Backend 12

Asszociatív tömbök, tömbfüggények és include

Rostagni Csaba

2025. október 7.

Tartalom

Asszociatív tömb

- Asszociatív tömb esetén nem számmal, hanem valamilyen kulcs segítségével indexeljük az elemeket.
- A kulcs lehet szöveg is.
- A **kulcs** és az **érték** között "nyíl" (=>) található

Asszociatív tömb felépítése:

Doksi

```
$asszociativTomb = [ kulcs => érték ];
```

Például:

PHP

```php
$hetnapjai = [
    "hétfő" => 1,
    "kedd" => 2,
    "szerda" => 3,
];
```

Asszociatív tömb példa

PHP
```php
$nagyAlex = [
    "nev" => "Nagy Alex",
    "kor" => 23,
    "atlag" => 4.27,
];
```

- A **$nagyAlex** egy asszociatív tömb, amiben három elem van.
- A **nev** szöveges, a **kor** egész, míg az **atlag** eleme valós szám.
- Kulcsok:
  - nev
  - kor
  - atlag
- Értékek:
  - Nagy Alex
  - 23
  - 4.27

Tartalom

Asszociatív tömb kiírása

- Előfordulhat, hogy szeretnénk látni egy tömb tartalmát és a felépítését.
- Tesztelési célzattal felfedhetjük a tömbünk adatait.

Rossz példa!

```
$at = [
    "nev" => "Nagy Alex","kor" => 23,"atlag" => 4.27,
];
echo $at;
```

Eredmény

```
Array
```

Figyelem!

A tömb egy összetett adatszerkezet, így egy egyszerű `echo` paranccsal nem tudjuk kiírni a tartalmát.

Asszociatív tömb kiírása print_r() segítségével

- A `print_r()` függvény rekurzívan írja ki a tömb tartalmát.
- Megjeleníti a tömb elemeihez tartozó kulcsokat is.

PHP

```php
$at = [
  "nev" => "Nagy Alex",
  "kor" => 23,
  "atlag" => 4.27,
];
print_r($at);
```

Eredmény

```
Array
(
    [nev] => Nagy Alex
    [kor] => 23
    [atlag] => 4.27
)
```

Figyelem!

Csak és kizárólag tesztelésre alkalmazható!

Asszociatív tömb kiírása var_dump() segítségével

- A `var_dump()` függvény rekurzívan írja ki a tömb tartalmát.
- Megjeleníti a tömb elemihez tartozó indexeket és a típusokat is.

PHP

```php
$at = [
  "nev" => "Nagy Alex",
  "kor" => 23,
  "atlag" => 4.27,
];
var_dump($at);
```

Eredmény

```
array(3) {
  ["nev"]=>
    string(9) "Nagy Alex"
  ["kor"]=>
    int(23)
  ["atlag"]=>
    float(4.27)
}
```

Figyelem!

Csak és kizárólag tesztelésre alkalmazható!

Asszociatív tömb elemeinek kiírása

PHP

```php
$at = [
    "nev" => "Nagy Alex",
    "kor" => 23,
    "atlag" => 4.27,
];

echo "$at[nev]\n";
echo "{$at["kor"]}\n";
echo $at["atlag"];
```

Eredmény

```
Nagy Alex
23
4.27
```

- Idézőjelek között kiírva a kulcs idézőjeleit el kell hagyni! (nev)
- Idézőjelen belül lévő kapcsos zárójelben viszont ki kell tenni!(kor)
- Ha nem szövegbe írjuk be, hanem magában, akkor ki kell tenni a kulcs köré az idézőjeleket. (atlag)

Tömb kiírása foreach ciklussal

"Nagy Alex"	23	4.27
"nev"	"kor"	"atlag"

PHP

```php
$at = ["nev" => "Nagy Alex","kor" => 23,"atlag" => 4.27,];

foreach($at as $adat)
{
    echo "$adat\n";
}
```

Eredmény

```
Nagy Alex
23
4.27
```

Tömb kiírása foreach ciklussal (+kulcs)

"Nagy Alex"	23	4.27
"nev"	"kor"	"atlag"

PHP

```php
$at = ["nev" => "Nagy Alex","kor" => 23,"atlag" => 4.27,];

foreach($at as $kulcs => $ertek)
{
    echo "$kulcs: $ertek\n";
}
```

