyi tipi geri qaytarır (sorğunu geri qaytarmaq):

```
<?php

$var = "5";
$var1 = 5;

echo(gettype($var));
echo"<br>";
echo(gettype($var1));
```

```
?>
```

Birinci halda PHP **string** geri qaytaracaq, ikinci halda **integer**. Həmçinin belə bir funksiya da mövcuddur: **settype**(), hansı ki, tipi təyin etməyə imkan verir :

```
<?php

$var = "5";

echo(gettype($var));

settype($var,integer);

echo"<br>";
echo(gettype($var));

?>
```

Yuxarıda yazdığım kod, bundan əvvəlki kodun nəticəsi ilə eyni olacaq. **settype**() funksiyasından başqa da PHP-də tipləri təyin etmək mümkündür. Belə ki, dəyişənin yeni tipini ona mənimsətmək lazımdır. Bunu etmək üçün aşağıdakı formada yazmaq lazımdır:

```
<?php

$var = (int)$var;

?>
```

Müvafiq olaraq, növbəti kodun yerinə yetirilməsi PHP-nin integer geri qaytarmasına səbəb olur:

```
<?php

$var = "5";             // tip string
$var = (int)$var;       // int dəyişdiririk

echo(gettype($var));

?>
```

Array

PHP-də **array** əsasən bir dəyişəndən istifadə edərək, bir neçə məlumatın daşınmasında istifadə olunur. Kodlara baxaq:

```
<?php

$ar=array("PHP","HTML","JAVASCRIPT");

// Burada $ar dəyişəni array sayılır, biz array-ın içindəki məlumatları bu dəyişən vasitəsilə ekrana çıxara bilərik.

echo'Birinci dəyişən '; echo$ar['0'];// Qeyd: PHP-də say sıfırdan başladığı üçün 0 yazdıq.
echo"İkinci dəyişən"; echo$ar['1'];
echo"Üçüncü dəyişən"; echo$ar['2'];

?>
```

Bu dəyərləri eyni vaxtda ekrana çıxara bilərik, lakin bu, **foreach** funksiyası ilə edilir. Bu funksiya ilə digər dərsdə tanış olacağıq.

if-else funksiyası

if-else, **elseif** funksiyası PHP-də çox istifadə olunan funksiyalardandır. Bu funksiyanın əsas mənası göndərilən məlumatın doğruluğunun yoxlanmasıdır.
Məsələn, **$a**-nın 5-ə bərabər olub-olmamasını yoxlayaq.

```
<?php

$a=0;

if($a==5) {
echo"a 5-ə bərabərdir!";
}
else {
echo"a 5-ə bərabər deyil!";
}
```

```
?>
```

elseif isə, əgər şərt səhvdirsə, yenidən yoxlamadan keçirməkdən ötrüdür. Məsələn:

```
<?php

$a=0;

if($a==5) {
echo"a 5-ə bərabərdir!";
}
elseif($a==0) {
echo"a 0-a bərabərdir!";
}
else {
echo"Hər iki şərt səhvdir!";
}

?>
```

GET və POST metodları

Bu metod vasitəsilə dəyişənlərin qiymətlərini daxil etmək olar. Bu zaman lokal kompüterdən serverə müraciət olunur və dəyişənlərin qiymətini serverə çağırır. Bu metodun yazılış qaydası aşağıdakı kimidir:

```
<form action="hesabla2.php"method="get">

Daxil et x: (bura istənilən sözü yazmaq olar) <input type="text"name="x" /><br />
Daxil et y: <input type="text"name="y" /><br />
<input type="submit"value="Hesabla" />

</form>
```

Burada **x** və **y** dəyişənlərinin qiymətini daxil edib "Hesabla" düyməsini basan kimi "hesabla2.php" faylı yüklənir.

```
$x=$_GET['x'];
```

```
$y=$_GET['y'];
```
Bu zaman GET metodu vasitəsilə **x** və **y** qiymətləri oxunaraq serverin yaddaşına yüklənir.

