7. ÓRA

Tömbök

A tömbök és a kezelésüket segítő eszközök nagy mértékben növelik a PHP 4 programok rugalmasságát. Ha jól értünk a tömbökhöz, képesek vagyunk nagy méretű, összetett adatokat tárolni és kezelni.

Ebben az órában bemutatjuk a tömböket és néhány beépített függvényt, amelyek segítik a velük való boldogulást. Az órában a következőket tanuljuk meg:

- Mik a tömbök és hogyan hozhatók létre?
- Hogyan érhetjük el a tömbökben tárolt adatokat?
- Hogyan rendezhetjük a tömbökben tárolt adatokat?

Mit nevezünk tömbnek?

Mint már tudjuk, a változó egy „vödör”, amiben egy értéket lehet tárolni. Változók segítségével olyan programot írhatunk, amely információt tárol, dolgoz fel és ír ki. 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. Ha a változó „vödör”, akkor a tömb „iratszekrény”, tehát olyan tároló, amelyben sok-sok elemet tárolhatunk.

ÚJDONSÁG A tömb változók listája. Több változót tartalmaz, amelyeket számok vagy karakterláncok segítségével azonosíthatunk, így a különböző értékeket egyetlen névvel tárolhatjuk, rendezhetjük és érhetjük el.

Természetesen ha öt értéket kell tárolnunk, megadhatunk öt változót is. Akkor miért jó tömböt használni változók helyett? Először is azért, mert a tömb rugalmasabb adatszerkezet. Lehet benne két vagy akár kétszáz érték és ennek elérése érdekében nem kell további változókat létrehozni. Másodszor, a tömbök elemeit könnyedén kezelhetjük egységként is. Ha akarjuk, végighaladhatunk a tömbön egy ciklussal vagy elérhetjük az elemeit egyenként. Lehetőségünk van a tömböt rendezni szám szerint, szótári rendezés szerint vagy saját rendezőelv alapján.

A tömb elemeit az index segítségével könnyen elérhetjük. Az index lehet szám, de akár karakterlánc is.

Alapértelmezés szerint a tömböket számokkal indexeljük, mégpedig úgy, hogy az első elem indexe 0. Ebből az következik, hogy az utolsó tömbelem indexe mindig eggyel kisebb a tömb méreténél. Tehát az öt elemű tömb utolsó eleme a 4-es indexű elem. Ezt ajánlatos mindig fejben tartani!

A 7.1. táblázatban a `felhasznalok` tömböt láthatjuk. Figyeljük meg, hogy például a tömb harmadik elemének indexe 2.

7.1. táblázat A felhasznalok tömb elemei

Index	*Érték*	*Hányadik elem*
0	Berci	Első
1	Mariska	Második
2	Aladár	Harmadik
3	Eleonóra	Negyedik

A karakterlánccal való indexelés akkor lehet hasznos, ha egyedi nevek (kulcsok) mellett más értékeket is tárolni kell.

A PHP 4 mind a számokkal, mind a „nevekkel” indexelt tömbök elérésére biztosít segédeszközöket. Ezek közül néhánnyal ebben a fejezetben is találkozni fogunk, másokat a tizenhatodik, adatműveletekkel foglalkozó fejezetben ismerünk majd meg.

Tömbök létrehozása

A tömbök alapértelmezés szerint értékek számmal indexelt listái. Értéket egy tömbhöz kétféleképpen is rendelhetünk: Az egyik mód az `array()` függvény, a másik a tömbazonosító használata szögletes zárójelekkel (`[]`). A következő két részben mind a kétféle változattal találkozni fogunk.

Tömbök létrehozása az array() függvény segítségével

Az `array()` függvény akkor hasznos, ha egyszerre több értéket szeretnénk egy tömbhöz rendelni. Hozzunk létre egy `$felhasznalok` nevű tömböt és rendeljünk hozzá négy elemet!

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

Most a `$felhasznalok` tömb harmadik elemét, melynek indexe 2, írassuk ki!

```
print $felhasznalok[2];
```

Ez a következő karakterláncot fogja kiírni: `"Aladár"`. Az indexet a tömb neve után közvetlenül következő szögletes zárójelbe kell írni. A tömbelem írásakor és olvasásakor is ezt a jelölést kell használni.

