

# **PHP Web Programlaşdırma dili**

**2-ci buraxılış**

**Şükür Hüseynov**

## Mündəricat

Müəllif haqqında.....	3
Proqramlaşdırma nədir?.....	4
Web proqramlaşdırma nədir?.....	5
Php üçün proqramların quraşdırılması.....	7
Php dilinə giriş.....	10
Php dilində şərhlər.....	11
Hesablamalar.....	12
Dəyişənlər.....	18
Şərt operatorları.....	27
Dövlər. While operatoru.....	28
Dövlər. For operatoru.....	31
Dövlər. Do while operatoru.....	33
Çoxluqlar.....	34
Funksiyalar.....	51
Isset(), Empty() və Unset() funksiyaları.....	56
Form daxili əməliyyatlar.....	57
Filter funksiyaları.....	60
Cookies.....	65
Sessiyalar.....	66
Zaman funksiyaları.....	68
String funksiyaları.....	71
Məlumatların şifrələnməsi.....	79
Php dilində \$_SERVER massivi.....	81
Fayl funksiyaları.....	82
Php fayl upload əməliyyatı.....	87
Php ilə şəkil hazırlamaq.....	89
Mail funksiyaları.....	91
HTTP funksiyaları.....	93
Include və require funksiyaları.....	95
Phpmyadmin və MySql.....	97
Php və MySql arasında əlaqə.....	103
Qeydiyyat formu.....	104
Məlumatların bazadan oxunması.....	106
Məlumatın form vasitəsilə silinməsi.....	107
Məlumata form vasitəsilə düzəliş edilməsi.....	109
Obyektyönlü proqramlaşdırma.....	110
Php və Ajax.....	118
Kitabın sonu.....	122

## **Müəllif haqqında**

Şükür Hüseynov, 1996-cı ildə Bakı şəhərində dünyaya gəlmişdir. 2014-cü ildə, Bakı şəhərində yerləşən 55 saylı tam orta məktəbi bitirmişdir və həmin il Azərbaycan Dövlət Neft və Sənaye Universitetinin Elektronika, Telekommunikasiya və Radiotexnika mühəndisliyi ixtisasına daxil olmuşdur. 2018-ci ildə universiteti başa vurmuşdur. Müəllif, 14-15 yaşlarından etibarən proqramlaşdırma ilə məşğuldur. İndiyə qədər Veb və stolüstü proqramlaşdırma, həvəskar səviyyədə isə mobil, oyun və qurğu proqramlaşdırması ilə məşğul olmuşdur. Daha çox stolüstü, oyun və sistem proqramlaşdırmasına maraq duymaqladır.

Müəllif, Proqramlaşdırmadan əlavə Psixologiya, Poeziya və Musiqi ilə də məşğul olmaqladır. Bu kitabdan əlavə elektron variantda yayımlanan həvəskar Php kitabının və bir şeir kitabının müəllifidir.

Əlavə olaraq qeyd edək ki, “Şükür Hüseynovla Proqramlaşdırma” adlı Youtube kanalımızda proqramlaşdırmaya aid videolar yayımlanmaqdadır. Videolar arasında kitabdakı bəzi mözular da yer almaqladır.

Əlaqə: [Shukur.Huseynov.1996@mail.ru](mailto:Shukur.Huseynov.1996@mail.ru)

## **Proqramlaşdırma nədir?**

Proqramlaşdırma 20-ci əsrdə həyatımıza sanki bir möcüzə kimi daxil olmuşdur. Bu möcüzənin nəticəsində bugün biz komputerlə saysız-hesabsız əməliyyatlarımızı yerinə yetirə, şəkillər, sənədlər, 3 ölçülü modellər hazırlayara, oyunlar oynayaraq vaxtımızı əyləncəli şəkildə keçirə bilirik. 20-ci əsrdə hesablama üçün yaradılmış qurğular vasitəsilə bugün oyunların oynanılmasını, saysız-hesabsız əməliyyatların yerinə yetirilməsini yalnız möcüzə adlandırmaq olar.

İlk olaraq 20-ci əsrdə mikroprosessorlar yaradılmışdır və bu prosessorlarla müxtəlif əməliyyatlar yerinə yetirilirdi. Zaman keçdikcə bu prosessorlar çox inkişaf etdirilmişdir və bugünkü səviyyəyə çatmışdır. Mikroprosessorların ilk çıxdığı vaxtlarda təəssüf ki, C, Php kimi dillər yox idi. O zaman ilk çıxan Assembler dili var idi və bu dillə də proqramları yazmaq çox çətin idi. Marağ üçün deyim ki, Github üzərində Nasanın 1969-cu ildə aya göndərdiyi Apollo 11-in elektronika hissəsinin Assemblerdə yazılmış kodları mövcuddur. Təbii ki, o kodlara baxdıqda qorxmamaq mümkün olmur :)

İlk başlarda Mikroprosessorları proqramlaşdırmaq çox çətin olsa da, zaman keçdikcə inkişaf etdirilmişdir və bu gün artıq C++, Java, C#, Delphi kimi dillərə nisbətən daha asan proqramlar hazırlamağımız mümkündür. Zamanla komputer qrafikası da çox güclənmişdir və bugün oyunlarda buna şahid ola bilirik.

## Web proqramlaşdırma nədir?

C++, Java kimi dillərlə öz komputerimizdə sadə və ya mürəkkəb olmaqla bir çox proqram təminatını yarada bilirik. Məsələn Photoshop, Ms Office tipli proqramlarda məlumatlar sadəcə öz komputerimizdə yerləşir. Bu proqramlarda əməliyyatları aparırıq və faylları öz komputerimizdə yadda saxlayırıq. Ancaq bəzi proqramlar var ki, bu proqramlarda bəzi məlumatlar öz komputerimizdə saxlanılsa da, bəzi məlumatlar başqa bir komputerdə, yəni, serverdə saxlanılır və biz ora qoşulduqda məlumatlardan serverdən komputerimizə göndərir. Server də adi bir komputerdür, sadəcə, adi komputerə nisbətən daha güclü və aylarla söndürülmədən işlədilən komputerdür.

Məlumatların serverdə saxlanılmasının əsas səbəblərindən biri məlumatların daima dəyişdirilməsi ola bilər. Beləliklə, məlumatlar dəyişdirildikdə, istifadəçinin proqramı məlumatları serverdən götürdüyü üçün istifadəçinin proqramında da dəyişən məlumatlar görünəcək. Başqa bir səbəb də çoxsaylı istifadəçinin öz aralarında məlumat mübadiləsi apara bilməsidir. Məsələn, məşhur Whatsapp proqramını nəzərdən keçirək. Biz bu proqramla kiməsə mesaj yazdıqda bu mesajımız birbaşa onun telefonuna getmir. Mesaj əvvəlcə serverə gedir, sonra isə serverdən qarşı tərəfin telefonuna göndərilir. Gördüyümüz kimi, bu tip tətbiqlərdə serverin istifadəsi çox vacibdir.

Başqa bir nümunə olaraq Point Blank oyununu göstərə bilirik. Oyunda yüzlərlə, minlərlə istifadəçi mövcud olur və onları olaraq serverə məlumat ötürürlər və başqa məlumatları qəbul edirlər. Bunun nəticəsində oyunda oyunçular yerlərini dəyişə və bir çox əməliyyatları apara bilirlər.

Göstərdiyimiz Whatsapp və Point Blank nümunələrində proqramlar ilk öncə mütləq komputerə və ya telefona yüklənməlidirlər. Ancaq belə bir şey də mövcuddur ki, proqram telefona yüklənməsin, istifadəçilər bir proqram vasitəsilə serverə qoşulub müəyyən əməliyyatları aparsınlar. Serverə qoşulacağımız proqram bizim brauzerimizdir.

Deməli ilk öncə serverə bir fayl yerləşdirilir. Sonra brauzerimiz o serverə qoşulur. Biz saytın adın yazanda, məsələn, Google.az yazdıqda brauzerimiz google-un öz serverinə qoşulur və oradan lazımı faylları bizə gətirir. Brauzerimiz o faylda olan Html, Css, və Javascript kodlarına əsasən bir görünüş yaradır ki, bu görünüşdə faylı serverə yerləşdirən proqramçının nəzərdə tutduğu görünüşdür və o, bu görünüşü Html, Css və Javascript vasitəsilə yaratmışdır. Burada bilməli olduğumuz əsas hissə, Html, Css və Javascript kodlarının serverdən brauzerimizə göndərildikdən sonra brauzerimizdə görünüşə çevrilməsidir. Bəzi brauzerlərdə bəzi saytların fərqli görünməsinin əsas səbəbi budur. Məsələn, köhnə brauzerlərlə bəzi saytları yoxlasanız, onlar brauzerdə çox bərbad bir vəziyyətdə açılacaq. Çünki, son illər yaradılmış Html, Css və Javascript xüsusiyyətlərin yeni brauzerlər dəstəkləməkdədir. Bu hissəni Front end development adlandırırlar və Front end developerdən əsas tələb olunan şeylər Html, Css və Javascript olur.

Front end developmentdən sonra əsas bir hissə Back end developmentdir. Veb səhifənin Back End tərəfi isə Php, Asp.net, Jsp kimi dillərlə yazılmaqdadır. Burada

əsas bir şeyə diqqət yetirək ki, Back end tərəfinin kodları server tərəfdə işlədilir, html səhifəsinin içinə yerləşdirilir və istifadəçiyə göndərilir. Yəni, istifadəçiyə php kodlarının özü yox, kod işlədikdən sonra alınan nəticələr göndərilir. Ona görə də bu hissə Back end adlanır. Eyni zamanda, istifadəçi front end hissənin kodlarına, yəni, veb səhifənin Html, Css və Javascript kodlarına baxa, müdaxilə edə bilər, ancaq, back end kodlarının ancaq nəticəsini əldə etdiyinə görə o kodlara baxa bilmir.

Biz bu kitabda Php vasitəsilə back end proqramlaşdırmanı öyrənəcəyik. Back end dillərində mənfi bir cəhət odur ki, hər-hansı bir əməliyyat aparıldıqda və ya başqa səhifəyə keçid etdikdə səhifənin yenilənməsidir. Məsələn, bir sayta girib oradakı bir xəbərə girsək səhifə yeniləndikdən sonra yeni məlumatları görəəcəyik. Məsələn, oyunlarda bir çox şey canlı olmalı olduğu üçün oyun proqramlaşdırmasında bu tip şeylər problem yarada bilər. Ancaq, Javascript və php dilinin qarşılıqlı işlədilməsi nəticəsində bu problem aradan qaldırılır, məlumatlar səhifə yenilənmədən səhifədə dəyişə bilər.

Nə qədər tam canlılıq olmasa da, back end tərəfi üstünlük təşkil edən oyunlara misal olaraq combats.com oyununu göstərə bilərik. Oyunda bir çox hərəkətlər səhifə yenilənərək baş versə də, oyun strateji olduğu üçün böyük oyunçu kütləsinə malik olmuşdur və bir zamanların ən məşhur oyunlarından olmuşdur. Günümüzdə belə bu oyun oynanmaqdadır. Azərbaycan istehsalı olan Meydan.az oyununu da bu tip oyunlara misal göstərmək olar.

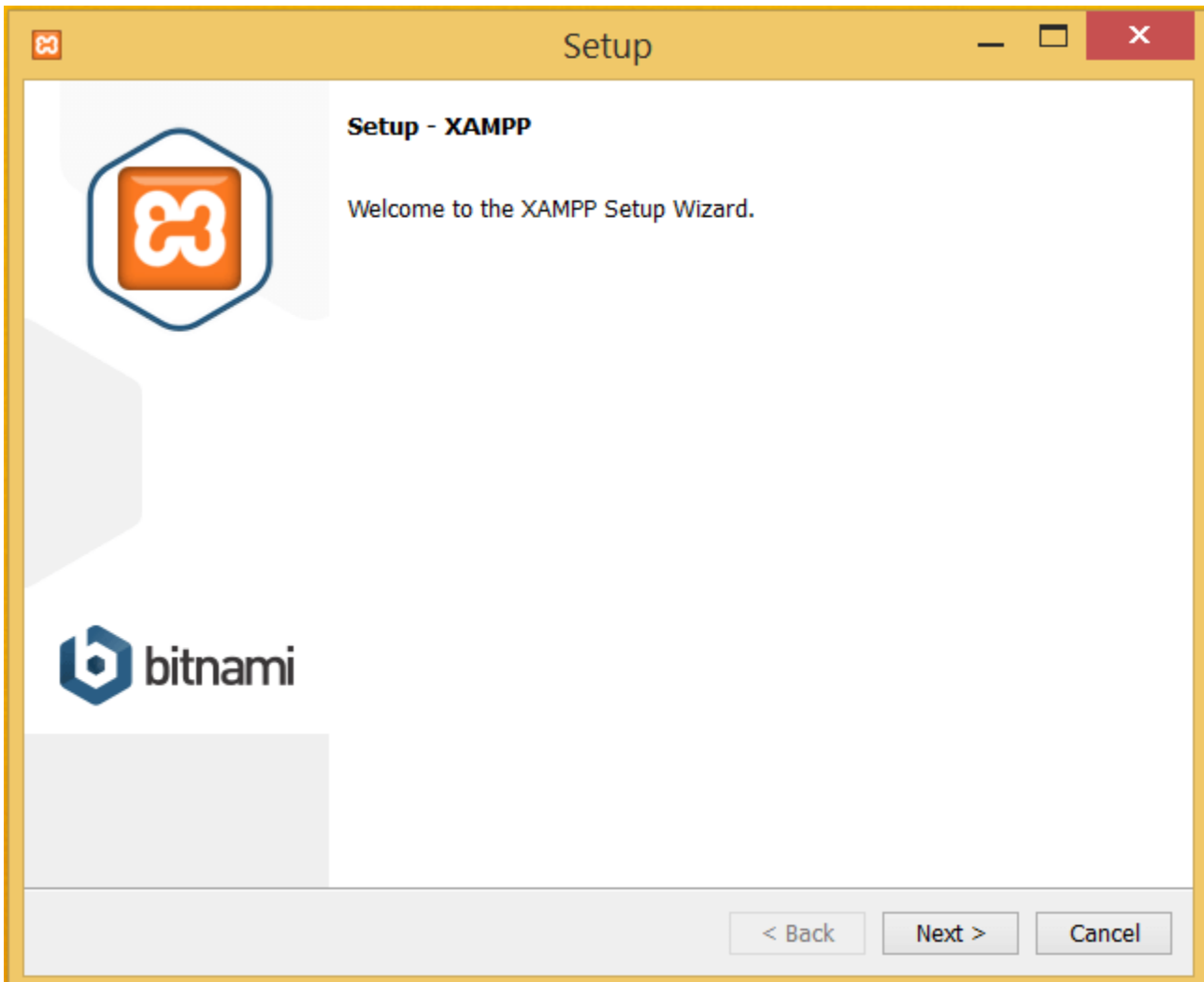
Artıq proqramlaşdırma və veb proqramlaşdırmanı müəyyən qədər izah etdikdən sonra php dilini öyrənməyə başlaya bilərik. Qeyd edim ki, php öyrənmədən öncə oxucunun html dilini biraz da bilməsi lazımdır. Çünki, dediyimiz kimi php kodlarının nəticəsi Html kodlarının arasına yerləşdirilərək istifadəçiyə göndərilir. Buna görə də, Css və Javascript olmasa belə, Php dilinə başlamazdan öncə oxucunun Html dilini bir qədər bilməsi çox yaxşı olar.

Başlıqakı sualımıza gəldikdə, bütün bunlardan sonra deyə bilərik ki, web proqramlaşdırma dedikdə, proqramın veb səhifə üzərində proqramlaşdırılması nəzərdə tutulur.

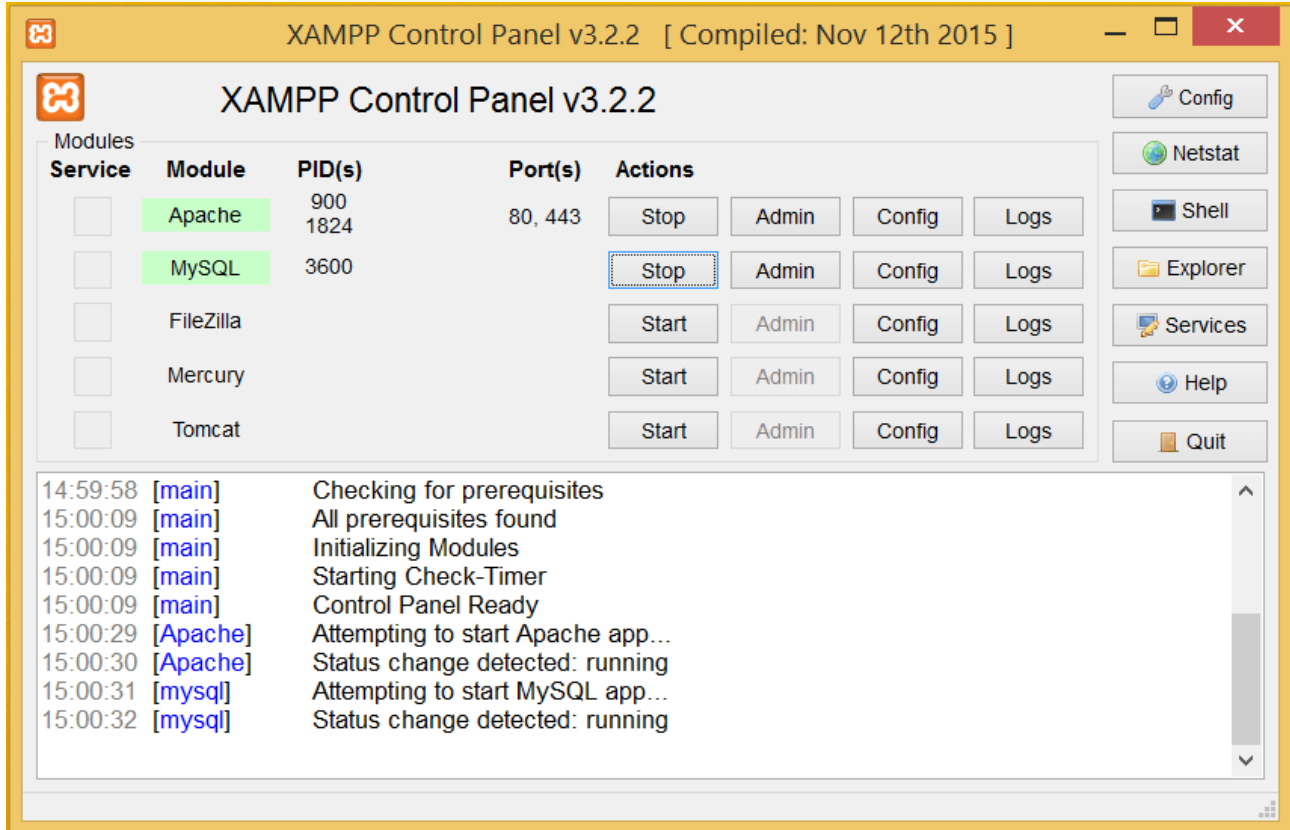
## Php üçün proqramların quraşdırılması

Php kodlarının server tərəfdə işləməsi üçün bir proqrama ehtiyac duyulur. Bu proqrama misal olaraq Xampp server, Wamp server, Lamp server və s. Proqramları göstərmək olar. Biz php öyrənməyə başlamazdan öncə bu proqramlardan birini kompüterimizə quraşdırmalıyıq. Əlavə olaraq deyək ki, proqram quraşdırmadan online kompilyatorlar vasitəsilə də php kodlarımızı yazıb yoxlaya bilərik. Ancaq, real işlər üçün kompilyatorun quraşdırılması məsləhət görülür. Mən ilk zamanlar kompüterimdə kompilyatoru quraşdırma bilmədiyim üçün kodları vərəqdə yazırdım. Sonra isə pulsuz hosting olan [1freehosting.com](http://1freehosting.com) saytıda yazmışdım. Bunu deməyimin əsas səbəbi odur ki, əgər kompilyatorun quraşdırılmasında problem çıxarsa narahat olmağa dəyməz :) Kodları hosting üzərində və ya onlayn kompilyatorlardan birində yazıb yoxlaya bilərsiniz. Google üzərində Php online compiler yazıb axtarsanız bir çox onlayn kompilyatorlar çıxacaq. Həmin kompilyatorlar vasitəsilə kodlarınızı yazma və işlədə bilərsiniz.

İndi isə gəlin Xampp serveri kompüterimizdə quraşdıraq. Quraşdırma qaydası asandır, ilk olaraq [apachefriends.org](http://apachefriends.org) saytıdan Xampp serveri yükləyək. Yüklədikdən sonra yüklədiyimiz proqrama daxil olaq.



Burada standart olaraq Next, Next edərək proqramı quraşdırırıq. Proqramı quraşdırdıqdan sonra əgər ekranımızda proqramın adı görünürsə daxil oluruq. Adı görünməsə belə komputerin axtarış hissəsində xampp yazaraq proqramı tapırıq və daxil oluruq. Nəticədə aşağıdakı pəncərə açılır.



Buradan Apache və Mysql sözlərinin qarşısındakı start düymələrinə basırıq. Bu düymələrə basdıqdan sonra Apache və Mysql yazıları yaşıl rəngdə oldusa, deməli artıq komputerimizdə php kodlarını işlədə bilərik.

Bundan sonra brauzerimizə daxil olub adres yerinə localhost yazıb enter düyməsinə klik edirik. Nəticədə Xampp serverin qarşılama səhifəsi açılacaq. Bundan sonra C diskinə, oradan xampp qovluğuna, oradan isə htdocs qovluğuna daxil oluruq. Əgər xampp qovluğu C diskində olmazsa, C diskində program files qovluğunda xampp qovluğunun olub-olmadığını yoxlaya bilərsiniz. Sonda htdocs qovluğuna daxil olduqdan sonra qarşımıza index.php adlı bir fayl çıxacaq. Brauzer üzərində localhost yazıb daxil olduqda bizə göstərilən səhifə bu index.php səhifəsidir. Burada faylın uzantısının html yox, php olmasına diqqət edin. Həmin index.php faylını Notepad vasitəsilə açmalıyıq. Faylı Notepad vasitəsilə açmaq üçün Notepada daxil olub File menyusundakı open seçimi vasitəsilə bu qovluğa gəlib faylı açmağa bilərik. Başqa bir üsul də, faylın üzərində sağ düyməni klik etdikdən sonra "Open With" seçimindən faylı açmağımızdır. Faylda girdikdən sonra fayldakı bütün kodları silək və aşağıdakı kodu fayla yazmaq.



```
<?php  
echo "Hello World!";  
?>
```

Bundan sonra faylı yadda saxlayaq və yenidən brauzerdə localhost yazaraq daxil olaq. Nəticədə "Hello World!" yazısı ekrana çıxacaq. Beləliklə, ilk kodumuzu işlətməmiş olduq.

Kodu yazarkən daha rahat olması üçün, komputerə Notepad++, Sublime Text kimi kod redaktorlarından birini də yükləyə bilərsiniz.

## Php dilinə giriş

Lazım olan proqramı komputerimizə quraşdıqdan sonra php-i öyrənə bilərik. Php kodları rahat bir şəkildə Html kodlarımızın arasına yerləşdirilir. Yəni, html kodlarımızı yazıb arasına php kodlarımızı yazmağa bilərik. Php kodlarımız <?php və ?> arasında yazılır. Php daxilində öyrənəcəyimiz ilk funksiya **echo** funksiyasıdır. Bu funksiya yazıları ekranda çap etmək üçündür. Funksiyamızı, yəni echo funksiyasının adını yazdıqdan sonra məsafə qoyuruq, dırnaq açırıq və yazmaq istədiyimiz yazını yazırıq. Sonda isə nöqtə vergül qoyuruq. Bir çox proqramlaşdırma dilində olduğu kimi php dilində də sətirlərin sonunda nöqtə vergül qoyulur.

```
<?php  
echo "Php dilində ilk yazı";  
?>
```

Nəticədə "Php dilində ilk yazı" sözü ekrana çap olunmuş olur. Azərbaycan şrifflərinin səhifədə işləyə bilməsi üçün php kodlarından əvvəl səhifədə <meta charset="utf-8"> hissəsini yazmağı unutmamalıyıq. Ədədləri çap edərkən dırnaq işarəsi qoymağımıza ehtiyac olmur. Məsələn:

```
<?php  
echo 5;  
?>
```

Bu şəkildə ədədimizi çap edirik. Bundan əlavə, echo daxilində html teqlərinin də yazıla bildiyini qeyd edək. Məsələn:

```
<?php  
echo "Php";  
echo "<br><font color='red'>Php</font>";  
?>
```

Burada <font color='red'> hissəsində yazdığımız dırnaq işarəsinin tək olduğuna diqqət edək. Əgər əvvəldə cüt dırnaq işarəsindən istifadə etməliyiksə, yazının içində tək dırnaqdan istifadə etməliyik. Nəticədə php daxilində yazılmış html kodu işləyəcək və sarı rəngdə yazı ekrana çıxacaq. Əslində burada php ilə digər yazılar olduğu kimi html-lə yazılmış kod da çap olunur, sadəcə brauzer onu yenidən görünüşə çevirir.

## Php dilində şərhlər

Bəzi proyektlərdə kodlarımız həddindən artıq qarışıq bir şəkllə düşməkdədir. Kodlarımız o qədər qarışıq bir hala gəlirik ki, ikinci dəfə baxanda o kodu özümüz belə başa düşə bilmirik. Buna görə də, proqramlaşdırmada şərhlərdən istifadə olunur. Şərhlər heç bir əməliyyat yerinə yetirmir, nəzərə alınmır, sadəcə, kodu növbəti dəfə oxuduqda rahat, anlaşılıqlı olması üçün yazılır. Tək sətirlik şərhləri yazmaq üçün sadəcə sətirdən sonra // işarələri qoyuruq və şərtimizi yazırıq. Nümunəmizə baxaq:

```
<?php  
echo "Şərhlər"; // Echo funksiyası yazını çap etmək üçündür.  
?>
```

Bu şəkildə şərhimizi yazmış oluruq.

