Tömbök

A könyv előző részében a változót egy fiókhoz hasonlítottuk, amely egy értéket képes tárolni. Egy összetett tároló szerkezet a Tömb, amelyet egy fiókos szekrényhez hasonlítunk. A fiókos szekrényben a fiókoknak sorszámuk van, és a fiókokban tárolódnak az értékek, továbbra is egy fiókban egy érték helyezkedik el.

A változó is a memóriában helyezkedik el és a tömb is a memóriában helyezkedik el, azaz hosszútávú tárolásra alkalmatlan.

5-6 értéket szeretnénk tárolni, az eddigi imsereteinkkel, 5-6 változót kellene létrehozni és kezelni, nehéz lenne egy egységként kezelni. Szeretnénk egy táblázat értékeit egy egységként kezelni, egy ciklusban kiíratni ehhez tömböt kell létrehozni. Egy ciklusban nem lehet 5 különböző változót kezelni, ez csak egyenként menne, ami nem egy hatékony kifejezési mód.

Tömbök fajtái

A tömböket több féle csoportosításban is nézhetjük.

Sorszámozott ➔indexelt tömb

Címkézett ➔ asszociatív tömb

Vegyes ➔ indexelt és asszociatív tömb is egyszerre.

Másik lehetséges csoportosítás:

Egy dimenziós (vector)

Két dimenziós(mátrix)

Sok dimenziós (multi dimenziós)

A folytatásban először a sorszámozott tömb kezelését nézzük meg, majd áttérünk a címkézett vagy asszociatív tömb használatára.

Tömb létrehozása

```
<?php
// Első mód
$tomb = array(5,6,7,8,9);
// Másik mód
$tomb1 = [5,6,7,8,9];
```

Mindkét változat megfelelő, az első kicsit hosszabb és többet kell írni hozzá.

Tömbhöz elemek hozzáadása

```
<?php
//üres tömb inicializálása
$tomb = [];
$tomb[] = 5;
$tomb[] = 76;
```

A fenti példában az első sorban felmerül az üres tömb fogalma, tömb szerkezet, de nincs egy eleme sem. Az azt követő sorokban a tömb elemeinek megadásakor automatikusan sorszámozódik a tömb indexe.

Tömb feltöltése array_fill függvény segítségével. Ez a függvény tól-ig indexek alapján egyforma értékekkel tölt fel egy tömböt. Ezt akkor érdemes használni, ha nagy mennyiségű azonos értékkel töltünk fel egy tömböt, pl.: sok nullával.

```
<?php
$a = array_fill(5, 6, 'banana');
print_r($a);
```

Tömb elemeinek elérése

```
<?php
$tomb = [4,5,64,12];
print($tomb[3]);
```

Tömb méretének lekérése count függvénnyel.

```
<?php
$tomb = [4,5,64,12];
print(count($tomb));
```

Tömb utolsó elemének lekérése a count függvény segítségével.

```
<?php
$tomb = [4,5,64,12];
$utolsoIndex = count($tomb)-1;
print($tomb[$utolsoIndex]);
```

Figyeljük meg, hogy egyet ki kellett vonni a hosszból, mivel a tömbök 0 -tól sorszámozódnak, emiatt a hossz az utolsó elem utánra mutat, így mindig ki kell vonni belőle egyet.

A tömb utolsó elemének lekérése az end függvény segítségével.

```
<?php
$tomb = [4,5,64,12];
print(end($tomb));
```

A tömb méretének lekérdezése akkor sima ügy, ha szépen sorfolytonosan van indexelve, ha ez nem így van, akkor megkapjuk a méretét , de nem tudjuk sorfolytonosan bejárni. A PHP ben eléggé rugalmas a tömb kezelés, így ez előfordulhat.

```
<?php
$tomb = [0=>4,34=>5,39=>64,45=>12];
$db = count($tomb);
print($db);
```

Tömb elemeinek bejárása, for ciklussal vagy kifejezetten erre a célra való foreach ciklussal.

```
<?php
$tomb = [4,5,64,12];
for($index = 0; $index < count(); $index++){
      print($tomb[$index];
}
```

```
<?php
$tombElemek = [4,5,64,12];
foreach($tombElemek as $key => $tombElem){
      print($tombElem);
}
```

Tömb elemeinek kiírása hibakeresés céljából var_dump függvénnyel.

```
<?php
$tombElemek = [4,5,64,12];
var_dump($tombElemek);
```

A két bejárási mód közötti különbség, hogy a for ciklusnál, nem kötelező egyesével növelni az index értékét, azaz könnyedén lehet csak a páros indexeket lekérni, addig a foreach mindenképen bejárja a tömböt.