Ne felejtsük el, hogy a tömbök indexelése nullától indul, így bármely tömbelem indexe eggyel kisebb a tömbelem tömbbéli helyénél. (Az első elem indexe 0, a második elem indexe 1.)

Tömb létrehozása vagy elem hozzáadása a tömbhöz szögletes zárójel segítségével

Tömböket úgy is létrehozhatunk, illetve meglevő tömbökhöz új elemeket adhatunk, ha a tömb neve után olyan szögletes zárójelpárt írunk, amelynek belsejében nincs index.

Hozzuk létre újra a `$felhasznalok` nevű tömböt:

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

Figyeljük meg, hogy nem kellett számot írnunk a szögletes zárójelbe. A PHP 4 automatikusan meghatározza az indexértéket, így nem kell nekünk bajlódni azzal, hogy kiszámítsuk a következő olyan indexet, amelyben még nincs érték.

Persze írhattunk volna számokat is a szögletes zárójelbe, de ez nem tanácsos. Nézzük a következő programrészletet:

```
$felhasznalok[0] = "Berci";
$felhasznalok[200] = "Mariska";
```

A tömbnek így mindössze két eleme van, de az utolsó elem indexe 200. A PHP 4 nem fog értéket adni a köztes elemeknek, ami félreértésekre adhat okot az elemek elérésekor.

Tömbök létrehozása mellett a tömbváltozók szögletes zárójele segítségével az `array()` függvénnyel létrehozott tömb végéhez új elemet is adhatunk. Az alábbi kis programrészletben létrehozunk egy tömböt az `array()` függvény segítségével, majd új elemet adunk a tömbhöz szögletes zárójellel.

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

Asszociatív tömbök

A számmal indexelt tömbök akkor hasznosak, ha abban a sorrendben szeretnénk tárolni az elemeket, amilyen sorrendben a tömbbe kerültek. Néha azonban jó lenne, ha a tömb elemeit meg tudnánk nevezni. Az asszociatív tömb egy karakterláncokkal indexelt tömb. Képzeljünk el egy telefonkönyvet: melyik a jobb megoldás: a név mezőt a 4-gyel vagy a "név"-vel indexelni?

ÚJDONSÁG A karakterlánccal indexelt tömböket asszociatív tömböknek (néha hash-nek) hívják.

Asszociatív tömbök létrehozása a számokkal indexelt tömbökkel megegyezően történik, az `array()` függvény vagy a szögletes zárójelek segítségével.

A számmal és a karakterlánccal indexelt tömbök közti határvonal a PHP-ben nem éles. Nem különböző típusok, mint a Perlben, ennek ellenére helyes az a hozzáállás, ha külön-külön kezeljük őket, mert az egyes típusok más hozzáférési és kezelési módot igényelnek.

Asszociatív tömbök létrehozása az array() függvény segítségével

Ha asszociatív tömböt szeretnénk létrehozni az `array()` függvény segítségével, minden elemnek meg kell adni a kulcsát és az értékét. Az alábbi programrészlet egy `$karakter` nevű asszociatív tömböt hoz létre négy elemmel.

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

Most elérhetjük a `$karakter` elemeit (mezőit):

```
print $karakter["eletkor"];
```

Asszociatív tömbök létrehozása és elérése közvetlen értékadással

Asszociatív tömböt úgy is létrehozhatunk vagy új név–érték párt adhatunk hozzá, ha egyszerűen a megnevezett elemhez (mezőhöz) új értéket adunk. Az alábbiakban újra létrehozzuk a `$karakter` nevű tömböt, úgy, hogy az egyes kulcsokhoz egyenként rendelünk értékeket.

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

Többdimenziós tömbök

Eddig azt mondtuk, hogy a tömbök elemei értékek. A `$karakter` tömbünkben az elemek közül három karakterláncot tartalmaz, egy pedig egy egész számot. A valóság ennél egy kicsit bonyolultabb. Egy tömbelem valójában lehet érték, objektum vagy akár egy másik tömb is. A többdimenziós tömb valójában tömbök tömbje. Képzeljük el, hogy van egy tömbünk, amelynek tömbök az elemei. Ha el akarjuk érni a második elem harmadik elemét, két indexet kell használnunk:

```
$tomb[1][2]
```

ÚJDONSÁG A tömböt, amelynek elemei tömbök, többdimenziós tömbnek hívjuk.