Eredmény

```
nev: Nagy Alex
kor: 23
atlag: 4.27
```

Tartalom

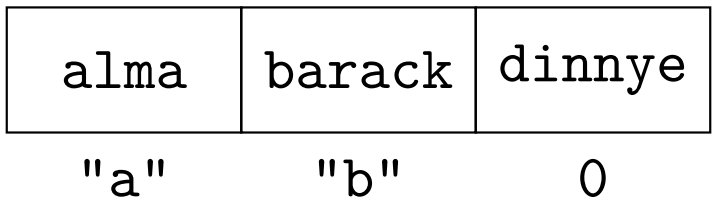
Elem hozzáadása asszociatív tömbhöz

- A tömb elemeire szögletes zárójelekkel hivatkozhatunk.
- Előfordulhat, hogy felülírunk egy régi értéket.
- Ha kulcsot elhagyjuk sorszámozni kezdi a tömböt.

PHP

```php
$t = [
    "a" => "alma",
    "b" => "banán",
];

$t["b"]  =  "barack";
$t[]     = "dinnye";
```



Tartalom

array_key_exists()

```
array_key_exists(string|int $key, array $array): bool
```

- Az első pataméter a keresett kulcs
- A második paraméter a tömb, amiben keresendő
- A **visszatérési értéke** (`bool`) meghatározza, az adott kulcs megtalálható -e a tömbben

PHP

```php
$t = ["a", "b", "c"];
var_dump(array_key_exists(1, $t));
```

Eredmény

```
bool(true)
```

Linkek:

- array_key_exists() - php.net

Tartalom

Tartalom

array_push()

```
array_push(array &$array, mixed ...$values): int
```

- Egy vagy több elemet fűz a tömb **végéhez**
- Az első paraméter a tömb, ez referencia szerint kerül átadásra
- A további paraméterek a beszúrandó értékek
- A visszatérési értéke a tömb új mérete

PHP

```php
$t = ["a", "b"];
array_push($t, "c");              // Egy elem hozzáadása
array_push($t, "d", "e", "f");    // Több elem hozzáadása
```

Eredmény

```
["a", "b", "c", "d", "e", "f"]
```

Linkek:

- array_push() - php.net

array_pop()

```
array_pop(array &$array): mixed
```

- Kiveszi a tömb **utolsó** elemét
- Az egyetlen paramétere a tömb, ez referencia szerint kerül átadásra
- A visszatérési értéke a tömb utolsó eleme
- Üres tömb esetén `NULL`-t ad vissza

PHP

```php
$t = ["a", "b", "c"];
$utolso = array_pop($t);
var_dump($utolso);
var_dump($t);
```

Eredmény

```
string(1) "c"
array(2) {
    [0]=> string(1) "a"
    [1]=> string(1) "b"
}
```

Linkek:

- array_pop() - php.net

array_shift()

```
array_shift(array &$array): mixed
```

- A tömb **legelső** elemét kiveszi
- A többi elemét eggyel előrébb csúsztatja
- A numerikus indexeket újra számozza 0-tól
- A visszatérési értéke a tömb **legelső** eleme

PHP

```php
$t = ["a", "b", "c"];
$elso = array_shift($t);
var_dump($elso);
var_dump($t);
```

Eredmény

```
string(1) "a"
array(2) {
    [0]=> string(1) "b"
    [1]=> string(1) "c"
}
```

Linkek:

- array_shift() - php.net

array_shift() példa

PHP
```php
$t = [
    99 => "a",
    10 => "b",
    42 => "c"
];
var_dump($t);
```

Eredmény
```
string(1) "a"
array(2) {
    [99]=> string(1) "a"
    [10]=> string(1) "b"
    [42]=> string(1) "c"
}
```

PHP
```php
$else = array_shift($t);
var_dump($else);
var_dump($t);
```

Eredmény
```
string(1) "a"
array(2) {
    [0]=> string(1) "b"
    [1]=> string(1) "c"
}
```

- A tömb numerikus indexeit újra számozta

array_unshift()

```
array_unshift(array &$array, mixed ...$values): int
```

- Egy vagy több elemet fűz a tömb **elejéhez**
- Az első paraméter a tömb, ez referencia szerint kerül átadásra
- A további paraméterek a beszúrandó értékek, sorrendjük megmarad
- A numerikus indexeket újra számozza 0-tól
- A visszatérési értéke a tömb új mérete