Tömb elem(ek) törlése.

```
<?php
$tombElemek = [4,5,64,12];
unset($tombElemek[2]);
```

ennek is van alternatívája az array_splice függvénnyel, lehet egy indextől meghatározott számű elemet kitörölni.

```
<?php
$autok = ["Mercedes", "BMW", "Ford"];
array_splice($autok, 1, 1);
```

Az utolsó elem törlése

```
<?php
$autok = ["Mercedes", "BMW", "Ford"];
array_pop($autok);
```

Az első elem törlése

```
<?php
$autok = ["Mercedes", "BMW", "Ford"];
array_shift($autok);
```

Elem törlése array_diff függvény használatával (itt az eredeti tömbből nem kerül ki az elem, hanem egy új változóba kerül a tömb, a kijelölt elem nélkül.

```
<?php
$autok = ["Mercedes", "BMW", "Ford"];
$masAutok = array_diff($autok, "BMW");
```

A “BMW” érték nélkül kerül bele a masAutok változóba a tömb összes eleme. Az eredeti tömb megmarad.

Asszociatív tömb kezelése és használata

Asszociatív tömb elemek felvétele

```
<?php
$asszocTombElemek = ["cimke1" =>4, "cimke2"=>5, "cimke3"=>64,
"cimke4"=>12];
var_dump($asszocTombElemek);
```

Asszociatív tömb elemek felvétele máshogyan

```
<?php
$asszocTombElemek["cimke1"] = "érték 1";
$asszocTombElemek["cimke2"] = "érték 2";
$asszocTombElemek["cimke3"] = "érték 3";

var_dump($asszocTombElemek);
```

Asszociatív tömb gyakorlati haszna, hátránya és a for ciklus egy másik változata.

Az asszociatív tömb elemeit címkékkel láthatjuk el, ez alapján rekordokat is lehet velük ábrázolni. (Rekord / Record : egy oszlopokkal ellátott sor azaz egy táblázat egy soraként kell elképzelni).

Név	Cím	Telefonszám	Magasság
Kikel Molly	Bp 2. Kerület Sziriátoszlopai utca 2	+36(79)7123-45-67	182

```
<?php
$adatok["nev"]              = "Kikel Molly";
$adatok["cim"]              = "Bp 2. Kerület Sziriátoszlopai utca 2";
$adatok["telefonszam"]      = "+36(79)7123-45-67";
$adatok["magassag"]         = "182";

var_dump($adatok);
```

Ennek a tömb típusnak a hétránya, hogy sima for ciklussal nem lehet bejárni, mivel nem számokkal van indexelve, hanem címkékkel, amely egy string. Ennek a feloldására készült el a foreach ciklus.

Címkézett vagy asszociatíív tömb bejárása

```
<?php
$asszocTombElemek = ["cimke1" =>4, "cimke2"=>5, "cimke3"=>64,
"cimke4"=>12];
Foreach($asszocTombElemek as $kulcs => $asszocTombElem) {
     print($asszocTombElem);
}
```

A fenti példában láthatjuk a foreach ciklus lehetőségét, bejárja a teljes tömböt és a kulcs változóba teszi a címkét az asszocTombElem változóba pedig az aktuális értéket teszi.

Tömbök lehetőségei (több dimenzió , kevert , hierarchia)

Két dimenziós tömb

A két dimenziós tömböt több feladatra is lehet használni, táblázat tárolása, 2 dimenziós kép ábrázolására. A fenti tömböt fel lehet fogni, úgy hogy tömb a tömbben. A tömb egyik eleme egy másik tömb. A PHP eléggé rugalmas, az egyes tömbök akár eltérő hosszúságúak is lehetnek.

Két dimenziós tömb egy elemére hivatkozás:

```
<?php
$d2Tomb =[0=>[1,2,3], 1=>[4,5,6]];
print($d2Tomb[1][2]);
```

```
Eredmény: 6
```

A két dimenziós tömb méretét lekérdezve nem a teljes tömb méretét kapjuk meg, csak az egyik dimenzió hosszát.