A tény, hogy egy tömbelem lehet tömb is, lehetőséget biztosít arra, hogy kifinomultabb adatszerkezeteket kezeljünk viszonylag egyszerűen. A 7.1. példaprogram egy tömböt ír le, amelynek elemei asszociatív tömbök.

7.1. program Többdimenziós tömb létrehozása

```
 1: <html>
 2: <head>
 3: <title>7.1. program Többdimenziós tömb
    létrehozása</title>
 4: </head>
 5: <body>
 6: <?php
 7: $karakter = array
 8:  (
 9:  array (
10:                 "nev" => "János",
11:                 "tevekenyseg" => "szuperhős",
12:                 "eletkor" => 30,
13:           "kulonleges kepesseg" => "röntgenszem"
14:                 ),
```

7.1. program (folytatás)

```
15:            array (
16:                  "nev" => "Szilvia",
17:                  "tevekenyseg" => "szuperhős",
18:                  "eletkor" => 24,
19:                  "kulonleges kepesseg" => "nagyon erős"
20:                  ),
21:            array (
22:                  "nev" => "Mari",
23:                  "tevekenyseg" => "főgonosz",
24:                  "eletkor" => 63,
25:                  "kulonleges kepesseg" =>
                     "nanotechnológia"
26:                  )
27:            );
28:
29: print $karakter[0]["tevekenyseg"]; //kiírja, hogy
    szuperhős
30: ?>
31: </body>
32: </html>
```

Figyeljük meg, hogy `array()` függvényhívásokat építettünk egy `array()` függvényhívásba. Az első szinten adjuk meg a tömböt, melynek minden eleme asszociatív tömb.

A `$karakter[2]` tömbelemhez történő hozzáféréskor a legfelső szintű tömb harmadik elemét, egy asszociatív tömböt kapunk. Az asszociatív tömb bármely elemét elérhetjük, például a `$karakter[2]["nev"]` értéke `"Mari"` lesz, a `$karakter[2]["eletkor"]` pedig `63`.

Ha a fogalmak tiszták, asszociatív és hagyományos tömbök párosításával egyszerűen hozhatunk létre összetett adatszerkezeteket.

Tömbök elérése

Eddig azt láttuk, hogyan kell tömböket létrehozni és elemeket adni azokhoz. Most végignézünk néhány lehetőséget, amit a PHP 4 a tömbök elérésére biztosít.

Tömb méretének lekérdezése

A tömb bármely elemét elérhetjük indexének használatával:

```
print $felhasznalo[4];
```

A tömbök rugalmassága miatt nem mindig egyszerű feladat kideríteni, hány elem van a tömbben. A PHP a `count()` függvényt biztosítja erre a feladatra.
A `count()` a tömbben levő elemek számával tér vissza. Az alábbi programrészben egy számokkal indexelt tömböt hozunk létre, majd a `count()` függvényt használjuk a tömb utolsó elemének elérésére.

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

Figyeljük meg, hogy egyet ki kellett vonnunk a `count()` által visszaadott értékből. Ez azért van így, mert a `count()` nem az utolsó elem indexét adja vissza, hanem a tömbelemek számát.

Semmi sem garantálja, hogy ezzel a módszerrel bármilyen (számokkal indexelt) tömb utolsó elemét megkapjuk. Vegyük például az alábbi programot:

```
$tomb = array("Ecc, pecc"); // Egyelemű tömb!
$tomb[2] = "kimehetsz";
print "Az utolsó elem: " . $tomb[count($tomb)-1];
```

A fenti kódot futtatva azt tapasztaljuk, hogy az `"Az utolsó elem: "` szöveg után semmi nem íródik ki. Ez azért van, mert a tömbnek nincs 1 indexű eleme. Az első elem a 0 indexen található, a második a 2 indexen, mivel az értékadáskor az indexet közvetlenül határoztuk meg.
A tömbök indexelése alapértelmezés szerint nullától indul, de ezt meg lehet változtatni. Ha azonban világos és következetes programokat szeretnénk írni, ne éljünk ezzel a lehetőséggel.

