

Web-technológia II.

PHP: típusok, változók, operátorok, vezérlési szerkezetek,
függvények, hatókör, fájlok

Hatwágner F. Miklós

Széchenyi István Egyetem, Győr

2019. február 21.

proba.php

```
<!DOCTYPE html>
<html>
  <head>
    <title>Első PHP weboldalunk</title>
    <meta charset="utf-8" />
  </head>
  <body>
    <?php
      print("  <p>Juhééé, működik a PHP oldalam!</p>\n");
    ?>
  </body>
</html>
```

PHP blokk jelölése: `<?php // kód ?>`

Ha csak PHP kódot tartalmaz a fájl, a záróelem elhagyható (CLI-nél ajánlott elhagyni)

Elavult megoldások:

`<? ?>` rövid elem

`<% %>` ASP stílus

`<script language="php"> </script>` Script elem

Kódolási stílusok:

- Szinte csak HTML, néhány rövid közbeszúrt PHP blokk
- Bonyolultabb (MVC) oldalaknál: tiszta PHP kód, kódból nyomtatott/fájlból beszúrt HTML tartalmak

Szöveges tartalmak (PHP, HTML, stb.) beszúrása:

`include` Ha nem sikerül a beszúrás, warning-ot ad

`require` Fatális hibával programleállás sikertelenség esetén

`include_once`

Mint `include`, de mindenképp csak egyszer helyettesít be

`require_once`

Mint `require`, de mindenképp csak egyszer helyettesít be

Fájlok tartalmának beszúrása

pelda01.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Első PHP weboldalunk</title>
    <meta charset="utf-8" />
  </head>
  <body>
```

pelda01.php

```
<?php
// pelda01.html betöltése
require 'pelda01.html';
// html fájl hiányzó végének előállítása
print "   <p>Juhééé, működik a PHP oldalam!</p>\n   </body>\n</html>\n";
```

Megjegyzések, karakterláncok

Megjegyzések:

- `/*` Több
 `* soros`
 `* megjegyzés */`
- `//` Egysoros megjegyzés
- `#` Egysoros megjegyzés

Nyomtatás nyelvi elemekkel:

`print` 1-et ad vissza, nincs paraméterlista

`echo` Nincs v.t. érték, paraméterlista, rövid alak (`<?="Szoveg"??`, `short_open_tag`)

Nyomtatás függvényekkel: `printf()`, `sprintf()`, `vprintf()`...

Karakterláncok jelzése:

- aposztróf: változtatás nélküli nyomtatás
- idézőjel: változók értékét, escape-szekvenciákat behelyettesíti
- Here document (Heredoc, van helyettesítés) / Now document (Nowdoc, nincs helyettesítés)

pelda02.php

```
<?php
    $nev = "Gabi"; // Változó definíció

    print 'Név: $nev<br>\n'; // Nincs helyettesítés
    print "Név: $nev<br>\n"; // Van helyettesítés
    print("Név: $nev<br>\n"); // Fv.-szerű alak, bár nyelvi elem
    // Nowdoc
    print <<<'NOW'
        Név: $nev<br>\n
NOW;
    // Heredoc
    print <<<HERE
        Név: $nev<br>\n
HERE;
    echo "Több sorba
        tördelt forrásszöveg.";
?>

HTML <?='PHP'?> HTML
```

Névadás:

- Első karakter: \$
- Második karakter: betű vagy hangosköz
- További karakterek: betű, hangosköz és számjegy
- Kis- és nagybetűre érzékeny

Legfontosabb alaptípusok:

- bool/boolean
- int/integer
- float/double
- string

Gyenge típusosság:

- Elég közvetlenül a használat előtt deklarálni
- A típus az értékadásnál dől el
- Bármikor típust válthat egy változó

`var_dump()` Változó típusának, értékének és struktúrájának nyomtatása

`gettype()` Típust adja vissza string-ként

pelda03.php