PHP

```php
$t = ["a", "b"];
array_unshift($t, "x", "y", "z");  // Több elem hozzáadása
```

Eredmény

```
["x", "y", "z", "a", "b"]
```

Linkek:

- array_unshift() - php.net

Tartalom

array_keys()

```
array_keys(array $array, mixed $filter_value, bool $strict
↪  = false): array
```

- Az első pataméter a tömb
- A második (opcionális) paraméter a szűrő
  - Azokat a kulcsokat adja vissza, ahol az érték megegyezik a szűrővel
- A harmadik (opcionális) paraméter azt szabja meg, hogy szigorúan típusos ellenőrzést végezzen (===)
- A **visszatérési értéke** a kulcsokat tartalmazó tömb

PHP

```php
$t = [
    "nev" => "Piroska",
    "kor" => 22,
    "szin" => "piros"
];
var_dump(array_keys($t));
```

Eredmény

```
array(3) {
    [0] => string(3) "nev"
    [1] => string(3) "kor"
    [2] => string(4) "szin"
}
```

array_keys() példa

PHP
```php
$t = [
    "nev" => "Piroska",
    "kor" => 22,
    "szin" => "piros"
];
var_dump(array_keys($t, "piros"));
```

Eredmény
```
array(1) {
    [1] => string(4) "szin"
}
```

PHP
```php
$t = [
    "nev" => "Piroska",
    "kor" => 22,
    "szin" => "piros"
];
var_dump(array_keys($t, "22"));
```

Eredmény
```
array(1) {
    [0] => string(3) "kor"
}
```

array_keys() példa

PHP
```php
$t = [
    "nev" => "Piroska",
    "kor" => 22,
    "szin" => "piros"
];
var_dump(array_keys($t, "22", true));
```

Eredmény
```
array(0) {
}
```

- A harmadik paramétere szerint pontos egyezést vár el (===)
- 22 !== "22"

PHP
```php
$t = [
    "nev" => "Piroska",
    "kor" => 22,
    "szin" => "piros"
];
var_dump(array_keys($t, 22, true));
```

Eredmény
```
array(1) {
  [0] => string(3) "kor"
}
```

array_values()

```
array_values(array $array): array
```

- Az első pataméter a tömb
- A **visszatérési értéke** az értékeket tartalmazó tömb

PHP

```php
$t = [
    "nev" => "Piroska",
    "kor" => 22,
    "szin" => "piros"
];
var_dump(array_values($t));
```

Eredmény

```
array(3) {
  [0] => string(7) "Piroska"
  [1] => int(22)
  [2] => string(5) "piros"
}
```

Linkek:

- array_values() - php.net

Tartalom

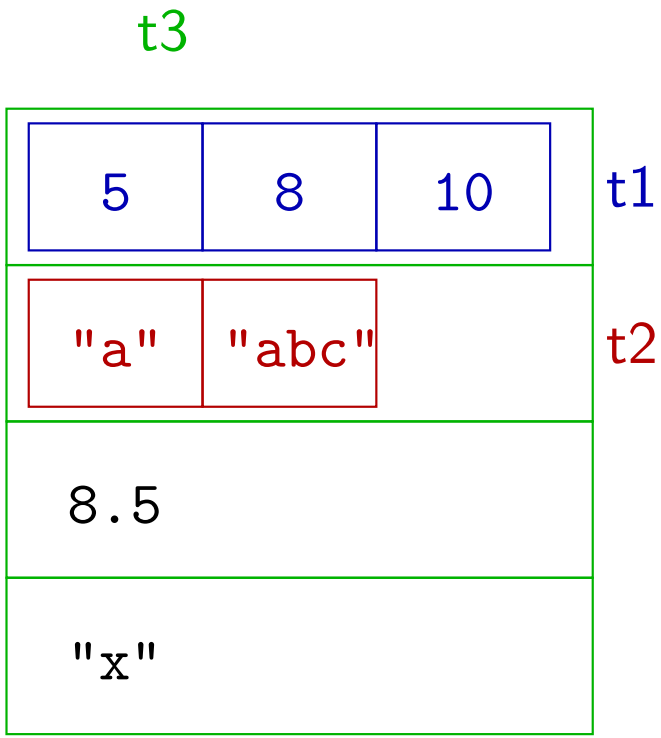
Több dimenziós tömbök

Elkészíthetjük a tömböket külön-külön, majd egymásba pakolhatjuk...