Tömb bejárása

Több módja van annak, hogy egy tömb minden elemét végigjárjuk. Mi most a PHP 4 igen hatékony `foreach` szerkezetét használjuk, de a tizenhatodik órában további módszereket is megvizsgálunk.

A foreach a PHP 4-es változatában került a nyelvbe.

A számmal indexelt tömbökre a `foreach` szerkezetet így kell használni:

```
foreach($tombnev as $atmeneti)
{
}
```

`$tombnev` a tömb neve, amit végig szeretnénk járni, az `$atmeneti` pedig egy változó, amelybe minden ismétlés során a tömb egy-egy eleme kerül.

Az alábbi példában egy számmal indexelt tömböt hozunk létre és a `foreach` vezérlési szerkezet segítségével érjük el annak elemeit:

```
$felhasznalok = array ("Berci", "Marci", "Ödön");
$felhasznalok[10] = "Télapó";
foreach($felhasznalok as $szemely)
       {
       print"$szemely<br>";
       }
```

A program kimenetét a 7.1-es ábrán láthatjuk.

7.1. ábra
Tömb bejárása

A tömb minden eleme átmenetileg a `$szemely` változóba került, amit a program kiír a böngészőbe. A Perlben gyakorlattal rendelkezők vigyázzanak, a `foreach` szerkezet a két nyelvben jelentősen különbözik! A Perlben ha az átmeneti változó értékét megváltoztatjuk, a megfelelő tömbelem is megváltozik.

Ha ugyanezt az előző példában tesszük, a `$felhasznalok` tömb értéke nem fog megváltozni. A tizenhatodik órában látni fogjuk, hogyan módosíthatjuk a `foreach` segítségével a tömbök tartalmát.

Figyeljük meg, hogy annak ellenére, hogy a tömb „lyukas”, a program a `foreach` szerkezet segítségével bejárt tömb minden elemét kiírta és az utolsó sor előtt nem jelentek meg üres sorok, tehát a 3 és 9 indexek közötti tömbelemeket (melyek nem meghatározottak, vagyis üres karakterláncok) nem írja ki a program.

Ahol csak mód van rá, ne használjunk tömbindexeket. A tömböket sokszor csak egyszerű listaként használjuk, ahol nem számít az, hogy egy adott elemnek mi az indexe. A PHP a tömbök elérésére a tömbindexek használatánál hatékonyabb és áttekinthetőbb függvényeket biztosít, programunk ezáltal rugalmasabb, könnyebben módosítható és jól olvasható lesz.

Asszociatív tömb bejárása

Ha az asszociatív tömb kulcsát és értékét is el szeretnénk érni, egy kicsit másképpen kell a `foreach` szerkezet használnunk.

Asszociatív tömböknél a `foreach` így alkalmazható:

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

ahol `$tomb` a tömb neve, amin végig szeretnénk menni, `$kulcs` a változó, amelyben az elem kulcsa jelenik meg, a `$ertek` pedig a kulcshoz tartozó tömbelem értéke.

A 7.2. példaprogramban létrehozunk egy asszociatív tömböt, majd egyesével elérjük és kiírjuk a tömb elemeit

7.2. program Asszociatív tömb bejárása a foreach segítségével

```
 1: <html>
 2: <head>
 3: <title>7.2 példaprogram Asszociatív tömb bejárása
    a foreach segítségével</title>
 4: </head>
 5: <body>
 6: <?php
 7: $karakter = array (
 8:              "nev" => "János",
 9:              "tevekenyseg" => "szuperhős",
10:              "eletkor" => 30,
11:              "kulonleges kepesseg" => "röntgenszem"
12:              );
13:
14: foreach ( $karakter as $kulcs => $ertek )
15:       {
16:       print "$kulcs = $ertek<br>";
17:       }
18: ?>
19: </body>
20: </html>
```

A 7.2. példaprogram kimenetét a 7.2. ábrán láthatjuk.