```
<?php
// Hogyan jelenik meg a logikai érték?
$v = true;
var_dump($v);
echo " Érték: $v, Típus: ", gettype($v), "<br>\n";

$v = false;
var_dump($v);
echo " Érték: $v, Típus: ", gettype($v), "<br>\n";

$v = 42;
var_dump($v);
echo " Érték: $v, Típus: ", gettype($v), "<br>\n";

$v = 3.14;
var_dump($v);
echo " Érték: $v, Típus: ", gettype($v), "<br>\n";

$v = "szöveg";
var_dump($v);
echo " Érték: $v, Típus: ", gettype($v), "<br>\n";
```


Explicit típuskonverziók:

- `settype()`
- (új típus)változó

pelda04.php

```
<?php
    $v = '42';
    settype($v, 'integer');
    var_dump($v);

    $d = (double)$v;
    echo '<br>$d típusa: ', gettype($d);
```

Aritmetikai operátorok

pelda05.php

```
<?php
// Definíciók
$a = 10;
$b = 3;
// Aritmetikai operátorok
echo "$a + $b = ", $a+$b, "<br>\n";
echo "$a - $b = ", $a-$b, "<br>\n";
echo "$a * $b = ", $a*$b, "<br>\n";
echo "$a / $b = ", $a/$b, "<br>\n";
echo "$a % $b = ", $a%$b, "<br>\n";
echo "$a ** $b = ", $a**$b, "<br>\n";
echo "-$a = ", -$a, "<br>\n";
// Implicit típuskonverzió (type juggling)
$a = "10";
echo "$a + $b = ", $a+$b, "<br>\n";
```

Kerekítés: `round()`

Relációs operátorok

pelda06.php

```
<?php
// Relációs operátorok
echo "1 < 2 --> ", var_dump(1 < 2), "<br>\n";
echo "1 > 2 --> ", var_dump(1 > 2), "<br>\n";
echo "1 <= 2 --> ", var_dump(1 <= 2), "<br>\n";
echo "1 >= 2 --> ", var_dump(1 >= 2), "<br>\n";
// Spaceship operator
echo "1 <=> 2 --> ", var_dump(1 <=> 2), "<br>\n";
echo "1 <=> 1 --> ", var_dump(1 <=> 1), "<br>\n";
echo "2 <=> 1 --> ", var_dump(2 <=> 1), "<br>\n";
// Egyezőségi operátorok
echo "1 == '1' --> ", var_dump(1 == '1'), "<br>\n";
// Nincs implicit típuskonverzió!
echo "1 === '1' --> ", var_dump(1 === '1'), "<br>\n";
echo "1 != '1' --> ", var_dump(1 != '1'), "<br>\n";
echo "1 <> '1' --> ", var_dump(1 <> '1'), "<br>\n";
// Nincs implicit típuskonverzió!
echo "1 !== '1' --> ", var_dump(1 !== '1'), "<br>\n";
```

További operátorok

pelda07.php

```
<?php
// Logikai operátorok
echo "true && false == ", var_dump(true&&false), "<br>\n";
echo "true || false == ", var_dump(true||false), "<br>\n";
echo "!true == ", var_dump(!true), "<br>\n";
// Karakterláncok összefűzése
echo 'Nagyon '. 'hosszú '. 'szöveg.'. "<br>\n";
// Értékadás és összetett operátorok
$v = 42; echo "\$v == $v<br>\n";
$v += 2; echo "\$v == $v<br>\n";
$v -= 3; echo "\$v == $v<br>\n";
$v *= 2; echo "\$v == $v<br>\n";
$v /= 3; echo "\$v == $v<br>\n";
$v .= ' ló'; echo "\$v == $v<br>\n";
// Növelés és csökkentés
$v = 5;
$v++; echo "\$v == $v<br>\n";
$v--; echo "\$v == $v<br>\n";
```

Precedencia és asszociativitás

A **tömb** asszociatív a PHP-ben, kulcsának típusa lehet

- integer, vagy
- string.

Rugalmas, megvalósítható vele pl.

- dinamikus tömb,
- hash tábla,
- szótár,
- verem,
- sor, stb.

Tömbök egymásba ágyazhatók, így többdimenziós tömbök is létrehozhatók.

Létrehozás módjai:

- `array()` nyelvi elemmel,
- literállal.

pelda08.php