Çoxsətirli şərhləri isə aşağıdakı şəkildə yazmağa bilərik:

```
<?php  
echo "Şərhlər";  
/*  
Şərhlər buraya yazılır  
*/  
?>
```

Beləliklə çoxsətirli şərhləri yazmış oluruq.

## Hesablamalar

Demək olar ki, bütün proqramlaşdırma dillərində olduğu kimi php daxilində də toplama, çıxma, vurma və bölmə kimi ilkin hesablamaların aparılması mümkündür. Aşağıdakı nümunəyə baxaq:

```
<?php
echo 5+7; // Toplama
echo "<br>";
echo 5*7; // Vurma
echo "<br>";
echo 5-7; // Çıxma
echo "<br>";
echo 5/7; // Bölmə
?>
```

Nəticədə əməliyyatlar hesablanaraq alt-alta yazılır. Gördüyümüz kimi, + (toplama), - (çıxma), \* (vurma) və / (bölmə) işarələri ilə lazımi əməliyyatları yerinə yetiririk. Diqqət edilməli bir əsas məqam da odur ki, yazılan ifadə hesablama olduğu üçün dırnaq işarəsi içində yazılmır. Əgər dırnaq işarəsi içində yazılsa, hesablamının nəticəsi yox, sadəcə söz kimi "5+7" yazılacaq. Hesablama bir neçə mərhələdən ibarətdirsə, ilk öncə riyaziyyatda olduğu kimi vurma və bölmə, sonra isə toplama və çıxma yerinə yetirilir. Məsələn:

```
<?php
echo 5+8/4;
?>
```

Burada nəticə olaraq, 7 çıxacaq əgər biz ilk öncə toplamanın yerinə yetirilməyin istəyiriksə 5+8 hissəsini mötərizə daxilində yazmalıyıq. Məsələn:

```
<?php
echo (5+8)/4;
?>
```

Burada ilk öncə 5 və 8 toplanacaq, sonra isə 4-ə bölünəcək. Əsas 4 hesablama olduğu kimi, digər mürəkkəb hesablamaları da yerinə yetirmək olar. Məsələn sqrt funksiyasını işlədərək ədədin kök altısını tapa bilərik. Məsələn:

```
<?php
echo sqrt(9);
?>
```

Burada ekrana 3 çıxacaq. Kök altından əlavə php dilində bir çox riyazi funksiyalar var. Bu funksiyalar aşağıda göstərilir. Bu funksiyaların bəzilərinə aşağıda göstərək. Ancaq qeyd edək ki, bu funksiyaları əzbərləməyə ehtiyac yoxdur. Sadəcə belə funksiyaların olduğunu bilməyimiz kifayətdir. Lazım olduqda google üzərindən bu funksiyaları axtararaq tapa bilərik. Axtarış sistemlərinin proqramçının ən böyük köməkçilərindən olduğunu qeyd etmək yerinə düşər. Funksiyalar hədsiz çox olduqda onlar istəməz yaddan çıxıb bilər. Bu zaman onu google üzərində axtararaq tapa bilərik. Ən yaxşı nəticələr ingiliscə axtarıqda tapılmaqdadır.

1) abs funksiyası - Bu funksiya ədədin modulunu tapır. İstifadə qaydası aşağıdakı kimidir:

```
<?php  
echo abs(-5); // Nəticədə ekrana 5 çıxacaq,  
?>
```

2) acos funksiyası - Bu funksiya riyaziyyatdakı arccos funksiyasıdır. İşlənmə qaydası aşağıdakı şəkildədir:

```
<?php  
echo acos(0.5);  
?>
```

Proqramlaşdırma dillərində triqonometrik funksiyalarda vahid olaraq radiandan istifadə olunur. Yuxarıdakı funksiyanın nəticəsi də radianla veriləcək. Nəticəni dərəcəyə çevirmək üçün lazım olan funksiya ilə irəlidə tanış olacağıq.

3) acosh funksiyası - Hiperbolik kosinus funksiyasıdır. İşlənmə qaydası digər riyazi funksiyalarla eynidir.

4) asin funksiyası - Arcsin funksiyasıdır. Nəticə radianla verilir. Məsələn:

```
<?php  
echo asin(0.5);  
?>
```

5) asinh funksiyası - Hiperbolik sinus funksiyasıdır. İşlənmə qaydası digər riyazi funksiyalarla eynidir.

6) atan funksiyası - Arctg funksiyasıdır. Daxiletmə radianlardır. Məsələn:

```
<?php  
echo atan(21);  
?>
```

7) atanh funksiyası - Hiperbolik tg funksiyasıdır. İşlənmə qaydası digər riyazi funksiyalarla eynidir.

8) Base\_convert funksiyası-Bu funksiya bir say sistemində olan ədədi digərinə çevirir. Məsələn:

```
<?php
$hexadecimal = 'A37334';
echo Base_convert($hexadecimal, 16, 2);
?>
```

Burada 16-lıq say sistemindəki ədəd 2-lik say sistemindəki ədədə çevrilir.

9) ceil funksiyası - Bu funksiya ədədi yuxarı yuvarlaqlaşdırır. Məsələn:

```
<?php
echo ceil(4.3);
echo ceil(2.7);
?>
```

Nəticədə uyğun olaraq 5 və 3 çıxacaq.

10) cos funksiyasıdır. - Kosinus funksiyasıdır. Nəticə radianlardır. Məsələn:

```
<?php
echo cos(30); // 0.15425144988758
?>
```

11) cosh funksiyası - Hiperbolik kosinus funksiyasıdır. İşlənmə qaydası digər riyazi funksiyalarla eynidir.

12) Deg2rad funksiyası - Bu funksiya dərəcəni radiana çevirir. Məsələn:

```
<?php
echo deg2rad(45); // 0.785398163397
?>
```

13) exp funksiyası - Bu funksiya e üstü hər hansı bir ədədi tapmaq üçündür. Məsələn:

```
<?php
echo exp(2); // Nəticədə 7.3890560989307 çıxacaq
?>
```

14) floor funksiyası- Bu funksiya ədədi aşağı yuvarlaqlaşdırır. Məsələn:

```
<?php
echo floor(4.8); // Nəticədə ekrana 4 çıxacaq.
```

?>

15) log10 funksiyası- Bu funksiya 10 əsasdan loqarifmanı tapmaq üçündür. Məsələn:

```
<?php
echo log10(1000); // Nəticədə 3 çıxacaq
?>
```

16) Log funksiyası-Natural loqarifma funksiyasıdır. Məsələn:

```
<?php
echo Log(10); // Nəticədə 2.302585092994 çıxacaq.
?>
```

17) max funksiyası - Daxil edilmiş ədədlərdən ən böyüyünü tapmaq üçündür. Məsələn:

```
<?php
echo max(4,7); // Nəticədə 7 çıxacaq
echo max(7,4,8,9,1); // Nəticədə 9 çıxacaq
?>
```

Bu funksiyada ədədlərlə sözlər də müqayisə oluna bilər. Məsələn:

```
<?php
echo max('word',-3);
?>
```

Burada -3 rəqəmi 0-dan kiçik olduğu üçün maksimum olaraq word sözü olacaq.

18) Min funksiyası - Verilmiş ədədlərdən ən kiçiyini tapmaq üçün istifadə olunur. İstifadə qaydası max funksiyası ilə eynidir.

19) pi funksiyası - Pi ədədinin qiymətini verir. Məsələn:

```
<?php
echo pi();
?>
```

Bu funksiyanın yerinə M\_PI işlədilə bilər. Məsələn:

```
<?php
echo M_PI; // Nəticədə 3.14159265358979323846 alınır
echo M_PI_2; // Nəticədə Pi/2 alınır
echo M_PI_4; // Nəticədə Pi/4 alınır
```

```
echo M_1_Pi;// Nəticədə 1/Pi alınır  
echo M_2_Pi; // Nəticədə 2/Pi alınır  
?>
```

20) pow funksiyası - Üstlü funksiyaları hesablamaq üçündür. Məsələn:

```
<?php  
echo pow(2,8); // Nəticədə 256 alınacaq  
echo pow (-1,10); // Nəticədə 1 alınacaq  
?>
```

21) rad2deg funksiyası - Radianı dərəcəyə çevirmək üçün istifadə olunur. Məsələn:

```
<?php  
echo rad2deg(1); // Nəticədə 57.295779513082 alınacaq  
?>
```

22) rand funksiyası-Təsadüfi ədəd istehsal etmək üçün istehsal olunur. Məsələn:

```
<?php  
echo rand();  
?>
```

Nəticədə təsadüfi bir ədəd çap olunacaq. Bu ədəd müəyyən aralıqdan da seçilə bilər. Məsələn:

```
<?php  
echo rand(5,15);  
?>
```

Nəticədə 5 və 15 aralığından təsadüfi bir ədəd seçiləcək.

23) round funksiyası-Bu funksiya ədədi yuvarlaqlaşdırmaq üçün istifadə olunur. Məsələn:

```
<?php  
echo round(5.7); //Nəticədə 6 alınacaq  
echo round(5.3); //Nəticədə 5 alınacaq  
?>
```

24) sin funksiyası-Sinus funksiyasıdır. Daxiletmə radianladır. Məsələn:

```
<?php  
echo sin(2); // Nəticədə 0.90929742682568 alınacaq
```



?>

Bir neçə funksiyanı bir yerdə işlətmək mümkündür. Yuxarıda göstərilmiş deg2rad funksiyası ilə sin funksiyasını bir yerdə işlədərək sinusu dərəcə ilə hesablaya bilərik. Məsələn:

```
<?php  
echo sin(deg2rad(30)); Nəticədə 0.5 alınacaq  
?>
```

Burada ilk öncə içəridəki deg2rad funksiyası öz işini görür və 30 dərəcəni radiana çevirir. Sinus isə alınmış radianla qiyməti hesablayır.

25) sinh funksiyası- Hiperbolik sinus funksiyasıdır. İstifadə qaydası digər riyazi funksiyalarla eynidir.

26) tan funksiyası - Tg funksiyasıdır. Daxiletmə radianlardır. Məsələn:

```
<?php  
echo tan(deg2rad(45)); // Nəticədə 1 alınacaq  
?>
```

27) tanh funksiyası - Hiperbolik tg funksiyasıdır. İstifadə qaydası digər riyazi funksiyalarla eynidir.

## Dəyişənlər

Dəyişən anlayışına ilk olaraq elə təbiətdə rast gəlməkdəyik. Məsələn, otaqdakı temperatur bir dəyişəndir. Əvvəl 20 dərəcə olur, 1 saat sonra 21 dərəcə olur, 3 ay keçir 5 dərəcə olur. Yəni zaman keçdikcə dəyişən dediyimiz obyekt dəyişməkdədir. Və ya insanın yaşını götürə bilərik, insanın yaşı da bir dəyişəndir, hər il artmaqdadır. Gördüyümüz kimi, dəyişən zaman keçdikcə müxtəlif şeylərdən asılı olaraq dəyişməkdədir. Riyaziyyatdan da x, y dəyişənlərini xatırlaya bilərik.

Proqramlaşdırmada da dəyişənlər bir çox şey üçün istifadə olunmaqdadır. Məsələn, oyunçunun canı dəyişəndir, əvvəl 100 olur, sonra nəsə olanda azalıb 60 olur, 0-a düşdükdə ölür. Veb üzərində göstərək, məsələn istifadəçinin saytda qalma vaxtı, əvvəlcə 10 saniyə olur, sonra 1 dəqiqə olur və s. Saytda xəbərin oxunma sayı da bir dəyişəndir, əvvəlcə 0 olur, sonra hər dəfə oxunduqca artır. Biz də dəyişənlərdən müxtəlif məqsədlərlə istifadə edəcəyik. Ancaq, ilk öncə php dilində dəyişənlərin istifadə olunmasına baxaq. Php dilində dəyişənləri ifadə etmək üçün \$ işarəsi qoyulur, sonra dəyişənin adı yazılır, sonra da bərabərlik qoyularaq dəyişənin aldığı dəyər yazılır. Aşağıdakı nümunəyə baxaq:

```
<?php  
$a=5;  
?>
```

Burada a dəyişənin adı, 5 isə a dəyişənin aldığı qiymətdir.

Dəyişənin adı üçün bir sıra qaydalar mövcuddur.

1) Dəyişən adları rəqəmlə başlaya bilməz;

2) Dəyişən adları daxilində yalnız hərflər, rəqəmlər və \_ işarəsi ola bilər;

3) Dəyişən adlarında böyük və kiçik hərflər ayrı nəzərdə tutulur. Məsələn \$a və \$A dəyişəni ayrı-ayrı dəyişənlər sayılır. Məsələn:

```
<?php  
$a=5;  
$A=7;  
?>
```

Burada \$a dəyişənin qiyməti 5, \$A dəyişənin qiyməti isə 7 olur. Dəyişənlər echo funksiyası vasitəsilə çap oluna bilər. Məsələn:

```
<?php  
$start=15;  
echo $start;  
?>
```

Burada \$start dəyişəni çap olunaraq ekrana 15 çıxacaq. Ədədlərdən əlavə dəyişənlər yazı vtipində də ola bilərlər. Məsələn:

```
<?php
$start="Yazılar";
echo $start;
?>
```

Dəyişənə yazı ötürüldükdə echo funksiyasında olduğu kimi yazılar dırnaq arasında yazılmaqdadır.

Kod ərzində dəyişənin qiyməti olaraq, onun aldığı ən son qiymət nəzərdə tutulur. Məsələn:

```
<?php
$a=4;
$a=8;
echo $a;
?>
```

Burada \$a dəyişəninin qiyməti 8 sayılır və ekrana 8 çap olunur. Adi ədədlərlə hesablama aparıldığı kimi dəyişənlərlə də aparıla bilər. Məsələn:

```
<?php
$a=8;
$b=3;
$c=$a+$b;
echo $c;
?>
```

Burada \$a dəyişəninə 8, \$b dəyişəninə isə 3 qiymətini veririk. \$c dəyişəninə isə \$a və \$b dəyişənlərini cəmini veririk və \$c dəyişəninə çap edirik. Nəticədə 11 alınır. Dəyişənlərə funksiyaların qiymətlərini də verə bilərik. Məsələn:

```
<?php
$a=sqrt(25);
$b=pow(2,8);
$c=sin(deg2rad(30));
echo $a, $b, $c;
?>
```

Bir neçə dəyişəni bir yerdə çap edərkən yuxarıda olduğu kimi vergüllə ayıraraq tək bir echo funksiyası ilə çap edə bilərik. Nəticədə qiymətlər yan-yanı yazıldığı üçün qarışıqlıq alınacaq. Burada echo ilə yeni bir yazılış şəklinə baxacağıq. Bir neçə ifadəni eyni echo funksiyasında çap edərkən, onlar aralarında nöqtə qoyularaq da yazıla bilər. Məsələn:

```
<?php
$a=sqrt(25);
$b=pow(2,8);
$c=sin(deg2rad(30));
echo $a."<br>".$b."<br>".$c;
?>
```

Göründüyü kimi, ifadələrin aralarında nöqtə qoyaraq bir neçə ifadəni çap etmiş olduq. Yazdığımız <br> etiketləri mətn tipli olduğu üçün, onları dırnaq arasında yazmalı oluruq.

## Şərt operatorları

Şərtlərin rolu həyatımızda çox böyükdür. Biz özümüz də fikir vermədən bir çox şərtəndən istifadə edirik. Məsələn, düz gedəndə baxırıq, əgər qarşımızda divar varsa sağa və ya sola dönürük. Gördüyümüz kimi, burada şərt qoyuruq, əgər divar varsa sağa dönəcəyik. Yoxdursa dönməyəcəyik. Əgər yağış yağarsa çətiri götürək, yağmırsa götürməyək. Burada da şərt qoyuruq, əgər yağış yağarsa çətiri götürürük. Buna aid saysız-hesabsız nümunə göstərə bilərik. Bu şərtlərdən proqramlaşdırmada da istifadə edirik. Php dilində şərt elan etmək üçün **if** operatorundan istifadə olunur. Aşağıdakı koda baxaq:

```
<?php
if(5>4) echo "Ok";
?>
```

Burada if yazırıq, sonra isə mötərizə daxilində şərtimizi yazırıq. Şərtimiz 5 rəqəminin 4-dən böyük olması şərtidir. Bu sadə bir şərtəndir, biz bilirik 5 rəqəmi 4 rəqəmindən böyükdür. Ancaq, proqram bunu yoxlayır və şərt doğru olarsa “Ok” yazısını ekrana yazır. Əgər şərtəndə “5<4” yazsaq o zaman şərt ödənmədiyinə görə heçnə çap olunmayacaq. Çünki, 5 rəqəmi 4 rəqəminə kiçik deyil. Gördüyümüz kimi, şərt daxilindəki əməliyyatlar yalnız şərtin ödəndiyi hallarda yerinə yetirilir.

Əgər şərtin doğru olması halında yerinə yetiriləcək əməliyyat birdən çoxdursa, o zaman, əməliyyatlar { ' və ' } işarələri arasında yazılır. Məsələn:

```
<?php
if(5>4){
echo "Şərt ödəndi";
echo "<br>ikinci";
}
?>
```

İndi isə dəyişənlərdən istifadə edərək if operatoruna aid başqa bir nümunə göstərək:

```
<?php
$a=8;
$b=10;
if($b>$a){
echo "Şərt ödəndi. Əməliyyatlar yerinə yetirilir.";
echo "Şərt doğrudur.";
}
?>
```

Burada a və b dəyişənlərinə qiymət verdik. Sonra isə b dəyişəninin a dəyişənindən böyük olub-olmamasını yoxladıq. Şərt doğru olduğu üçün '{' və '}' işarələri arasındakı əməliyyatlar yerinə yetiriləcək. '}' işarəsindən sonra yazılan əməliyyatlar isə şərt operatorunun əməliyyatlarına daxil olmur. Məsələn:

```
<?php
$a=8;
$b=10;
if($b>$a){
echo "Şərt ödəndi. Əməliyyatlar yerinə yetirilir.";
echo "Şərt doğrudur.";
}
echo "Bu sətir və növbəti sətirlər isə yuxarıdakı şərtə daxil olmadığı üçün yuxarıdakı şərtədən asılı olmayaraq yerinə yetiriləcək.";
?>
```

İndi isə 2 şərt operatorundan istifadə edək:

```
<?php
$a=8;
$b=10;
if($b>$a){
echo "Şərt ödəndi. Əməliyyatlar yerinə yetirilir.";
echo "Şərt doğrudur.";
}
if($b<$a){
echo "Şərt ödənmir.";
}
?>
```

Burada iki dəfə if operatorundan istifadə etdik. İkinci şərtə birincinin əksini göstərdik. Ancaq əksini göstərmək üçün iki dəfə if operatorundan istifadə etməyə ehtiyac yoxdur. Şərt ödənmədikdə digər bütün hallarda olacaq əməliyyatları göstərmək üçün else yazılır. Məsələn:

```
<?php
$a=8;
$b=10;
if($b>$a){
echo "Şərt ödəndi. Əməliyyatlar yerinə yetirilir.";
echo "Şərt doğrudur.";
}
else{
```

```
echo "Burada isə yuxarıdakı şərt ödənməsə qalan bütün hallar üçün olacaq əməliyyatlar yazılır.";
}
?>
```

Şərtin ödənmədiyini bütün digər hallar üçün əməliyyatlar yerinə yetirmək istəyiriksə if ilə bir yerdə else istifadə etməliyik. Yuxarıdakı şərtin ödənmədiyini hallar \$b dəyişəninin \$a dəyişənindən kiçik olması və \$b dəyişəninin \$a dəyişəninə bərabər olması hallarıdır. Ancaq biz böyük olanda bir əməliyyat, bərabər olanda başqa bir əməliyyat, digər hallarda isə yenə başqa bir əməliyyatı yerinə yetirmək istəyiriksə else if istifadə etməliyik. Məsələn:

```
<?php
$a=8;
$b=10;
if($a>$b){
echo "a dəyişəninin b dəyişənindən böyük olduğu hal";
}
else if($a==$b){
echo "a dəyişəninin b dəyişəninə bərabər olduğu hal";
}
else{
echo "Qalan bütün hallar.";
}
?>
```

Burada əvvəlcə \$a dəyişəninin \$b dəyişənindən böyük olma halı yoxlanılır. Şərt ödənmədiyini üçün şərt daxilindəki əməliyyatlara baxılmadan ikinci şərtin, yəni, \$a dəyişəninin \$b dəyişəninə bərabər olma halı yoxlanılır. Bu hal da doğru olmadığına görə bu şərt daxilindəki əməliyyatlara baxılmadan else hissəsinə keçilir. Burada isə heç bir şərt yoxlanılmadan else daxilindəki əməliyyatlar yerinə yetirilir. Ancaq burada şərt sadə olduğuna görə yoxlanılmayan son halın \$a<b\$ olduğu bizə məlumdur. Eyni zamanda çox sayda else if istifadə edə bilərik. Məsələn:

```
<?php
$a=100;
if($a==1){
echo "Dəyişənin qiyməti 1-dir.";
}
else if($a==2){
echo "Dəyişənin qiyməti 2-dir.";
}
else if($a==3){
```

```

echo "Dəyişənin qiyməti 3-dür.";
}
else if($a==4){}
echo "Dəyişənin qiyməti 4-dür.";
)
else{
echo "Dəyişənin qiyməti fərqli bir ədəddir.";
}
?>

```

Burada nəticə olaraq "Dəyişənin qiyməti fərqli bir ədəddir." sözü ekrana çıxacaq. İndi isə daha bir kodu nəzərdən keçirək:

```

<?php
$a=100;
if($a==1){
echo "Dəyişənin qiyməti 1-dir.";
}
else if($a==2){
echo "Dəyişənin qiyməti 2-dir.";
}
else{
echo "Dəyişənin qiyməti fərqli bir ədəddir.";
}
if(5>3){
echo "Bu yeni şərt operatorudur və şərt doğrudur.";
}
?>

```

Burada axırda işlədilən if operatorunun yuxarıdakı if, else və else if ilə heç bir əlaqəsi yoxdur. Yuxarıda işlədilən if, else if və else bir-birilə əlaqəlidir. Else if və else ilk işlədilən if-in əks hallarıdır. Ancaq, sonda işlədilən if operatoru daxilində yeni şərt və yeni əməliyyatlardır və bu şərt və əməliyyatlar yuxarıdakı operator və şərtlərdən asılı deyil. Daha bir kod nümunəsinə baxaq:

```

<?php
$a=100;
if($a==1){
echo "Dəyişənin qiyməti 1-dir.";
}
else if($a==2){
echo "Dəyişənin qiyməti 2-dir.";
}

```



```

else{
echo "Dəyişənin qiyməti fərqli bir ədəddir.";
}
if(5>3){
echo "Bu yeni şərt operatorudur və şərt doğrudur.";
}
$a=7;
if($a>4){
echo "Digər şərt.";
}
?>

```

Burada sonda \$a dəyişəninə yeni qiymət verilmiş və şərtlə yoxlanılmışdır. Kodun ilk əvvəlində \$a dəyişəninə 100 qiyməti verilmişdir. Sonra if operatoru işlədilərək \$a dəyişəni yoxlanılmışdır. \$a dəyişəninə 100 etdikdən sonra yazdığımız ilk şərtlər a dəyişəninə qiymətini 100 olaraq yoxlayır. Sonda isə a dəyişəninə 7 qiymətini veririk. 7 qiymətini verdikdən sonra yazdığımız şərt isə \$a dəyişəninə qiymətini 7 olaraq yoxlayır. Yəni, Yuxarıda yazdığımız ilk operatora \$a dəyişəninə qiyməti 7 olaraq görülmür. Ümumiyyətlə proqramlaşdırmada operator öz işini görərkən özündən sonrakı sətirləri nəzərə almır. Yalnız özündən əvvəlki sətirlər nəzərə alınır. Məsələn 2-ci sətirdə \$a dəyişəninə 5 qiyməti verilmişdir. 8-ci sətirdə \$a dəyişəni if operatoru vasitəsilə yoxlanılmışdır və əməliyyatlar yerinə yetirilmişdir. 12-ci sətirdə isə yenidən \$a dəyişəninə 10 qiyməti verilmişdirsə 8-ci sətirdəki if operatoru a dəyişəninə yalnız 2-ci sətirdəki qiymətinə baxır. 12-ci sətir öz sətirindən sonra gəldiyinə görə operator bu sətirə baxmır.

Real layihələrdə operator daxilində onlarla, hətta yüzlərlə şərt yazmaq lazım gələ bilər. Bu zaman if operatorundan istifadə etmək çətin olur. Şərtlərin sayı çox olduqda digər şərt operatoru olan switch operatorundan istifadə olunur. İstifadə qaydası aşağıdakı şəkildədir:

```

<?php
switch($a){
case 'hal1': {Bu halda};
break;
case 'hal2': {Digər hal};
break;
case 'hal3': {Digər hal};
break;
default: {Qalan bütün hallar};
}
?>

```

Burada yoxlanılacaq dəyişən \$a dəyişənidir. Switch yazıldıqdan sonra mötərizə açırıq və yoxlanılacaq dəyişəni yazırıq. Case sözündən sonra halı yazıb iki nöqtə qoyuruq. Sonra isə yerinə yetiriləcək əməliyyatları yazırıq. Növbəti sətirdə break yazmalıyıq. Əgər break yazmasaq, şərt ödənsə belə yenə digər hallara baxılacaq. Bu səbəbdən break yazmaq vacibdir. Sonda isə default yazırıq və qarşısında digər bütün hallarda yerinə yetiriləcək əməliyyatları yazırıq. Məsələn:

```
<?php
$a=3;
switch($a){
case 1: echo "Doğru olmayan hal";
break;
case 2: echo "Doğru olmayan hal";
break;
case 3: echo "Doğru hal";
break;
default: echo "Digər hallar";
}
?>
```

Burada \$a dəyişəni 3 olduğu üçün 3-cü hal doğru olur və müvafiq əməliyyat yerinə yetirilir. İndi isə switch operatoruna aid daha bir nümunə yazaq. Fərz edək ki, a dəyişəni istifadəçinin forma daxil etdiyi sözdür (Bunu irəlidə öyrənəcəyik). Həmin söz ingiliscə daxil olunub və biz həmin sözün Azərbaycan dilində qarşılığın yazmalıyıq. O zaman kod aşağıdakı kimi olar:

```
<?php
$a="apple";
switch($a){
case "apple": echo "Alma";
break;
case "Orange": echo "Portağal";
break;
default: echo "Axtarılan söz tapılmadı.";
}
?>
```

Burada düzgün hala uyğun müvafiq nəticə çıxacaq.

