Internetes alkalmazás fejlesztés

PHP

Mi a PHP?

- A PHP nyelv túlnőtt eredeti jelentőségén. Születésekor csupán egy makró készlet volt, amely személyes honlapok karbantartására készült. Innen ered neve is: Personal Home Page Tools. Később a PHP képességei kibővültek, így egy önállóan használható programozási nyelv alakult ki, amely képes nagyméretű webes adatbázis-alapú alkalmazások működtetésére is.

Mi a PHP?

- Tulajdonképpen kiszolgáló-oldali programozási nyelv, amit jellemzően HTML oldalakon használnak.
- A hagyományos HTML lapokkal ellentétben azonban a kiszolgáló a PHP parancsokat nem küldi el az ügyfélnek, azokat a kiszolgáló oldalán a PHP- értelmező dolgozza fel.

A PHP előnyei

- A programozási szakasz érezhetően gyorsabb
- A PHP,mint nyílt forráskódú termék jó támogatással rendelkezik, amit a képzett fejlesztői gárda és az elkötelezett közösség nyújt számunkra.
- A legfontosabb operációs rendszerek bármelyikén képes futni, a legtöbb kiszolgálóprogrammal együttműködve.
- Hordozhatósság
- Teljesítmény

A PHP telepítése

- Milyen platformokat, kiszolgálókat és adatbázisokat támogat a PHP?
- Honnan szerezhetjük be a PHP-t és más nyílt forráskódú programokat?
- Hogyan telepíthető a PHP Linux rendszerre?
- Hogyan állíthatók be a fontosabb paraméterek?
- Hol található segítség, ha nem sikerül a telepítés?

A PHP telepítése

- A PHP teljesen platform független, ami azt jelenti, hogy fut Windows operációs rendszeren,a legtöbb UNIX rendszeren . beleértve a Linuxot ., sőt még Macintosh gépeken is. A támogatott kiszolgálók köre igen széles.
- A legnépszerűbbek: Apache (szintén nyílt forráskódú és platform független), Microsoft Internet Information Server

A PHP telepítése

- A PHP fordítható önálló alkalmazássá is, így az értelmező parancssorból is hívható.
- A PHP-t alapvetően úgy tervezték, hogy könnyen összhangba hozható legyen a különböző adatbázisokkal.
- A PHP beszerzése:
  - www.php.net
  - WAMP szerver http://www.wampserver.com/en/
  - Magyar változat (nem a legfrissebb) http://wamp5.extra.hu/download.html

A PHP telepítése Linux rendszeren

- Jelentkezz be **root** felhasználóval
- A PHP-t kétféleképpen lehet Apache modulként előállítani. Egyrészt újrafordíthatjuk a webkiszolgálót és beépíthetjük a PHP-értelmezőt, másrészt a PHP-t dinamikusan megosztott objektumként (DSO, Dynamic Shared Object) is fordíthatjuk.
- le kell töltenünk a PHP legfrissebb változatát.
- ki kell csomagolnunk: *tar -xvzf php-4.x.x.tar.gz*
- cd ../php-4.x.x

A PHP telepítése Linux rendszeren

- A kapcsolók:
- **--enable-track-vars**
  - Ez a szolgáltatás automatikusan előállítja számunkra a PHP oldalakon kívülről érkező adatokhoz tartozó asszociatív tömböket. Ezek a GET vagy POST kéréssel érkezett adatok, a visszaérkező süti-értékek, a kiszolgálói és környezeti változók.
- **--with-gd**
  - A --with-gd paraméter engedélyezi a GD könyvtár támogatását. Amennyiben a GD könyvtár telepítve

Az Apache beállítása

- Miután sikeresen lefordítottuk az Apache-t és a PHP-t, módosítanunk kell az Apache beállításait tartalmazó *httpd.conf* fájlt. Ez az Apache könyvtárának conf alkönyvtárban található. A következő sorok hozzáadása szükséges:
  - AddType application/x-httpd-php .php .php3
  - AddType application/x-httpd-php-source .phps
- Ha ügyfeleink miatt esetleg a hagyományos oldalaknál megszokott .html kiterjesztést