PHP

```php
$t1 = [5, 8, 10];
$t2 = ["a", "abc"];

$t3 = [
    $t1,
    $t2,
    8.5,
    "x"
];
```



Figyelem!

Tömbben csak akkor használunk $ jelet, ha változót szeretnénk beleírni.

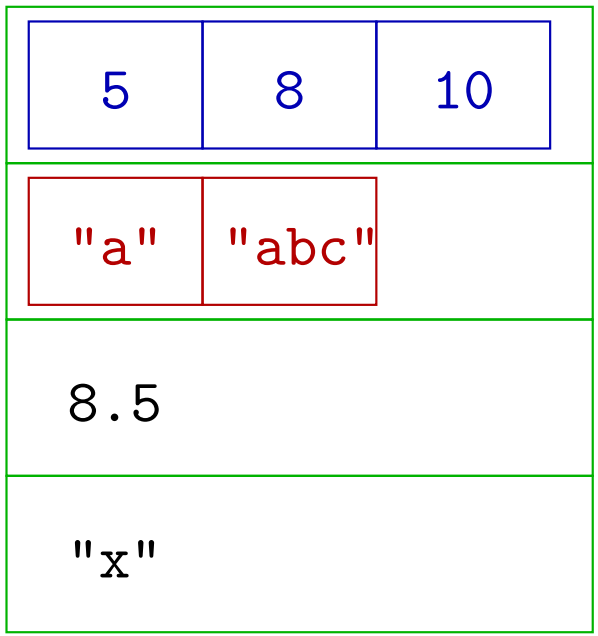
Több dimenziós tömbök

... vagy közvetlen létrehozhatjuk a tömbben a tömböt.

PHP

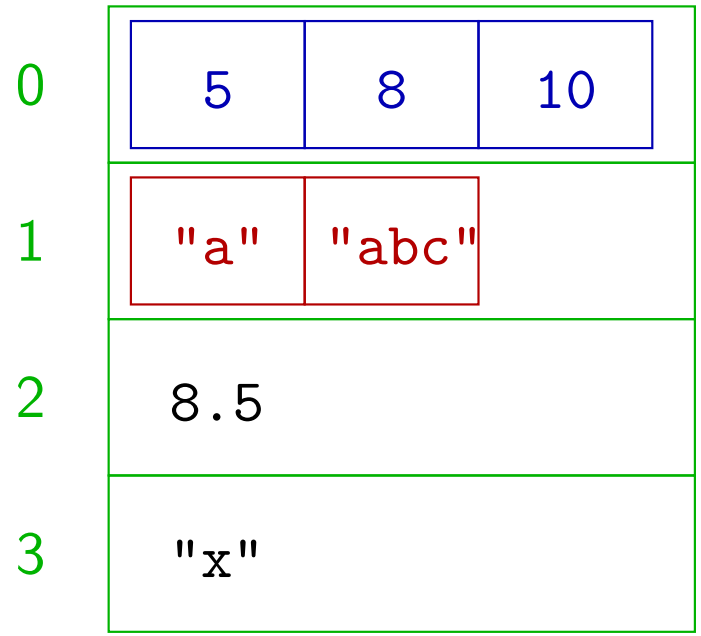
```php
$t3 = [
    [5, 8, 10],
    ["a", "abc"],
    8.5,
    "x"
];
```