7.2. ábra

Asszociatív tömb bejárása

Többdimenziós tömb bejárása

Most már ismerünk olyan módszereket, amelyek segítségével a 7.1. példaprogramban létrehozott többdimenziós tömböt bejárhatjuk. A 7.3. példaprogramban létrehozunk egy többdimenziós tömböt és a `foreach` segítségével végigjárjuk az elemeit.

7.3. program Többdimenziós tömb bejárása

```
 1: <html>
 2: <head>
 3: <title>7.3. program - Többdimenziós tömb
          bejárása</title>
 4: </head>
 5: <body>
 6: <?php
 7: $karakterek = array
 8:              (
 9:              array (
10:                   "nev" => "János",
11:                   "tevekenyseg" => "szuperhős",
12:                   "eletkor" => 30,
13:                   "kulonleges kepesseg" => "röntgenszem"
14:                   ),
15:              array (
16:                   "nev" => "Szilvia",
17:                   "tevekenyseg" => "szuperhős",
18:                   "eletkor" => 24,
19:                   "kulonleges kepesseg" => "nagyon erős"
20:                   ),
21:              array (
22:                   "nev" => "Mari",
23:                   "tevekenyseg" => "főgonosz",
24:                   "eletkor" => 63,
25:                   "kulonleges kepesseg" =>
                      "nanotechnológia"
26:                   )
27:              );
```

7.3. program (folytatás)

```
28: foreach ( $karakterek as $karakter )
29:              {
30:              foreach ( $karakter as $kulcs => $ertek )
31:                   {
32:                   print "$kulcs: $ertek<br>";
33:                   }
34:              print "<br>";
35:              }
36: ?>
37: </body>
38: </html>
```

A 7.3. példaprogram kimenetét a 7.3. ábrán láthatjuk. Két `foreach` ciklust használunk. A külső ciklusban a számmal indexelt `$karakterek` tömb elemeit vesszük végig, az egyes elemek pedig a `$karakter` nevű változóba kerülnek. A `$karakter` maga is egy tömb, ezért ennek értékeit is egy `foreach` ciklussal járjuk végig. Mivel a `$karakter` egy asszociatív tömb, a `foreach` szerkezetet az ennek megfelelő formában használjuk, a kulcs-érték párok pedig a `$kulcs` és az `$ertek` változókba kerülnek.

Annak érdekében, hogy ez a módszer jól működjön, biztosnak kell lennünk abban, hogy a `$karakter` változó valóban mindig tömböt tartalmaz. A kódot megbízhatóbbá tehetnénk, ha meghívnánk a `$karakter` változóra az `is_array()` függvényt. Az `is_array()` függvény számára egy paramétert kell megadnunk. A visszatérési érték `true` lesz, ha a változó típusa tömb, minden más esetben `false` értéket ad.

Bonyolultabb programok esetében gyakran előfordul, hogy kíváncsiak vagyunk egy tömb vagy valamilyen más összetett változó tartalmára. Ilyenkor általában nem kell saját tömblistázó kódot használnunk, hiszen a PHP 4 biztosítja számunkra a print_r() függvényt, amely a paramétereként megadott változó tartalmát írja ki.

Műveletek tömbökkel

Most már fel tudunk tölteni tömböket elemekkel, elérhetjük azokat, de a PHP rengeteg függvénnyel segíti a tömbök feldolgozását is. A Perlben jártas olvasó bizonyára ismerősnek fogja találni ezen függvények nagy részét.

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. Az alábbi példában létrehozunk két tömböt, egyesítjük azokat, majd végigjárjuk a keletkező harmadik tömböt:

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

A `$harmadik` tömb az `$elso` és a `$masodik` tömbök elemeinek másolatát tartalmazza. A `foreach` ciklus ennek a tömbnek az elemeit (`"a"`, `"b"`, `"c"`, `1`, `2`, `3`) írja ki a böngészőbe. Az egyes elemeket a `<br>` (sortörés) választja el egymástól.

Az array_merge() függvény a PHP 4-es változatában került a nyelvbe.