- A PHP működését a fordítás vagy telepítés után is befolyásolhatjuk, a php.ini használatával. UNIX rendszereken az alapbeállítású könyvtár a php.ini fájl számára a /usr/local/lib, Windows rendszereken a Windows könyvtára.
- Emellett a feldolgozásra kerülő PHP oldal könyvtárában . a munkakönyvtárban . Elhelyezett php.ini fájlban felülbírálhatjuk a korábban beállított értékeket, így akár

Az első PHP program

```
<?php
print ("Hello Web!");
?>
```

PHP blokkok kezdése és befejezése

Kezdő és záró elemek		
Elnevezés	Kezdő elem	Záró elem
Hagyományos	<?php	?>
Rövid	<?	?>
ASP stílus	<%	%>
Script elem	<SCRIPT Language=„PHP”>	</script>

A rövid és ASP stílusú elemeket engedélyezni kell a php.ini fájlban.

- Ha PHP-ben egysoros kódot írunk, nem kell külön sorba tennünk a kezdő- és záró elemeket , illetve a programsort:
- <?php print("Hello Web!"); ?>

HTML és PHP kód egy oldalon

```
<html>
   <head>
        <title> PHP program HTML
   tartalommal</title>
   </head>
   <body>
        <b>
        <?php
        print ("Hello Web!");
        ?>
```

Megjegyzések a PHP kódokban

- // Ez megjegyzés
- # Ez is megjegyzés
- /*

Ez egy megjegyzés.

A PHP-értelmező

ezen sorok egyikét

sem fogja feldolgozni.

*/

A print() függvény

- A print() függvény kiírja a kapott adatokat. A legtöbb esetben minden, ami a print() függvény kimenetén megjelenik, a böngészőhöz kerül.

A PHP nyelv alkotó elemei

- Mik a változók és hogyan használjuk azokat?
- Hogyan hozhatunk létre változókat és hogyan érhetjük el azokat?
- Mik azok az adattípusok?
- Melyek a leggyakrabban használt műveletek?
- Hogyan hozhatunk létre kifejezéseket műveletjelek használatával?
- Hogyan határozhatunk meg állandókat és hogyan használhatjuk azokat?

Változók

- A változók különleges tárolók, amiket abból a célból hozunk létre, hogy értéket helyezzünk el bennük.
- A változók egy dollárjelből ($) és egy tetszőlegesen választott névből tevődnek össze.
- A név betűket, számokat és aláhúzás karaktereket (_) tartalmazhat , számmal azonban nem kezdődhet!
- Szóközöket és más olyan karaktereket,

Változók

- Változó létrehozásához (deklarálásához, vagyis bevezetéséhez) egyszerűen csak bele kell írni azt a programunkba.
- Létrehozáskor általában rögtön értéket is szoktunk adni a változónak.
- *$szam1 = 8;*
- *$szam2 = 23;*

Dinamikus változók

- Szokatlan, ám hasznos, hogy a változó nevét is tárolhatjuk változóban.
- Tehát ha az alábbi módon értéket rendelünk egy változóhoz
- *$felhasznalo = "Anna";*
- az ekvivalens ezzel:
- *$tarolo = "felhasznalo";*
- *$$tarolo = "Anna";*

Dinamikus változók

- Dinamikus változókat karakterlánc-konstanssal is létrehozhatunk.
- Ekkor a változó nevéül szolgáló karakterláncot kapcsos zárójelbe tesszük:
- *${"felhasznalonev"} = "Anna";*

```
<html>
   <head>
      <title>4.1. program Dinamikusan beállított
   és elért változók</title>
   </head>
   <body>
   <?php
      $tarolo = "felhasznalo";
      $$tarolo = "Anna";
      // lehetne egyszerűen csak:
```

```
<html>
   <head>
      <title>4.2. program Változók érték szerinti
   hozzárendelése</title>
   </head>
   <body>
   <?php
      $egyikValtozo = 42;
      $masikValtozo = $egyikValtozo;
      // $masikValtozo ha $egyikValtozo
```

```
<html>
   <head>
      <title>4.3. program Változóra mutató
   hivatkozás</title>
   </head>
   <body>
   <?php
      $egyikValtozo = 42;
      $masikValtozo = &$egyikValtozo;
      // $masikValtozo ha $egyikValtozo re
```

Adattípusok

Típus	Leírás	Példa
Integer	Egész szám	5
Double	Lebegő pontos	3,3456
String	Karakterek sorozata	„Hello”
Boolean	Logikai változó, értéke igaz vagy hamis (true vagy false)	
Object	Objektum	
Array	Tömb	