```
<?php
// egészekkel indexelt tömbök
$nemetBalna = ['Mercedes', 'BMW', 'Audi'];
$egyebBalna = array('Jaguar', 'Lexus');
$francia = [ // Az indexelés nem feltétlenül sorfolytonos
    1 => 'Peugeot',
    3 => 'Citroen',
    7 => 'Renault',
    -5 => 'DS' // és lehet negatív is
];

// karakterláncokkal indexelt tömbök
$hallgatok = [
    'ABC123' => 'Nemoda Buda',
    'QWE456' => 'Remek Elek',
    'A1B2C3' => 'Trab Antal'
];
```

pelda08.php

```
// Több dimenziós tömb
$tantargy = array(
    'NGB_IN023_1' => array( // Kulcsok/értékek típusainak nem
        'nev' => 'Web-technológia I.', // kell azonosnak lennie
        'kredit' => 4,
        'eloadas' => 3,
        'gyakorlat' => 0,
        'labor' => 0
    ),
    'NGB_IN023_2' => array(
        'nev' => 'Web-technológia II.',
        'kredit' => 4,
        'eloadas' => 0,
        'gyakorlat' => 3,
        'labor' => 0
    )
);
```

Tömbök tartalma rekurzívan nyomtatható:

- `var_dump()`
- `print_r()` Méret és típus adatokat nem közöl

Elem hozzáadása:

- új kulcs - érték pár megadásával
- `[]` a tömb végére illeszti az új elemet

Elem (vagy bármilyen más változó) törlése: `unset()`

Az `isset()` igazat ad vissza, ha a változó létezik és nem NULL.

Elemszám meghatározása: `count()`

Az `empty()` igazat ad, ha a paramétere

- üres string,
- zérus értékű egész/lebegőpontos szám vagy a 0 karaktert tartalmazó string,
- NULL,
- FALSE,
- **üres tömb**, vagy
- deklarált, de nem inicializált változó.

pelda08.php

```
// Tömb megjelenítése
echo '<pre>';
var_dump($francia);
echo "</pre>\n";

echo '<pre>';
print_r($tantargy);
echo "</pre>\n";

// Tömelemek utólag módosíthatók
$nemetBalna[0] = 'Mercedes-Benz';
// Tömbök utólag bővíthetők
$francia[42] = 'Matra';
$egyebBalna[] = 'Volvo'; // a sor végére teszi az új elemet
$tantargy['NGB_IN001_1'] = array(
    'nev' => 'Programozás I.',
    'kredit' => 5,
    'eloadas' => 2,
    'gyakorlat' => 2,
    'labor' => 0
);
```

pelda08.php

```
echo "<p>\$nemetBalna elemszáma: ", count($nemetBalna), "</p>\n";
echo "<p>Van a \$francia tömbnek 42-es indexű eleme? ",
    isset($francia[42])?'Van.': 'Nincs.', "</p>\n";
unset($francia[42]);
echo "<p>Még mindig van? ",
    isset($francia[42])?'Van.': 'Nincs.', "</p>\n";
echo "<p>Az \$egyebBalna üres? ",
    empty($egyebBalna)? 'Igen.': 'Nem.', "</p>\n";
```

Keresés az elemek között

- `in_array()` csak a létezésről tájékoztat
- `array_search()` elem indexét adja vissza

pelda08.php