Két dimenziós tömb mérete

```
<?php
$d2Tomb =[0=>[1,2,3], 1=>[4,5,6]];
print(count($d2Tomb));
```

```
Eredmény: 2
```

Ha a másik dimenzió hosszát akarnánk lekérdezni, akkor az egyik indexet meg kell adni, az abban elhelyezkedő tömb hosszát kapjuk meg.

```
<?php
$d2Tomb =[0=>[1,2,3], 1=>[4,5,6]];
print(count($d2Tomb[0]));
```

```
Eredmény: 3
```

Két dimenziós tömb bejárása

```
<?php
$d2Tomb =[0=>[1,2,3], 1=>[4,5,6]];
for($sor=0; $sor < count($d2Tomb); $sor ++){
      for($oszlop = 0; $oszlop < count($d2Tomb[$sor]); $oszlop++){
```

```
            print($d2Tomb[$sor][$oszlop]);
        }
}
```

“Kevert” tömb alatt az indexelt és címkézett tömböt értjük. Ez lehet index és címke egymás után. Lehet olyan tömb, ahol indexelve Vannak a címkézett tömbök, jellemzően egy táblázat sorait tároljuk így.

```
<?php
$vegyesTomb = [0 => "elem", "cimke"=>34]
```

Itt vegyesen van az index és a címke.

```
<?php
$karakterek = [
["nev"=>"Kiss József", "telefonszam" => "+3675333-33-33", "cim" => "Budapest Tolnai Lajos utca 567."],
["nev"=>"Nagy Péter", "telefonszam" => "+3682444-55-66", "cim" => "Budapest Hernád utca 900."],
["nev"=>"Veress Károly", "telefonszam" => "+3699888-88-88", "cim" => "Budapest Merényi Károly utca 423."]
]
```

Ez egy sorszámozott / indexelt tömb, amelynek elemei címkézett / asszociatív tömbök.

Műveletek és lehetőségek a tömbökkel

A PHP igen gazdagon el van látva tömb függvényekkel. Tömbök egyesítése, rendezése, tömb részletek kiemelése, tömb feltöltése véletlen elemekkel, tömb eleminek (szám esetén) összegzése.

Két tömb egyesítése array_merge függvénnyel.

```
<?php
$tomb1 = ["szin" => "piros", 5, 7, 2, 4];
$tomb2 = ["a", "b", "szin" => "zöld", "forma" => "trapezoid", 9];
$eredmeny = array_merge($tomb1, $tomb2);
var_dump($eredmeny);
?>
```

A fenti példában érdemes megfigyelni, hogy a két tömbben szereplő azonos kulcsok esetén a az utolsó kulcshoz tartozó értéket veszi fel az új egyesített tömbbe.

```php
<?php
$tomb1 = [0 => 'zero_a', 2 => 'two_a', 3 => 'three_a'];
$tomb2 = [1 => 'one_b', 3 => 'three_b', 4 => 'four_b'];
$eredmeny = $tomb1 + $tomb2;
var_dump($eredmeny);
```

Ha az első tömbhöz szeretnénk hozzáadni a második tömbből az összes elemet, úgy hogy ne írja felül az első tömb elemeit, és nem szeretnénk újraindexelni az egész szerkezetet, akkor használjuk a + tömbegyesítő operátort:

Egyszerre több elem hozzáadása a tömb végéhez. Az array_push függvény használatával.

```php
<?php
$verem = ["narancs", "alma", "körte"];
array_push($verem, "banán", "málna");
var_dump($verem);
```

Ha a tömb elejéhez szeretnénk hozzáfűzni elemeket, akkor használjuk az array_unschift függvényt.

```php
<?php
$sor = ["narancs", "alma", "körte"];
array_unshift($sor, "banán", "málna");
var_dump($sor);
```

A tömb egy részének kinyerése array_slice függvénnyel

Tömbök rendezése

Tömb szűrése

Tömb elem keresése

Több diemnziós tömb egy oszlopának kiemelése

Tömb kulcsainak kinyerése

Tömb értékeinek kinyerése

Adott kulcs létezése a tömbben

Sorfolytonos számozás ellenörzése (array_is_list)

Első / utolsó kulcs visszaadása

Tömb bejárásához kapcsolodó függvények

Current

Prev

Next

Reset

Gyakorlatok

Bankszámlaszám ellenőrzése

Taj szám ellenőrzése

- Alapvető programozási tételek
  - Összegzés
  - Megszámlálás
  - Eldöntés
  - Kiválasztás
  - Keresés
  - Maximumkiválasztás