- A változó típusának meghatározására a PHP 4 beépített gettype() függvényét használhatjuk.
- Ha a függvényhívás zárójelei közé változót teszünk, a gettype() egy karakterlánccal tér vissza, amely a változó típusát adja meg.

```
<html>
   <head>
      <title>4.4. program Változó típusának
   vizsgálata</title>
   </head>
   <body>
   <?php
      $proba = 5;
      print gettype( $proba ); // integer
      print "<br>"; // új sor, hogy ne follyanak
   össze a típusnevek
```

- A PHP a változó típusának módosítására a settype() függvényt biztosítja.
- A settype()-ot úgy kell használnunk, hogy a megváltoztatandó típusú változót, valamint a változó új típusát a zárójelek közé kell tennünk, vesszővel elválasztva.

```
<html>
   <head>
      <title>4.5. program Változó típusának módosítása a settype() függvény segítségével</title>
   </head>
   <body>
   <?php
      $ki_tudja_milyen_tipusu = 3.14;
      print gettype( $ki_tudja_milyen_tipusu );
   // double
      print " - $ki_tudja_milyen_tipusu<br>"; //
```

- A változó neve elé zárójelbe írt adattípus segítségével a változó értékének általunk meghatározott típusúvá alakított másolatát kapjuk.
- A lényegi különbség a settype() függvény és a típus-átalakítás között, hogy az átalakítás során az eredeti változó típusa és értéke változatlan marad,
- míg a settype() alkalmazása során az eredeti változó típusa és értéke az új adattípus

Műveletjelek és kifejezések

elnevezés	példa
hozzárendelés	$nev = ”aaaa”
összeadás	4 + 4
kivonás	6 - 3
szorzás	3 *4
osztás	10 / 2
maradék	10 % 3
összefűzés	”alma” . ”fa”

A műveletek kiértékelési sorrendje

- Az értelmező a kifejezéseket általában balról jobbra olvassa. A több műveletjelet tartalmazó összetettebb kifejezéseknél már bonyolultabb a helyzet.
- Mi a helyzet a következő esetben?
- 4 + 5 * 2

Állandók

- Egy adott név alatt tárolt érték ne változzon a program futása során, létrehozhatunk állandókat (konstansokat) is.
- Ezt a PHP beépített define() függvénye segítségével tehetjük meg. Miután az állandót létrehoztuk, annak értékét nem szabad (nem tudjuk) megváltoztatni.
- Az állandó nevét, mint karakterláncot és az értéket vesszővel elválasztva a zárójeleken belülre kell írnunk.

Vezérlési Szerkezetek

Elágazások

Elágazások

- **Az if utasítás**

```
if ( kifejezés )
   {
   // ha a kifejezés értéke igaz,
   // ez a blokk végrehajtódik
   }
```

- **Az if utasítás else ága**

```
if (feltétel)
   {
   // itt következik az a programrész, amely akkor
   kerül
```

```
<html>
   <head>
      <title>5.2. program Az else ággal
   kiegészített if utasítás</title>
   </head>
   <body>
   <?php
      $hangulat = "szomorú";
      if ( $hangulat == "boldog" )
      {
      print "Hurrá, jó kedvem van!";
```

Elágazások

- **Az if utasítás elseif ága**

```
if ( feltétel )
    {
    // ez a rész akkor fut le, ha a feltétel igaz
    }
    elseif ( másik feltétel )
        {
        // ez a rész akkor fut le, ha a másik feltétel igaz,
        // és minden elõzõ feltétel hamis
        }
        // itt még tetszõleges számú elseif rész következhet
```

Elágazások

- **A switch utasítás**

```
switch ( kifejezés )
    {
        case érték1:
        // ez történjen, ha kifejezés értéke érték1
        break;
        case érték2:
        // ez történjen, ha kifejezés értéke érték2
        break;
        default:
        // ez történjen, ha a kifejezés értéke
        // egyik felsorolt értékkel sem egyezett meg
```

```
switch( $hetnapja )
    {
        case "Péntek":
           print "Kikapcsolni a vekkert, holnap nem kell
           dolgozni!<br>";
        case "Hétfõ":
        case "Szerda":
           print "Ma délelõtt dolgozom<br>";
           break;
        case "Kedd":
        case "Csütörtök":
           print "Ma délután dolgozom<br>";
           break;
        case "Vasárnap":
           print "Bekapcsolni a vekkert!<br>";
```