```
echo "<p>A Volvo is bálna? ",
    in_array('Volvo', $egyebBalna)? 'Igen.': 'Nem.', "<br>\n";
echo 'Mi a kulcsa? ', ($i=array_search('Volvo', $egyebBalna))==FALSE
    ? 'Nincs kulcsa.': $i, "</p>\n";
echo "<p>A Trabant is bálna? ",
    in_array('Trabant', $egyebBalna)? 'Igen.': 'Nem.', "<br>\n";
echo 'Mi a kulcsa? ', ($i=array_search('Trabant', $egyebBalna))==FALSE
    ? 'Nincs kulcsa.': $i, "</p>\n";
```

Tömbök helyben rendezése

Függvény	Rendezés célja	Kulcstársítást megőriz	Sorrend
sort	érték	nem	növekvő
rsort	érték	nem	csökkenő
asort	érték	igen	növekvő
arsort	érték	igen	csökkenő
ksort	kulcs	igen	növekvő
krsort	kulcs	igen	csökkenő
usort	érték	nem	növekvő

pelda08.php

```
$nevek = $nevek1 = $nevek2 = $nevek3 = $nevek4 =  
    ['Aladár', -3 => 'Zsuzsi', 'Álmos', 50 => 'Mihály', 'Béla', 'Réka'];  
echo "<p>Eredeti tartalom:</p>\n<pre>"; print_r($nevek);  
    echo "</pre>\n";  
sort($nevek1);  
echo "<p>sort() hatása:</p>\n<pre>"; print_r($nevek1);  
    echo "</pre>\n";  
asort($nevek2);  
echo "<p>asort() hatása:</p>\n<pre>"; print_r($nevek2);  
    echo "</pre>\n";  
ksort($nevek3);  
echo "<p>ksort() hatása:</p>\n<pre>"; print_r($nevek3);  
    echo "</pre>\n";  
setlocale(LC_ALL, "hu_HU.UTF-8");  
usort($nevek4, "strcoll");  
echo "<p>usort() hatása:</p>\n<pre>"; print_r($nevek4);  
    echo "</pre>\n";
```

További tömbkezelő függvények

- `array_keys()` visszaadja a kulcsokat
- `array_values()` visszaadja az értékeket
- `array_merge()` tömbök összefűzése
- `array_shift()` első elem eltávolítása
- `array_unshift()` beszúrás első helyre
- `array_pop()` utolsó elem eltávolítása
- `array_push()` beszúrás utolsó helyre
- `array_unique()` ismétlődések eltávolítása
- `shuffle()` véletlenszerű sorrend

- `if (feltétel) utasítás1 [else utasítás2];`
- `else if` \equiv `elseif`
- `if (feltétel): ... else: ... endif;`

pelda09.php

```
<?php
$homerseklet = 23;
if($homerseklet < -10) {
    echo "<p>Farkasordító hideg.</p>\n";
} else if($homerseklet < 0) {
    echo "<p>Zima van.</p>\n";
} elseif($homerseklet < 10) { // elseif!
    echo "<p>Friss az idő.</p>\n";
} else if($homerseklet < 20) {
    echo "<p>Elkél egy pulóver.</p>\n";
} else if($homerseklet < 30){
    echo "<p>Irány a természet!</p>\n";
} else {
    echo "<p>Klímákat bekapcsolni!</p>\n";
}
```

pelda09.php

```
$bejel = true;
if($bejel) {
?>
  <p>Ön bejelentkezett a fiókjába.</p>
<?php
  } else {
?>
  <p>Ön vendégként használja az oldalunkat.</p>
<?php
  }

if($bejel): ?>
  <p>Ön bejelentkezett a fiókjába.</p>
<?php else: ?>
  <p>Ön vendégként használja az oldalunkat.</p>
<?php endif; ?>
```

```
switch (kifejezés) {  
    case cimke1:  
    case cimke2:  
        utasítások1;  
        [break;]  
    case cimke3:  
    ...  
    case cimkeN:  
        utasítások2;  
        [break;]  
    default:  
        utasítások3;  
        [break;]  
}
```

pelda10.php