Misal:

Birinci fayl:

```
<html>
<head>
<meta http-equiv="content-type"content="text/html; charset=iso-8859-1">
<title>Hesabla1</title>
</head>

<body>

<form action="hesabla2.php"method="get">

Daxil et x: (bura istənilən sözü yazmaq olar) <input type="text"name="x" /><br />
Daxil et y: <input type="text"name="y" /><br />
<input type="submit"value="Hesabla" />

</form>

</body>
</html>
```

İkinci fayl:

```
<html>
<head>
<meta http-equiv="content-type"content="text/html; charset=iso-8859-1">
<title>Hesabla2</title>
</head>

<body>

<?php
```

```
$x=$_GET['x'];
$y=$_GET['y'];
$result=$x*$y;

echo"y=$result<br>";

$result=sqrt($x);

echo$result

?>

</body>
</html>
```

Post metodu

Bu dərsimizdə sizinlə birlikdə PHP-də **post** funksiyasını öyrənəcəyik. PHP-də **post** funksiyası müəyyən məlumatın daşınması üçün istifadə olunur. Bunun üçün HTML kod ilə qutucuq yaradırıq:

```
<?php

echo"<input name=\"php\">";

?>
```

İndi isə, **post** metodunu daha dəqiq öyrənək. Kodlara baxaq:

```
<?php

// post əmri gəldikdə görsənməsi üçün

if($_POST) {
echo"$_POST['php']";
}
```

```
elseif"<form action=\"hansı səhifəyə yönlənməsi üçün biz bu səhifədə iş gördüyümüz üçün buranı boş saxlayın\" method=\"POST\">"; {
echo"<input name=\"php\">";        // Qutucuq ("php" sözünü istədiyiniz sözlə əvəz edə bilərsiniz.)
echo"<input type=\"submit\" value=\"Göndər\">";
}

?>
```

Gördüyünuz kimi, əsas tərəfdə kodları yazdıq, ikinci tərəfdə isə, echo $_POST['postun adı']; ilə ekrana çıxardıq.

While funksiyası

while funksiyası funksiyanın təkrarlanması üçün istifadə olunur. Məsələn, bir dəyişənin hər təkrarda artması funksiyasını yazaq. Kodlar:

```
<?php

$a=0;

while($a<100) {
$a++;  // Dəyərin artması üçün
echo"$a<br/>";

?>
```

Bu funksiya əsasən, məntiqi proqramların yazılmasında istifadə olunur. Məsələn, sıfır və yüz arasında beşə bölünən ədədləri ekrana çıxaraq:

Foreach

foreach funksiyası PHP-də gələn məlumatları seçdirmək üçün istifadə olunur. Məsələn, bir array var və içində 100-dən artıq məlumat var. Əlbəttə, onu tək-tək seçib yazmaq çətin olur, bu vaxt **foreach** işə yarayır.
Koda baxaq:

```
<?php
```

```php
$ar=array("facebook.com","ilkaddimlar.com","mobiz.az","google.az");
echo"Dəyərin içindəki məlumatlar:<br/>";
$a=0;

foreach($arasmurik=>$site) {
$a++;
echo"$a)$site<br/>";

?>
```

Sabitlər (Define)

Sabitlər adından göründüyü kimi, sabit qalır. Məsələn, dəyişənlərdə hər hansı bir dəyişənə müdaxilə etmək olurdu.

```php
<?php

$a=5;
$a=0;

echo"$a";

?>
```

yazdıqda, bu vaxt ekrana 0 çıxacaq, sabitlərdə isə bu, istisna haldır. Kodlara baxaq:

```php
<?php

$a=0;

define("ad","Murad");
define("ad","Şükürlü");

?>
```

Gördüyümüz kimi, sabitlərdə dəyişiklik olmadi.

printf() и sprintf() funksiyaları

Qeyd etmək lazımdır ki, hər iki funksiya nəticələrin verilməsi formatını müəyyən etmək məqsədi üçündür. **printf()** funksiyası nəticə selinin brauzerdə verilməsi, **sprintf()** funksiyası isə sətirin veriməsi üçün uyğun formatın seçilməsi üçündür. Yazılış qaydası belədir:

```
int printf(string format [, mixed args])
string sprintf(string format [, mixed args])
```

Ümumiyyətlə, bu funksiyaların yazılışı belə olur:

```
<?php

printf("Hello!");                 // "Hello!" yazılır.

sprintf("Hello!");                // Heç nə verilmir.

$str = sprintf("Hello!");         // Həmin sətri dəyişənə mənimsədir.

printf($str);                     // Həmin söz ekrana verilir.

?>
```

Request metodu

Bu metod da GET və ya POST metodu kimi dəyişənləri serverə göndərmək üçündür.