## Dövlər. While operatoru.

Bir otaqda 100 ədəd qutu olduğunu və bu qutuladan neçəsinin boş, neçəsinin dolu olduğunu öyrənmək istədiyimizi fərz edək. Bunun üçün qutuları bir-bir açıb qutuların dolu olub-olmadığına baxmalıyıq. Buna oxşar əməliyyatları proqramlaşdırmada da yerinə yetirərkən eyni üsuldən istifadə etməliyik. Yazacağımız xəyali proqram qutuları bir-bir açıb hansının dolu, hansının boş olmasını yoxlamalıdır. 100 dəfə qutuları yoxlamaq isə bir xeyli vaxtımızı alacaq. Proqramlaşdırmada sırf bu cür ardıcıl əməliyyatları qısa şəkildə yerinə yetirmək üçün, eyni bir əməliyyatı bir neçə dəfə təkrarlamaq üçün bir sıra operatorlar mövcuddur. Bu operatorlardan biri də **while** operatorudur. Bu operator, bir şərt ödəndiyi müddətcə bir əməliyyatı yerinə yetirmək üçündür. Operatorun istifadə qaydası if operatoruna oxşardır. Əvvəlcə operatorun adı yazılır, sonra mötərizə daxilində şərt, sonra isə fiqurlu mötərizələr arasında əməliyyatlar yazılır. Nümunə bir proqramda bu operatorndan istifadə edək.

```
<?php
$a=0;
while($a<5){
echo "Ok <br>";
}
?>
```

Kodu işlətdiyimizdə səhifədə sonsuz bir dövrə yaranmış olacaq. Yəni alt-alta yazı yazılmağa davam edəcək və bu heç avxt bitməyəcək. Buna görə də səhifəni nə qədər tez bağlasaq o qədər yaxşı olar :)

Bunun səbəbi  $a$  dəyişənin qiymətinin həmişə 5-dən kiçik olmasıdır. «While» operatorunda ilk olaraq yazdığımız şərt yoxlanılır. Burada yazdığımız şərt  $a$  dəyişənin 5-dən kiçik olmasıdır. Şərt ödəndiyi üçün ilk olaraq əməliyyatlar yerinə yetirilir. Əməliyyatlar bir dəfə yerinə yetirildikdən sonra, operator yenidən əvvəlcə qayıdaraq şərti bir daha yoxlayır. Şərt doğru olarsa yenidən əməliyyatlar aparılır. Bu proses daimi olaraq davam edir. Yuxarıdakı nümunədə də şərt həmişə ödənəcəyinə görə, əməliyyatlar həmişə yerinə yetirilir. Ancaq, biz hər dövrün sonunda  $a$  dəyişənin qiymətin bir vahid artırırsa, onda, nəticə başqa cür olacaq.

İndi isə, gəlin əməliyyatları sonsuz sayda deyil, 5 dəfə yerinə yetirək. Bunun üçün kodumuzda kiçik bir dəyişiklik edək.

```
<?php
$i=0;
while($i<5){
echo "Ok <br>";
$i++;
```

```
}  
?>
```

Burada ilk olaraq  $i$  dəyişənin qiyməti 0-dır. İlk şərtə dəyişənin qiymətinin 5-dən kiçik olub-olmaması yoxlanılır. Həqiqətən də 0 rəqəmi 5-dən kiçikdir. Şərt ödəndiyinə görə əməliyyatlar yerinə yetirilməyə başlanır. İlk əməliyyatımız sözün çap olunmasıdır. Sonrakı əməliyyatımız isə dəyişənin qiymətinin 1 vahid artmasıdır. İlk dövrün sonunda dəyişənin qiyməti 1 vahid artaraq 1 olur.

Əməliyyatlar bir dəfə yerinə yetirildikdən sonra operator başa qayıdır və şərti yenidən yoxlayır. Bu dəfə  $i$  dəyişənin qiyməti 1-ə bərabərdir. Həqiqətən də 1 rəqəmi 5-dən kiçik olduğu üçün şərt ödənilir və operator əməliyyatları yerinə yetirməyə başlayır, eyni zamanda əməliyyatların sonunda  $++$  ifadəsi ilə dəyişənin qiymətini 1 vahid artıraraq 2 edir.

Artıq əməliyyatlar iki dəfə yerinə yetirildi. Növbəti dəfə yenə operator əvvəlcə qayıdaraq  $i$  dəyişənin qiymətinin 5-dən kiçik olub-olmamasını yoxlayır. Dəyişənin hal-hazırkı qiyməti 5-dən kiçik olduğu üçün, yəni şərt ödəndiyi üçün, əməliyyatlar yerinə yetirilməyə başlayır və eyni zamanda  $i$  dəyişənin qiyməti 1 vahid artaraq 3 olur.

Üçüncü dəfə əməliyyatlar yerinə yetirildikdən sonra, program yenidən əvvəlcə qayıdır və  $i$  dəyişənin qiymətinin 5-dən kiçik olub-olmamasını yoxlayır. Bu dəfə də  $i$  dəyişənin qiyməti, yəni 3, 5-dən kiçik olduğu üçün şərt ödənilir və əməliyyatlar yerinə yetirilməyə başlayır, eyni zamanda  $i$  dəyişənin qiyməti 1 vahid artaraq 4 olur.

Dördüncü dəfə də əməliyyatlar yerinə yetirildikdən sonra, operator yenidən əvvəlcə qayıdır və  $i$  dəyişənin 5-dən kiçik olub-olmamasını yoxlayır. Bu dəfə də  $i$  dəyişənin qiyməti, yəni 4, 5-dən kiçik olduğu üçün şərt ödənilir və əməliyyatlar yerinə yetirilməyə başlayır, eyni zamanda  $++$  yazıldığı üçün,  $i$  dəyişənin qiyməti 1 vahid artaraq 5 olur.

Artıq əməliyyatlar 5 dəfə yerinə yetirilib və  $i$  dəyişənin qiyməti 5-ə bərabər olub. 5-ci dəfə əməliyyatlar yerinə yetirildikdən sonra, operator yeindən əvvəlcə qayıdaraq  $i$  dəyişənin qiymətinin 5-dən kiçik olub olmadığını yoxlayır. Bu dəfə  $i$  dəyişənin qiyməti 5-dir və 5-dən kiçik deyil, 4 olsa idi doğru olacaqdı. Beləliklə şərt ödənmir, şərt ödənmədiyinə görə, görə operator bu dəfə əməliyyatları yerinə yetirmir və operatorun işi sona çatmış olur. Bundan sonra artıq operatorun sonrakı əməliyyatlar yerinə yetirilməyə başlayır.

Bu operatora aid bir neçə nümunə daha yazaraq operatorun işini daha yaxşı anlamağa çalışaq. Gəlin 1-dən 10-a qədər ədədləri alt-alta çap edən bir program yazaq.

```
<?php
```

```
$i=1;
while($i<=10){
echo $i."<br>";
$i++;
}
?>
```

Dəyişənimizin qiyməti əvvəlcə 1 olaraq təyin olunmuşdur. İlk olaraq operatorunda dəyişənin qiymətinin 11-dən kiçik-bərabər olub-olmaması yoxlanılır. Şərt doğru olduğu üçün əməliyyatlar yerinə yetirilməyə başlayır. Əməliyyatlarımız i dəyişənin çap olunmasından və sonda bir vahid artırılmasından ibarətdir. İlk olaraq i dəyişənin qiyməti 1 çap olunur və bir vahid də artaraq 2 olur. Sonra yenidən operator başa qaydır və şərt yoxlanılır. Doğru olduğu üçün əməliyyatlar yerinə yetirilir, i-nin qiyməti çap olunur və bir vahid artaraq 3 olur. Bu proses i-nin qiyməti 11 olana qədər davam edir. Dəyişənin qiyməti 11 olduqda artıq şərt ödənmədiyinə görə əməliyyatlar başlamır və operator öz işini bitirmiş olur.

İndi isə, 1-dən 5-ə qədər ədədlərin cəmini tapan bir program yazaq.

```
<?php
$i=1;
$cem=0;
while($i<=10){
$cem=$cem+$i;
$i++;
}
echo $cem;
?>
```

Burada hər dövrdə i-nin qiyməti 1 vahid artmış olur və i-nin yeni qiyməti cem dəyişənin üzərinə gəlinir. Cem dəyişənin ilk qiyməti 0-dır. İlk dövrün sonunda 1 olur. Növbəti dövrdə 3 olur, növbəti dövrdə 6, növbəti dövrdə 10, sonuncu dövrdə isə 15 olur və operator öz işini bitirmiş olur.

İndi isə, 1-dən 5-ə qədər ədədlərin hasilini tapan bir program yazaq.

```
<?php
$i=1;
$hasil=1;
while($i<=5){
$hasil=$hasil*$i;
$i++;
}
```

```
echo $hasil;  
?>
```

Burada hasil dəyişəninin ilk qiyməti 1 olaraq təyin olunmuşdur. İlk dövrdə hasil dəyişəni 1-ə vurulur, ikinci dövrdə 2-ə vurularaq 2 olur, üçüncü dövrdə 3-ə vurularaq 6 olur, dördüncü dövrdə 4-ə vurularaq 24 olur, beşinci və sonuncu dövrdə isə 5-ə vurularaq 120 olur və operator öz işini tamamlayır.

Burada diqqət etməli olduğumuz əsas məqam, cəmin tapılarkən ilk qiymətin 0, hasilin tapılarkən ilk qiymətin 1 olaraq təyin olunmasıdır. Cəm tapılarkən ilk qiymət 0 olaraq təyin olunur, çünki, növbəti ədəd 0-ın üzərinə gəlinəcək və cəm alınacaq. Hasilə isə 1 olur, çünki növbəti ədəd 1-ə vurulacaq. Əgər hasilə də ilk qiymət 0 olsa, onda növbəti ədəd 0-a vurular və yeni qiymət də 0 olardə. Nəticədə isə dəyişənin son qiyməti 0 olaraq qalardı və istədiyimiz nəticəni ala bilməzdik.

İndi isə, gəlin 1 ilə 100 arasında olan cüt ədədləri çap edək. Əvvəlcə yalnız 1 ədədin cüt olub-olmadığını yoxlamaq üçün lazım olan koda baxaq:

```
<?php  
$i=1;  
$hasil=1;  
while($i<=100){  
if($i%2==0){  
    echo $i."<br>";  
}  
$i++;  
}  
?>
```

Burada əsas if hissəsinə diqqət edək. «i%2» ifadəsi, dəyişənin 2-ə bölünməsindən alınan qalıq göstərir. Məsələn, ədədimiz 51-dirsə, 2-ə bölünməsindən alınan qalıq 1 olur. Çünki, 51 tək ədəddir. Cüt ədədlərdə isə, qalıq 0 olur. Bundan istifadə edərək ədədin cüt yoxsa tək olmasını ayırd edə bilirik. Yəni, əgər ədədin 2-ə bölünməsindən alınan qalıq 0 olarsa ədəd cütdür, 1 olarsa ədəd təkdir. Beləliklə 1-dən 100-ə qədər olan cüt ədədləri çap etmiş oluruq. Şərt hissəsində **if(\$i%2==0){** yazmaqla, 1-dən 100-ə qədər olan tək ədədləri çap edə bilərik.

## Dövlər. For operatoru.

Yuxarıdakı nümunələrdə dövr yaratmaq üçün «while» operatorundan istifadə etdik. Ancaq, dövr yaratmaq üçün «while» operatorundan əlavə «for» operatorundan da istifadə olunur. «While» operatorunda dövr sayı əvvəlcədən bilinməsə də dövrü başlatmaq olurdu və dövr sayı sonsuz da ola bilərdi. For operatoru isə əsasən dövr sayı məlum olduqda istifadə olunur. Operatorun istifadə qaydasına aid bir nümunə göstərək:

```
<?php
for($i=1;$i<=5;$i++){
    echo $i."<br>";
}
?>
```

While operatorunda  $i$  dəyişəninin artımını yalnız operatorun əməliyyatların daxilində göstərə bilirdik. Ancaq, for operatorunda, əməliyyatlar başlamazdan öncə  $i$  dəyişəninin ilk qiymətini təyin edirik. İlk parametrdə  $i$  dəyişəninin ilk qiyməti təyin olunur. İkinci parametrdə şərt yazılır, şərtimiz dəyişənin 5-dən kiçik-bərabər olmasıdır. Üçüncü parametr isə  $i$ -nin artımıdır. Operatorun işləmə qaydası da while operatorunun işləmə qaydasına oxşardır. İlk olaraq  $i$ -nin ilk qiyməti təyin olunduqdan sonra  $i$ -nin qiymətinin 5-dən kiçik-bərabər olub-olmaması yoxlanılır. Şərt ödənilsə əməliyyatlar başlayır, sonda isə  $i$ -nin qiyməti 1 vahid artırılır və yenidən əvvələ qayıdılır.  $i$ -nin qiyməti 1 vahid artdıqdan sonra şərt yenidən yoxlanılır. Şərt ödənilməyənə qədər proses eyni qaydada yerinə yetirilir. Bu proses şərt ödənməyə qədər davam edir. Dəyişənin qiyməti 5-ə çatdıqda əməliyyatlar şərt ödəndiyinə görə yerinə yetirilir və dəyişənin qiyməti bir vahid artırılaraq altı olur. Yenidən əvvələ qayıdılır və şərt yoxlanılır. Şərt ödənilmədiyinə görə əməliyyatlar yerinə yetirilmir və operator öz işini bitirir.

While operatoru ilə yazdığımız bir çox nümunəni eynilə for operatoru ilə də yazmağımız mümkündür. Məsələn, 1-dən 100-ə qədər olan cüt ədədlərin tapılması üçün lazım olan proqramı for operatorundan istifadə edərək də yaza bilərik.

```
<?php
for($i=1;$i<=100;$i++){
    if($i%2==0){
        echo $i."<br>";
    }
}
```

```
}  
}  
?>
```

Nəticədən 1-dən 100-ə qədər olan cüt ədədlər alt-alta çap olunmuş olacaq. Dövr sayı əvvəlcədən məlum olan dövrlər üçün for operatorunun istifadə olunması daha məqsədəuyğundur.

For operatoru ilə başqa bir nümunə də yazaq. Məsələn, ədədin faktorialını hesablamaq üçün for operatorundan istifadə edə bilərik. Faktorialın nə olduğunu xatırlayaq, faktorial 1-dən təyin etdiyimiz ədədə qədər olan ədədlərin hasilidir. Məsələn, 4 faktorial  $4*3*2*1$  deməkdir. Deməli bir ədədin faktorialını tapmaq üçün, 1-dən başlayaraq həmin ədədə qədər olan ədədləri bir-birinə vurmaliyiq. Aşağıdakı nümunədə proqramla 5 faktorialı hesablayaq.

```
<?php  
$faktorial=1;  
for($i=1;$i<=5;$i++){  
    $faktorial=$faktorial*$i;  
}  
echo $faktorial;  
?>
```

Yazdığımız bu kod vasitəsilə 5 faktorialı hesablamış oluruq. İlk olaraq bir faktorial dəyişəni elan edirik və onun qiymətini 1-ə bərabər edirik. Sonra isə for operatoru ilə i dəyişənini 1-dən 5-ə qədər sıralayırıq və faktorial dəyişənini i-nin aldığı hər bir qiymətə vururuq, yəni faktorial dəyişənini 1-dən 5-ə qədər bütün ədədlərə vururuq və nəticədə 5 faktorialı tapmış oluruq.



## Dövrələr. Do while operatoru

Dövr operatorlarından biri də «do while» operatorudur. Bu operator «while» operatoruna çox oxşardır. Ancaq əsas bir fərqi mövcuddur. «while» operatorunda şərt ödənmədiyi halda əməliyyatlar yerinə yetirilmirdi. «do while» operatorunda isə şərt ödənməsə belə əməliyyatlar ən azı 1 dəfə yerinə yetirilir və operator öz işini bitirir. Bəzi yerlərdə şərt ödənməsə belə əməliyyatların ən azı bir dəfə yerinə yetirilməsinə ehtiyac duyularsa, «do while» operatorundan istifadə oluna bilər. Operatorun istifadəsinə aid kiçik bir nümunə göstərək:

```
<?php
do{
echo "Ok";
} while(1==2);
?>
```

Gördüyümüz kimi, ilk olaraq do yazılır, fiqurlu mötərizələr açılır və bu mötərizələrin içində əməliyyatlar yazılır. Əməliyyatlar bitdikdən sonra fiqurlu mötərizə bağlanır və while(şərt) yazılır. Yuxarıdakı nümunədə şərtimiz 1-in 2-ə bərabər olmasıdır. Təbii ki, 1 və 2 bərabər deyillər, şərt ödənmir. Şərt ödənmədiyinə görə də operator öz işini bitirir. Ancaq yenə də, şərtə baxmadan öncə operator əməliyyatları bir dəfə yerinə yetirmiş olur. Belə deyə bilərik ki, şərtədən asılı olmadan operator əməliyyatları bir dəfə yerinə yetirir, əməliyyatları bir dəfə yerinə yetirdikdən sonra şərtə baxır.

## Çoxluqlar

Çoxluq anlayışı yəqin ki, riyaziyyatdan sizə tanış olar. Məsələn, bir A çoxluğu göstərək:

$$A=\{4,3,5\}$$

Gördüyümüz kimi, bir A çoxluğudur və içərisində üç element mövcuddur. Bu elementlər 4, 3 və 5-dir. İlk olaraq riyaziyyatda göstərsək də əslində, çoxluqları həyatın hər bir nöqtəsində görə bilərik. Riyaziyyatda çoxluğun elementləri yalnız ədədlər olaraq göstərilərsə də, praktikada ədədlərin yerini real elementlər, zərrəciklər canlılar ala bilər. Məsələn, bütün insanlığı 7 milyard elementli olan bir çoxluq olaraq görə bilərik. Bütün Azərbaycanı 10 milyonluq bir çoxluq olaraq görə bilərik. Günəş sistemini içərisində 9 elementi, yəni 9 planeti olan bir çoxluq olaraq görə bilərik. Kainatı içərisində saysız-hesabsız planet, meteorit, zərrəcik, element olan çoxluq olaraq görə bilərik. Gördüyümüz kimi, çoxluqlar hər yerdə var. Hər yerdə olduğu kimi, proqramlaşdırmada da çoxluq anlayışı var və istifadəsi də çox genişdir. Proqramlaşdırmadakı çoxluqlar adətən ədədlərdən, simvollarından, yazılardan ibarət ola bilər.

İndi isə, php dilində çoxluğun yaradılma qaydasına baxaq. Php dilində çoxluğu yaratmaq üçün **array** sözündən istifadə olunur. Aşağıdakı nümunəyə baxaq.

```
<?php
$A=array("birinci element","ikinci element","3-cu element");
echo $A[0];
?>
```

Yazdığımız kodu izah etməyə çalışaq. İlk olaraq A çoxluğunu elan etdik. A çoxluğunda 3 element var. Elementlər array daxilində vergüllə bir-birindən ayrılır.

Növbəti sətirdə echo vasitəsilə \$A[0] yazaraq çoxluğumuzun ilk elementini çap etdik. İlk element 0-cı elementdir, buna görə də 0 yazırıq. Bəs niyə 1 yox, 0 yazdıq? Çünki proqramlaşdırmada sayma 1-dən yox, 0-dan başlayır. Buna görə də, ilk elementin nömrəsi 0 olur. Əgər \$A[1] yazsa idik, o zaman çoxluğun ikinci elementi çap olunacaqdı, proqramlaşdırma dilində isə birinci elementi :)

Php dilində çoxluğun elementlərini açar sözlə də təyin edə bilərik. Aşağıdakı nümunəyə baxaq.

```
<?php
$A=array("red"=>"qirmizi","green"=>"Yasil","blue"=>"goy");
echo $A['red'];
?>
```

Burada ən əsas olan "red"=>"qirmizi" hissəsinə diqqət edək. Burada "red" elementin açar sözü, adı da deyə bilərik, "qirmizi" isə onun aldığı dəyərdir. Gördüyümüz kimi, \$A['red'] yazaraq "red" açar sözünə aid olan dəyəri əldə etmiş oluruq.

Çoxluğun bütün elementlərini tək-tək yox, hamısını bir yerdə çap etmək üçün **print\_r** funksiyasından istifadə olunur. Aşağıdakı nümunəyə baxaq.

```
<?php
$A=array("birinci element","ikinci element","3-cu element");
print_r($A);
?>
```

Nəticədə çoxluq ekrana çıxarılır.

Çoxluqlarla işləyərkən bəzən onların sayını tapmağa da ehtiyac duyulur. Bu zaman **count** funksiyasından istifadə edə bilərik. Aşağıdakı nümunəyə baxaq.

```
<?php
$A=array("birinci element","ikinci element","3-cu element");
echo count($A);
?>
```

Nəticədə çoxluqdakı element sayı çap olunur.

Php dilində çoxluqlarla işləmək üçün bir çox hazır funksiyalar mövcuddur. Onların bəzilərini aşağıda təqdim edirik. Ancaq qeyd edək ki, bu funksiyaları əzbər bilməyə ehtiyac yoxdur. Sadəcə, belə funksiyaların olduğunu və lazım olan zaman işlədə biləcəyimizi bilməyiniz kifayətdir.

1) **array\_change\_key\_case()** funksiyası - Çoxluqdakı açar sözlərin hərflərini böyütmək və ya kiçiltmək üçün istifadə olunur. İstifadə qaydası aşağıdakı şəkildədir:

```
<?php
$massiv=array("one" => "element1","two" => "Element2","three" => "Element3");
print_r(array_change_key_case($massiv,CASE_UPPER));
?>
```

Burada nəticə olaraq massivin açar sözlərinin hərfləri böyük hərflərə çevrilir. **CASE\_UPPER** əvəzinə **CASE\_LOWER** yazsaq hərflər kiçik olacaq.

2) **array\_chunk()** funksiyası - Çoxluqları bir neçə çoxluğa bölmək üçün istifadə olunur. İstifadə qaydası aşağıdakı şəkildədir:

```
<?php
$scars=array("Volvo","BMW","Toyota","Honda","Mercedes","Opel");
```

```
print_r(array_chunk($cars,2));  
?>
```

Nəticədə çoxluğu 2 çoxluğa böldük. Ancaq əgər biz elementləri açar sözlə yazmışıqsa yuxarıdakı kimi yazsaq açar sözlər müvafiq ədədlərlə əvəz olunacaq. Bu səbəbdən əgər elementləri açar sözlərlə daxil etmişiksə funksiyanı aşağıdakı şəkildə işlətməliyik:

```
<?php  
$fname=array("ad" => "Peter", "ad2" => "Ben", "ad3" => "Joe");  
print_r(array_chunk($fname,2, true));  
?>
```

Belə yazdıqda açar sözlər saxlanacaq.

3) array\_combine() funksiyası - Bu funksiya çoxluqları birləşdirərək yeni çoxluq yaratmaq üçün istifadə olunur. Məsələn:

```
<?php  
$fname=array("Peter","Ben","Joe");  
$age=array("35","37","43");  
$c=array_combine($fname,$age);  
print_r($c);  
?>
```

Burada birinci çoxluğun elementləri yeni çoxluğun açar sözləri olur. İkinci çoxluğun elementləri isə yeni çoxluğun elementləri olur.

4) array\_count\_values() funksiyası - Bu funksiya çoxluqdakı bütün elementləri saymaq üçün istifadə olunur. İstifadə qaydası aşağıdakı şəkildədir:

```
<?php  
$a=array("A","Cat","Dog","A","Dog");  
print_r(array_count_values($a));  
?>
```

Nəticə olaraq hər elementdən neçə ədəd olduğu çap olunur.

5) array\_diff() funksiyası - Bu funksiya çoxluqları müqayisə etmək üçün istifadə olunur. İstifadə qaydası aşağıdakı şəkildədir:

```
<?php
$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");
$a2=array("e"=>"red","f"=>"green","g"=>"blue");
$result=array_diff($a1,$a2);
print_r($result);
?>
```

Nəticədə \$a1 çoxluğunda olan, ancaq digər çoxluqda olmayan elementlər çap olunacaq. Başqa bir nümunə:

```
<?php
$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");
$a2=array("e"=>"red","f"=>"black","g"=>"purple");
$a3=array("a"=>"red","b"=>"black","h"=>"yellow");
$result=array_diff($a1,$a2,$a3);
print_r($result);
?>
```

Nəticədə \$a1 çoxluğunda olan və digər çoxluqlarda olmayan elementlər çap olunacaq.