```
<?php  
$ho = 'Február';  
switch($ho) {  
    case 'Január':  
    case 'Február':  
    case 'December':  
        echo "<p>Tél</p>\n";  
        break;  
    case 'Március':  
    case 'Április':  
    case 'Május':  
        echo "<p>Tavaszi</p>\n";  
        break;  
    case 'Június':  
    case 'Július':  
    case 'Augusztus':  
        echo "<p>Nyár</p>\n";  
        break;  
    case 'Szeptember':  
    case 'Október':  
    case 'November':  
        echo "<p>Ősz</p>\n";  
        break;  
    default:  
        echo "<p>Ismeretlen hónap</p>\n";  
        break;  
}
```


Elöltesztelő:

- `while (feltétel) [utasítás];`
- `for(inicializálás; feltétel; módosítás) [utasítás];`
- `foreach(tömb as [kulcs =>] érték) [utasítás];`

Hátultesztelő:

- `do [utasítás]; while(feltétel);`

`break`, `continue` használható, de kerülendő.

Segédfüggvények:

- `ord()` karakter ASCII kódját adja
- `chr()` adott kódú karaktert adja
- `number_format()` számok formázása

pelda11.php

```
<?php
    $sor = 3;
    $oszlop = 2;

    echo "<table>\n";
    for($s=0; $s<$sor; $s++) {
        echo "    <tr>\n";
        $o=0;
        while($o < $oszlop) {
            echo "        <td>", chr($s+ord('A')).$o, "</td>\n";
            $o++;
        }
        echo "    </tr>\n";
    }
    echo "</table>\n";
```

pelda11.php

```
$tartalom = [  
    ['Beosztás', 'Fizetés'],  
    ['Főnök', 1000000],  
    ['Beosztott', 100000],  
    ['Feketemunkás', 10000]  
];  
  
echo "<table>\n";  
foreach($tartalom as $sor) {  
    echo "    <tr>\n";  
    foreach($sor as $cella) {  
        echo "        <td>",  
        gettype($cella) === 'integer'?  
            number_format($cella, 0, ',', ', '):$cella,  
        "</td>\n";  
    }  
    echo "    </tr>\n";  
}  
echo "</table>\n";
```

- Függvények listája
- Hossz lekérdezése: `strlen()`
- Fehér (vagy specifikált egyéb) karakterek levágása balról: `ltrim()`
- Ugyanez jobbról: `rtrim()`
- Ugyanez mindkét oldalról: `trim()`
- Kisbetűs alakra konvertálás (**helyi beállítások** alapján): `strtolower()`
- Nagybetűs alakra hozás: `strtoupper()`
- Rész-karakterláncok lecserélése: `str_replace()`
- Ugyanaz, de reguláris kifejezésekkel (lassú): `preg_replace()`
- Rész-karakterlánc előállítás: `substr()`

pelda12.php

```
<?php
echo setlocale(LC_ALL, 'hu_HU.UTF-8')==FALSE ?
    "Hiba a hely beállítása során!<br>\n" :
    "Magyar hely beállítva.<br>\n";
$s = 'Micimackó'; // Mennyi az annyi? Miért?
echo "'$s' karaktereinek száma: ", strlen($s), "<br>\n";
$s = trim('    Tartalom    ');
echo "'$s' karaktereinek száma: ", strlen($s), "<br>\n";
$s = 'Árvíztűrő Tükörfúrógép';
echo 'Eredeti karakterlanc: ', $s,
    ', kisbetűkkel: ', strtolower($s),
    ', nagybetűkkel: ', strtoupper($s), "<br>\n";
$s = 'Életem egyetlen szerelme, Lujza';
$mit = 'Lujza';
$mire = 'Ilonka';
echo str_replace($mit, $mire, $s), "<br>\n";
$s = 'abcdef';
echo 'Első három betű: ', substr($s, 0, 3),
    ', utolsó három: ', substr($s, -3),
    ', középső kettő: ', substr($s, 2, 2), "<br>\n";
```

- Rész-karakterlánc előfordulásának keresése: `strstr()` / `strpos()`
- Karakterlánc darabolása egy tömbbe: `explode()`
- Tömb elemeinek összefűzése karakterlánccá: `implode()` / `join()`
- md5 hash előállítás: `md5()` / `md5_file()`
- sha1 hash előállítás: `sha1()` / `sha1_file()`

Megjegyzés: `md5`, `sha1` algoritmusok elavultak →
`password_hash()`, `password_verify()`

Jelszavak ajánlott kódolásáról bővebben [itt](#) olvashatnak.

pelda13.php