Elágazások

- **A ?: műveletjel**

(feltétel) ? érték_ha_a_feltétel_igaz :
érték_ha_a_feltétel_hamis ;

Vezérlési szerkezetek

Ciklusok

Ciklusok

- **A while ciklus**
- A while ciklus szerkezete rendkívül hasonlít az if elágazáséhoz:

```
while ( feltétel )
    {
    // valamilyen tevékenység
    }
```

```
<html>
    <head>
       <title>5.6. program A while ciklus</title>
    </head>
    <body>
        <?php
            $szamlalo = 1;
            while ( $szamlalo <= 12 )
                {
                print "$szamlalo kétszerese
                  ".($szamlalo*2)."<br>";
                $szamlalo++;
```

Ciklusok

- **A do..while ciklus**
- A do..while ciklus kicsit hasonlít a while-hoz. A lényegi különbség abban van,hogy ebben a szerkezetben először hajtódik végre a kód és csak azután értékelődik ki a feltétel:

```
do
{
// végrehajtandó programrész
}
while ( feltétel );
```

```
<html>
    <head>
    <title>5.7. program A do..while ciklus</title>
    </head>
    <body>
    <?php
        $szam = 1;
        do
        {
        print "Végrehajtások száma: $szam<br>\n";
        $szam++;
```

Ciklusok

- **A for ciklus**

```
for ( változó_hozzárendelése; feltétel;
   számláló_növelése)
{
// a végrehajtandó programblokk
}
```

- Az ezt megvalósító egyenértékű while:

```
változó_hozzárendelése;
while ( feltétel )
{
```

Ciklusok

- Következő ismétlés azonnali elkezdése a continue utasítás segítségével

```
<html>
    <head>
        <title>5.11. program A continue utasítás használata</title>
    </head>
    <body>
        <?php
        $szamlalo = -4;
```

Ciklusok

- **Egymásba ágyazott ciklusok**
  - A ciklusok törzsében is lehetnek ciklusok. Ez a lehetőség különösen hasznos, ha futási időben előállított HTML táblázatokkal dolgozunk.

```
<html>
	<head>
	    <title>5.12. program Két for ciklus egymásba ágyazása</title>
	</head>
	<body>
		<?php
		print "<table border=1>\n"; // HTML táblázat kezdete
		for ( $y=1; $y<=12; $y++ )
			{
			print "<tr>\n"; // sor kezdete a HTML táblázatban
			for ( $x=1; $x<=12; $x++ )
				{
				print "\t<td>"; // cella kezdete
				print ($x*$y);
				print "</td>\n"; // cella vége
				}
			print "</tr>\n"; // sor vége
			}
		print "</table>\n"; // táblázat vége
		?>
	</body>
</html>
```

Függvények

Függvények

- Kétféle függvény létezik: a nyelvbe beépített függvény és az általunk létrehozott függvény.

valamilyen_fuggveny ($elso_parameter, $masodik_parameter);

- A print() abból a szempontból tipikus függvény, hogy van visszatérési értéke.
- A legtöbb függvény, ha nincs .értelmes. visszatérési értéke, információt ad arról, hogy munkáját sikeresen befejezte-e. A print() visszatérési értéke logikai típusú (true, ha sikeres volt)

Függvények

```
<html>
	<head>
	    <title>6.2. program Függvény létrehozása</title>
	</head>
	<body>
	<?php
		function nagyHello()
			{
			print "<h1>HELLO!</h1>";
			}
		nagyHello();
	?>
	</body>
</html>
```

Függvények

```
<html>
	<head>
	<title>6.3. program Egy paramétert váró függvény létrehozása</title>
	</head>
	<body>
	<?php
		function sorKiir( $sor )
			{
			print ("$sor<br>\n");
			 }
		sorKiir("Ez egy sor");
		sorKiir("Ez már egy másik sor");
		sorKiir("Ez megint egy új sor");
	?>
	</body>
</html>
```

- **Függvények visszatérési értéke**
  - A függvényeknek visszatérési értéke is lehet, ehhez a return utasításra van szükség. A return befejezi a függvény futtatását és az utána írt kifejezést küldi vissza a hívónak.

```
<html>
	<head>
		<title>6.4. program Visszatérési értékkel rendelkező függvény</title>
	</head>
	<body>
	<?php
		function osszead( $elsoszam, $masodikszam )
			{
			$eredmeny = $elsoszam + $masodikszam;
			return $eredmeny;
			}
		print osszead(3,5); // kiírja, hogy "8„
	?>
	</body>
</html>
```