6) array\_diff\_key() funksiyası - Çoxluqları açar sözlərinə görə müqayisə etmək üçün istifadə olunur. İstifadə qaydası aşağıdakı şəkildədir:

```
<?php
$a1=array("a"=>"red","b"=>"green","c"=>"blue");
$a2=array("a"=>"red","c"=>"blue","d"=>"pink");

$result=array_diff_key($a1,$a2);
print_r($result);
?>
```

Yalnız \$a1 çoxluğunda "b" açar sözü olduğuna görə yalnız bu açar söz və elementi çap olunacaq. Aşağıdakı kodda çoxluqda açar sözlər təyin olunmayıb:

```
<?php
$a1=array("red","green","blue","yellow");
$a2=array("red","green","blue");
$result=array_diff_key($a1,$a2);
print_r($result);
?>
```

Nəticədə yellow çap olunacaq, çünki 3-cü nömrə (Proqramlaşdırmada sıralama 0-dan başlayır) yalnız \$a1 çoxluğunda var. Başqa bir nümunə:

```
<?php
$a1=array("a"=>"red","b"=>"green","c"=>"blue");
$a2=array("c"=>"yellow","d"=>"black","e"=>"brown");
$a3=array("f"=>"green","c"=>"purple","g"=>"red");

$result=array_diff_key($a1,$a2,$a3);
print_r($result);
?>
```

Nəticədə a və b açar sözlərinin elementləri çap olunacaq.. Çünki a və b açar sözləri yalnız birinci çoxluqda var.

7) array\_flip() funksiyası – çoxluqdakı açar sözlərlə elementlərinin yerini dəyişmək üçün istifadə olunur. İstifadə qaydası aşağıdakı şəkildədir.

```
<?php
$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");
$result=array_flip($a1);
print_r($result);
?>
```

Nəticədə çoxluğun açar sözləri ilə elementlərinin yerləri dəyişəcək.

8) array\_intersect() funksiyası - Çoxluqları müqayisə etmək üçün istifadə olunur. Digər müqayisə funksiyalarından fərqli olaraq burada alınmış yeni çoxluq müqayisə olunmuş çoxluqların ortaq elementlərindən təşkil olunacaq.

```
<?php
$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");
$a2=array("e"=>"red","f"=>"green","g"=>"blue");
$result=array_intersect($a1,$a2);
print_r($result);
?>
```

Nəticədə alınmış çoxluq: Array ( [a] => red [b] => green [c] => blue )

Nəticədə alınmış yeni çoxluqda yuxarıdakı iki çoxluğun ortaq elementləri olacaq. Başqa bir nümunə göstərək:

```
<?php
$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");
```

```
$a2=array("e"=>"red","f"=>"black","g"=>"purple");
$a3=array("a"=>"red","b"=>"black","h"=>"yellow");
$result=array_intersect($a1,$a2,$a3);
print_r($result);
?>
```

Nəticədə alınmış çoxluq: Array ( [a] => red )

Əlavə olaraq deyək ki, müqayisə olunarkən yalnız elementlər müqayisə olunur və açar sözlərə baxılmır.

9) array\_intersect\_assoc() - Bu funksiya çoxluqları eyni zamanda həm elementlərə, həm də elementlərin açar sözlərinə görə müqayisə etmək üçün istifadə olunur. Nəticədə ortaq elementlər seçilir. İstifadə qaydası aşağıdakı şəkildədir:

```
<?php
$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");
$a2=array("a"=>"red","b"=>"green","c"=>"blue");
$result=array_intersect_assoc($a1,$a2);
print_r($result);
?>
```

Nəticədə alınmış çoxluq: Array ( [a] => red [b] => green [c] => blue )

Başqa bir nümunə göstərək:

```
<?php
$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");
$a2=array("a"=>"red","b"=>"green","g"=>"blue");
$a3=array("a"=>"red","b"=>"green","g"=>"blue");
$result=array_intersect_assoc($a1,$a2,$a3);
print_r($result);
?>
```

Nəticədə alınmış çoxluq: Array ( [a] => red [b] => green )

Çoxluqlardakı 3-cü elementlər eyni olsalar da, açar sözlər fərqli olduğuna görə 3-cü element yeni çoxluğa daxil olmur.

10) array\_intersect\_key() funksiyası - Bu funksiya çoxluqları açar sözlərinə görə müqayisə etmək üçün istifadə olunur. Burada da ortaq elementlər seçilir. Məsələn:

```
<?php
$a1=array("a"=>"red","b"=>"green","c"=>"blue");
$a2=array("a"=>"red","c"=>"blue","d"=>"pink");
$result=array_intersect_key($a1,$a2);
```

```
print_r($result);
?>
```

Nəticədə alınmış yeni çoxluq: Array ( [a] => red [c] => blue )

Başqa bir nümunə:

```
<?php
$a1=array("a"=>"red","b"=>"green","c"=>"blue");
$a2=array("c"=>"yellow","d"=>"black","e"=>"brown");
$a3=array("f"=>"green","c"=>"purple","g"=>"red");
$result=array_intersect_key($a1,$a2,$a3);
print_r($result);
?>
```

Nəticədə alınmış çoxluq: Array ( [c] => blue )

Açar sözlər olmadıqda isə massivlər sıra nömrələrinə görə müqayisə olunur:

```
<?php
$a1=array("red","green","blue","yellow");
$a2=array("red","green","blue");
$result=array_intersect_key($a1,$a2);
print_r($result);
?>
```

Nəticədə alınmış çoxluq: Array ( [0] => red [1] => green [2] => blue )

11) array\_key\_exists() funksiyası - Bu funksiya çoxluqda açar sözə görə bir elementin olub-olmadığını yoxlamaq üçün istifadə olunur. Funksiyanı istifadə edərkən şərt operatorundan da istifadə edəcəyik. Məsələn:

```
<?php
$a=array("Volvo"=>"XC90","BMW"=>"X5");
if (array_key_exists("Volvo",$a))
{
    echo "Açar söz mövcuddur!";
}
else
{
    echo "Açar söz mövcud deyil!";
}
?>
```



Burada ilk öncə çoxluq yaradılır. Sonra isə şərt operatoru vasitəsilə \$a çoxluğunda Volvo açar sözünün olub-olmadığı yoxlanılır. Əgər şərt doğru olsa “Açar söz mövcuddur” çap olunacaq, əks halda isə “Açar söz mövcud deyil” deyil sözü çap olunacaq. İndi isə açar söz olmadan elementin olub-olmamağını yoxlayaq. Bunun üçün elementin sıra nömrəsindən istifadə edəcəyik. Məsələn:

```
<?php
$a=array("Volvo","BMW");
if (array_key_exists(0,$a))
{
echo "Açar mövcuddur!";
}
else
{
echo "Açar mövcud deyil!";
}
?>
```

Burada isə 0-cı elementin (Proqramlaşdırmada sayma 0-dan başlayır) olub-olmamağı yoxlanılır. Şərt doğru olduğu üçün “Açar mövcuddur” sözü çap olunacaq.

12)array\_keys() funksiyası - Mövcud çoxluğun açar sözlərindən ibarət yeni çoxluq yaratmaq üçün istifadə olunur. Məsələn:

```
<?php
$a=array("key1"=>"element1","key2"=>"Element2","key3"=>"element3");
$b=array_keys($a);
print_r($b);
?>
```

Burada \$b çoxluğu \$a çoxluğunun açar sözlərindən ibarət yeni çoxluqdur.

13)array\_merge() funksiyası - Bu funksiya iki və ya daha çox çoxluğu birləşdirərək yeni çoxluq yaratmaq üçün istifadə olunur. Məsələn:

```
<?php
$a1=array("red","green");
$a2=array("blue","yellow");
$a3=array_merge($a1,$a2);
print_r($a3);
?>
```

Burada \$a3 çoxluğu \$a1 və \$a2 çoxluqlarının elementlərindən ibarət yeni çoxluq olur.

14)array\_multisort() funksiyası - Çoxluq elementlərini sıralamaq üçün istifadə olunur. Elementləri müxtəlif cür sıralamaq olar.İstifadə qaydası aşağıdakı şəkildədir:

```
<?php
$a=array("Dog","Cat","Horse","Bear","Zebra");
array_multisort($a);
print_r($a);
?>
```

Burada çoxluq elementləri əlifba sırasına görə sıralanır. Elementlər ədədlər olduqda kiçikdən böyüyə sıralanır. Məsələn:

```
<?php
$a=array(10,8,15,11,20);
array_multisort($a);
print_r($a);
?>
```

Burada ədədlər kiçikdən böyüyə sıralanacaq.

15)array\_pop() funksiyası - Çoxluğun son elementini silmək üçün istifadə olunur. Məsələn:

```
<?php
$a=array("red","green","blue");
array_pop($a);
print_r($a);
?>
```

Nəticədə son element olan “blue” elementi massivdən çıxarılır.

16)array\_product() funksiyası - Bu funksiya çoxluq elementlərini bir-birinə vurub nəticə əldə etmək üçün istifadə olunur. Məsələn:

```
<?php
$a=array(5,5,8);
echo(array_product($a));
?>
```

Nəticədə 200 çap olunacaq. Burada massiv yox, sadəcə bir ədəd çap olunduğuna görə print\_r funksiyasından yox, echo funksiyasından istifadə olunur.

17)array\_push() funksiyası - Çoxluğa yeni elementlər əlavə etmək üçün istifadə olunur. Məsələn:

```
<?php
$a=array("red","green");
array_push($a,"blue","yellow");
print_r($a);
?>
```

Burada çoxluğa “blue” və “yellow” elementləri əlavə olunur.

18)array\_reverse() funksiyası - Bu funksiya çoxluq elementlərini tərs qaydada düzmək üçün istifadə olunur. Məsələn:

```
<?php
$a=array("a"=>"Volvo","b"=>"BMW","c"=>"Toyota","d"=>"Honda");
$b=array_reverse($a);
print_r($b);
?>
```

Nəticədə alınmış çoxluq: Array ( [d] => Honda [c] => Toyota [b] => BMW [a] => Volvo )

19)array\_search() - Çoxluqdakı elementin açar sözünü tapmaq üçün istifadə olunur. Məsələn:

```
<?php
$a=array("a"=>"red","b"=>"green","c"=>"blue");
echo array_search("red",$a);
?>
```

“Red” elementinin açar sözü “a” olduğu üçün “a” çap olunacaq.

20)array\_shift() funksiyası - Çoxluqdakı ilk elemnti silmək üçün istifadə olunur. Məsələn:

```
<?php
$a=array("a"=>"red","b"=>"green","c"=>"blue");
array_shift($a);
print_r ($a);
```

?>

Nəticədə alınmış çoxluq: Array ( [b] => green [c] => blue )

Funksiyanı, indeks nömrələri ilə olan çoxluq üçün istifadə edək:

```
<?php
$a=array(0=>"red",1=>"green",2=>"blue");
array_shift($a);
print_r($a);
?>
```

Nəticədə alınmış çoxluq: Array ( [0] => green [1] => blue )

Eyni zamanda ilk elementi silərkən həmin elementi ekrana çap edə bilərik. Məsələn:

```
<?php
$a=array("a"=>"red","b"=>"green","c"=>"blue");
echo array_shift($a);
print_r($a);
?>
```

Burada massivlə yanaşı silinən elementdə ekrana çap olunur.

21)array\_slice() funksiyası - Hazırki çoxluğun bir hissəsindən ibarət yeni massiv yaratmaq üçün istifadə olunur. Məsələn:

```
<?php
$a=array("red","green","blue","yellow","brown");
$b=array_slice($a,2);
print_r($b);
?>
```

Burada \$b çoxluğu \$a çoxluğunun 2-ci elementi də daxil olmaqla (proqramlaşdırmada sayma 0-dan başladığı üçün 2-ci element dedikdə 3-cü yerdə duran element nəzərdə tutulur) növbəti elementlərindən ibarət yeni çoxluqdur. Yuxarıdakı kimi yazdıqda \$b çoxluğuna \$a çoxluğunun 2-cidən başlayaraq bütün elementləri daxil ola bilər. Ancaq biz 2-ci elementdən başlayaraq istədiyimiz sayda elementi yeni çoxluğa daxil edə bilərik. Məsələn:

```
<?php
$a=array("red","green","blue","yellow","brown");
$b=array_slice($a,2,1);
print_r($b);
?>
```

Belə yazdıqda 2-ci elementdən başlayaraq yalnız bir element yeni çoxluğa daxil olacaq. Sıralamağa əvvəldən başladığımız kimi axırdan da başlaya bilərik Məsələn əgər biz axırdan 2-ci elementdən başlayaraq elementləri seçmək istəyiriksə onda aşağıdakı kimi yazmalıyıq

```
<?php
$a=array("red","green","blue","yellow","brown");
$b=array_slice($a,-2,2);
print_r($b);
?>
```

Nəticədə \$b çoxluğu \$a çoxluğunun sondan 2-ci elementindən başlayaraq 2 elementindən ibarət olur. -1-ci element dedikdə sonuncu element, -2-ci element dedikdə isə sondan ikinci element nəzərdə tutulur.

22)array\_sum() funksiyası - Ədədlərdən ibarət çoxluğun elementlərinin cəmini tapmaq üçün istifadə olunur. Məsələn:

```
<?php
$a=array(5,15,25);
echo array_sum($a);
?>
```

Nəticədə ekrana 45 çıxacaq.

23)array\_unique() funksiyası - Çoxluqda olan eyni elementləri silmək üçün istifadə olunur Əgər çoxluqda bir neçə eyni element varsa birincisi saxlanılır, digərləri isə silinir. Məsələn:

```
<?php
$a=array("a"=>"red","b"=>"green","c"=>"red","e"=>"red");
$b=array_unique($a);
print_r($b);
?>
```

Nəticədə alınmış \$b çoxluğu: Array ( [a] => red [b] => green )

24)array\_unshift() funksiyası - Çoxluğa yeni element daxil etmək üçün istifadə olunur. Məsələn:

```
<?php
$a=array("a"=>"red","b"=>"green");
```

```
array_unshift($a,"blue");
print_r($a);
?>
```

Nəticədə \$a çoxluğuna “blue” elementi daxil olur. Aşağıdakı kimi yazdıqda isə element a çoxluğuna daxil olduqdan sonra a çoxluğunun element sayı çap olunur:

```
<?php
$a=array("a"=>"red","b"=>"green");
echo array_unshift($a,"blue");
echo "<br>";
print_r($a);
?>
```

Son sətirdə isə print\_r() funksiyası vasitəsilə yeni a çoxluğu çap olunur. Funksiyanı istifadə etdikdə yeni element massivin ilk elementi olur. Elementlər indeks nömrələri ilə olduqda yenə yeni element çoxluğun ilk elementi olur və 0 indeksini alır. Məsələn:

```
<?php
$a=array(0=>"red",1=>"green");
array_unshift($a,"blue");
print_r($a);
?>
```

Burada “blue” elementi 0 indeksini aldı və digər elementlərin indeksləri bir vahid artdı.

25)array\_values() funksiyası - Çoxluq elementlərini indeks nömrələri ilə almaq üçün istifadə olunur. Məsələn:

```
<?php
$a=array("Name"=>"Peter","Age"=>"41","Country"=>"USA");
$b=array_values($a);
print_r($b);
?>
```

Nəticədə \$a çoxluğundakı elementlərin açar sözləri olsa da, \$b çoxluğundakı elementlər indeks nömrələri ilə olur.

26)Current() funksiyası - Çoxluqdakı ilk elementi seçmək üçün istifadə olunur, Məsələn:

```
<?php
$people = array("Peter", "Joe", "Glenn", "Cleveland");
echo current($people) . "<br>";
?>
```

Nəticədə çoxluğun ilk elementi çap olunur,

27)End() funksiyası - Çoxluğun son elementini seçmək üçün istifadə olunur, Məsələn:

```
<?php
$people = array("Peter", "Joe", "Glenn", "Cleveland");
echo end($people);
?>
```

Nəticədə massiv son elementi çap olunacaq,

28)Extract() funksiyası-Çoxluqdakı açar sözlərlə adlanan, uyğun olaraq massiv elementlərinə bərabər olan dəyişənlər yaratmaq üçün istifadə olunur, Məsələn:

```
<?php
$array=array("key1"=>"ilkelement","key2"=>5,"key3"=>"Sonuncu");
extract($array);
echo $key1."<br>".$key2."<br>".$key3;
?>
```

Burada \$key1, \$key2, \$key3 adlı dəyişənlər yaranır və uyğun dəyərlərinə bərabər olur.

29)In\_array() funksiyası-Bir elementin çoxluqda olub olmadığını yoxlamaq üçün istifadə olunur. Məsələn:

```
<?php
$people = array("Peter", "Joe", "Glenn", "Cleveland");
if (in_array("Joe", $people))
{
    echo "Element tapıldı";
}
else
{
    echo "Element tapılmadı";
}
?>
```

Burada “Joe” elementi çoxluqda olduğu üçün Ekranə “Element tapıldı” yazısı çap olunacaq.

30)List() funksiyası - Çoxluq elementlərini dəyişənlərə köçürmək üçün istifadə olunur. Məsələn:

```
<?php
$people = array("Peter", "Joe", "Glenn", "Cleveland");
list($a,$b,$c,$d)=$people;
echo $a."<br>".$b."<br>".$c."<br>".$d;
?>
```

Burada uyğun olaraq çoxluq elementlərinə bərabər olan \$a, \$b, \$c, \$d dəyişənləri yaradıldı. Çoxluğun sadəcə bir neçə elementinə aid dəyişənlər yaratmaq mümkündür. Məsələn:

```
<?php
$people = array("Peter", "Joe", "Glenn", "Cleveland");
list($a,$b)=$people;
echo $a."<br>".$b;
?>
```

Burada yalnız çoxluğun ilk iki elementinə uyğun dəyişənlər yaradıldı və çap olundu.

31)Next() funksiyası - Çoxluğun növbəti elementlərini seçmək üçün istifadə olunur. Məsələn:

```
<?php
$people = array("Peter", "Joe", "Glenn", "Cleveland");
echo next($people);
?>
```

Burada ikinci element çap olunacaq. Məsələn 4-cü elementi çap etmək istəsək, 3 dəfə next funksiyasından istifadə etməliyik:

```
<?php
$people = array("Peter", "Joe", "Glenn", "Cleveland");
next($people);
next($people);
echo next($people);
?>
```

Nəticədə 4-cü element çap olunacaq.



32)Prev() funksiyası - əvvəlki elementi seçmək üçün istifadə olunur. Məsələn:

```
<?php
$people =array("Peter", "Joe", "Glenn", "Cleveland");
echo next($people) . "<br>";
echo prev($people);
?>
```

Burada ilk öncə next() funksiyası vasitəsilə növbəti element, yəni, ikinci element seçilərək çap edilir Növbəti addımda isə prev() funksiyası vasitəsilə bir vahid əvvəlki element ,yəni, 1-ci element çap edilir.

33)Range() funksiyası - Seçilmiş iki ədəd arasındakı ədələrdən ibarət yeni Çoxluq yaratmaq üçün istifadə olunur. Məsələn:

```
<?php
$number = range(0,5);
print_r ($number);
?>
```

Nəticə: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 )

34)Reset() funksiyası — Çoxluğu yeniləmək üçün istifadə olunur. Məsələn:

```
<?php
$people = array("Peter", "Joe", "Glenn", "Cleveland");
echo next($people) . "<br>";
echo next($people) . "<br>";
echo reset($people);
?>
```

Burada iki dəfə next() funksiyasını işlədərək 3-cü elementə getmiş oluruq. Ancaq sonra reset() funksiyasını işlədərək yenidən 1-ci elementə qayıdırıq.

35)Shuffle() funksiyası - Çoxluq elementlərini təsadüfi şəkildə sıralamaq üçün istifadə olunur. Məsələn:

```
<?php
$my_array = array("red","green","blue","yellow","purple");
shuffle($my_array);
print_r($my_array);
?>
```

Burada hər dəfə səhifə yeniləndikdə çoxluq elementləri təsadüfi olaraq yerini dəyişəcək.

## Funksiyalar

İlk olaraq funksiyaların istifadəsini riyaziyyatda görə bilərik. Məsələn,  $x$  kvadratı funksiyasını aşağıdakı şəkildə yazma bilərik:

$$Y=F(x)=x*x;$$

Burada,  $x$ -ə müxtəlif qiymətlər verdikdə  $y$  də müxtəlif qiymətlər alır. Məsələn,  $x=2$  olduqda  $y=2*2=4$  olur,  $x=3$  olduqda  $y=3*3=9$  olur. Təbiətdə demək olar hər şey funksiyalarla təyin olunur. Məsələn, gedilən yolu tapmaq üçün funksiya:

$$Y=F(\text{sürət,zaman})=\text{sürət}*zaman$$

Təbiətdə də bir çox qiymətlər müxtəlif funksiyalarla təyin olunur. Funksiyada ən əsas olan onun parametrləri, funksiyanın özünün riyazi düsturu və funksiyanın nəticədə alacağı qiymətdir. Məsələn,  $x$ -in kvadratı funksiyasında  $y$  alınacaq nəticə,  $x$  parametrləri,  $x*x$  isə funksiyanın düsturudur.

Programlaşdırmada da funksiyaların işləmə prinsipi eynidir. Sadəcə, funksiyanı müəyyən sintaksislə yazmaq lazımdır. Php dilində funksiyanın elan olunma qaydası aşağıdakı şəkildədir:

```
<?php
function funksiya_adi(arqumentlər){
//Əməliyyatlar
}
?>
```

Burada arqumentlər hissəsinə istifadə olunacaq arqumentlər, əməliyyatlar hissəsinə isə lazımi əməliyyatları yazmalıyıq. İlk öncə arqumentsiz funksiyalarla tanış olaq. Sadəcə ekrana yazı yazan bir funksiya yaradaq:

```
<?php
function funksiya(){
echo "First function";
}
?>
```

Burada artıq funksiya yaradıldı. Ancaq burada ekrana heç yazılmayacaq çünki biz funksiyanı çağırılmamışıq. Funksiyanı çağırmaq üçün sadəcə növbəti sətirlərdə funksiyanın adını yazırıq. Aşağıdakı şəkildə funksiyanı çağırmaq.

```
<?php
function funksiya(){
echo "First function";
```

```
}  
funksiya();  
?>
```

Nəticədə funksiya öz işini görür və ekrana yazı yazılır. Burada funksiyanı bir neçə dəfə çağıraraq funksiya əməliyyatlarını bir neçə dəfə yerinə yetirə bilərik.

Bu yazdığımız arqumentsiz funksiya idi. İndi isə arqumentli funksiyalarla tanış olaq. Arqumentli funksiya yaradarkən funksiyada mötərizə daxilində arqumentləri daxil edirik və funksiya daxilindən həmin arqumentlər üzərində müəyyən əməliyyatlar aparırıq. Funksiyalar çağırıldıqda isə arqumentlər müəyyən dəyərlərlə əvəz olunur və əməliyyatlar həmin dəyərlər üzərində aparılır. Məsələn:

```
<?php  
function funksiya($a){  
echo $a;  
}  
funksiya("First variable");  
?>
```

Burada \$a arqument olur və sonra funksiya çağırıldıqda arqument dəyərlə əvəzlənir. Nəticədə arqumentin yerinə yazdığımız dəyər çap olunur. İndi isə bir neçə arqumentli funksiya yazaq:

```
<?php  
function funksiya($a,$b,$c){  
echo $a+$b+$c;  
}  
funksiya(5,4,8);  
?>
```

Burada \$a,\$b və \$c dəyişən olur. Sonra funksiya çağırıldıqda Arqumentlər ədədlərlə əvəz olunur. İndi isə bu funksiyanı müxtəlif dəyərlərlə bir neçə dəfə işlədək:

```
<?php  
function funksiya($a,$b,$c){  
echo $a+$b+$c."<br>";  
}  
funksiya(4,5,8);  
funksiya(1,2,3);  
funksiya(7,8,5);
```

```
?>
```

Nəticədə müxtəlif qiymətlər çap olunur. Yazdığımız funksiya daxilində digər hazır funksiyalarıda işlədə bilərik. Məsələn:

```
<?php
function funksiya($a){
echo sqrt($a);
}
funksiya(4);
?>
```

Nəticədə 2 çap olunur. Ancaq bir şeyi nəzərə almaq lazımdır ki funksiya daxilində echo ilə ekrana nəticəni yazdığımızda, sonra biz həmin funksiyanı çağırdıqda nəticəni ədəd kimi istifadə edə bilməyəcəyik. Bunun səbəbi echo funksiyasının nəticəni sadəcə ekrana yazmasıdır. Aşağıdakı koda baxaq:

```
<?php
function funksiya($a,$b){
echo $a+$b;
}
$a=funksiya(5,4);
$b=funksiya(3,2);
echo $a+$b;
?>
```

Burada biz düzgün nəticə ala bilməyəcəyik, çünki, nəticədə sadəcə 9 və 5 yan-yana yazılacaq. Əgər funksiya daxilindəki əməliyyatlardan sonra alınmış nəticə üzərində əməliyyatlar aparmaq istəyiriksə, funksiya daxilində echo yox, return istifadə edərək müəyyən bir dəyər qaytarmalıyıq. Bu halda nəticə üzərində əməliyyat aparmaq mümkün olacaq. Məsələn:

```
<?php
function funksiya($a,$b){
return $a+$b;
}
$a=funksiya(5,4);
$b=funksiya(3,2);
echo $a+$b;
?>
```

Burada isə funksiyadan alınan qiymətlər üzərində əməliyyatlar aparmaq mümkün olur, yəni funksiya konkret bir qiymətə bərabər olur. Funksiyaya aid bir neçə nümunə yazaq.

Nümunə 1: Əgər bir arqumentli funksiyada arqument müsbətdirsə 1 dəyərini, əks halda isə 0 dəyərini qaytaraq, Sonra isə bu dəyəri yoxlayaraq ekrana yazı yazdıraq. Kod aşağıdakı kimi olacaq:

```
<?php
function funksiya($a){
if($a>0){
return 1;
}
else{
return 0;
}
}
$check=funksiya(5);
if($check==1){
echo "Musbet";
}
else{
echo "Menfi ve ya sifir";
}
?>
```

Kodu izah edək. İlk öncə funksiyanı elan etdik və \$a arqumentini təyin etdik. Funksiya daxilində if operatoru vasitəsilə bu arqumentin 0-dan böyük olub-olmamasını yoxladıq. Əgər şərt doğru olsa 1 qiyməti qayıdacaq. Əks halda isə 0 qiyməti qayıdacaq. Funksiya sona çatır. Deməli əgər arqument 0-dan böyük olsa 1, 0-dan kiçik olsa 0 qiyməti qayıdacaq, Növbəti addımda isə funksiya 5 arqumentini verərək \$check dəyişəninə mənimsədirik. Buradan asanlıqla görə bilərik ki, 5 ədədi 0-dan böyük olduğu üçün funksiyada 1 dəyəri qayıdacaq və \$check dəyişəni 1-ə bərabər olacaq. Növbəti olaraq isə \$check dəyişəninə yoxlayaraq ekrana müvafiq yazını yazırıq.