```
<?php
$dal = 'Az én kedvesem egy olyan lány, akit
farkasok neveltek és
táncolt egy délibábbal,
majd elillant csendesen
az én kedvesem.
Ő az én kedvesem.';
$szo = 'kedvesem';
echo "<p>Teljes dalszöveg:</p>\n<pre>$dal</pre>\n
      <p>'$szo' előfordulásainak indexei:</p>\n
      <ul>\n";
$kezd = -1;
while (($kezd = strpos($dal, $szo, $kezd+1)) !== FALSE) {
    echo "<li>$kezd</li>\n";
}
echo "</ul>\n";

$t = explode(',', 'Opel,Volkswagen,Ford,Suzuki,Honda,Toyota');
echo "<p>A fellelt márkák:</p>\n<ul>\n";
foreach($t as $m) {
    echo "<li>$m</li>\n";
}
echo "</ul>\n<p>Összefűzve: ", $s=implode(' ', $t), ".</p>\n";

echo "<p>md5: ", md5($s), ", sha1: ", sha1($s), ", password_hash: ",
password_hash($s, PASSWORD_DEFAULT), "</p>\n";
```

`function név(param1, param2, ..., paramN)` utasítások [return]
Hátulról előre összefüggő paraméter-listához alapértelmezett
(híváskor elhagyható) értékek adhatók meg

pelda14.php

```
// Nem szükséges a hívás előtt deklarálni
function fejlec($info, $meret=1) {
    echo "<h$meret>$info</h$meret>\n";
}

fejlec('Üdvözöllek dicső lovag');
fejlec('Szép a ruhád, szép vagy magad', 3);
```


Változó számú paraméterlista is használható:

- `func_num_args()` paraméterlista elemszáma
- `func_get_arg()` paraméterlista egy elemét adja
- `func_get_args()` paraméterlistát tömbben adja vissza

pelda14.php

```
function osszead() {  
    $db = func_num_args();  
    $osszeg = 0;  
    for($i=0; $i<$db; $i++) {  
        $osszeg += func_get_arg($i);  
    }  
    return $osszeg;  
}  
  
echo "<p>1+2+3=", osszead(1, 2, 3), "</p>\n";
```

pelda14.php

```
function kiir($a, $b) {  
    echo "<p>\$a értéke: $a, \$b értéke: $b</p>\n";  
}  
// érték szerinti paraméter-átadás  
function csereErtek($a, $b) {  
    $csere = $a;  
    $a = $b;  
    $b = $csere;  
}  
// referencia szerinti paraméter-átadás  
function csereRef(&$a, &$b) {  
    $csere = $a;  
    $a = $b;  
    $b = $csere;  
}  
$szam1 = 3; $szam2 = 5;  
kiir($szam1, $szam2);  
csereErtek($szam1, $szam2);  
kiir($szam1, $szam2);  
csereRef($szam1, $szam2);  
kiir($szam1, $szam2);
```

Type hinting – paraméterek és visszatérési érték típusának előírása
static két hívás között megőrzi a változó értékét

pelda15.php

```
<?php
declare(strict_types=1); // A fájl első utasítása legyen!