Şərti keçid operatoru (if)

Bir çox hallarda qoyulmuş şərtdən asılı olaraq, müxtəlif kodları yerinə yetirmək tələb olunur. Bu halda **if** əmrindən istifadə edilir. Yazılış qaydası aşağıdakı kimidir:

```
if (şərt)
ifadə_1;
else
ifadə_2;

<?php

$i = 10;
$j = 5*2;
```

```
if ($i == $j)

echo'$i və $j bərabərdir';

else

echo'$i və $j bərabər deyil';

?>
```

Göründüyü kimi, **$i** və **$j** arasındakı şərt == (iki bərabərlik) işarəsi vasitəsilə yerinə yetirilir. Beləliklə, şərtlərin yoxlanması aşağıdakı işarələr vasitəsilə göstərilir:
Burada **if/else** operatoru şərti operator kimi istifadə olunur. Ümumilikdə bu bele görünür:
!= - bərabər deyil;
< - kiçik;
> - böyük;
< = - kiçik bərabər;
> = - böyük bərabər.

```
<?php

$i = 10;
$j = 5*2;

if ($i == $j)

echo'$i və $j dəyişənlərinin eyni qiyməti var';

?>

<?php

$i = 10;
$j = 11;

if ($i>$j)
$diff = $j - $i;
```

```
echo'$j $i -dən böyükdür; $j və $i arasındakı fərq: '. $diff; // Səhvdir!

?>
```

Bir çox hallarda qoyulmuş şərtə əsasən, bir neçə əmrlər ardıcıllığı yerinə yetirmək tələb olunur. Bu zaman həmin əmrlər fiqurlu mötərizə daxilində yazılır:

```
<?php

$i = 10;
$j = 11;

if ($i>$j) {
$diff = $j - $i;

echo'$j $i -dən böyükdür; $j və $i arasındakı fərq: '. $diff;
}

?>
```

Qeyd etmək lazımdır ki, hətta təkcə bir əmri yerinə yetirmək lazımdırsa, onda da bir sətri fiqurlu mötərizə daxilində yazmaq məsləhətdir. Bu zaman təsadüfi səhvlərə yol vermək şansı azalır. Bundan başqa, şərt daxilində yeni şərt vermək olar və bu zaman yazılış qaydası bu cür olur: if... else if... else.

```
<?php

$i = 10;
$j = 11;

if ($i>$j) {

echo'$i $j -dən böyükdür';

} elseif ($i<$j) {

echo'$i $j -dən kiçikdir';

} else { // Bu halda bərabərlikdən başqa bir şey qalmır :)
```

```
echo'$i və $j bərabərdirlər';
}

?>
```

Dəyişənlərin qiymətini serverə göndərərkən ilkin olaraq, həmin dəyişənin elan olunmamasına görə ekrana bu haqda sorğu çıxacaq. Belə vəziyyəti aradan götürmək üçün şərti keçid əmrindən istifadə olunur.

```
<html>
<head>
<meta http-equiv="content-type"content="text/html; charset=iso-8859-1">
<title>Untitled Document</title>
</head>

<body>

<form action="vurmaq.php"method="post">

<input type="text"name="a"size="20"maxlength="8" /><br />
<input type="text"name="b"size="20"maxlength="8" /><br />
<input type="submit"value="Vurmaq"></form>

<?php

if (isset($_POST["a"])) {$a=$_POST["a"]; }

else {$a=0;};

if (isset($_POST["b"])) {$b=$_POST["b"];

} else {$b=0;};

echo"<br> hasil=".$a*$b;

?>

</body>
```

```
</html>
```