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. Figyeljük meg, hogy az `array_merge()` függvénnyel ellentétben az `array_push()` megváltoztatja első paraméterének értékét. E függvény visszatérési értéke a tömbben lévő elemek száma (miután az elemeket hozzáadta). Hozzunk létre egy tömböt, majd adjunk hozzá néhány elemet:

```
$elso = array( "a", "b", "c" );
$elemszam = array_push( $elso, 1, 2, 3 );

print "Összesen $elemszam elem van az \$elso tömbben<P>";
foreach( $elso as $ertek )
    {
    print "$ertek<br>";
    }
```

Mivel az `array_push()` függvény visszaadja a módosított tömb elemeinek számát, az értéket (ami ebben az esetben 6) egy változóban tárolhatjuk és kiíratjuk a böngészőbe. Az `$elso` tömb ekkor az eredeti három betűt tartalmazza, melyekkel az első sorban feltöltöttük, illetve a három számot, amit az `array_push()` függvény segítségével adtunk hozzá. Az így létrehozott tömb elemeit a `foreach` ciklus segítségével kiíratjuk a böngészőbe.

Figyeljük meg, hogy a `"$elso"` karakterlánc elé egy fordított perjelet kellett tennünk. Ha a PHP egy macskakörmös (kettős idézőjeles) karakterláncban egy dollárjelet követő betű- és/vagy számsorozatot talál, változóként próbálja értelmezni azt és megkísérli az értékét beírni. A fenti példában mi a `'$elso'` karaktersorozatot szerettük volna megjeleníteni, ezért kellett a dollárjel elé a fordított perjel karakter. Így a PHP már nem próbálja meg változóként értelmezni, hanem kiírja a karaktereket. Ezt a módszert a szakirodalomban gyakran `escape`-ként emlegetik.

A Perlben gyakorlattal rendelkezők figyeljenek arra, hogy ha a PHP-ben az array_push() második paramétereként tömböt adnak meg, a tömb egy tömb típusú elemmel bővül, ezáltal többdimenziós tömb keletkezik. Ha két tömb elemeit szeretnénk egyesíteni, használjuk az array_merge() függvényt.

Ha egy karakterláncban nem szeretnénk, hogy a változók behelyettesítődjenek, a karakterlánc jelölésére egyszeres idézőjeleket használjunk; ekkor a ’ és a \ karakterek helyett \’-t és \\-t kell írnunk.

Az első elem eltávolítása az array_shift() függvény segítségével

Az `array_shift()` eltávolítja a paraméterként átadott tömb első elemét és az elem értékével tér vissza. A következő példában az `array_shift()` függvény segítségével eltávolítjuk a tömb első elemét. Ezt a műveletet egy `while` ciklusba tesszük és a műveletet addig ismételjük, amíg el nem fogy a tömb. Annak ellenőrzésére, hogy van-e még elem a tömbben, a `count()` függvényt használjuk.

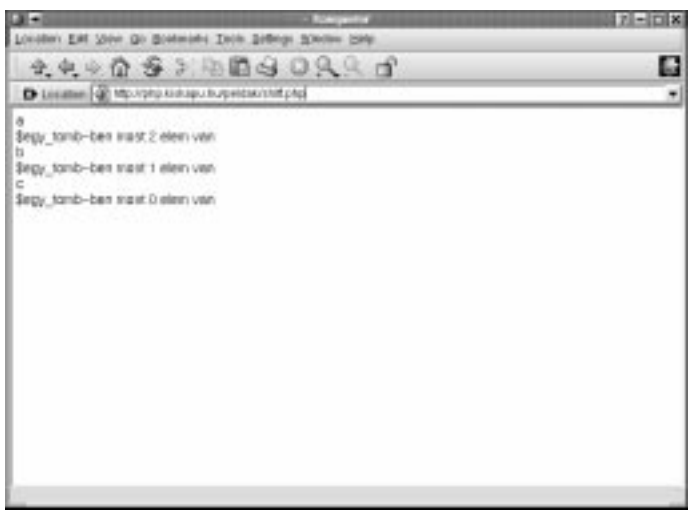
```
<?php
$egy_tomb = array( "a", "b", "c" );

while ( count( $egy_tomb ) )
    {
    $ertek = array_shift( $egy_tomb );
    print "$ertek<br>";
    print '$egy_tomb-ben most ' . count( $egy_tomb )
          . 'elem van<br>';
    }
?>
```

A program kimenetét a 7.4. ábrán láthatjuk.