t3



Több dimenziós tömbök

PHP
```php
$t3 = [
    [5, 8, 10],
    ["a", "abc"],
    8.5,
    "x"
];
```

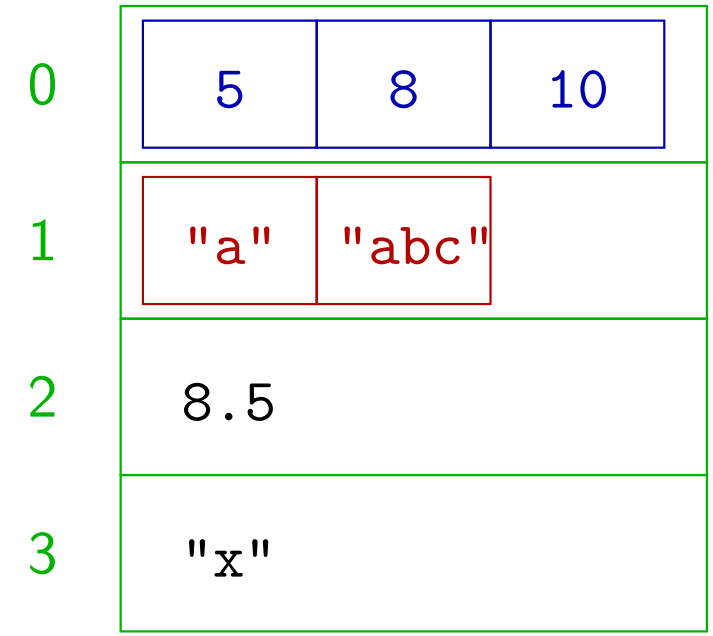


PHP
```php
print_r($t3[0]);
```

Eredmény
```
Array
(
    [0] => 5
    [1] => 8
    [2] => 10
)
```

Több dimenziós tömbök

PHP
```php
$t3 = [
    [5, 8, 10],
    ["a", "abc"],
    8.5,
    "x"
];
```

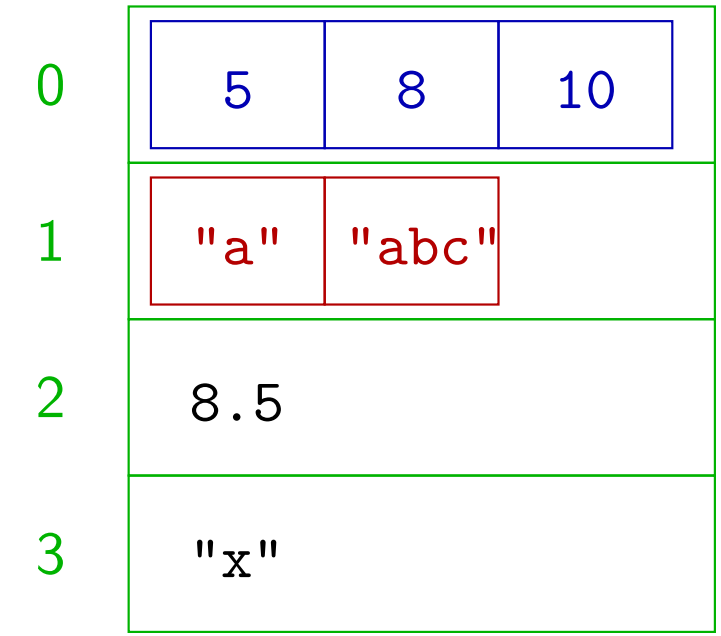


PHP
```php
var_dump($t3[0][1]);
```

Eredmény
```
int(8)
```

Több dimenziós tömbök

PHP
```php
$t3 = [
    [5, 8, 10],
    ["a", "abc"],
    8.5,
    "x"
];
```

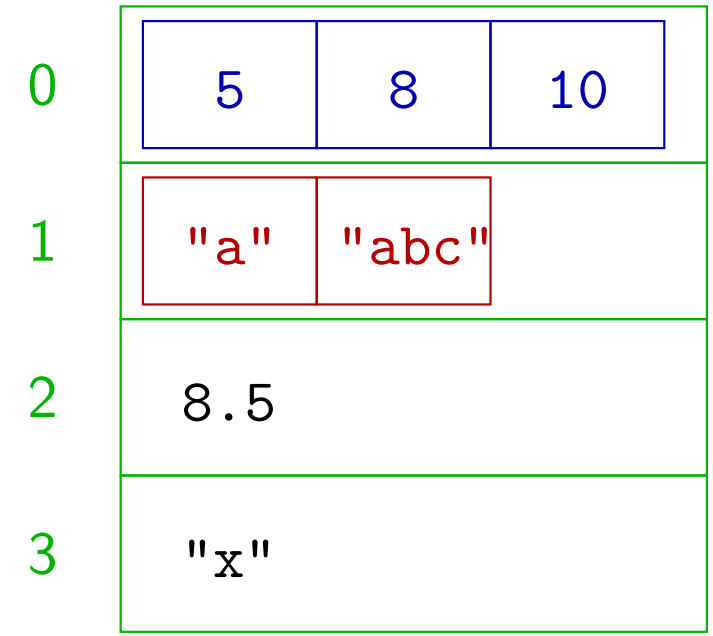


PHP
```php
print_r($t3[1]);
```

Eredmény
```
Array
(
    [0] => a
    [1] => abc
)
```

Több dimenziós tömbök

PHP
```php
$t3 = [
    [5, 8, 10],
    ["a", "abc"],
    8.5,
    "x"
];
```

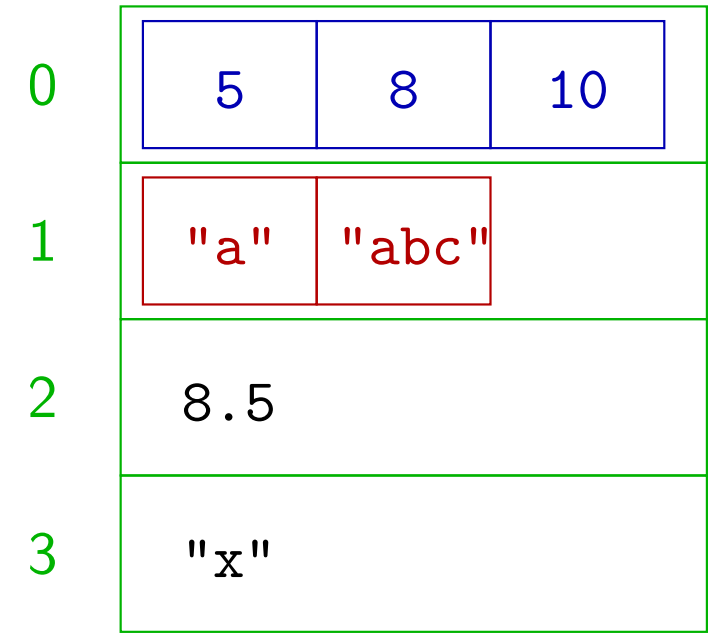


PHP
```php
var_dump( $t3[1][1]);
```

Eredmény
```
string(3) "abc"
```

Több dimenziós tömbök

PHP
```php
$t3 = [
    [5, 8, 10],
    ["a", "abc"],
    8.5,
    "x"
];
```

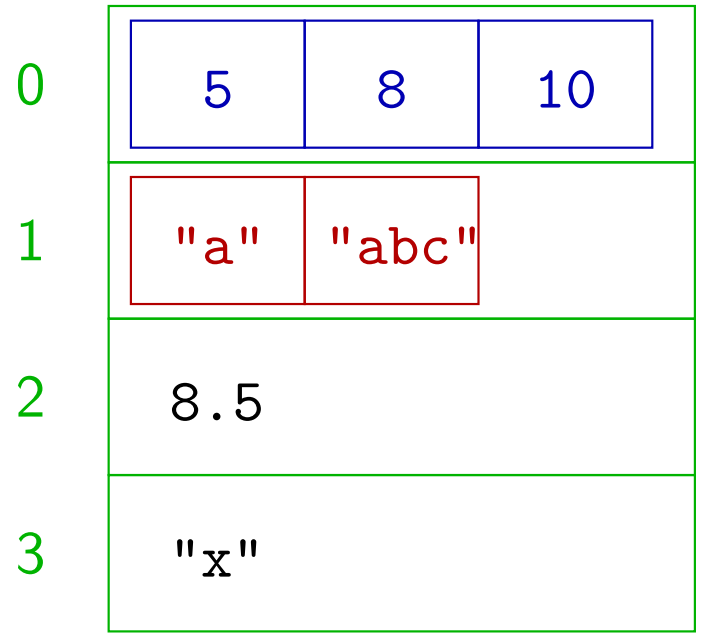


PHP
```php
var_dump( $t3[1][1][2]);
```

Eredmény
```
string(1) "c"