Bu yazılış qaydası hər üç - GET, POST və REQUEST metodları üçün doğrudur.

```
<html>
<head>
<meta  http-equiv="content-type"content="text/html;  charset=iso-8859-1">
<title>Mühazirə 3</title>
</head>

<body>

<form method="post"action="muhaz3.php">

<input name="i"type="text"size="30" />
<input name="j"type="text"size="20" />
<input type="image"src="../photos/search.jpg"border="0" />

</form>

<?php

if (isset($_POST["i"])) {$i=$_POST["i"];}

else {$i=0;};

if (isset($_POST["j"])) {$j=$_POST["j"];}

else {$j=0;};

$i=$_POST["i"];
$j=$_POST["j"];

if ($i>$j)
{

$diff=$j+$i;
```

```
echo"Cəm".$diff;
}

else {

$diff=$j-$i;

echo"Fərq".$diff;
}

?>

</body>
</html>
```

Switch

Bir çox hallarda eyni dəyişənin qiymətindən asılı olaraq, proqramın icrasını şaxələndirmək lazım gəlir. Bu zaman **switch** əmrindən istifadə olunur.

Məsələn, aşağıdakı əmrlər toplusunun əvəzinə

```
if ($i==1) {
ifadə 1
} elseif ($i==2) {
ifadə 2
} elseif ($i==3) {
ifadə 3
}
```

Daha əhəmiyyətli yazılış qaydasından istifadə etmək olar:

```
<form action="switch.php"method="post">

<input type="text"name="i" />
<input type ="submit"value="Ok" />

</form>
```

```php
<?php

if (isset($_POST["i"])) {$i=$_POST["i"];}

else {$i=1;};

switch ($i) {

case1:
echo'Bir';
break;

case2:
echo'İki';
break;

case3:
echo'Üç';
break;

default:
echo'$i > 3';

};

?>
```

switch əmrindən sonra yazılmış dəyişənin qiymətindən asılı olaraq, ekranda **case** əmrindən sonra göstərilmiş qiymətə uyğun sətir yerinə yetirilir. Burada yazılan ifadəni fiqurlu mötərizənin içərisinə almaq lazım deyil. Burada proqramın icrası **break** əmri vasitəsilə yekunlaşır. **default** əmri qalan bütün qiymətlərdə doğruluğunu göstərir. Bu əmr sonda olduğu üçün **break** əmrinə ehtiyac yoxdur. Bu əmrə ehtiyac yoxdursa, onda yazmamaq olar.

```php
<?php

$i = 1;

switch ($i) {
```

```
case0: // "break" qəsdən yazılmayıb!

case1:
echo'Sıfır və ya bir';
break;

case2:
echo'İki';
break;

case3:
echo'Üç';
break;

}

?>
```

Dövrlər

Dövr nə üçündür. Bəzən hansısa bir misalı yoxlayanda şərtə uyğun olaraq dəfələrlə təkrarlanmasını və uyğun qiymət aldıqda bizə göstərməsi üçün istifadə olunur. Aşağıda sadə misallara baxaq.

```
<?php

for ($i=1; $i<=5; $i++)
  {
    if ($i==4) echo $i. 'tapdiq<br/>';
    else echo $i. 'tapilmadi<br/>';
  }

?>
```

Bu şərtimizdə belədir.
$i=1 - bərabərdir, $i<=5 kiçik bərabərdir və hərdəfə $i++ (yəni +1) dəfə artır.

əgər if ($i==4) bərabər olduqda ödənilsin, (tapdıq) sözü çıxsın. əgər ödənilməsə tapılmadı çıxsın.

Nəticə:
1 tapilmadi
2 tapilmadi
3 tapilmadi
4 tapdiq
5 tapilmadi

---

while dövrü operatoru da eyni ilə for kimidir. Aşağıdakı misalda string dəyişənində \"a\" hərfinin neçənci yerdə dayandığını tapmaq üçün dövrdən istifadə edirik. 0-dan başlayıb uyğun qiyməti tapana kimi hər dəfə bir vahid artaraq dövr təkrarlanır.

```
<?php

$i=0;

   $string='qwerrtuyua';
   $str='';

while ($str != 'a') {

   $str=substr($string,$i,1);
   $i++;
}

echo 'a herfi '.$i.'-ci dir<br/>';

?>
```