Nümunə 2: Fərz edək ki funksiya dəyəri olaraq 1-7 arası ədədlərdən biri verilir və biz bu qiymətə əsasən həftənin gününü ekrana yazmalıyıq. Burada şərt çox olduğu üçün

switch operatorundan istifadə edəcəyik. Burada sadəcə ekrana yazma olacaq deyə return istifadə etməyəcəyik. Kod aşağıdakı kimi olacaq:

```
<?php
function funksiya($a){
switch($a){
case 1: echo "Birinci gün";
break;
case 2: echo "İkinci gün";
break;
case 3: echo "Üçüncü gün";
break;
case 4: echo "Dördüncü gün";
break;
case 5: echo "Beşinci gün";
break;
case 6: echo "Altıncı gün";
break;
case 7: echo "Bazar günü";
break;
default: echo "Daxiletme sehvidir!";
break;
}
}
funksiya(3);
?>
```

Nəticədə "Üçüncü gün" sözü çap olunacaq.

Nümunə 3: Funksiya vasitəsilə 1-dən arqumentə qədər ədələri çap edək. Bunun üçün for operatorundan istifadə edəcəyik. Kod aşağıdakı kimi olacaq:

```
<?php
function funksiya($a){
for ($i=1;$i<=$a;$i++)
echo $i."<br>";
}
funksiya(10);
?>
```

Nəticədə 1-dən 10-a kimi ədələr alt-alta çap olunacaq.

## Isset(), Empty() və Unset() funksiyaları

Bir dəyişənin dəyərə malik olub-olmadığını yoxlamaq üçün isset() funksiyasından istifadə olunur. İstifadə qaydası aşağıdakı şəkildədir:

```
<?php
$a=5;
if(isset($a)){
echo "Dəyişən mövcuddur!";
}
?>
```

Isset() funksiyasının tam əksi isə unset() funksiyasıdır. Əgər dəyişən dəyərə malik deyilsə doğru sayılır.

```
<?php
$a=5;
if(empty($a)){
echo "Dəyişən mövcud deyil!";
}
else{
echo "Dəyişən mövcuddur!";
}
?>
```

Dəyişəni silmək üçün unset() funksiyasından istifadə olunur.

```
<?php
$a=5;
unset($a);
if(empty($a)){
echo "Dəyişən mövcud deyil";
}
else{
echo "Dəyişən mövcuddur";
}
?>
```

Burada unset() funksiyası ilə dəyişən silindiyinə görə şərt doğru olacaq və «Dəyişən mövcud deyil» yazısı çap olunacaq.



## Form daxili əməliyyatlar. Post və Get metodları

Yaradılmış formlardan istifadəçi tərəfindən daxil edilmiş məlumatları götürmək və başqa vacib məlumatları əldə etmək üçün Post və Get metodlarından istifadə olunur. İlk öncə Get metodu ilə tanış olaq.

Get metodu. Bu metodla formdan məlumatları əldə etmək üçün yaratdığımız formanın method atributuna «Get» yazmalıyıq. Əvvəlcə formanı yaradaq:

```
<form method="Get">
Adınız: <input type="text" name="ad">
<br>
<input type="submit" value="Gonder">
</form>
```

Formanı hazırladıq. Əməliyyatlar eyni səhifədə baş verəcəyi üçün formada action daxil etmirik. Burada yaratdığımız inputların adlarına xüsusi fikir vermək lazımdır, çünki, inputlara bu adlarla müraciət edəcəyik. Formanı yaratdıq. İndi isə inputa daxil olunmuş yazını ekrana yazmağa çalışaq. Inputa daxil olunmuş yazı, Get metodu ilə aşağıdakı şəkildə götürülür:

```
$a=$_GET['inputun_adi'];
```

Burada inputa daxil edilmiş yazı \$a dəyişəninə mənimsədilmiş olur. Ümumi kod isə bu formada olacaq:

```
<form method="Get">
Adınız: <input type="text" name="ad">
<br>
<input type="submit" value="Gonder">
</form>
<?php
$a=$_GET['ad'];
echo $a;
?>
```

Nəticədə inputa daxil olunmuş yazı çap olunacaq. Biz burada birbaşa yazını çap etdik. Ancaq əslində göndər düyməsinə basıldıqdan sonra inputun dəyərini götürməliyik. Bunun üçün isə isset() funksiyası vasitəsilə göndər düyməsinə basılmağını yoxlamalıyıq. Yuxarıdakı php kodunda aşağıdakı dəyişikliyi edək:

```
<?php
if(isset($_GET['daxil'])) {
$a=$_GET['ad'];
echo $a;
```

```
}  
?>
```

Burada artıq inputa daxil edilmiş yazı Göndər basıldıqdan sonra əldə olunacaq və çap olunacaq.

Get metodunun mənfi bir xüsusiyyəti var ki, get metodundan istifadə etdikdə formaya daxil edilmiş dəyərlər brauzerin adres hissəsində görünür. Məsələn, yuxarıdakı kodda Göndər düyməsinə basdıqda brauzerdə adres hissəsi aşağıdakı şəkildə olur:

**1.php?ad=Daxiletme&daxil=Gonder**

Belə olduqda isə təhlükəsizlik cəhətdən problemlər yaranar bilər. Bu səbəbdən formalarda digər metod olan post metodundan istifadə etmək daha əlverişlidir. Ancaq bu yazıların adres hissəsinə yazılmağından biz başqa yerlərdə də istifadə edə bilərik, Məsələn fərz edək ki, sayta 3 xəbər çap etmişik. Bu xəbərləri silmək üçün bizə əlavə iki səhifə lazım olacaq. Birinci səhifədə xəbərlərin siyahısı və silmə linki olacaq. İkinci səhifədə isə silmə əməliyyatı yerinə yetiriləcək. Birinci səhifəmiz:

```
<a href="sil.php?xeber=1">Sil</a>  
<br>  
<a href="sil.php?xeber=2">Sil</a>  
<br>  
<a href="sil.php?xeber=3">Sil</a>
```

İkinci səhifəmizin adı isə sil.php-dir. Sil.php səhifəsinə silmək istədiyimiz xəbərə uyğun ədəd göndəririk. Sil.php səhifəsində isə aşağıdakı kod yazılacaq və ədədə uyğun xəbər silinəcək:

```
<?php  
echo $_GET['xeber'];  
// Silmə əməliyyatları  
?>
```

Bu cür əməliyyatlarla irəlidəki mövzularda tanış olacağıq.

İndi isə Post metodu ilə tanış olaq.

Post metodu. Bu metodun istifadə qaydası Get metodu ilə eynidir. Üstün cəhəti isə formalarda daxil edilmiş məlumatlar brauzerin adres hissəsində göstərilmir. Bu zaman istə təhlükəsizlik problemləri yaranmır. Post metoduna aid bir nümunə:

```
<form method="POST">  
Adınız: <input type="text" name="ad">  
<br>  
<input type="submit" value="Gonder">  
</form>  
<?php
```

```
$a=$_POST['ad'];  
echo $a;  
?>
```

Burada məlumat adres hissəsinə yazılmadan səhifədə çap olunacaq.

## Filter funksiyaları

Dəyişənə hər hansı bir məlumatı ötürdükdən sonra həmin məlumatın doğruluğunu yoxlamaq lazımdır. Bu təhlükəsizlik baxımından vacibdir. Bir sıra filter funksiyaları mövcuddur. Bunlar aşağıdakılardır:

1)filter\_var() funksiyası-Hər hansı bir məlumatın doğru olub-olmadığını yoxlamaq üçün istifadə olunur. Məsələn aşağıda emailin doğruluğu yoxlanılır:

```
<?php
$email = "example@example.com";
if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
    echo "Valid!";
}
else {
    echo "Invalid!";
}
?>
```

Funksiyada emaili yoxlamaq üçün FILTER\_VALIDATE\_EMAIL ifadəsi işlənir. filter\_var() funksiyası ilə emaildən əlavə bir çox ifadələri yoxlamaq mümkündür.

İfadənin tam ədəd olub-olmadığını yoxlamaq üçün:

```
<?php
$a=123;
if (filter_var($a, FILTER_VALIDATE_INT)) {
    echo("True");
} else {
    echo("False");
}
?>
```

Burada \$a dəyişəni tam ədəd olduğu üçün True sözü çap olunacaq.

2.İfadənin məntiqi tip olub-olmadığını yoxlamaq üçün:

```
<?php
$a=true;
if (filter_var($a, FILTER_VALIDATE_BOOLEAN)) {
    echo("True");
} else {
    echo("False");
}
}
```

?>

İfadə məntiqi olduğu üçün true sözü çap olunur.

3.İfadənin onluq ədəd olub-olmadığını yoxlamaq üçün:

```
<?php
$a=5.4;
if (filter_var($a, FILTER_VALIDATE_FLOAT)) {
    echo("True");
} else {
    echo("False");
}
?>
```

Nəticədə true sözü çap olunacaq. Tam ədədlər də buraya daxildir.

4.İfadənin URL ünvan olub-olmadığını yoxlamaq üçün:

```
<?php
$a="http://example.com/";
if (filter_var($a, FILTER_VALIDATE_URL)) {
    echo("True");
} else {
    echo("False");
}
?>
```

5.İfadənin Email olub-olmadığını yoxlamaq üçün:

```
<?php
$a="email@email.com";
if (filter_var($a, FILTER_VALIDATE_EMAIL)) {
    echo("True");
} else {
    echo("False");
}
?>
```

6.İfadənin İP ünvan olub-olmadığını yoxlamaq üçün:

```
<?php
$a="1.1.1.1";
```

```
if (filter_var($a, FILTER_VALIDATE_IP)) {
    echo("True");
} else {
    echo("False");
}
?>
```

7. Email olaraq təqdim olunan ifadədən uyğun olmayan simvolları silmək üçün:

```
<?php
$email = "john(.doe)@exa//mple.com";
$email = filter_var($email, FILTER_SANITIZE_EMAIL);
echo $email;
?>
```

Nəticə: john.doe@example.com

8. Aşağıdakı filterləmədə addslashes() funksiyası ilə eyni iş görülür:

```
<?php
$var="Peter's here!";
echo(filter_var($var, FILTER_SANITIZE_MAGIC_QUOTES));
?>
```

Nəticə: Peter\'s here!

9. Onluq ifadədən uyğun olmayan simvolları silmək üçün:

```
<?php
$number="5-2f+3.3pp";
echo(filter_var($number, FILTER_SANITIZE_NUMBER_FLOAT,
FILTER_FLAG_ALLOW_FRACTION));
?>
```

Nəticə: 5-2+3.3

Funksiyada üçüncü parametrlər olaraq FILTER\_FLAG\_ALLOW\_FRACTION daxil etdik. Bu parametrlər ifadədə nöqtələrin qalmasına icazə verir. Bu ifadənin yerinə başqa iki ifadə də yazıla bilər. FILTER\_FLAG\_ALLOW\_THOUSAND ifadəsi vergülləri saxlamağa icazə verir. Məsələn:

```
<?php
$number="5-2f+4,3pp";
echo(filter_var($number, FILTER_SANITIZE_NUMBER_FLOAT,
FILTER_FLAG_ALLOW_THOUSAND));
```

?>

Nəticə: 5-2+4,3

FILTER\_FLAG\_ALLOW\_SCIENTIFIC ifadəsi isə ifadədə e və E simvollarını saxlamağa icazə verir. Məsələn:

```
<?php
$number="5-2f+4epp";
echo(filter_var($number, FILTER_SANITIZE_NUMBER_FLOAT,
FILTER_FLAG_ALLOW_SCIENTIFIC));
?>
```

Nəticə: 5-2+4e

10. Tam ifadədən uyğun olmayan simvolları silmək üçün:

```
<?php
$number="5-2+3.5pp";
echo(filter_var($number, FILTER_SANITIZE_NUMBER_INT));
?>
```

Nəticə: 5-2+35

11. Aşağıdakı filtrləmədə isə ifadədə olan xüsusi xarakterlər(məsələn html teqləri) adi söz kimi nəzərə alınır:

```
<?php
$url="Is Peter <smart> & funny?";
echo(filter_var($url, FILTER_SANITIZE_SPECIAL_CHARS));
?>
```

Nəticə: Is Peter <smart> & funny?

12. Aşağıdakı filtrləmədə mətndə olan html teqlər silinir:

```
<?php
$str = "<h1>Hello World!</h1>";
$newstr = filter_var($str, FILTER_SANITIZE_STRING);
echo $newstr;
?>
```

Nəticə: Hello World!

13.URL ünvandan uyğun olmayan ifadələri silmək üçün:

```
<?php
$var="http://www.w3schoolss.com";
echo(filter_var($var, FILTER_SANITIZE_URL));
?>
```

Nəticə: <http://www.w3schools.com>

2)filter\_list() funksiyası-Filter funksiyalarında dəstəklənən ifadələri özündə saxlayan massiv verir. Məsələn:

```
<?php
print_r(filter_list());
?>
```

3)filter\_id() funksiyası-Xüsusi filter adının ID nömrəsini göstərən funskiyadır. Məsələn:

```
<?php
$echo(filter_id("validate_email"));
?>
```

Nəticədə 274 çap olunur.



## Cookies

Php dilində bir məlumatı istifadəçinin brauzerinə yazaraq oradan oxumaq mümkündür. Bu əməliyyatlar cookie-lər vasitəsilə yerinə yetirilir. Belə ki, cookie yaradılarkən cookie-nin qalma vaxtı təyin olunur və həmin müddət boyunca cookie istifadəçinin brauzerində qalır. Cookie-lər ən çox saytın giriş bölməsində «yadda saxla» əməliyyatı üçün istifadə olunur. Bir şeyi də nəzərə almaq lazımdır ki, istifadəçi istədiyi vaxt brauzerdən həmin cookie-ləri silə bilər. Eyni zamanda əgər cookie A brauzerə yazılıbsa, B brauzeri ilə sayta daxil olduqda həmin cookie B brauzerində olmadığı üçün istifadə oluna bilməyəcək. Cookie-nin yaradılma qaydası aşağıdakı kimidir:

```
<?php
setcookie("cookie","value",time()+3600);
?>
```

Burada ilk parametrlər yaradacağımız cookie-nin adıdır. İkinci parametrlər cookie-nin dəyəridir. Üçüncü parametrlər isə cookie-nin brauzerdə qalacağı vaxtdır. Üçüncü parametrlər saniyə ilə təyin olunur.

Time()+3600 yazaraq cookie-nin qalma vaxtını 1 saat təyin edirik(1 saat=3600 saniyə). Əgər cookie-nin bir gün aktiv qalmasını istəyiriksə onda time()+3600\*24 yazacağımız kifayətdir. Eynilə olaraq cookie-nin qalma vaxtını daha da uzada bilərik. Yaradılmış cookie aşağıdakı qaydada çağırılır:

```
<?php
setcookie("cookie","value",time()+3600);
echo $_COOKIE['cookie'];
?>
```

Nəticədə cookie-nin dəyəri çap olunur. \$\_COOKIE[""] daxilində cookie-nin adı olmalıdır.

Əvvəlcədən yaradılmış cookie-ni silmək üçün zaman hissəsinə keçmiş saati yazmalıyıq. Beləliklə cookie-nin vaxtı dolmuş sayılacaq.

```
<?php
setcookie("cookie", "value", time() - 3600);
?>
```

Beləliklə yaradılmış cookie-nin vaxtı dolmuş sayılır. Bir cookie-nin olub olmadığını isset() funksiyası ilə yoxlaya bilərik:

```
<?php
setcookie("cookie","value",time()+3600);
if(isset($_COOKIE['cookie'])) {
```

```
echo "Cookie mövcuddur!";  
}  
?>
```

## Sessiyalar

Sessiyalar istifadəçinin brauzeri açıq saxladığı müddətcə aktiv olur və istifadə oluna bilər. Brauzer bağlandıqda sessiya itir. Sessiyanın yaradılma və çağırılma qaydası aşağıdakı kimidir:

```
<?php
session_start();
$_SESSION['name']="Php";
echo $_SESSION['name'];
?>
```

Sessiya istifadə olunan səhifənin əvvəlində mütləq `session_start()`; yazılmalıdır. Sessiyanı silmək üçün `unset()` funksiyasından istifadə olunur.

```
<?php
session_start();
$_SESSION['name']="Php";
unset($_SESSION['name']);
if(isset($_SESSION['name'])){
echo "Sessiya mövcuddur!";
}
else{
echo "Sessiya mövcud deyil!";
}
?>
```

Nəticədə sessiya boş olduğu üçün heç nə çap olunmayacaq. Sessiyanın olub-olmadığını yoxlamaq üçün `isset()` funksiyasından istifadə etməliyik:

```
<?php
session_start();
$_SESSION['name']="Php";
if(isset($_SESSION['name'])){
echo "Sessiya mövcuddur!";
}
?>
```

Sessiyalar ən əsas saytda istifadəçi girişinin proqramlaşdırılmasında istifadə olunur və ən vacib mövzulardan biridir. İrəlidəki mövzularda sessiyaların geniş istifadəsilə tanış olacağıq.

## Zaman funksiyaları

Php dilində zaman funksiyaları mövcuddur. Bu funksiyalar vasitəsilə saati, tarixi əldə etmək və s. Müxtəlif formalarda əldə etmək mümkündür. Bəzi zaman funksiyalarını işlətməzdən əvvəl `date_default_timezone_set('Asia/Baku');` funksiyasını işlətmək lazımdır. Yoxsa nəticə səhv olacaq.

Ən vacib zaman funksiyaları aşağıdakılardır:

1) `Date()` funksiyası-Bu funksiya vasitəsilə hal hazırki tarixi və saati əldə etmək mümkündür. Məsələn funksiya daxilində «d» yazmaqla hal-hazırkda ayın neçəsi olduğunu əldə edə bilərik:

```
<?php
date_default_timezone_set('Asia/Baku');
echo date("d");
?>
```

Nəticədə ayın neçəsi olduğu çap olunur. Gündən əlavə digər vaxtları da əldə etmək mümkündür. Bunun üçün funksiya daxilində nəyi əldə etmək istəyiriksə ona müvafiq hərfi yazmalıyıq. Ancaq burada böyük və kiçik hərflər fərqlənir. Bunlar aşağıdakılardır:

Hərflər	İzahı	Nümunə
d	Ayın gününü əldə etmək üçün istifadə olunur.	27
m	Hansı ayda olduğumuzu göstərir(rəqəmlə).	3
Y	Hansı ildə olduğumuzu göstərir.	2015
H	Saati göstərir.	14
i	Dəqiqəni göstərir.	21
s	Saniyəni göstərir	51
l	Həftənin gününün adını ingiliscə verir.	Saturday
F	Ayın adını ingiliscə verir.	Oct
D	Günün adını ilk 3 hərfini verir(İngiliscə)	Sat
N	Həftədəki günün nömrəsini verir. Məsələn Bazar ertəsi üçün 1, şənbə üçün 6	1.5
U	1970-ci ildən günümüzə qədər keçən vaxtı saniyə ilə verir.	1445084708

Bunlardan əlavə hərflər də mövcuddur, ancaq ən əsasları yuxarıdakı cədvəldə göstərilib. Funksiyanın müxtəlif hərflərlə işlənmə qaydası aşağıda göstərilib:

```
<?php
date_default_timezone_set('Asia/Baku');
echo date("d-m-y H:i:s");
?>
```

2)time() funksiyası-1970-ci ildən indiyə qədər keçən vaxtı saniyə ilə verir.

```
<?php
date_default_timezone_set('Asia/Baku');
echo time();
?>
```

3)getdate()-əsas zaman parametrləri daxil olan bir massiv yaradır. Bu massivdən lazımı tarixi, saati və s. Əldə edə bilərik.

```
<?php
date_default_timezone_set('Asia/Baku');
$a=getdate();
echo $a['seconds'];
?>
```

\$a['seconds'] yazaraq hal-hazırkı saniyəni əldə edirik. Uyğun sözü yazaraq digər tarixləridə əldə edə bilərik. Bu sözlər aşağıdakılardır:

Hərf	Funksiyası
seconds	Hal-hazırkı saniyəni əldə etmək üçündür.
minutes	Hal-hazırkı dəqiqəni əldə etmək üçündür.
hours	Hal-hazırkı saati əldə etmək üçündür.
mday	Ayın gününü əldə etmək üçündür.
wday	Həftənin gününü əldə etmək üçündür.
mon	Ayı əldə etmək üçün istifadə olunur.
year	İli əldə etmək üçün istifadə olunur.
yday	İlin gününü əldə etmək üçün istifadə olunur.
weekday	Həftənin günlərdə etmək üçün istifadə olunur.
month	Ayın adını əldə etmək üçün istifadə olunur.
0	1970-ci ildən günümüzə qədər olan vaxtı

	saniyə ilə verir.
--	-------------------

## String funksiyaları

Php dilində yazılar üzərində dəyişikliklər aparmaq üçün bir sıra funksiyalar mövcuddur. Bu funksiyalar aşağıdakılardır:

1)addslashes() funksiyası-Bu funksiya vasitəsilə yazıdakı dırnq işarəsinin əvvəlinə avtomatik olaraq \ işarəsi qoyulur. Bazaya məlumat göndərərkən təhlükəsizlik məqsədilə bu funksiyadan istifadə olunur. İstifadə qaydası aşağıdakı şəkildədir:

```
<?php
$str = addslashes('What does "yolo" mean?');
echo($str);
?>
```

Nəticə:

What does \"yolo\" mean?

2)bin2hex() funksiyası-Yazını 16-lıq say sisteminə çevirir. İstifadə qaydası aşağıdakı şəkildədir:

```
<?php
$str = bin2hex("Hello World!");
echo($str);
?>
```

3)chunk\_split() funksiyası-Yazıdakı xarakterləri bir neçə simvoldan bir ayırmaq üçün istifadə olunur. Məsələn:

```
<?php
$str = "Hello world!";
echo chunk_split($str,1,".");
?>
```

Nəticə:

H.e.l.l.o. .w.o.r.l.d.!

Burada hər bir simvoldan bir araya nöqtə Qoyulur. Nöqtənin əvəzinə digər simvollar da ola bilər.

4)htmlspecialchars() funksiyası-Bu funksiya vasitəsilə yazı daxilindəki html kodlar kod kimi yox sadəcə yazı kimi nəzərə alınır. Məsələn:

```
<?php
$str = "This is some <b>bold</b> text.";
echo htmlspecialchars($str);
?>
```

Nəticə:

This is some <b>bold</b> text.

5)lcfirst() funksiyası-Bu funksiya yazının ilk hərfini kiçiltmək üçün istifadə olunur. Məsələn:

```
<?php
echo lcfirst("Hello world!");
?>
```

Nəticə: hello world!

6)ltrim() funksiyası-Yazının sol tərəfindən qeyd olunmuş hissəni silmək üçün istifadə olunur. Məsələn:

```
<?php
$str = "Hello World!";
echo $str . "<br>";
echo ltrim($str,"Hello");
?>
```

Nəticə: World!

7)rtrim() funksiyası-Yazının sağ tərəfindən qeyd olunmuş hissəni silmək üçün istifadə olunur. Məsələn:

```
<?php
$str = "Hello World!";
echo $str . "<br>";
echo rtrim($str,"World!");
?>
```

Nəticə: Hello

8)md5() funksiyası-Yazını md5 şifrələmə alqoritmi ilə şifrələmək üçün istifadə olunur. Məsələn:



```
<?php
$str = "Hello";
echo md5($str);
?>
```

Nəticə: 8b1a9953c4611296a827abf8c47804d7

9)nl2br() funksiyası-\n ifadəsi olan hissədə növbəti sətirə keçir. Məsələn:

```
<?php
echo"One line.\nAnother line.";
?>
```

Nəticədə iki tərəf ayrı-ayrı sətirlərdə olur. Əsasən formalarda textarea daxilində istifadəçi yazı yazdıqda ortada enter basaraq növbəti sətirə keçsə də həmin yazılar yan-yana olur. Bu zaman yazıları yeni sətirdə etmək üçün nl2br() funksiyasından istifadə olunur.

10)sha1() funksiyası-Yazıları sha-1 alqoritmi ilə şifrələmək üçün istifadə olunur. Məsələn:

```
<?php
$str = "Hello";
echo sha1($str);
?>
```

Nəticə: f7ff9e8b7bb2e09b70935a5d785e0cc5d9d0abf0

11)similar\_text() funksiyası-İki mətdə eyni olan simvol sayını tapmaq üçün istifadə olunur. Məsələn:

```
<?php
echo similar_text("Hello World","Hello Peter");
?>
```

Nəticə: 7

Eyni zamanda nəticəni faizlə də hesablamaq olar. Bunun üçün funksiyanı aşağıdakı şəkildə istifadə etmək lazımdır:

```
<?php
similar_text("Hello World","Hello Peter", $percent);
```

```
echo $percent;  
?>
```

Nəticə: 63.636363636364

12)str\_pad() funksiyası-Yazını qeyd olunmuş simvol sayına qeyd olunmuş xarakterlə tamamlayır. Məsələn:

```
<?php  
$str = "Hello World";  
echo str_pad($str,20,".");  
?>
```

Nəticə:Hello World.....

Burada 20 simvol tamam olana qədər mətnin sonuna nöqtə yazır.

Eyni zamanda yazını tamamlarkən simvolları mətnin sol tərəfinə də yazmaq olar.