function szorzas(int $a, int $b) : int {
    return $a * $b;
}

echo "<p>1*2*3*4=", szorzas(szorzas(szorzas(1, 2), 3), 4), "</p>\n";

function osszead(int $a) : int {
    static $osszeg;
    return $osszeg += $a;
}

osszead(1); osszead(2); osszead(3);
echo "<p>1+2+3+4=", osszead(4), "</p>\n";
```

Változók hatásköre (scope)

A legtöbb programnyelvvél ellentétben a blokkon kívül deklarált változó hatásköre nem terjed ki a blokkra, ha csak nem deklaráljuk **global**-nak!

pelda16.php

```
<?php
    $globA = 1; $globB = 2;

    function fv() {
        global $globB;
        $lokalA = 3; $lokalB = 4;
        echo "<p>Fv-en belül: \$globA == $globA, \$globB == $globB, ",
            "\$lokalA == $lokalA, \$lokalB == $lokalB</p>\n";
    }
    fv();
    echo "<p>Fv-en kívül: \$globA == $globA, \$globB == $globB, ",
        "\$lokalA == $lokalA, \$lokalB == $lokalB</p>\n";
```

Foglalt konstansok, előredefiniált konstansok

pelda17.php

```
<?php
define("FELHASZNALO", "Andi"); // függvényben is deklarálható
echo "<p>Üdvözlöm, kedves ", FELHASZNALO, "!</p>\n";
echo "<p>A PHP értelmező verziószáma: ", PHP_VERSION, "</p>\n";
echo "<p>Ez a sor áll értelmezés alatt: ", __LINE__, "</p>\n";