```

Több dimenziós tömbök

PHP
```php
$t3 = [
    [5, 8, 10],
    ["a", "abc"],
    8.5,
    "x"
];
```

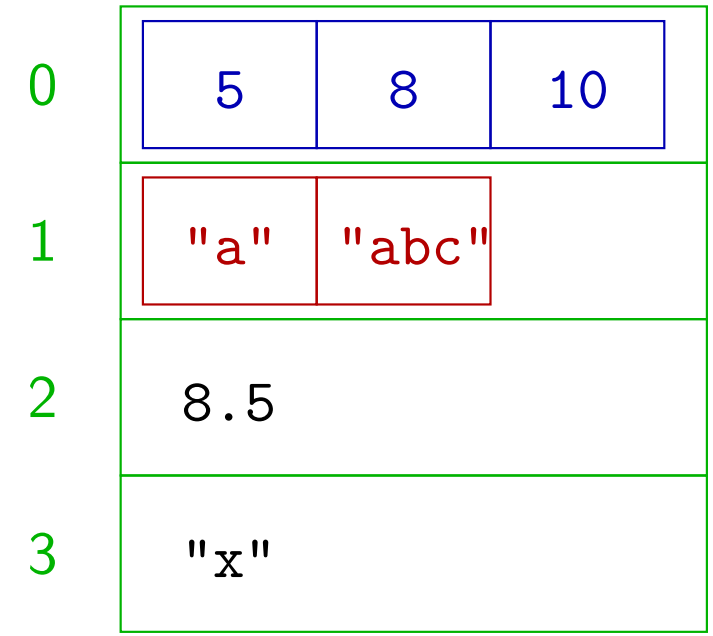


PHP
```php
var_dump( $t3[2]);
```

Eredmény
```
float(8.5)
```

Több dimenziós tömbök

PHP
```php
$t3 = [
    [5, 8, 10],
    ["a", "abc"],
    8.5,
    "x"
];
```



PHP
```php
var_dump( $t3[3]);
```

Eredmény
```
string(1) "x"
```

Több dimenziós tömbök (tanulók példa)

PHP

```php
$petra = [
    "nev" => "Juhász Petra",
    "kor" => 19,
    "targyak" => [
        "angol" => [5,2,5,4],
        "php" => [4,4,3,4],
    ],
];
$laszlo = [
    "nev" => "Budai László",
    "kor" => 18,
    "targyak" => [
            "angol" => [3,3,5,4],
            "php" => [5,2,1,5],
        ],
];
$tanulok = [$petra, $laszlo];
```

Összetettebb adatszerkezet esetén egyszerűbb lehet egy-egy blokkot külön változóként felvenni, majd összeépíteni egy nagy tömbbé.

Több dimenziós tömbök (tanulók példa)

PHP

```php
$t = [
  "petra" => [
    "nev" => "Juhász Petra",
    "kor" => 19,
    "targyak" => [
      "angol" => [5,2,5,4],
      "php" => [4,4,3,4],
    ],
  ],
  "laszlo" => [