Məsələn:

```
<?php  
$str = "Hello World";  
echo str_pad($str,20,".",STR_PAD_LEFT);  
?>
```

Nəticə: .....Hello World

Eyni zamanda simvolları yazının hər iki tərəfinə də yerləşdirmək olar. Məsələn:

```
<?php  
$str = "Hello World";  
echo str_pad($str,20,".",STR_PAD_BOTH);  
?>
```

Nəticə: ...:Hello World:...:

13)str\_repeat() funksiyası-Müəyyən bir yazını bir neçə dəfə təkrar yazmaq üçün istifadə olunur. Məsələn:

```
<?php  
echo str_repeat(".",13);  
?>
```

Nəticə: .....

14)str\_replace() funksiyası-Bir yazıdakı müəyyən hissəni başqa bir yazı ilə əvəz etmək üçün istifadə olunur. Məsələn:

```
<?php
echo str_replace("world","Peter","Hello world!");
?>
```

Nəticə: Hello Peter!

Nəticə «Hello world» ifadəsində «world» sözü «Peter» sözü ilə əvəz olunur.

15)str\_shuffle() funksiyası()-Yazıdakı simvolları təsadüfi sıralamaq üçün istifadə olunur. Məsələn:

```
<?php
echo str_shuffle("Hello World");
?>
```

Burada hər dəfə müxtəlif nəticələr çıxır. Nəticə 1: HoreWl oldd Nəticə 2: d erloHWoll  
Nəticə 3: eldo HWroll

16)str\_split() funksiyası-Bu funksiya yazının simvollarında ibarət olan massiv düzəldir. Məsələn:

```
<?php
print_r(str_split("Hello"));
?>
```

Nəticə: Array ( [0] => H [1] => e [2] => l [3] => l [4] => o )

Funksiyanın başqa cür istifadəsi də mövcuddur:

```
<?php
print_r(str_split("Hello",3));
?>
```

Burada massiv elementlərində 3 simvol olur.

17)str\_word\_count() funksiyası-Cümlədəki sözlərin sayını tapmaq üçün bu funksiyadan istifadə olunur. Məsələn:

```
<?php
echo str_word_count("Hello world!");
?>
```

Nəticə: 2

18)strip\_tags() funksiyası-Mətn daxilindəki html teqlərini silmək üçün istifadə olunur. Məsələn:

```
<?php  
echo strip_tags("Hello <b>world!</b>");  
?>
```

Nəticə: Hello world!

Bu funksiyadan istifadə edərkən bəzi teqlərin işlənməsinə icazə vermək olar. Məsələn:

```
<?php  
echo strip_tags("Hello <b><i>world!</i></b>", "<b>");  
?>
```

Nəticə: Hello **world!**

19)stripslashes() funksiyası-Mətn daxilindən \ işarəsini silmək üçün istifadə olunur. Məsələn:

```
<?php  
echo stripslashes("Who\'s Peter Griffin?");  
?>
```

Nəticə: Who's Peter Griffin?

20)stripos() funksiyası-Mətn daxilində işlənmiş bir sözün ilk yerini tapmaq üçün istifadə olunur. Məsələn:

```
<?php  
echo stripos("I love php, I love php too!", "PHP");  
?>
```

Nəticə: 7

21)strlen() funksiyası-Mətn daxilindəki simvol sayını tapmaq üçün istifadə olunur. Məsələn:

```
<?php
echo strlen("Hello");
?>
```

Nəticə: 5

22)stristr() funksiyası()-Mətn daxilində müəyyən sözün olub-olmadığını yoxlamaq üçün istifadə olunur. Məsələn:

```
<?php
if(stristr("String functions","functions")){
echo "Var";
}
?>
```

Nəticədə şərt doğru olduğuna görə Var sözü çap olunur. Bir şeyi də nəzərə almaq lazımdır ki bu funksiyada böyük və kiçik hərflər fərqlənir. Yəni yuxarıda «functions» yerinə «Functions» yazsaydıq şərt doğru olmayacaqdı. Böyük və kiçik hərflərin nəzərə alınmasını istəmiriksə stristr() funksiyasından istifadə edə bilərik.

23)stristr() funksiyası-Bu funksiya da mətn də müəyyən sözün olub-olmadığını yoxlamaq üçün istifadə olunur. Stristr() funksiyasından fərqli olaraq bu funksiyada böyük və kiçik hərflərin fərqi yoxdur. Məsələn:

```
<?php
if(stristr("String functions","Functions")){
echo "Var";
}
?>
```

Burada şərt doğru olduğu üçün «Var» sözü çap olunacaq.

24)ucfirst() funksiyası-yazının ilk xarakterini böyülmək üçün istifadə olunur. Məsələn:

```
<?php
echo ucfirst("hello world!");
?>
```

Nəticə: Hello world!

25)ucwords() funksiyası-Yazıdakı hər bir sözün ilk hərfini böyülmək üçün istifadə olunur. Məsələn:

```
<?php
echo ucwords("hello world");
?>
```

Nəticə: Hello World

26)lcfirst() funksiyası-Yazının ilk xarakterini kiçiltmək üçün istifadə olunur. Məsələn:

```
<?php
echo lcfirst("Hello world!");
?>
```

Nəticə: hello world!

27)strtolower() funksiyası-Yazının bütün hərflərini kiçiltmək üçün istifadə olunur. Məsələn:

```
<?php
echo strtolower("Hello WORLD.");
?>
```

Nəticə: hello world.

## Məlumatların şifrələnməsi

Təhlükəsizlik baxımından bazaya göndərilən məlumatların şifrələnməsi vacibdir. Php dilində şifrələmə üçün bir neçə funksiya mövcuddur.

1)md5() funksiyası-Bu şifrələmə funksiyasını istifadə etdikdə 32 simvollaq bir şifrə əldə olunur. Bu şifrəni yenidən açmaq üçün bir funksiya mövcud deyil. Nümunə:

```
<?php
$x = 123456;
echo md5($x);
?>
```

Nəticə: e10adc3949ba59abbe56e057f20f883e

2)sha1() funksiyası-Bu funksiya ilə şifrələmə nəticəsində 40 simvollaq şifrə əldə olunur. Bu şifrəni yenidən açmaq üçün bir funksiya mövcud deyil. Nümunə:

```
<?php
$x = 123456;
echo sha1($x);
?>
```

Nəticə: 7c4a8d09ca3762af61e59520943dc26494f8941b

3)crc32() funksiyası-Bu şifrələmə metodu ilə məlumat şifrələnərək bir tam ədəd əldə olunur.

Bu şifrəni yenidən açmaq üçün bir funksiya mövcud deyil. Nümunə:

```
<?php
$x = 123456;
echo crc32($x);
?>
```

Nəticə: 158520161

4)Base64 şifrələmə alqoritmində digərlərindən fərqli olaraq şifrələnmiş məlumatı yenidən deşifrə etmək üçün funksiya mövcuddur. Şifrələmək üçün base64\_encode() funksiyasından istifadə olunur. Şifrələnmiş məlumat deşifrə etmək üçün isə base64\_decode() funksiyasından istifadə olunur. Nümunə:

```
<?php
$x = 123456;
$a = base64_encode($x);
$b= base64_decode($a);
```

```
echo 'Şifrələnmiş məlumat:'. $a.'<br />';  
echo 'Deşifrə olunmuş məlumat:'. $b;  
?>
```

Burada \$a dəyişəni şifrələnmiş qiymətə bərabər olur, \$b dəyişənində isə məlumat yenidən deşifrə olunur. Tam təhlükəsizlik üçün yuxarıdakı funksiyalardan bir neçə dəfə istifadə oluna bilər. Məsələn bir qiyməti 3 dəfə md5() funksiyası ilə, iki dəfə isə sha1() funksiyası ilə şifrələmək olar. Məsələn:

```
<?php  
$a="Php";  
$b=sha1(sha1(md5(md5(md5($a)))));  
echo $b;  
?>
```

Artıq burada məlumat mürəkkəb şəkildə şifrələnir.



## Php dilində \$\_SERVER massivi

\$\_SERVER massivi serverə dair məlumatları özündə saxlayan bir massivdir. Bu massivdəki məlumatlar server tərəfindən emal olunur. Massivdəki məlumatlar aşağıdakılardır:

1) \$\_SERVER['PHP\_SELF'] — Olduğumuz səhifənin yolunu göstərir. Məsələn:

```
<?php  
echo $_SERVER['PHP_SELF'];  
?>
```

2) \$\_SERVER['GATEWAY\_INTERFACE'] - Serverin istifadə etdiyi CGI versiyasını göstərir.

3) \$\_SERVER['SERVER\_ADDR'] - Sayta aid IP adresini göstərir.

4) \$\_SERVER['SERVER\_NAME'] - Saytın domenini göstərir.

5) \$\_SERVER['SERVER\_PROTOCOL'] — Serverin http protokolunu yoxlayır.

6) \$\_SERVER['HTTP\_ACCEPT\_CHARSET'] — Saytın xarakter dilini göstərir.

7) \$\_SERVER['HTTP\_ACCEPT\_LANGUAGE'] — istifadəçinin brauzerinin dilini yoxlayır.

8) \$\_SERVER['REMOTE\_ADDR'] — İstifadəçinin IP adresini yoxlayır.

9) \$\_SERVER['QUERY\_STRING'] - Səhifə adından sonra gələn məlumatları göstərir.

## Fayl funksiyaları

Php dilində fayllar yaratmaq və onlar üzərində əməliyyatlar aparmaq mümkündür. Əvvəlcə faylların yaradılmasına baxaq. Php dilində fayl yaratmaq üçün touch() funksiyasından istifadə olunur. Aşağıda faylın yaradılması göstərilir:

```
<?php  
touch("fayl.txt");  
?>
```

Bu kod vasitəsilə fayl adlı mətn faylı yaradılır. Yaradılmış bu faylı unlink() funksiyası ilə silə bilərik:

```
<?php  
unlink("fayl.txt");  
?>
```

Nəticədə fayl silinir. Yaradılmış faylı açaraq həmin fayla məlumat yazmağa bilərik. Bu zaman fopen() funksiyasından istifadə olunur.

```
<?php  
$fp=fopen("1.txt","w");  
fwrite($fp,"Php");  
fclose($fp);  
?>
```

Fopen() funksiyası ilə faylı açırıq. Fwrite() funksiyası ilə fayla yazı yazdıqdan sonra isə fclose() funksiyası ilə fayl əməliyyatını bağlarıq. Fayla yazı yazarkən yazını növbəti sətirə keçirmək üçün \n ifadəsini yazmağa bilərik. \t ifadəsini yazaraq yazılar arasında 8 xarakter məsafə qoya bilərik. Fopen() funksiyasında ikinci argument olaraq w yazdıq. w Faylı yazmaq üçün açır. Faylı digər əməliyyatlar üçün də açmağa bilərik. Bunun üçün w əvəzinə digər müvafiq simvolları yazmaq lazımdır.

Simvol	Yerinə yetirdiyi funksiya
r	Faylı oxumaq üçün istifadə olunur.
r+	Faylı həm açmaq həm də fayla məlumat yazmaq üçün istifadə olunur.
w	Fayla məlumat yazmaq üçün istifadə olunur. Yeni məlumat yazılarkən fayldakı köhnə məlumatlar silinir.
w+	Faylı həm oxumaq həm də yazmaq üçün açır. Fayldakı köhnə məlumatlar silinmir.
a	Fayla məlumat yazmaq üçün istifadə olunur. Bu zaman fayldakı köhnə

	məlumatlar silinir.
a+	Fayla həm məlumat yazmaq həm də oxumaq üçün istifadə olunur. Məlumat yazılarkən fayldakı köhnə məlumatlar silinmir
x	Faylı yaradır və oxumaq üçün açır. Əgər fayl mövcuddursa false dəyərini qaytarır.
x+	Faylı yaradır, oxumaq və yazmaq üçün istifadə olunur. Əgər fayl mövcuddursa false dəyərini qaytarır.

Bir çox fayl funksiyaları mövcuddur. Bunlar aşağıdakılardır:

1)copy() funksiyası-Bir faylın tərkibini başqa fayla kompyalamaq üçün istifadə olunur. Məsələn:

```
<?php
copy("1.html","1.php");
?>
```

Burada 1.html faylında olan məlumat 1.php faylına kopyalanır.

2)file() funksiyası-Bu funksiya faylı sətir-sətir massivə yazır. Məsələn:

```
<?php
$massiv=array();
$massiv=file("1.txt");
print_r($massiv);
?>
```

3)file\_exists() funksiyası-Faylın mövcudluğunu yoxlamaq üçün istifadə olunur. Məsələn:

```
<?php
if(file_exists("1.php")){
echo "Fayl mövcuddur!";
}
?>
```

4)file\_get\_contents() funksiyası-Fayldakı məlumatı əldə etmək üçün istifadə olunur. Məsələn:

```
<?php
echo file_get_contents("1.txt");
?>
```

5)file\_put\_contents() funksiyası-Fayla məlumat yazmaq üçün istifadə olunur. Bu funksiya ilə fayla məlumat yazılarkən fayldakı köhnə məlumatlar silinir. Məsələn:

```
<?php
file_put_contents("1.html","Information");
?>
```

6)filesize() funksiyası-Faylın ölçüsünü göstərir. Məsələn:

```
<?php
echo filesize("test.txt");
?>
```

Nəticəni byte olaraq göstərir.

7)is\_dir() funksiyası-Bir qovluğun mövcud olub-olmadığını yoxlamaq üçün istifadə olunur. Məsələn:

```
<?php
if(is_dir("build")){
echo "Qovluq var!";
}
else{
echo "Mövcud deyil!";
}
?>
```

8)is\_readable() funksiyası-Faylı oxumağın mümkünlüyünü yoxlamaq üçün istifadə olunur. Məsələn:

```
<?php
if(is_readable("1.html")){
echo "Oxuna bilər!";
}
else{
echo "Oxunması mümkün deyil!";
}
?>
```

9)is\_writable()funksiyası-Faylın yazıla biləcəyini yoxlamaq üçün istifadə olunur. Məsələn:

```
<?php
```

```
if(is_writeable("test.txt")){
echo "Yazıla bilər!";
}
else{
echo "Yazılması mümkün deyil!";
}
?>
```

10)is\_executable() funksiyası-Faylın işləyə biləcəyin yoxlamaq üçün istifadə olunur. Məsələn:

```
<?php
if(is_executable("end.exe")){
echo "İşləyə bilər!";
}
else{
echo "İşləyə bilməz!";
}
?>
```

11)mkdir() funksiyası-Yeni qovluq yaratmaq üçün istifadə olunur. Məsələn:

```
<?php
mkdir("Qovluq");
?>
```

Nəticədə yeni qovluq yaranır.

12)rename() funksiyası-Bu funksiya faylın adını dəyişmək üçün istifadə olunur. Məsələn:

```
<?php
rename("1.txt","yeni.txt");
?>
```

Nəticədə 1.txt adlı faylın adı dəyişdirilərək yeni.txt olur.

13)stat() funksiyası-Bu funksiya ilə fayla aid məlumatlardan ibarət massiv yaradılır. Məsələn:

```
<?php
$stat=stat("1.php");
echo $stat['size'];
```

?>

Burada faylın ölçüsü göstərilir. Bu funksiya vasitəsilə ölçüdən əlavə fayla aid digər xüsusiyyətlərə də baxmaq olur. Burada size sözünün əvəzinə uyğun sözü yazaraq digər bir xüsusiyyətə baxmaq olar. Bu sözlər aşağıdakılardır:

- dev
- ino
- mode
- nlink
- uid
- gid
- rdev
- size
- atime
- mtime
- ctime
- blksize
- blocks

## Php fayl upload əməliyyatı

Php vasitəsilə faylı serverə yükləmək mümkündür. Bu əməliyyat üçün form aşağıdakı şəkildə yaradılmalıdır:

```
<form method="POST" enctype="multipart/form-data">
<input type="file" name="file" />
<button type="submit" name="submit">Yüklə</button>
</form>
```

Bu formda enctype="multipart/form-data" yazılmağı vacibdir. Fayl yükləmə əməliyyatı üçün bir çox hazır funksiyalar var. Bunlar aşağıdakılardır:

- 1) \$\_FILES['fayl']['name']-Faylın adını göstərir.
- 2) \$\_FILES['fayl']['type']-Faylın MIME tipini göstərir.
- 3) \$\_FILES['fayl']['size']-Faylın baytlarla ölçüsünü göstərir.
- 4) \$\_FILES['fayl']['tmp\_name']-Serverdə saxlanacaq faylın müvəqqəti adını göstərir.
- 5) \$\_FILES['fayl']['error']-Fayl yükləmə zamanı alınan xəta kodudur.

move\_uploaded\_file() funksiyası faylı serverə daşıyır. İndi isə bu funksiyalardan istifadə etməklə yükləmə əməliyyatına baxaq:

```
<form method="POST" enctype="multipart/form-data">
<input type="file" name="file" />
<button type="submit" name="submit">Yüklə</button>
</form>
<?php
if(isset($_POST['submit'])) {
$file = $_FILES['file'];
$qovluq = "images/";
$upload_file = $qovluq.$file['name'];
if(move_uploaded_file($file['tmp_name'], $upload_file)){
echo "<h3>Şəkil əlavə olundu!</h3>";
}
else {
echo "<h3>Şəkil əlavə olunmadı!</h3>";
}
}
?>
```

Burada şəkil images qovluğuna yüklənir. Ancaq burada heç bir yoxlama aparılmır. Biz şəklın ölçüsünü, şəklın tipini yoxlayaraq şəkli serverə əlavə edə bilərik:

```
<form method="POST" enctype="multipart/form-data">
<input type="file" name="file" />
```

```

<button type="submit" name="submit">Yüklə</button>
</form>
<?php
if(isset($_POST['submit'])) {
$file = $_FILES['file'];
$uzantilar = array("jpg", "png", "images/jpeg", "images/png");
$dizin="images/";
$upload_file = $dizin.basename($file['name']);
$size = $file['size'];
$uzanti = explode(".", $file['name']);
$uzanti = $uzanti[count($uzanti)-1];
$tip = $file['type'];
if($file['name'] != "") {
if(in_array($tip, $uzantilar) || in_array($uzanti, $uzantilar))
if($size < (1024*1024*3)) {
if(move_uploaded_file($file['tmp_name'], $upload_file)) {
echo "<h3>Şəkil əlavə olundu!</h3>"; // olumlu
}
} else {
echo "<h3>Şəkil əlavə olunmadı!</h3>"; // hata
}
} else {
echo "<h3>Şəkilin ölçüsü 3MB-dan çox olmamalıdır.</h3>"; // hata
}
} else {
echo "<h3>Yalnız Jpg və Png formatlar qəbul edilir.</h3>"; // hata
}
}
?>

```

Burada yalnız png və jpg formatında olan şəkillər əlavə oluna bilər və eyni zamanda şəkilin ölçüsü maksimum 3MB ola bilər.



## Php ilə şəkil hazırlamaq

Php vasitəsilə şəkillər hazırlamaq mümkündür. Php ilə şəkil yaratmaq əsasən saytlarda olan təsdiləmə kodunu hazırlayarkən və başqa yerlərdə istifadə olunur. Şəkil hazırlamaq üçün hazır funksiyalar yaradılmışdır. Bu funksiyalarla tanış olaq:

`ImageCreate()` funksiyası-Bu funksiya şəkli yaratmaq üçündür. İki parametr daxil edilməlidir. Birinci parametr olaraq şəklın eni ikinci parametr olaraq isə şəklın uzunluğu daxil edilir.

`ImageColorAllocate()` funksiyası-Bu funksiya rəngi seçmək üçündür. Dörd parametr qeyd olunmalıdır. Birinci parametr olaraq `ImageCreate()` funksiyası ilə yaradılan şəklın dəyişəni ;İkinci parametr olaraq qırmızı rəngin dəyəri; Üçüncü parametr olaraq yaşıl rəngin dəyəri; dördüncü parametr olaraq isə göy rəng dəyəri daxil olunmalıdır. Bu üç rəng vasitəsilə bütün rənglər əldə oluna bilər. Dəyərlər 0-255 arası olur.

`ImageString()` funksiyası-Bu funksiya şəklın üzərinə yazı yazmaq üçündür. Altı parametr daxil edilməlidir. Birinci parametr `ImageCreate()` ilə yaradılan şəklın dəyişəni;İkinci parametr olaraq yazının böyüklüyü;üçüncü parametr olaraq yazının x koordinatı;dördüncü parametr olaraq yazının y koordinatı;beşinci parametr olaraq yazılacaq yazı;altıncı parametr olaraq isə `ImageColorAllocate()` ilə yaradılan rəngin dəyəri daxil edilməlidir.

`ImageJpeg()` funksiyası-Hazırlanan şəklın Jpeg formada olması üçündür. Birinci parametr olaraq `ImageCreate()` ilə yaradılan şəklın dəyişəni; İkinci parametr olaraq əgər şəkil yaddaşda saxlanacaqsa saxlanılacaq şəklın adı; üçüncü parametr olaraq isə şəklın keyfiyyəti daxil edilməlidir.

`ImageGif()` funksiyası-Hazırlanan şəklın gif olması üçündür. İlk parametr olaraq `ImageCreate()` ilə yaradılan şəklın dəyişəni; ikinci parametr olaraq isə əgər şəkil yadda saxlanacaqsa saxlanılacaq şəklın adı daxil edilməlidir.

`ImagePng()` funksiyası-Hazırlanan şəklın png formatda olması üçündür. Dörd parametr daxil oluna bilər. Birinci parametr olaraq `ImageCreate()` ilə yaradılan şəklın dəyişəni İkinci parametr olaraq əgər şəkil yadda saxlanılacaqsa saxlanılacaq şəklın adı üçüncü parametr olaraq şəklın keyfiyyəti dördüncü parametr olaraq isə istifadə olunan filtr. Bunlardan birincisi mütləq daxil edilməlidir.

`ImageDestroy()` funksiyası-Yaddaşı təmizləyir. Faylın sonuna əlavə edilməlidir.

İndi isə bu funksiyaları birləşdirərək şəklın hazırlanmasına baxaq:

```
<?php  
header ("Content-type: image/jpeg");
```

```
$image= ImageCreate (350,350);  
$arxaplan=ImageColorAllocate($image , 30 , 144 , 255);  
$white= ImageColorAllocate ($image , 255 , 255 , 255);  
ImageString($image, 30 , 40 , 160 , "Image", $white);  
ImageJpeg($image,NULL,100);  
ImageDestroy($image);  
?>
```

Yuxarıda göstərilən funksiyalarla şəkli hazırladıq. İlk sətirdə header funksiyasını işlədərək səhifənin şəkil olduğunu bildirdik. İndi isə bu kodları yazdığımız səhifəni image.php adı ilə yadda saxlayaq və başqa bir səhifədə bu şəkli işlədək:

```

```

Bu şəkildə hazırladığımız şəkli səhifədə istifadə edə bilərik.

## Mail funksiyaları

Php dilində mail göndərmək üçün funksiyalar mövcuddur. Mail() funksiyası vasitəsilə hər hansı bir emailə məktub göndərə bilərik. İstifadə qaydası aşağıdakı şəkildədir:

```
<?php
mail("Email","Subject","Message");
?>
```

Eyni zamanda funksiyanı aşağıdakı şəkildə də işlətmək olar:

```
<?php
$to = "somebody@example.com";
$subject = "My subject";
$txt = "Hello world!";
$headers = "From: webmaster@example.com" . "\r\n" . "CC:
somebodyelse@example.com";
mail($to,$subject,$txt,$headers);
?>
```

Eyni zamanda məktub daxilinə html kodları da yazıla bilər. Məsələn:

```
<?php
$to = "somebody@example.com, somebodyelse@example.com";
$subject = "HTML email";

$message = "
<html>
<head>
<title>HTML email</title>
</head>
<body>
<p>This email contains HTML Tags!</p>
<table>
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>
<tr>
<td>John</td>
<td>Doe</td>
</tr>
</table>
```

```
</body>
</html>
";
$headers = "MIME-Version: 1.0" . "\r\n";
$headers .= "Content-type:text/html;charset=UTF-8" . "\r\n";
$headers .= 'From: <webmaster@example.com>' . "\r\n";
$headers .= 'Cc: myboss@example.com' . "\r\n";
mail($to,$subject,$message,$headers);
?>
```

## HTTP funksiyaları

HTTP funksiyaları aşağıdakılardır:

1)header() funksiyası- Səhifəni başqa bir səhifəyə yönləndirmək və digər əməliyyatlar üçün bu funksiyadan istifadə olunur. İstifadə qaydası aşağıdakı şəkildədir:

```
<?php
header("Location: 1.php");
?>
```

Bu zaman səhifə avtomatik olaraq 1.php səhifəsinə yönləndirilir. Eyni zamanda səhifəni müəyyən bir vaxtdan sonra da yönləndirə bilərik. Məsələn:

```
<?php
echo "Php";
header("refresh:2");
?>
```

Burada hər iki saniyədən bir səhifə yenilənəcək. Eyni zamanda müəyyən bir vaxtdan sonra da səhifəni başqa bir səhifəyə yönləndirə bilərik. Məsələn:

```
<?php
header("Refresh:2; url=page2.php");
?>
```

Burada 2 saniyədən sonra səhifə göstərilən səhifəyə yönləndirilir. Eyni zamanda header() funksiyası ilə səhifə tərkibini də müəyyən etmək olar. Məsələn:

```
<?php
header("Content-type: text/plain; charset=utf-8");
?>
```

Funksiya yuxarıdakı formada istifadə oluna bilər.