7.4. ábra

A tömb egyes elemeinek törlése és kiíratása az array_shift() függvény segítségével



Az `array_shift()` akkor lehet hasznos, amikor egy sor elemein kell valamilyen tevékenységet végezni, amíg a sor ki nem ürül.

Az array_shift() függvény a PHP 4-es változatában került a nyelvbe.

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 paraméterként átadott tömböt, hanem újat ad vissza, amelyben a hívó által kért elemek vannak.

Az alábbi példában létrehozunk egy tömböt, majd egy új, három elemű tömböt állítunk elő belőle:

```
$elso = array( "a", "b", "c", "d", "e", "f" );
$masodik = array_slice( $elso, 2, 3 );

foreach( $masodik as $ertek )
       {
       print "$ertek<br>";
       }
```

A fenti példa a `"c"`, `"d"` és `"e"` elemeket írja ki, a `<br>` kódelemmel elválasztva. Mivel kezdőpozícióként a kettőt adtuk meg, így az első elem, ami a `$masodik` tömbbe kerül, az `$elso[2]`.

Ha kezdőpozícióként negatív számot adunk meg, a tömbszelet első eleme hátulról a megadott számú elemtől kezdődik. Ez azt jelenti, hogy ha a második paraméter értéke -1, a tömbszelet az utolsó elemtől fog kezdődni.

Ha hossz paraméterként nullánál kisebb számot adunk meg, a visszakapott tömbszelet a tömb hátulról a megadott számú eleméig tart.

Az array_slice() függvény a PHP 4-es változatában jelent meg.

Tömbök rendezése

Talán a rendezés a legnagyszerűbb művelet, amit egy tömbön el lehet végezni. A PHP 4 függvényeinek köszönhetően egykettőre rendet teremthetünk a káoszban. A következő rész néhány olyan függvényt mutat be, amelyek számmal és karakterlánccal indexelt tömböket rendeznek.

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, szám szerint történik. A függvénynek nincs visszatérési értéke, a paraméterként kapott tömböt alakítja át.
Ebből a szempontból különbözik a Perl hasonló függvényétől. Az alábbi programrészlet egy karakterekből álló tömböt hoz létre, rendezi, majd a rendezett tömb elemeit egyenként kiírja:

```
$tomb = array( "x", "a", "f", "c" );
sort( $tomb );

foreach ($tomb as $elem)
{
     print "$elem<br>";
}
```

Nézzük egy kis példát egy tömb kétféle rendezési módjára!

```
$tomb = array( 10, 2, 9 );
sort( $tomb );

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

print '___Rendezés ábécésorrend szerint___<br>';
foreach ($tomb as $elem)
      {
      print "$elem<br>";
      }
```

A fenti példa kimenete:

```
___Rendezés szám szerint___
2
9
10
```

```
___Rendezés ábécésorrend szerint___
10
2
9
a
```

A sort() függvényt ne használjuk asszociatív (karakterlánccal indexelt) tömbökre! Ha mégis megtesszük, azt fogjuk tapasztalni, hogy a tömb elemei ugyan rendezettek lesznek, de az elemek kulcsai elvesznek, az egyes elemek így nullától kezdődő számokkal érhetők el, ugyanis a tömb számmal indexelt tömbbé alakul át.

Ha csökkenő sorrendben szeretnénk rendezni egy tömb elemeit, használjuk a `sort()` függvényhez hasonlóan működő `rsort()` függvényt.