    "nev" => "Budai László",
    "kor" => 18,
    "targyak" => [
      "angol" => [3,3,5,4],
      "php" => [5,2,1,5],
    ],
  ],
];
```

Ha átlátjuk a tömböket, akkor elsőre megírhatjuk egy nagy tömbként az összes tanuló adatait.

Figyelem!

Oda kell figyelni, hogy ne írjunk fölösleges $ jeleket!

Több dimenziós tömbök (tanulók példa)

PHP
```php
$t = [
  "petra" => [
    "nev" => "Juhász Petra",
    "kor" => 19,
    "targyak" => [
      "angol" => [5,2,5,4],
      "php" => [4,4,3,4],
    ],
  ],
  "laszlo" => [
    "nev" => "Budai László",
    "kor" => 18,
    "targyak" => [
      "angol" => [3,3,5,4],
      "php" => [5,2,1,5],
    ],
  ],
];
```

Hány éves Petra?

PHP
```php
echo $t["petra"]["kor"];
```

Eredmény

19

Több dimenziós tömbök (tanulók példa)

PHP
```php
$t = [
  "petra" => [
    "nev" => "Juhász Petra",
    "kor" => 19,
    "targyak" => [
      "angol" => [5,2,5,4],
      "php" => [4,4,3,4],
    ],
  ],
  "laszlo" => [
    "nev" => "Budai László",
    "kor" => 18,
    "targyak" => [
      "angol" => [3,3,5,4],
      "php" => [5,2,1,5],
    ],
  ],
];
```

Hány tantárgya van Petrának?