2)headers\_list() funksiyası-Səhifədə istifadə olunan başlıqları massiv şəkildə göstərir. Məsələn:

```
<?php
header("Content-type: text/plain; charset=utf-8");
print_r(headers_list());
?>
```

Notice:

Array

```
(  
    [0] => X-Powered-By: PHP/5.2.17  
    [1] => Content-type: text/plain; charset=utf-8  
)
```

## Include və require funksiyaları

Bir səhifəyə başqa səhifəni çağırmaq üçün include və require funksiyalarından istifadə olunur. Bu iki funksiya eyni işi görsə də müəyyən fərqləri var. İlk öncə include funksiyasına baxaq. Aşağıdakı səhifə 1.php səhifəsidir.

```
<?php
$a=5;
echo "1.php";
?>
```

1.php səhifəsini 2.php səhifəsinə çağırmaq:

```
<?php
include("1.php");
echo $a;
?>
```

Göründüyü kimi 1.php səhifəsi 2.php səhifəsinə çağırılır. Include funksiyasında çağırılan faylın olub-olmaması vacib deyil. Əgər həmin fayl olmasa belə kodlar növbəti sətirdən işləməyə davam edəcək.

Require funksiyasının da istifadə include funksiyası ilə eynidir:

```
<?php
require("1.php");
echo $a;
?>
```

Require funksiyasını işlədərkən əgər çağırılan fayl mövcud deyilsə bu funksiya digər kodların da işləməyinə icazə verməyəcək. Bu funksiyalara aid daha bir nümunə göstərək. Məsələn functions.php səhifəsində funksiyalarımız var:

```
<?php
function toplama($a,$b){
return $a+$b;
}
?>
```

İndi isə 1.php səhifəsinə functions.php səhifəsini çağırmaq və toplama funksiyasını istifadə edək:

```
<?php
include("functions.php");
echo toplama(4,5);
```

?>

functions.php səhifəsini 1.php səhifəsinə çağırdıq və oradakı funksiyanı istifadə etdik.



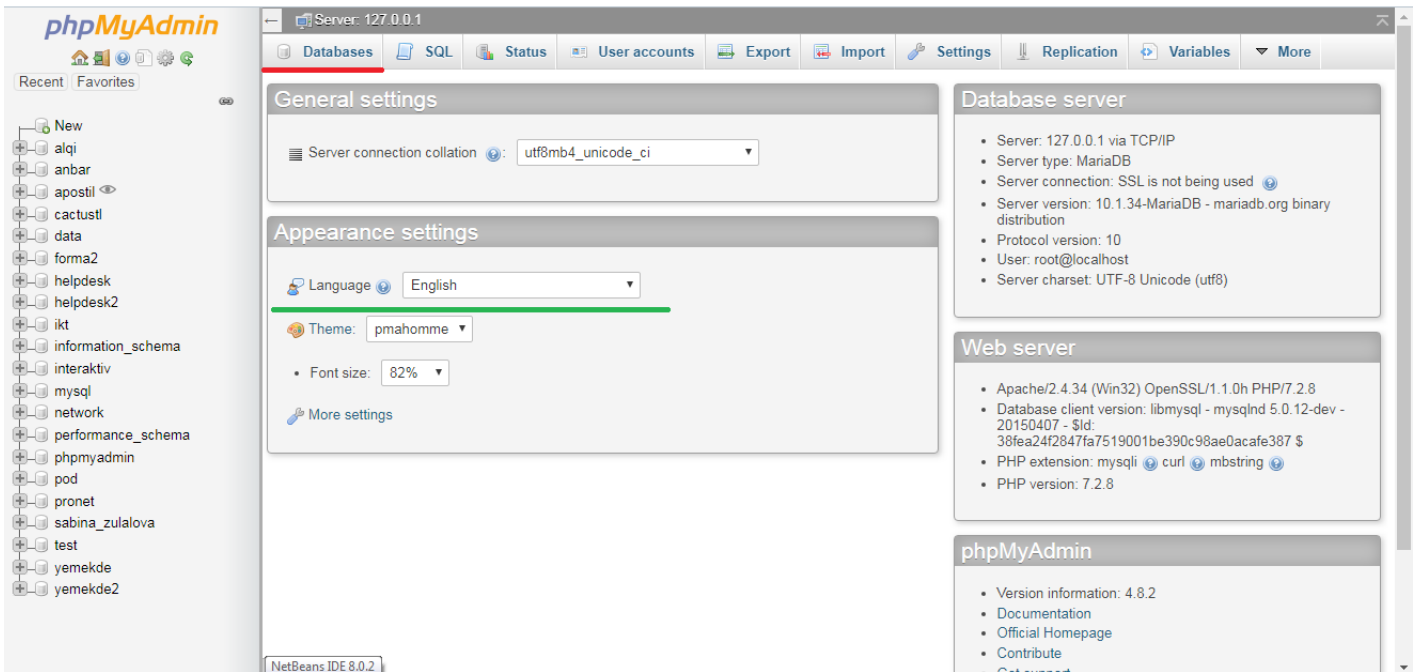
## Phpmyadmin və MySql

Sayt daxilindəki məlumatları, məsələn, xəbərləri, istifadəçi məlumatlarını və digər məlumatları yadda saxlamaq üçün verilənlər bazasına ehtiyac duyulur. Bunlara Oracle, Access, MySql və s. sistemləri misal göstərmək olar. Php ilə ən çox istifadə olunan verilənlər bazası sistemlərindən biri də MySql sistemidir.

İlk olaraq verilənlər bazasında lazım olan cədvəllər, əlaqələr, struktur yaradılır, lazım olan məlumatlar bazaya yerləşdirilir. Sonra isə proqramlaşdırma dilindəki funksiyalar vasitəsilə baza arasında əlaqə yaradılır və bazaya məlumatlar əlavə olunur və ya silinir.

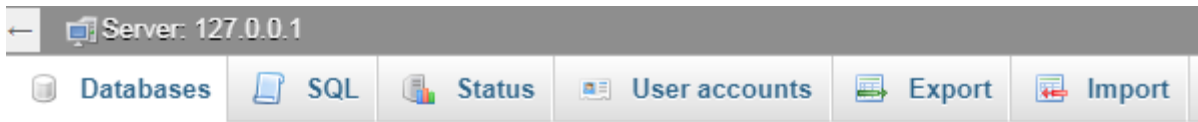
Php və baza arasında əlaqə yaradılmasını öyrənməzdən əvvəl, bazanın özünün idarə edilməsini öyrənməyimiz lazımdır. MySql bazasını brauzer üzərindən idarə etmək üçün yaradılmış sistemlərdən biri Phpmyadmin sistemidir. Phpmyadmin vasitəsilə biz bazamızı formalaşdırma bilərik.

Phpmyadminə daxil olmaq üçün serverimizi başlatdıqdan sonra brauzerdə ünvan hissəsinə “localhost/phpmyadmin” yazaraq phpmyadmin sisteminə daxil oluruq. Əgər istifadəçi adı və şifrə tələb olunarsa, login hissəsinə root yazırıq, şifrəni isə boş saxlayıb sistemə daxil oluruq.



Phpmyadmində məlumatlarımız cədvəllər daxilində saxlanılır. Cədvəlləri yaratmadan öncə mütləq bazanı yaratmalıyıq. Çünki, cədvələrimiz baza daxilində yerləşir. Bunun üçün, altından qırmızı xətt çəkilmiş “Databases” yazısına daxil oluruq. Əgər sistem rusca açılsa, o zaman, aşağıdan “Language” hissəsindən sistemin dilini dəyişə bilərsiniz. Bu hissə isə şəkildə yaşıl xəttlə göstərilmişdir.

“Databases” bölməsinə daxil olduqdan sonra aşağıdakı hissə açılacaq.



## Databases

**Create database**

Database name:  Collation:

**Filters**

Containing the word:

Database	Collation	Action
<input type="checkbox"/> alqi	latin1_swedish_ci	Check privileges
<input type="checkbox"/> anbar	latin1_swedish_ci	Check privileges

Burada “Database name” hissəsinə bir ad yazıb “Create” düyməsinə basırıq və yeni bazamız yaralmış olur. Yaratdıqdan sonra bazanın içərisinə daxil olmuş oluruq. Aşağı hissədə və sol tərəfdə yaradılmış bazaların adı göstərilir. Göstərilən siyahıdan da yaratdığımız bazanın adının üzərinə klikləyərək həmin bazaya daxil olmuş oluruq.

Server: 127.0.0.1 » Database: example

Structure SQL Search Query Export Import Operations

No tables found in database.

**Create table**

Name:  Number of columns:

Növbəti olaraq yuxarıdakı pəncərə açılmış olur. Yeni bir cədvəli yaratmaq üçün Name hissəsinin qarşısına cədvəlimizin adını yazırıq. Cədvəlimizin adı “users” olsun. Bu cədvəlimizdə istidadəçi məlumatları saxlanılacaq. Number of Columns sütununun qarşısına isə 3 yazırıq. Bu isə cədvəlimizdəki sütun sayını göstərəcək. Cədvəlimizdə yalnız login, parol və email olacağına görə 3 sütun yaratmağımız kifayətdir. Bundan sonra sağdakı Go düyməsinə basırıq. Növbəti olaraq aşağıdakı pəncərə açılır.

Server: 127.0.0.1 » Database: example » Table: users

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Table name: users Add 1 column(s) Go

Name	Type	Length/Values	Default	Collation	Attributes	Null	Index
login	VARCHAR	255	None			<input type="checkbox"/>	---
parol	VARCHAR	255	None			<input type="checkbox"/>	---
email	VARCHAR	255	None			<input type="checkbox"/>	---

Table comments: Collation: Storage Engine: InnoDB

PARTITION definition:

Partition by: ( Expression or column list )

Partitions:

Preview SQL Save

Buradakı bölmələri yuxarıdakı şəkildə dolduruq. Burada Name sütununa adları yazırıq. Type sütununda isə varchar seçirik. Varchar yazı tipidir, maksimum 255 simvol ala bilər. Üçüncü sütunda isə 255 yazırıq. Sonra isə Save düyməsinə basırıq və cədvəlimizi yaratmış oluruq.

Server: 127.0.0.1 » Database: example » Table: users

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Table structure Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	login	varchar(255)	latin1_swedish_ci		No	None			Change Drop More
2	parol	varchar(255)	latin1_swedish_ci		No	None			Change Drop More
3	email	varchar(255)	latin1_swedish_ci		No	None			Change Drop More

Check all With selected: Browse Change Drop Primary Unique Index Fulltext Add to central columns Remove from central columns

Cədvəlimizin strukturu və yerinə yetirilə biləcək əməliyyatlar şəkildəki kimi göstərilir. Növbəti olaraq cədvələ məlumat əlavə etmək üçün altından xətt çəkilərək göstərilmiş “insert” düyməsinə klik edirik.

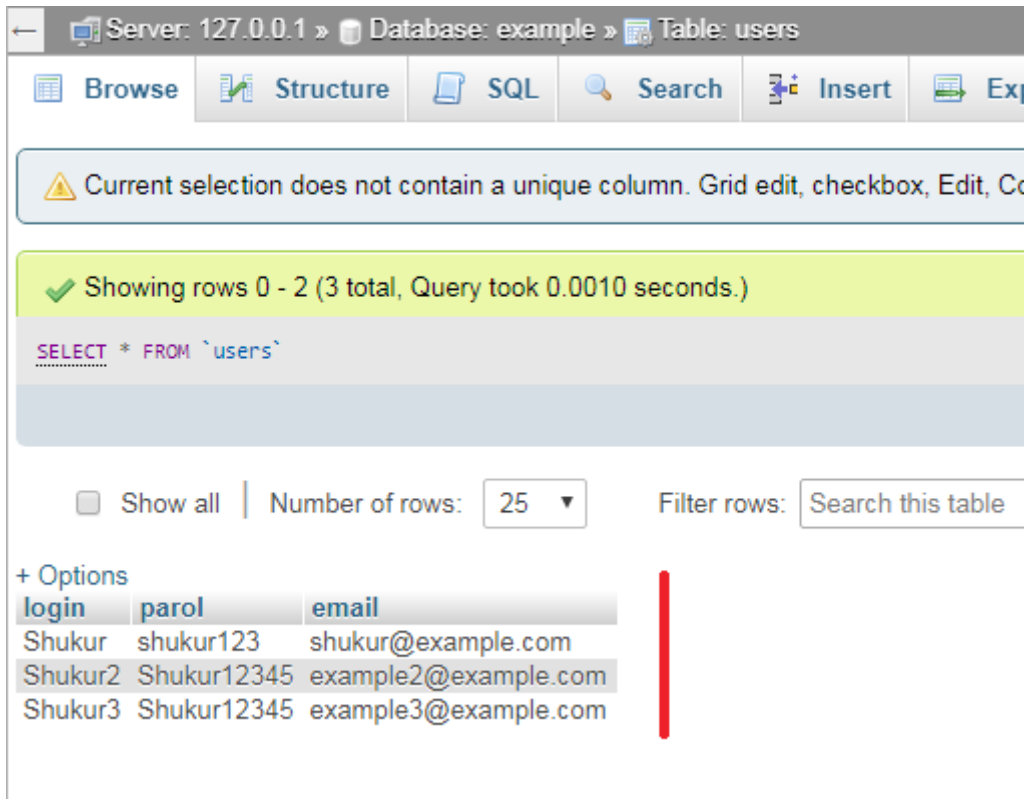
Server: 127.0.0.1 » Database: example » Table: users

Browse Structure SQL Search Insert Export Import Privileges

Column	Type	Function	Null	Value
login	varchar(255)			Shukur
parol	varchar(255)			shukur123
email	varchar(255)			shukur@example.com

Go

Sonra isə buradakı məlumatları doldurub Go düyməsinə basırıq. Bu əməliyyatı bir neçə dəfə təkrar edib müxtəlif istifadəçilər yaradıırıq. Sonda isə məlumatlara baxmaq üçün qırmızı xəttlə göstərilmiş “Browse” düyməsinə klik edirik.

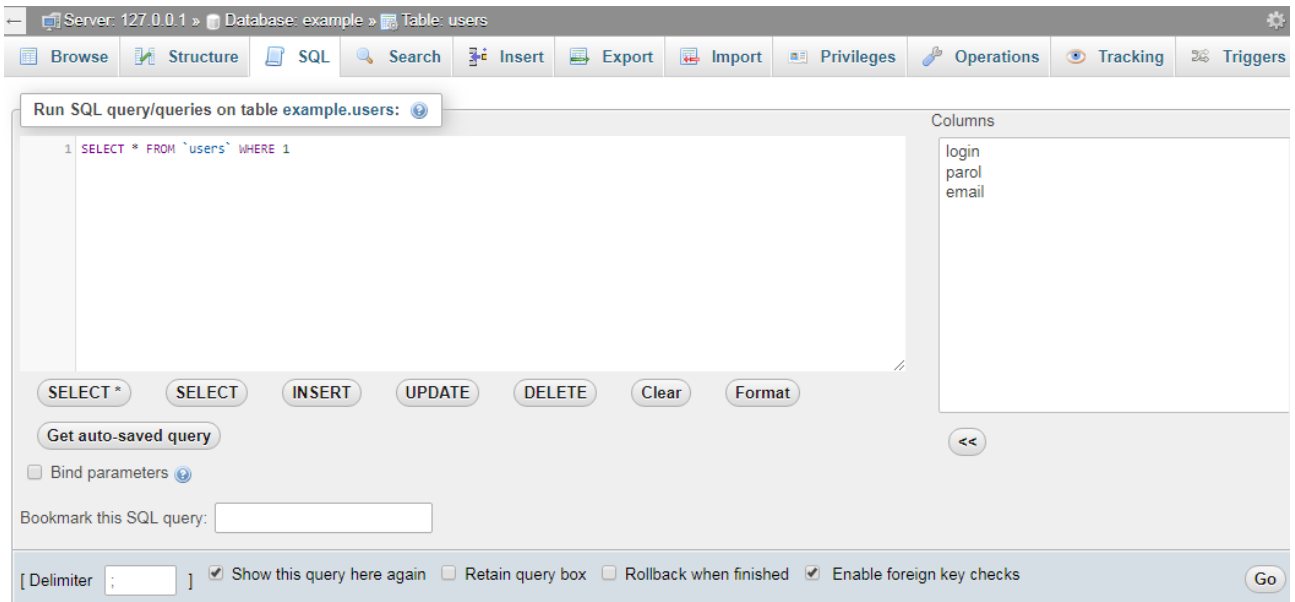


The screenshot shows a database management interface for a table named 'users'. The interface includes a toolbar with buttons for 'Browse', 'Structure', 'SQL', 'Search', 'Insert', and 'Export'. A warning message states: 'Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy, Paste, and Delete are disabled.' Below this, a green status bar indicates 'Showing rows 0 - 2 (3 total, Query took 0.0010 seconds.)'. The SQL query 'SELECT \* FROM `users`' is displayed. Below the query, there are controls for 'Show all', 'Number of rows: 25', and 'Filter rows: Search this table'. A red vertical line is positioned to the right of the data table. The data table has three columns: 'login', 'parol', and 'email'. The data rows are:

login	parol	email
Shukur	shukur123	shukur@example.com
Shukur2	Shukur12345	example2@example.com
Shukur3	Shukur12345	example3@example.com

Burada artıq məlumatlarımıza baxa bilirik. Cədvəlimizdəki məlumatlar bu şəkildə saxlanılır. Sütun adları cədvəlin yuxarisında yazılır. Növbəti hər sətir isə bir istifadəçiyə aid olur. Birinci sətirdəki məlumatlar ancaq bir istifadəçiyə aid olur. Digər sətirlərin isə hər biri bir istifadəçiyə aid olur. Bu məlumatları biz nümunə üçün özümüz bazada əlavə etdik. Ancaq, saytda bu məlumatlar istifadəçi tərəfindən formlara doldurulacaq və bizim yazacağımız kodlar vasitəsilə həmin məlumatlar bu cədvələ əlavə olunacaq.

Mysql sistemindən özünəməxsus dili mövcuddur ki, bu dil sql adlanır. Bu dillə müxtəlif sorğular yazılaraq müxtəlif əməliyyatlar yerinə yetirilir. Yazacağımız sorğularla bazada bir çox əməliyyatlar yerinə yetirə bilərik. Php ilə də bazaya qoşularkən sql sorğuları yazmalıyıq. Bu səbəbdən də, ən azı başlanğıc səvviyədə sql dilinə bələd olmalıyıq. Sql kodlarını yazmaq üçün yuxarı menyuda yerləşən “Sql” bölməsinə daxil oluruq və aşağıdakı pəncərə açılır.



Buradakı mətn sahəsinə kodlarımızı yazaraq müxtəlif əməliyyatları yerinə yetiririk. Sql daxilində ən əsas dörd operatoru öyrənməyimiz mütləqdir. Bu operatorlar select, delete, insert və update operatorlarıdır ki, bu operatorlarla məlumatları seçə, silə, əlavə edə və dəyişə bilərik.

İlk öncə **select** operatoruna baxaq. Bu operator vasitəsilə məlumatları seçə bilərik. Operatorun işlənmə qaydası aşağıdakı şəkildədir:

### **select \* from users**

Bu kodu yazıb Go düyməsinə basdıqda, cədvəldəki bütün məlumatlar seçilmiş olur. Burada users cədvəlin adını bildirir. Ulduz simvolu isə onu göstərir ki, cədvəldəki bütün sütunlar seçiləcək. Ulduz yerinə sütunların adını yazaraq yalnızca xüsusi sütunları da seçə bilərik.

### **select login, parol from users**

Beləliklə yalnız login və parol sütunları seçilmiş olur. Bu operator çox vacib operatorlardandır və geniş istifadə olunur. İrəlidə bu operatorun saytda istifadə olunmasına da baxacağıq.

Növbəti öyrənəcəyimiz operator **delete** operatorudur. Adından da göründüyü kimi bu operator cədvəldəki məlumatları silmək üçün istifadə olunur. Operatorun istifadə qaydası aşağıdakı şəkildədir:

### **delete from users**

Bu kodu sql bölməsində yazıb işlətdikdə cədvəlimizdəki məlumatlar silinəcək. Diqqət edək ki, burada **delete** ifadəsindən sonra ulduz simvolu qoyulmur.

İndi isə **insert** operatoruna baxaq. Bu operator cədvələ müxtəlif məlumatları əlavə etmək üçündür. İstifadə qaydası aşağıdakı şəkildədir.

```
INSERT INTO users(login, parol, email) VALUES ('username','parol','email@email.com')
```

Gördüyümüz kimi, burada birinci mətərizədən əvvəl cədvəlin adı, mətərizə daxilində isə sütunların adı yazılır. Növbəti olaraq “VALUES” açar sözündən sonrakı ikinci mətərizə daxilində birinci mətərizədəki sütunlara uyğun olaraq əlavə olunacaq məlumatlar dırnaq arasında yazılır. Bütün sütunlara əlavə etmədən də məlumatları əlavə edə bilərik. Bu zaman sətir daxilində həmin sütunlar boş qalacaq.

```
INSERT INTO users(login, parol) VALUES ('username','parol')
```

Burada email daxil etmədən yalnız login və parolu əlavə etmiş oluruq. Məlumatların əlavə olunduğu sətir üçün email sütunu boş olaraq saxlanılır.

Növbəti olaraq isə **update** operatoruna baxaq. Bu operator məlumatlarda dəyişiklik etmək üçün istifadə olunur. Operatorun istifadə qaydasına baxaq.

```
update users set email="yeni@yeni.com", parol="yeni_parol"
```

Bu kodu işlətdikdə, bütün sətirlərdəki email və parol dəyişdirilir. Burada update sözü yazıldıqdan sonra cədvəlin adı yazılır. Sonra isə set sözü yazılır. Bundan sonra sütunların adları və yeni qiymətləri qeyd edilir. Nəticədə məlumatlar dəyişdirilir.

Öyrənməli olduğumuz ən vacib operatorlardan biri də **where** operatorudur. Bu operator yalnız xüsusi məlumatları seçmək, silmək və dəyişmək üçün istifadə olunur. Məsələn, biz yalnız istifadəçi adı “Shukur1” olan istifadəçiləri seçmək istəyiriksə, o zaman bu operatorundan istifadə etməliyik. Aşağıdakı nümunəyə baxaq.

```
select * from users where login='Shukur1'
```

Gördüyümüz kimi, burada **where** sözünü yazdıqdan sonra login='Shukur1' şərtini yazırıq. Bu o deməkdir ki, yalnız logini ‘Shukur1’ olan istifadəçilər seçiləcək.

Yalnız logini ‘Shukur1’ olan istifadəçinin parol və emailin dəyişmək üçün:

```
update users set parol='sonparol', email='sonemail' where login='Shukur1'
```

Beləliklə yalnız bir sətirdə parol və email dəyişəcək. Bu sətir isə logini ‘Shukur1’ olan sətirdir.

Yalnız logini ‘Shukur1’ olan istifadəçini silmək üçün:

```
delete from users where login='Shukur1'
```

Beləliklə yalnız bir sətir silinəcək ki, bu sətir də logini ‘Shukur1’ olan sətirdir.

Bu Sql kodlarını Php ilə bir yerdə işlədərək istifadəçi qeydiyyatı, xəbər əlavə etmə, xəbər silmə kimi bir çox əməliyyatları yerinə yetirəcəyik.

Sql dilini daha geniş şəkildə bir çox mənbədən, youtube üzərindən və ya w3schools.com saytıdan öyrənmə bilərsiniz.

## Php və MySql arasında əlaqə

Phpmyadminlə tanış olduqdan sonra Php ilə Phpmyadmin arasında əlaqə yaratmağı öyrənməliyik. Biz bu əlaqəni yaradıb bazadakı silmə, seçmə, yaratma əməliyyatlarını php vasitəsilə edəcəyik. Bunun üçün php dilində bir çox funksiyalar mövcuddur. İlk istifadə edəcəyimiz funksiya baza ilə əlaqə yaratmaq üçün mövcud olan `mysqli_connect` funksiya funksiyasıdır. Funksiyanın istifadə qaydasına baxaq.

```
<?php
$connect=mysqli_connect("localhost","root","","example");
if (mysqli_connect_errno())
{
    echo "Error!: " . mysqli_connect_error();
}
else{
    echo "Connected!";
}
mysqli_close($connect);
?>
```

Burada `mysqli_connect` funksiyası 4 parametrlə alır. Birincisi serverin ünvanıdır. Baza öz kompüterimizdə olduğu üçün bu ünvan `localhost` olur. İkinci istifadəçi adı, üçüncü parol, dördüncü isə bazanın adıdır. Parol bizdə boş olduğu üçün parolu boş saxlayırıq. Burada `mysqli_connect_errno()` funksiyası isə qoşulmada xətanın olub olmadığını yoxlayır. Əgər xəta olarsa `Error` yazılır, əks halda isə `Connected` yazılır. Sonda isə `mysqli_close` funksiyası vasitəsilə baza ilə əlaqə bağlanır.

Öyrənəcəyimiz çox vacib funksiyalardan biri də `mysqli_query` funksiyasıdır. Funksiya iki parametrlə alınır. Birinci parametrlə bazayla əlaqə dəyişəni, ikinci parametrlə isə yazacağımız sorğudur. Aşağıdakı nümunəyə baxaq.

```
<?php
$connect=mysqli_connect("localhost","root","","example");
$sql=mysqli_query($connect,'delete from users');
if($sql){
    echo "Melumatlar silindi.";
}
mysqli_close($connect);
?>
```

Kodu yazıb səhifəni işlətdikdən sonra sorğu yerinə yetirilir və cədvəldəki məlumatlar silinmiş olur. Burada where operatorundan da istifadə edərək yalnız bir sətiri silə bilərik.

## Qeydiyyat formu

İstifadəçinin qeydiyyatdan keçməsi üçün aşağıdakı formu yaradaq və formdan götürdüyümüz məlumatları bazaya əlavə edək.

```
<form method="post">
<input type="text" name="login">
<br>
<input type="password" name="parol">
<br>
<input type="text" name="email">
<br>
<input type="submit" name="submit" value="gonder">
</form>
<?php
if(isset($_POST['submit'])) {
$login=$_POST['login'];
$parol=$_POST['parol'];
$email=$_POST['email'];
$connect=mysqli_connect("localhost","root","","example");
$sql=mysqli_query($connect,'insert into users(login,parol,email) values ("'.
$login.'"',''. $parol.'"',''. $email.'")');
if($sql){
    echo "Məlumatlar əlavə olundu.";
}
mysqli_close($connect);
}
?>
```

Nəticədə formdan götürülmüş məlumatlar bazaya əlavə olunur. Əsas hissəyə diqqət edək:

```
$sql=mysqli_query($connect,'insert into users(login,parol,email) values ("'.
$login.'"',''. $parol.'"',''. $email.'")');
```

Gördüyümüz kimi, burada ikinci hissədə əlavə edəcəyimiz qiymətlər yerinə formdan götürdüyümüz dəyişənləri yazırıq və onları əlavə edirik.