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 => $ertek )
{
     print "$kulcs = $ertek<br>";
}
```

Ennek a programocskának a kimenetét a 7.5. ábrán láthatjuk.

7.5. ábra

Asszociatív tömb érték szerinti rendezése az asort() függvény segítségével

Ha csökkenő sorrendben szeretnénk rendezni egy asszociatív tömb elemeit, használjuk a `arsort()` függvényt.

Asszociatív tömb rendezése kulcs szerint a ksort() függvény segítségével

A `ksort()` a paraméterben megadott asszociatív tömböt rendezi kulcs szerint. A többi tömbrendező függvényhez hasonlóan ez a függvény sem ad vissza semmiféle visszatérési értéket, hanem a tömböt alakítja át:

```
$tomb = array( "x" => 5, "a" => 2, "f" => 1 );

ksort( $tomb );

foreach( $tomb as $kulcs => $ertek )
{
     print "$kulcs = $ertek<BR>";
}
```

A program kimenetét a 7.6. ábrán láthatjuk.

7.6. ábra

Asszociatív tömb rendezése kulcs szerint a ksort() függvény segítségével

Összefoglalás

Ebben az órában a tömbökről és a hozzájuk kapcsolódó eszközökről tanultunk, melyeket a PHP 4 a rajtuk végzett műveletekhez nyújt. Most már tudunk normál (számmal indexelt) és asszociatív (karakterlánccal indexelt) tömböket is létrehozni, illetve tömböket bejárni a `foreach` ciklus segítségével.
Többdimenziós tömböket is létre tudunk hozni, a bennük levő információt pedig ki tudjuk nyerni egymásba ágyazott `foreach` ciklusokkal. A tömbökhöz egyszerre több elemet is adhatunk, illetve onnan kivehetünk. Végül megtanultuk, hogy a PHP 4-es változata milyen lehetőségeket nyújt a tömbök rendezésére.

Kérdések és válaszok

Ha a foreach ciklus csak a PHP 4-esben került a nyelvbe, akkor a PHP 3-as változatát használó programozók hogyan jártak végig egy tömböt?
A PHP 3-asban az `each()` függvényt használták egy `while()` ciklusban. Erről bővebben a tizenhatodik órában olvashatunk.

Van olyan tömböt kezelő függvény, amellyel nem foglalkoztunk ebben az órában?
A PHP 4-es változatában nagyon sok tömbkezelő függvény van. Erről bővebben a tizenhatodik órában fogunk tanulni. A kézikönyv erről szóló része megtalálható az Interneten, a `http://www.php.net/manual/en/ref.array.php` címen. (A kézikönyv magyar változata a `http://hu.php.net/manual/hu/` oldalon található.)

Ha a tömb elemeinek számát ismerem, normál tömb bejárására a `for` ciklust használjam? (Ekkor számláló segítségével címzem meg a tömb elemeit.)
Nagyon óvatosan kell bánni ezzel a megközelítéssel. Nem lehetünk biztosak abban, hogy az általunk használt tömb indexei 0-tól indulnak és egyesével nőnek.

Műhely

A műhelyben kvízkérdések találhatók, melyek segítenek megszilárdítani az órában szerzett tudást. A válaszokat az A függelékben helyeztük el.

Kvíz

1. Melyik függvénnyel hozhatunk létre tömböket?

2. Az alább létrehozott tömb utolsó elemének mi az indexe?
```
$torpek = array( "Tudor", "Vidor", "Kuka" );
```

3. Függvények használata nélkül hogyan tudnánk `"Hapci"`-t az imént létrehozott `$torpek` nevű tömbhöz adni?

4. Melyik függvény segítségével tudnánk `"Morgó"`-t a `$torpek` nevű tömbbe felvenni?

5. Hogyan lehet egy tömb elemszámát megtudni?

6. A PHP 4-es változatában mi a legegyszerűbb módja egy tömb bejárásának?

7. Milyen függvény segítségével lehet két tömböt egyesíteni?

8. Hogyan rendeznénk egy asszociatív tömböt kulcsai szerint?

Feladatok

1. Hozzunk létre egy többdimenziós tömböt, amely filmeket tartalmaz, zsáner szerint rendezve! Pontosabban hozzunk létre egy asszociatív tömböt, amelynek kulcsai filmtípusok (`"Scifi"`, `"Akció"`, `"Romantikus"`…)! A tömb elemei legyenek tömbök, amelyek filmcímeket tartalmaznak (`"2001"`, `"Alien"`, `"Terminator"`,...)!

2. Járjuk végig az előző feladatban létrehozott tömböt és írjuk ki az összes filmtípust és a hozzá tartozó filmcímeket!