- Lehetőségünk van rá, hogy karakterláncba tegyük egy függvény nevét és ezt a változót pontosan úgy tekintsük, mint ha maga a függvény neve lenne.

```
<html>
    <head>
    <title>6.5. program Dinamikus
      függvényhívás</title>
```

Változók hatásköre

- A függvényben használt változók az adott függvényre nézve helyiek maradnak.
- Más szavakkal: a változó a függvényen kívülről vagy más függvényekből nem lesz elérhető.
- Nagyobb projektek esetén ez megóvhat minket attól, hogy véletlenül két függvény felülírja azonos nevű változóik tartalmát.

Hozzáférés változókhoz a global kulcsszó segítségével

- Alapértelmezés szerint a függvényeken belülről nem érhetjük el a máshol meghatározott változókat.
- Ha mégis megpróbáljuk ezeket használni, akkor helyi változót fogunk létrehozni vagy elérni.
- Mindent figyelembe véve ez egy jó dolog. Megmenekültünk az azonos nevű változók ütközésétől és a függvény paramétert igényelhet, ha meg szeretne tudni valamit a

```
<html>
    <head>
    <title>6.8. program Globális változó elérése a global
       kulcsszó segítségével</title>
    </head>
    <body>
    <?php
        $elet=42;
        function eletErtelme()
        {
        global $elet;
        print "Az élet értelme: $elet<br>";
        }
```

```
<html>
    <head>
        <title>6.10. program Függvényhívások közötti
        állapot megőrzése a static kulcsszó
        használatával</title>
    </head>
    <body>
    <?php
        function szamozottCimsor( $cimszoveg )
            {
            static $fvHivasokSzama = 0;
            $fvHivasokSzama++;
            print "<h1>$fvHivasokSzama.
```

- Ha egy függvényen belül egy változót a static kulcsszóval hozunk létre, a változó az adott függvényre nézve helyi marad. Másrészt a függvény hívásról hívásra emlékszik a változó értékére.

Paraméterek alapértelmezett értéke

- A PHP nyelv remek lehetőséget biztosít számunkra rugalmas függvények kialakításához.
- Eddig azt mondtuk, hogy a függvények többsége paramétert igényel.
- Néhány paramétert elhagyhatóvá téve függvényeink rugalmasabbá válnak.

- A 6.11. példaprogram létrehoz egy hasznos kis függvényt, amely egy karakterláncot egy HTML font elembe zár.
- A függvény használójának megadja a lehetőséget, hogy megváltoztathassa a font elem size tulajdonságát, ezért a karakterlánc mellé kell egy $meret nevű paraméter is.

```
<html>
    <head>
        <title>6.11. program Két paramétert igénylő függvény</title>
    </head>
    <body>
    <?php
        function meretez( $szoveg, $meret )
            {
            print "<font size=\"$meret\„ face=\"Helvetica,Arial,Sans-Serif\">
               $szoveg</font>";
            }
        meretez("Egy címsor<br>",5);
        meretez("szöveg<br>",3);
        meretez("újabb szöveg<BR>",3);
        meretez("még több szöveg<BR>",3);
    ?>
    </body>
</html>
```

```
<html>
    <head>
        <title>6.11. program Függvény elhagyható paraméterrel </title>
    </head>
    <body>
    <?php
        function meretez( $szoveg, $meret = 3)
            {
            print "<font size=\"$meret\„ face=\"Helvetica,Arial,Sans-Serif\">
               $szoveg</font>";
            }
        meretez("Egy címsor<br>",5);
        meretez("szöveg<br>");
        meretez("újabb szöveg<BR>");
        meretez("még több szöveg<BR>");
    ?>
    </body>
</html>
```

Függvények

- Amikor a függvényeknek paramétereket adunk át, a helyi paraméter-változókba a paraméterek értékének másolata kerül.
- Az e változókon végzett műveleteknek a függvényhívás befejeződése után nincs hatásuk az átadott paraméterekre.