Ancaq, burada götürdüyümüz məlumatları yoxlanışdan keçirmirik və təhlükəsizlik üçün heç nə etmirik. Belə olan halda, istifadəçi bizim bazaya hücum edə bilər. Bu səbəbdən də bəzi tədbirlər almağımız mütləqdir. Koda aşağıdakı əlavələri edək.

```
<form method="post">
<input type="text" name="login">
<br>
<input type="password" name="parol">
<br>
<input type="text" name="email">
<br>
<input type="submit" name="submit" value="gonder">
</form>
<?php
if(isset($_POST['submit'])) {
$connect=mysqli_connect("localhost","root","","example");
$login=htmlspecialchars(mysqli_real_escape_string($connect,$_POST['login']));
$parol=htmlspecialchars(mysqli_real_escape_string($connect,$_POST['parol']));
$email=htmlspecialchars(mysqli_real_escape_string($connect,$_POST['email']));
if(strlen($login)<6 or strlen($login)>12){
    echo "Login 6 ve 12 simvol arasinda olmalidir.";
}
else if(strlen($parol)<6 or strlen($parol)>12){
    echo "Parol 6 ve 12 simvol arasinda olmalidir.";
}
else if(!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    echo "Email sehvidir.";
}
else {
$sql=mysqli_query($connect,'insert into users(login,parol,email) values ("'.
$login.'"','.$parol.'"','.$email.'"));
if($sql){
    echo "Melumatlar elave olundu.";
}
mysqli_close($connect);
}
?>
```

Buradakı şərtlərdə **strlen** funksiyasından istifadə edərək login və parolun simvol sayını yoxlayırıq. Buradakı şərtimiz login və parolun 6 və 12 simvol arası olmalıdır. Növbəti olaraq **filter\_var(\$email, FILTER\_VALIDATE\_EMAIL)** yazaraq emailin doğruluğunu yoxlayırıq. Dəyişənləri götürərkən işətdiyimiz funksiyalara baxaq:

```
$login=htmlspecialchars(mysql_real_escape_string($connect,$_POST['login']));
```

Burada `mysql_real_escape_string` funksiyası vasitəsilə **sql injection** boşluğunu aradan qaldırırıq. Növbəti olaraq istifadə etdiyimiz **htmlspecialchars** funksiyası vasitəsilə html kodlarının bazaya getməsinə əngəlləyirik. Beləliklə **XSS** boşluğunu aradan qaldırılmış olur.

## Məlumatların bazadan oxunması

Məlumatları bazadan oxumaq üçün aşağıdakı kodları yazıb bilərik.

```
<?php
$connect=mysql_connect("localhost","root","","example");
$sql=mysql_query($connect,'select * from users');
$array=mysql_fetch_array($sql);
print_r($array);
mysql_close($connect);
?>
```

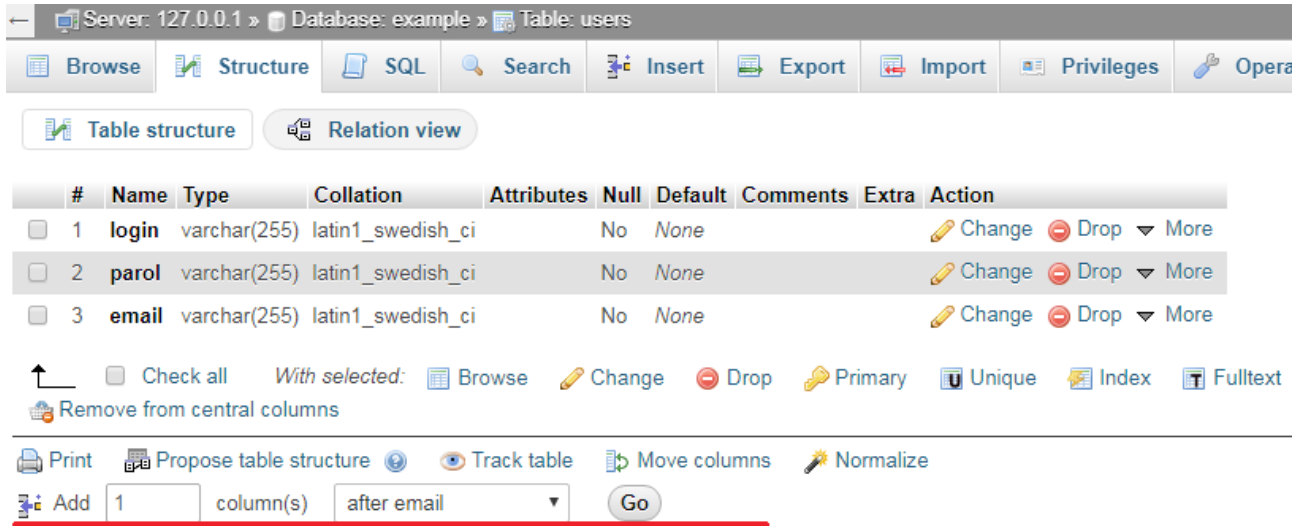
Burada **select** operatoru vasitəsilə cədvəldəki məlumatlar seçilir. Növbəti olaraq **mysql\_fetch\_array** funksiyası vasitəsilə cədvəldəki məlumatlar çoxluq olaraq götürülür və **print\_r** funksiyası vasitəsilə çap edilir. Məlumatların sətir-sətir götürülməsi üçün funksiya dövrə daxilində işlədilməli və sətirlər bitənə qədər çap olunmalıdır.

```
<?php
$connect=mysql_connect("localhost","root","","example");
$sql=mysql_query($connect,'select * from users');
echo "<table border=1>";
while($read=mysql_fetch_array($sql)){
echo "<tr><td>".$read['login']."</td><td>".$read['parol']."</td><td>".
$read['email']."</td></tr>";
}
echo "</table>";
mysql_close($connect);
?>
```

Bu şəkildə məlumatlar sətir-sətir oxunmuş olur.

## Məlumatın form vasitəsilə silinməsi

İlk öncə cədvəlimizə sətirin nömrəsini bildirən id dəyərini əlavə edək. Bu dəyər avtomatik olaraq artan dəyər olacaq və hər sətir üçün fərqli olacaq. Bunun üçün bazada “Structure” bölməsinə daxil olaq.



Server: 127.0.0.1 » Database: example » Table: users

Table structure | Relation view

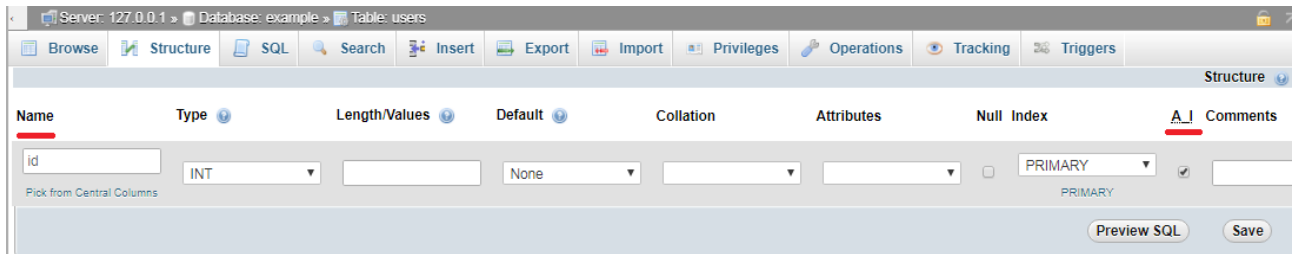
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	login	varchar(255)	latin1_swedish_ci	No	None			Change  Drop  More
<input type="checkbox"/>	2	parol	varchar(255)	latin1_swedish_ci	No	None			Change  Drop  More
<input type="checkbox"/>	3	email	varchar(255)	latin1_swedish_ci	No	None			Change  Drop  More

Check all With selected: Browse Change Drop Primary Unique Index Fulltext  
Remove from central columns

Print Propose table structure Track table Move columns Normalize

Add 1 column(s) after email Go

Bundan sonra sütun əlavə etmək üçün “Go” düyməsinə klik edirik.



Server: 127.0.0.1 » Database: example » Table: users

Structure

Name	Type	Length/Values	Default	Collation	Attributes	Null	Index	A_I	Comments
id	INT		None			<input type="checkbox"/>	PRIMARY	<input checked="" type="checkbox"/>	

Pick from Central Columns

Preview SQL Save

Buda ad hissəsinə id yazırıq. Qırmızı xəttlə işarələnmiş A\_I (Auto\_Increment) hissəsini isə seçili edirik və ok düyməsinə basırıq. Beləliklə id dəyəri hər sətir üçün özü avtomatik olaraq artacaq. Bundan sonra save edirik və id sütunu cədvəlimizə əlavə olunmuş olur.

İndi isə məlumatın silinməsi üçün bir forma yaradaq.

```
<?php
$connect=mysqli_connect("localhost","root","","example");
$sql=mysqli_query($connect,'select * from users');
echo "<form method=post>";
echo "<select name='select'>";
```

```

while($read=mysqli_fetch_array($sql)){
echo "<option value=".$read['id'].">".$read['login']."</option>";
}
echo "</select>";
echo "<input type=submit name=submit>";
echo "</form>";
if(isset($_POST['submit'])){
    $id=intval($_POST['select']);
    $sql=mysqli_query($connect,'delete from users where id="'.$id.'");
    if($sql){
        echo "Silindi.";
    }
}
mysqli_close($connect);
?>

```

Beləliklə seçdiyimiz select dəyərində görə seçilmiş məlumat silinir. Burada id dəyərinin götürülmə hissəsinə baxaq.

**\$id=intval(\$\_POST['select']);**

Burada **intval** funksiyası vasitəsilə dəyərin yalnız ədəd hissəsi saxlanılır. Bu da təhlükəsizliyi təmin edə bilər. **\$\_POST['select']** yazdıqda isə seçilmiş option elementinin value dəyəri götürülür. Bu səbəbdən də option elementinin value dəyərində istifadəçinin id dəyərini veririk.

## Məlumatı form vasitəsilə düzəliş edilməsi

Məlumatın silinməsinə oxşar qaydada məlumatın dəyişikliyi də edə bilərik.

Aşağıdakı koda baxaq.

```
<?php
$connect=mysqli_connect("localhost","root","","example");
$sql=mysqli_query($connect,'select * from users');
echo "<form method=post>";
echo "<select name='select'>";
while($read=mysqli_fetch_array($sql)){
echo "<option value=" . $read['id'] . ">" . $read['login'] . "</option>";
}
echo "</select>";
echo "<br><input type=text name=email><br>";
echo "<input type=submit name=submit>";
echo "</form>";
if(isset($_POST['submit'])){
    $id=intval($_POST['select']);
    $email=htmlspecialchars(mysqli_real_escape_string($connect,
$_POST['email']));
    $sql=mysqli_query($connect,'update users set email="' . $email . '" where id="' .
$id . '"');
    if($sql){
        echo "Deyisdirildi.";
    }
}
mysqli_close($connect);
?>
```

Beləliklə id dəyərindən istifadə edərək seçilmiş istifadəçinin emailini dəyişmiş oluruq.

## Obyektyönümlü proqramlaşdırma

Əsasən böyük layihələrdə obyektyönümlü proqramlaşdırmadan istifadə olunur. Obyektyönümlü proqramlaşdırmada dəyişənlər və funksiyalar sinif daxilində yazılır və istifadə olunur. Sinif aşağıdakı şəkildə elan olunur:

```
<?php
class sinif{
//Operations...
}
?>
```

Sinif bu şəkildə yaradılır. Sinif daxilində dəyişənlər var açar sözü ilə elan olunur.

```
<?php
class sinif{
var $a;
var $b;
}
?>
```

Burada sinif daxilində iki dəyişən elan olundu. İndi isə bu sinifdə olan dəyişənlərdən istifadə edək. Bunun üçün obyekt yaradıırıq:

```
<?php
class sinif{
var $a;
}
$obyekt=new sinif();
$obyekt->a=5;
echo $obyekt->a;
?>
```

«new» açar sözü ilə obyekt yaradıırıq. Dəyişənə müraciət üçün \$obyekt->a yazırıq. Bu dəyişənə müraciət zamanı \$ işarəsini yazmırıq. Başqa bir nümunə də göstərək:

```
<?php
class sinif{
var $a;
var $b;
}
$obyekt=new sinif();
$obyekt->a=5;
$obyekt->b=10;
```

```
echo $obyekt->a+$obyekt->b;  
?>
```

Burada sınıf dəyişənlərinə qiymət verib onların cəmini tapırıq. Sınıf daxilində sabitlərdən də istifadə edə bilərik. Məsələn:

```
<?php  
class sinif{  
const sabit="Php";  
}  
echo sinif::sabit;  
?>
```

Const açar sözü ilə sabiti elan edirik sonra isə sinif::sabit ifadəsi ilə sabitə müraciət edirik.

İndi isə sınıf daxilində funksiyaların elan olunmasına baxaq:

```
<?php  
class sinif{  
function funksiya(){  
echo "Class";  
}  
}  
$obyekt=new sinif();  
$obyekt->funksiya();  
?>
```

Burada isə sınıf daxilində funksiyanı yaratdıq və sonra istifadə etdik. İndi isə eyni sınıf daxilində həm funksiyadan həm də dəyişənlərdən istifadə edək:

```
<?php  
class sinif{  
var $ad;  
function funksiya($name){  
$this->ad=$name;  
}  
}  
$obyekt=new sinif();  
$obyekt->funksiya("Name");  
echo $obyekt->ad;  
?>
```

Burada ilk öncə dəyişəni elan edirik. Dəyişənri elan etdikdən sonra funksiyanı elan edirik. Funksiyada 1 parametr var. Sınıf daxilində dəyişənlərə müraciət etmək üçün \$this->ad kimi yazırıq. Funksiya daxilində isə sınıf dəyişənini funksiyanın parametrinə bərabər edirik. Sonda isə funksiyanı istifadə edirik və dəyişənə qiymət vermiş oluruq. Daha bir nümunəyə baxaq:

```
<?php
class sınıf{
var $a;
var $b;
function toplama(){
echo $this->a+$this->b;
}
}
$obyekt=new sınıf();
$obyekt->a=5;
$obyekt->b=10;
$obyekt->toplama();
?>
```

Burada da sınıf vasitəsilə toplama əməliyyatı yerinə yetirilir. Sınıf daxilində sabitə aşağıdakı şəkildə müraciət olunur:

```
<?php
class sınıf{
const sabit="Php";
function funksiya(){
echo self::sabit;
}
}
$obyekt=new sınıf();
$obyekt->funksiya();
?>
```

Burada self açar sözü ilə sınıf daxilində sabitə müraciət etdik. Sınıf daxilində \_\_construct adı ilə yaradılmış funksiya obyekt yaradılanda avtomatik şəkildə çağrılır. Məsələn:

```
<?php
class sınıf{
function __construct(){
echo "Function";
}
}
```



```
}  
$obyekt=new sinif();  
?>
```

Burada obyekt yaradılarda artıq funksiya avtomatik şəkildə çağrılır. Bu funksiyaaya aid başqa bir nümunəyə də baxaq:

```
<?php  
class sinif{  
function __construct(){  
echo "Function";  
}  
function funksiya(){  
echo "Funksiya";  
}  
}  
$obyekt=new sinif();  
$obyekt->funksiya();  
?>
```

Burada görə bilərik ki \_\_construct funksiyası biz çağırdığımız funksiyaadan əvvəl avtomatik çağrıldı.

Sinif daxilində \_\_destruct adı ilə yaradılmış funksiya çağrılmış bütün funksiyaalardan sonra avtomatik çağrılır. Məsələn:

```
<?php  
class sinif{  
function __destruct(){  
echo "Function";  
}  
function funksiya(){  
echo "Funksiya";  
}  
}  
$obyekt=new sinif();  
$obyekt->funksiya();  
?>
```

Burada isə çağırdığımız funksiyaadan sonra \_\_destruct funksiyası avtomatik olaraq çağırıldı.

İndi isə \_\_construct və \_\_destruct funksiyaalarından da istifadə etmək MySql baza ilə əlaqə yaradaq:

```

<?php
class baza{
var $connect;
function __construct(){
$this->connect=mysqli_connect("localhost","root","","example");
}
function query($query){
mysqli_query($this->connect,$query);
}
function __destruct(){
mysqli_close($this->connect);
}
}
$obyekt=new baza();
$obyekt->query("delete from users");
?>

```

Burada \_\_construct funksiyasında baza ilə əlaqə yaradırıq. Obyekt yaradılanda funksiya avtomatik çağrılır və baza ilə əlaqə yaradılır. Növbəti olaraq sorğunu yazmaq üçün query funksiyasını yaratdıq və çağırdıq. Növbəti olaraq isə \_\_destruct funksiyasında əlaqəni bağladıq. Sonda \_\_destruct funksiyası avtomatik çağrılacağı üçün baza ilə əlaqə avtomatik olaraq bağlanacaq.

İndi isə private və public anlayışları ilə tanış olaq. Public olaraq elan edilən dəyişən və funksiyalar ümumi olur yəni bu formada elan olunan dəyişənlərə kənardan qiymət vermək mümkün olur. Bizim yuxarda yazdığımız kodlarda bütün dəyişən və funksiyalar biz public yazmağımızdan asılı olmayaraq public hesab olunurdu. Məsələn aşağıdakı kodlar eynidir:

```

<?php
class sinif{
public $a;
}
?>

```

```

<?php
class sinif{
var $a;
}
?>

```

Yəni burada public yazmasaq da bu dəyişən public sayılır. Yuxarıda qeyd etdiyimiz kimi public dəyişənlərə kənarından qiymət vermək mümkündür. Private olan dəyişənlərə isə yalnız sinif daxilində qiymət verilə bilər. Aşağıdakı koda baxaq:

```
<?php
class sinif{
public $a;
}
$obyekt=new sinif();
$obyekt->a=5;
echo $obyekt->a;
?>
```

Burada dəyişəni elan edərkən public yerinə private yazsaq:

```
<?php
class sinif{
private $a;
}
$obyekt=new sinif();
$obyekt->a=5;
echo $obyekt->a;
?>
```

Artıq burada səhifəni işlədən zaman səhv çıxacaq. Bunun səbəbi yuxarıda da qeyd etdiyimiz kimi private olan dəyişənlərə yalnız sinif daxilində qiymət verilə bilər. Aşağıdakı kod isə doğrudur:

```
<?php
class sinif{
private $a;
public function funksiya($var){
echo $this->a=$var;
}
}
$obyekt=new sinif();
$obyekt->funksiya(5);
?>
```

Burada dəyişənə funksiyanın qiyməti verildi. Sinifləri genişləndirmək üçün extends açar sözündən istifadə olunur. Nümunə:

```
<?php
```

```

class sinif1 {
public $a;
}
class sinif2 extends sinif1 {
function funksiya($var){
$this->a=$var;
echo $this->a;
}
}
$obyekt=new sinif2();
$obyekt->funksiya("Hello World!");
?>

```

Burada sinifi genişləndirərək sinif2-də sinif1-in dəyişənindən istifadə etdik. Mücərrəd siniflər anlayışı mövcuddur. Mücərrəd sinif daxilində dəyişənlər və funksiyalar elan olunur, əməliyyatlar isə genişləndirilən siniflərdə yazılır. Mücərrəd siniflər abstract açar sözü ilə yazılır. Nümunə:

```

<?php
abstract class sinif1 {
public $a;
abstract public function funksiya($var);
}
class sinif2 extends sinif1 {
public function funksiya($var){
$this->a=$var;
echo $this->a;
}
}
$obyekt=new sinif2();
$obyekt->funksiya("Hello World!");
?>

```

Mücərrəd sinifdən bu şəkildə istifadə etdik.

Mücərrəd siniflərdən əlavə onlara çox oxşar olan interface siniflər də mövcuddur. Bu tip siniflərdə funksiyalar elan olunur. Əməliyyatlar isə genişləndirilmiş sinifdə yazılır. Bu tip siniflərdə dəyişənlər və funksiyalar public olmalıdır. Bu tip siniflər interface açar sözü ilə yaradılır və siniflər genişləndirildikdə implements açar sözündən istifadə olunur. Nümunə:

```

<?php
interface sinif1 {
public function funksiya();
}

```

```
class sinif2 implements sinif1 {
public function funksiya() {
echo "Hello World!";
}
}
$obyekt=new sinif2();
$obyekt->funksiya();
?>
```

Mücərrəd və interface sinifləri bir yerdə işlədə bilərik. Məsələn:

```
<?php
abstract class sinif1 {
public $var;
}
interface sinif2 {
public function funksiya($var);
}
class sinif3 extends sinif1 implements sinif2 {
public function funksiya($var) {
$this->var=$var;
echo $this->var;
}
}
$obyekt=new sinif3();
$obyekt->funksiya("Hello World!");
?>
```

Burada mücərrəd və interface siniflərini bir yerdə istifadə etdik.

## Php və Ajax

Ajax Javascript vasitəsilə səhifəni yeniləmədən əməliyyatlar aparmaq və digər səhifələrə sorğular göndərmək üçün yaradılmış texnologiyadır. Ajax ilə səhifə yenilənmədən əməliyyatlar aparmaq mümkündür. Bu texnologiyayı istifadə etmək üçün bir çox hazır kitabxanalar mövcuddur. Biz bu dərsdə JQuery kitabxanasından istifadə edəcəyik. Bunu üçün səhifəyə JQuery faylını çağırmalıyıq. Ajax istifadəsinə baxaq:

```
<html>
<head>
<script src="http://code.jquery.com/jquery-2.2.0.js"></script>
<script>
$(function(){
$.ajax({
type: "post",
url: "ajax.php",
data: {"name":"Ad"},
success: function(e){
alert(e);
}
})
})
</script>
</head>
<body>
</body>
</html>
```

ajax.php səhifəsi:

```
<?php
$ad=$_POST['name'];
echo $ad;
?>
```

İndi isə izahına keçək. İlk səhifədə jquery faylını çağırdıqdan sonra ajax metodundan istifadə edirik. İlk parametr type parametridir. Php səhifəsində post metodundan istifadə edəcəyimizə görə burada post yazırıq. İkinci parametr url parametridir. Burada məlumatı göndərəcəyimiz səhifəni yazırıq. Üçüncü parametr data parametridir. Burada isə göndərəcəyimiz məlumatları yazırıq. Dördüncü parametr isə success parametridir. Məlumat göndərildikdən sonra yerinə yetiriləcək funksiyayı buraya yazırıq. Ajax.php səhifəsində isə post metodu ilə məlumatı götürürük və çap edirik. Nəticədə ilk səhifədə alert mesajı ilə ajax.php səhifəsində çap olunan məlumat

göstərilir. İndi isə Formdan gələn məlumatları səhifə yenilənmədən bazaya əlavə edək. Əvvəlcə ilk səhifəni hazırlayaq:

```
<html>
<head>
<script src="http://code.jquery.com/jquery-2.2.0.js"></script>
<script>
$(function(){
$("#button").click(function(){
var login=$("#text").val();
var parol=$("#text2").val();
var email=$("#text3").val();
$.ajax({
type: "post",
url: "ajax.php",
data: {"login":login,"parol":parol,"email":email},
success: function(e){
alert(e);
}
})
})
})
})
</script>
</head>
<body>
<form method=post>
Login: <input type="text" name="login" id="text">
<br>
Parol: <input type="text" name="parol" id="text2">
<br>
Email: <input type="text" name="email" id="text3">
<br>
<button type="button" id="button">Send</button>
</form>
</body>
</html>
```

ajax.php səhifəsi:

```
<?php
$connect=mysqli_connect("localhost","root","","example");
$login=htmlspecialchars(mysqli_real_escape_string($connect,$_POST['login']));
$parol=htmlspecialchars(mysqli_real_escape_string($connect,$_POST['parol']));
```

```

$email=htmlspecialchars(mysqli_real_escape_string($connect,$_POST['email']));
$sql=mysqli_query($connect,'insert into users(login,parol,email) values ("'.
$.login.'"','.$parol.'"','.$email.'"));
if($sql){
    echo "Məlumatlar əlavə olundu.";
}
mysqli_close($connect);
?>

```

Beləliklə məlumatları bazaya əlavə edirik. Burada ilk səhifəni aşağıdakı şəkildə də yazmış olarıq.

```

<html>
<head>
<script src="http://code.jquery.com/jquery-2.2.0.js"></script>
<script>
$(function(){
$("#button").click(function(){
$.ajax({
type: "post",
url: "ajax.php",
data: $("#form").serialize(),
success: function(e){
alert(e);
}
})
})
})
</script>
</head>
<body>
<form method=post id="form">
Login: <input type="text" name="login">
<br>
Parol: <input type="text" name="parol">
<br>
Email: <input type="text" name="email">
<br>
<button type="button" id="button">Send</button>
</form>
</body>
</html>

```



Beləliklə input elementlərinə id dəyəri verməyə ehtiyac olmur, sadəcə **serialize** metodu ilə formdakı bütün məlumatlar php səhifəsinə göndərilir. Həmin səhifədə input məlumatları name dəyərinə görə götürülür.

## **Kitabın sonu**

Bu kitabın php kitabının ikinci buraxılışı olaraq təqdim olundu. Kitabda bəzi mövzular yenidən yazıldı, bəzi mövzulara isə əlavələr və düzəlişlər edildi. Ümid edirik ki, kitab php öyrənmək istəyənlərə kömək olacaq. Növbəti kitablarda görüşənədək :)