const ELET_ERTELME = 42; // csak globális szinten deklarálható
echo "<p>Az élet értelme még mindig ", ELET_ERTELME, "</p>\n";
```

Ritkán használtak (adatbázis-kezelő rendszerek előnyben), de néha szükséges (pl. tárolt fájlok mozgatása, átnevezése, metaadatok kiolvasása, stb.)

- `file_exists()` fájl létezését ellenőrzi
- `filesize()` méretet adja meg, 32 bites int-ben!
- `is_file()`, `is_dir()` megállapítják, hogy a bejegyzés fájl/könyvtár-e
- `is_readable()`, `is_writable()`, `is_executable()` olvasási/írási/futtatási jogosultság ellenőrzése
- `chmod()`, `chown()`, `chgrp()` jogosultság, tulajdonos és tulajdonos csoport állítása
- `fileatime()`, `filemtime()`, `filectime()` fájl utolsó elérésének, módosításának és inode módosításának időbélyege
- `unlink()` fájl törlése
- `rename()` fájl átnevezése

Fájl tartalmának felhasználása:

- 1 `fopen()` megnyitás
- 2 `fgets()`, `fwrite()`, `fread()` szövegsor olvasása/szöveg vagy bináris tart. írása/bináris olvasás
- 3 `fclose()` fájl lezárása

Magasabb szintű műveletek:

- `file_get_contents()`, `file_put_contents()` teljes fájl beolvasása/írása
- `fgetcsv()` megnyitott CSV fájlból beolvas egy sort, parzolja, tömbként visszaadja a cellákat
- `str_getcsv()` CSV sort parzol és ad vissza tömbként

- Készítsenek a **tanszéki munkatársakat** bemutatóhoz hasonló weboldalt! Az oldal tetején álljon címsorként a „Munkatársak” felirat, alatta külön táblázatokban az egyes kollégák (3-5 fő) adatai, fényképe! (Ezek felfoghatók kulcs-érték párokba rendezett adatoknak is.) A „részletes adatok”-kal most még nem kell foglalkozni. Az adatokat egyelőre jobb híján tárolják PHP tömbökben, és készítsenek olyan függvényt, ami képes egy-egy oktató adatai alapján generálni a megfelelő HTML oldalrészlet, illetve olyat is, ami az összes oktatóra meghívja egyesével ezt a függvényt! Készüljenek fel rá, hogy nem mindenki rendelkezik pontosan ugyanolyan adatokkal (pl. nincs fax készüléke). Az oldal legyen valid HTML, és CSS-sel formázott is!
- Továbbfejleszthetik a feladatot oly módon, hogy az oktatói adatokat CSV fájlban tárolják a szerveren, és a PHP ebből olvassa ki az adatokat.
- Bár nem kapcsolódik szorosan a PHP-hez, de átalakíthatják a programot úgy, hogy a nevet a táblázatosan megjelenített adatok fölött jelenítik meg, és ha valaki erre kattint, akkor elrejtí/megjeleníti az oktató összes többi adatát az oldal (némi CSS/JS kódot igényel)!

- Írjon egy PHP programot, ami az Északi Sarki Szeretetszolgálat munkáját segíti magánszemélyek és ajándéktárgyak hozzárendelésével!
A program először olvassa be a magánszemélyek adatait a `jolviselkedok.txt` fájlból! Minden adat két sorból áll: az 1. sor tartalmazza a személy teljes nevét, a 2. sor pedig egy tőrtszámot 0 és 100 között, amely százalékban fejezi ki a személy idén elért teljesítményét a hivatalos JólviselkedésTM skálán.
A személyek adatai után az ajándéktárgyak adatai következnek az `ajandekok.txt` fájlból. Ezek is egyenként 2 sorból állnak: egy névből és egy névleges értékből.
A bemenet beolvasása után a program határozza meg a hozzárendelést a személyek és ajándéktárgyak között. A szervezet vezetője, Sz. Miklós kérésének megfelelően, bármely két magánszemély közül a nagyobb JólviselkedésTM értékeléssel rendelkező kapjon nagyobb névleges értékű ajándéktárgyat, azaz a legmagasabb értékelésű kapja a legértékesebbet, a 2. legjobb a 2. legértékesebbet, stb. A program írja ki a hozzárendeléseket egy számozott listában, személynév - ajándéknév formátumban, a JólviselkedésTM skála szerinti csökkenő sorrendben!
- Fejlessze tovább úgy a feladatot, hogy a lista alatt elhelyez egy "Részletek" feliratú hivatkozást, melyre kattintva olyan oldalra lehet eljutni, ahol a párosításon kívül a személyek JólviselkedésTM értékét, és az ajándékok értékét is meg lehet tekinteni egy HTML táblázatban! Ennek is ugyanúgy rendezettnek kell lennie, mint a listának. Ügyeljen rá, hogy a magyar ékezetes betűket tartalmazó nevek is a megfelelő helyen szerepeljenek!

Írjon sportfogadások adatait kezelő programot! Először olvassa be az elérhető meccsek adatait a `meccs.txt` fájlból, ami a következő mezőket tartalmazza szóközzel elválasztva:

- 1 hazai csapat neve
- 2 vendég csapat neve
- 3 hazai győzelem odds
- 4 döntetlen odds
- 5 vendég győzelem odds

Ezután olvassa be a játékos fogadásait a `fogadasok.txt` fájlból! Ebben a játékos először megadja a meccs sorszámát (a számozás 1-gyel kezdődik), majd szóközzel elválasztva a kimenetelt H, D, vagy V karakterrel, ami a Hazai győzelem, Döntetlen, és Vendég győzelem rövidítése. A kombinált fogadásban nem szükséges minden meccsre fogadni. A fogadások lezárását a 0 meccs sorszám megadása jelzi.

A program írja ki a kiválasztott meccsek csapatait a választott kimenetekkel (H/D/V), és a fogadások kombinált oddsát, azaz az egyes oddsok szorzatát, 2 tizedesjegy pontossággal! Pl.

A kiválasztott fogadasok:

Brighton - Liverpool: V

Chelsea - Newcastle: D

A kombinalt odds: 7.15

Antonio Lopez
Learning PHP 7

Hivatalos PHP referencia