Függvények

- Lehetőségünk van arra is, hogy ne a változó értékét, hanem arra mutató hivatkozást adjunk át. (Ezt cím szerinti paraméterátadásnak is hívják.)
- Ez azt jelenti, hogy a változóra mutató hivatkozással dolgozunk a függvényben és nem a változó értékének másolatával.
- A paraméteren végzett bármilyen művelet megváltoztatja az eredeti változó értékét is.
- Hivatkozásokat függvényeknek úgy adhatunk

Érték vs. címszerinti paraméter átadás

```
<html>
<head>
<title>6.13. program
   Függvényparaméter
   érték szerinti
   átadása</title>
</head>
<body>
<?php
function ottelTobb( $szam
   )
```

```
<html>
<head>
<title>6.14. program Cím
   szerinti
   paraméterátadás
   változóra mutató
   hivatkozás
   segítségével</title>
</head>
<body>
<?php
```

Tömbök

- Egy változóban azonban sajnos csak egy értéket tárolhatunk.
- A tömb olyan különleges szerkezetű változó, amelyben nincs ilyen korlátozás.
- Egy tömbbe annyi adatot lehet beletenni, amennyit csak akarunk (amennyi memóriánk van).
- Minden elemet egy szám vagy egy karakterlánc segítségével azonosíthatunk.
- A tömb elemeit az index segítségével könnyen

Tömbök

- Tömbök létrehozása az *array()* függvény segítségével:

```
$felhasznalok = array ("Berci", "Mariska", "Aladár", "Eleonóra");
```

- Tömb létrehozása vagy elem hozzáadása a tömbhöz szögletes zárójel segítségével:

```
$felhasznalok[] = "Berci";
$felhasznalok[] = "Mariska";
$felhasznalok[] = "Aladár";
$felhasznalok[] = "Eleonóra";
```

Asszociatív tömbök

- Az asszociatív tömb egy karakterláncokkal indexelt tömb.
- Képzeljünk el egy telefonkönyvet: melyik a jobb megoldás: a név mezőt a 4-gyel vagy a névvel indexelni?

Asszociatív tömbök

- Asszociatív tömbök létrehozása az array() függvény segítségével:

```
$karakter = array
(
"nev" => "János",
"tevekenyseg" => "szuperhős",
"eletkor" => 30,
"kulonleges kepesseg" => "röntgenszem"
);
```

Asszociatív tömbök

- Asszociatív tömbök létrehozása és elérése közvetlen értékadással:

```
$karakter["nev"] => "János";
$karakter["tevekenyseg"] => "szuperhős";
$karakter["eletkor"] => 30;
$karakter["kulonleges kepesseg"] =>
    "röntgenszem";
```

- Asszociatív tömböt úgy is létrehozhatunk vagy új név-érték párt adhatunk hozzá, ha egyszerűen a megnevezett elemhez (mezőhöz)

Többdimenziós tömbök

```
<html>
	<head>
	<title>7.1. program Többdimenziós tömb létrehozása</title>
	</head>
<body>
	<?php
	$karakter = array
		(
		array (
		"nev" => "János",
		"tevekenyseg" => "szuperhős",
		"eletkor" => 30,
		"kulonleges kepesseg" => "röntgenszem„
		),
		array (
		"nev" => "Szilvia",
		"tevekenyseg" => "szuperhős",
		"eletkor" => 24,
		"kulonleges kepesseg" => "nagyon erős„
		),
```

Többdimenziós tömbök

```
    array (
    "nev" => "Mari",
    "tevekenyseg" => "főgonosz",
    "eletkor" => 63,
    "kulonleges kepesseg" =>
    "nanotechnológia„
    )
    );
print $karakter[0]["tevekenyseg"]; //kiírja, hogy
  szuperhős
?>
</body>
```

Tömbök elérése

- Tömb méretének lekérdezése
- A PHP a *count()* függvényt biztosítja erre a feladatra.
- A *count()* a tömbben levő elemek számával tér vissza.

```
$felhasznalok = array ("Berci", "Marci", "Ödön", "Télapó");
$felhasznalok[count($felhasznalok)-1];
```

Tömbök elérése

- Tömb bejárása
- Több módja van annak, hogy egy tömb minden elemét végigjárjuk.
- A számmal indexelt tömbökre a foreach szerkezetet így kell használni:

```
foreach($tombnev as $atmeneti)
{
}
$felhasznalok = array ("Berci", "Marci", "Ödön");
$felhasznalok[10] = "Télapó";
```