Nəticə: a herfi 10-ci dir

Əgər verilənlər tam məlum olmasa o zaman aşğıdakı kimi də yaza bilərik:

```
<?php

$i=0;

   $string='qwerrtuyua';
   $str='';

do
{

   $str=substr($string,$i,1);
   $i++;
}

while ($str != 'a');

echo 'a herfi '.$i.'-ci dir<br/>';

?>
```

İstənilən proqramlaşdırma dilində bu və digər şəkildə dövrlərdən istifadə olunur. Bu baxımdan PHP dilində 3 cür dövr vardır:

while
Bu əmrin yazılış qaydası aşağıdakı kimidir:

```
<?php

$i = 1;
```

```
while($i<10) {
echo$i . "<br>\n";
$i++;
}

?>
```

Dövr **while** əmrindən sonra gələn şərti yoxlayır və ondan asılı olaraq, növbəti ifadə və ya fiqurlu mötərizənin içərisindəki ifadələr yerinə yetirilir.

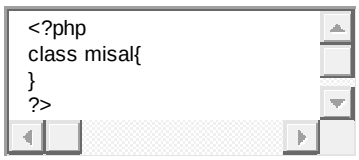
do..while

```
<?php

$i = 1;

do {
echo$i . "<br>\n";

} while ($i++ <10);

?>
```

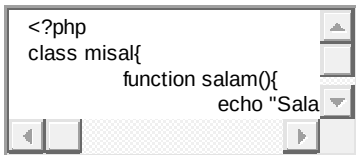
Kodlamalarda class-lardan istifadə etsək yazdığımız
kodlar çox aydın görünəcək hemde php fayllarının ağırlığı ortadan qalxacaq.
Class yeni yaranan OOP-nin təməlidir.
class öyrənərək OOP-yə keçmiş olacaqsınız.
Sual: Bəs bu OOP nədir?
Cavab: OOP(object oriented programing). Yəni açıqlamasındada deyildiyi kimi, Əşya Tərəfli Proqramlama deməkdir.
OOP haqqında irəlidə ətraflı yazı yazacam.

Class-lar

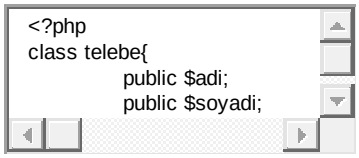
İndi isə keçək Class-ların işlədilməsinə.



Burada misal adlı bir class yaratdıq. ancaq class-ın içi boşdur deyə heş bir işə yaramır. indi isə class-ın içini dolduraq və işə salaq.



Bu sadə örnəktə kodları işlətsək ekranda Salam Dünya yazılacaqdır. Gördüyünüz kimi class-ın işlədilməsi çox asantdır həmdə görnüşü gözəldir.
Detallı bir örnək daha.



Ekranda:
Adı: Ramiz
Soyadı: Qasımov
Yaşı:20
yazılacaqdır.

Açıqlama:

Yuxarıda yazdığımız "public $adi" funksiyaların arasında işlədilən global dəyişən kimidir.
Ancaq bir ferq var ki, oda class-larda public bir dəfə yazılır, funksiyalarda isə hər dəfə yazılır.
Dəyişəni public ederek bütün class-ın her yerində istifadə edə bilirsən.
public dən savayı başqalarıda var. (olnar barədə digə raddımlarda danışılacaq.
"$this->adi" gördüyünüz kimi bu dəyişən adi adlı dəyişəni içinə almışdır.
$this-> də bir global dəyişən kimidir.
Class-ın hər yerində $this->adi olaraq dəyişən tanınmalıdır.
$mekteb=new telebe();
bu kodla $mekteb adlı dəyişənə dedik ki, yeni yaranmış telebe adlı class-ı öz öhdənə götür.
Və mən bunu bu səhifədə istifadə edəcəm.
$mekteb->adini_soyle();

$this də ki, kimi $mekteb dəyişənini səhifənin hər yerində istifadə edə bilirik.
Bütün CMS Sistemlərindədə PHP Class-lardan istifadə olunur.
Misal("WordPress,Joomla,DLE,Drupal və,s").
Sizdə öz yaratdığınız sistemdə class-lardan istifadə edərək həm kodlarınızı azaltmış olacaqsınız həmdə sisteminiz bir o qədər sürətlənəcək.
Unutmayın ən yaxşı proqramist az kodla çox iş görəndir.