PHP
```php
echo count($t["petra"]["targyak"]);
```

Eredmény

2

Több dimenziós tömbök (tanulók példa)

Mennyi László átlaga angolból?

PHP

```php
$osszeg=0;
$db=count($t["laszlo"]["targyak"]["angol"]);
for($i = 0; $i < $db; $i++ )
{
    $osszeg += $t["laszlo"]["targyak"]["angol"][$i];
}
echo $osszeg/$db;
```

Eredmény

```
3.75
```

Több dimenziós tömbök (tanulók példa)

Jelenítse meg a tanulók nevét és tárgyanként a jegyeiket.

PHP

```php
foreach($t as $tanulo){
    echo $tanulo['nev'] . "\n";

    foreach($tanulo['targyak'] as $targy => $jegyek ) {
        echo "\t{$targy}:";

        foreach($jegyek as $jegy)
        {
            echo "$jegy,";
        }
        echo "\n";
    }
}
```

Több dimenziós tömbök (tanulók példa)

Eredmény

```
Juhász Petra
    angol:5,2,5,4,
    php:4,4,3,4,
Budai László
    angol:3,3,5,4,
    php:5,2,1,5,
```

Több dimenziós tömbök (tanulók példa)

Egy kis trükkel megoldható, hogy ne legyen a sorok végén fölösleges vessző.

PHP

```php
foreach($t as $tanulo){
    echo $tanulo['nev'] . "\n";

    foreach($tanulo['targyak'] as $targy => $jegyek ) {
        echo "\t{$targy}:";

        for($i = 0; $i < count($jegyek)-1; ++$i)
        {
            echo "$jegyek[$i],";
        }
        echo $jegyek[count($jegyek)-1] . "\n";
    }
}
```

Több dimenziós tömbök (tanulók példa)

Eredmény

```
Juhász Petra
    angol:5,2,5,4
    php:4,4,3,4
Budai László
    angol:3,3,5,4
    php:5,2,1,5
```

Tartalom

Az include() függvény

Az include függvény a hívás helyére "másolja" a megadott fájlban lévő kódot és értelmezi is.

- Elsőként a megadott útvonalon keresi a fájlt, amennyiben létezik.
- Utána a konfigban beállított **include_path** mappában keresi.
- Végezetül az aktuális mappában keresi a fájl.

A második lépést kihagyja, ha gyökérkönyvtárral (c:\, vagy /home), egy vagy két ponttal kezdjük a hivatkozást.

PHP
```php
<?php include('menu.php') ?>
```

Az include() és a hatókör (scope)

- A külső fájlban található **változók** hatóköre ugyanaz lesz, mintha az adott sorba írtuk volna őket
- A függvények hatóköre globális lesz.

Linkek:

- PHP dokumentáció: include()
- PHP dokumentáció: include_path

include() vs. require()

```php
<?php include("test.php"); ?>
<?php require("test.php"); ?>
```

- Míg az `include()` hibás fájl esetén figyelmeztetést ad, addig a `require()` esetén megáll a szkript futása végzetes hibával.
- Az oldal működéséhez feltétlen szükséges részeket érdemes a `require()` segítségével hozzá adni a kódunkhoz.

include() vs. include_once()

PHP
```php
<?php include("reklam.php"); ?>
<?php include_once("fuggvenyek.php"); ?>
```

- Míg az `include()` annyiszor "bemásolja" és végrehajtja a fájlt ahányszor szerepel a kódban, addig az `include_once()` csak az első előfordulásakor.
- Ahol gondot okoz az, hogy már szerepelt korábban (például függvényeknél, amiket nem lehet újra definiálni) ott az `include_once`-t célszerű alkalmazni.

include, require, once összefoglaló

Közös: A függvény hívás helyére bemásolja a fájl tartalmát, értelmezi a benne található kódot.

- include()
  - Többszöri híváskor újra végrehajtódik.
  - Hibás fájl esetén figyelmeztet.
- include_once()
  - Többszöri híváskor NEM hajtódik újra végre.
  - Hibás fájl esetén figyelmeztet.
- require()
  - Többszöri híváskor újra végrehajtódik.
  - Hibás fájl esetén elszáll hibával.
- require_once()
  - Többszöri híváskor NEM hajtódik újra végre.
  - Hibás fájl esetén elszáll hibával.