Tömbök elérése

- Asszociatív tömb bejárása

```
foreach( $tomb as $kulcs => $ertek )
{
// a tömbelem feldolgozása
}
```

```
<html>
	<head>
	<title>7.2 példaprogram Asszociatív tömb bejárása a foreach segítségével</title>
	</head>
	<body>
	<?php
		$karakter = array (
		"nev" => "János",
		"tevekenyseg" => "szuperhõs",
		"eletkor" => 30,
		"kulonleges kepesseg" => "röntgenszem„
		);
		foreach ( $karakter as $kulcs => $ertek )
		{
		print "$kulcs = $ertek<br>";
		 }
		?>
	</body>
</html>
```

Műveletek tömbökkel

- Két tömb egyesítése az array_merge() függvény segítségével
  - Az array_merge() függvény legalább két paramétert vár: az egyesítendő tömböket. A visszatérési érték az egyesített tömb lesz.

```
$elso = array( "a", "b", "c" );
$masodik = array( 1, 2, 3 );
$harmadik = array_merge( $elso, $masodik );
foreach( $harmadik as $ertek )
{
print "$ertek<br>";
```

Műveletek tömbökkel

- Egyszerre több elem hozzáadása egy tömbhöz az array_push() függvény segítségével
  - Az array_push() függvény egy tömböt és tetszőleges számú további paramétert fogad. A megadott értékek a tömbbe kerülnek.
  - E függvény visszatérési értéke a tömbben lévő elemek száma (miután az elemeket hozzáadta).

```
$elso = array( "a", "b", "c" );
$elemszam = array_push( $elso, 1, 2, 3 );
print "Összesen $elemszam elem van az \$elso
   tömbben<P>";
```

Műveletek tömbökkel

- Az első elem eltávolítása az array_shift() függvény segítségével
  - Az array_shift() eltávolítja a paraméterként átadott tömb elsõ elemét és az elem értékével tér vissza.

```
<?php
    $egy_tomb = array( "a", "b", "c" );
    while ( count( $egy_tomb ) )
    {
    $ertek = array_shift( $egy_tomb );
    print "$ertek<br>";
```

Műveletek tömbökkel

- Tömb részének kinyerése az array_slice() függvény segítségével
  - Az array_slice() függvény egy tömb egy darabját adja vissza.
  - A függvény egy tömböt, egy kezdőpozíciót (a szelet első elemének tömbbeli indexét) és egy (elhagyható) hossz paramétert vár.
  - Ha ez utóbbit elhagyjuk, a függvény feltételezi, hogy a kezdőpozíciótól a tömb végéig tartó szeletet szeretnénk megkapni.
  - Az array_slice() nem változtatja meg a

Tömbök rendezése

- Számmal indexelt tömb rendezése a sort() függvény segítségével:
  - A sort() függvény egy tömb típusú paramétert vár és a tömb rendezését végzi el.
  - Ha a tömbben van karakterlánc, a rendezés (alapbeállításban angol!) ábécésorrend szerinti lesz, ha a tömbben minden elem szám, akkor szám szerint történik.
  - A függvénynek nincs visszatérési értéke, a paraméterként kapott tömböt alakítja át.

```
$tomb = array( 10, 2, 9 );
sort( $tomb );
print '___Rendezés szám szerint___'<br>;
foreach ($tomb as $elem)
{
print "$elem<br>";
}
$tomb[]="a";
sort( $tomb );
```

- Asszociatív tömb rendezése érték szerint az asort() függvény segítségével
  - Az asort() függvény egy asszociatív tömb típusú paramétert vár és a tömböt a sort() függvényhez hasonlóan rendezi. Az egyetlen különbség, hogy az asort() használata után megmaradnak a karakterlánc-kulcsok:

```
$a_tomb = array( "elso"=>5, "masodik"=>2,
   "harmadik"=>1 );
asort( $a_tomb );
foreach( $a_tomb as $kulcs => $ertek )
```

- Asszociatív tömb rendezése kulcs szerint a ksort() függvény segítségével
- A ksort() a paraméterben megadott asszociatív tömböt rendezi kulcs szerint.

```
$tomb = array( "x" => 5, "a" => 2, "f" => 1 );
ksort( $tomb );
foreach( $tomb as $kulcs => $ertek )
{
print "$kulcs = $ertek<BR>";
}
```