

WEB PROGRAMOZÁS

1.ELŐADÁS

Dr. Pál László

Sapientia EMTE, Csíkszereda, 2015 - 2016 tanév, I. Félév

Előadás tematika

1.Előadás:	PHP alapok (adattípusok, operátorok, vezérlési szerkezetek, tömbök, függvények)
2. Előadás:	Objektumorientált programozás PHP-ben
3. Előadás:	Űrlapok kezelése, feldolgozása
4. Előadás:	Sütik és munkamenetek használata
5. Előadás:	Adatbázis-kezelés PHP-ben
6. Előadás:	Internet szolgáltatások (email küldés, stb.)
7. Előadás:	Fájl-kezelés PHP-ben

Labor tematika- KGI

1.Labor:	PHP nyelvi elemek (változó, típusok, kiíratások, műveletek)
2.Labor:	PHP nyelvi elemek (ciklusok, függvények, tömbök)
3.Labor:	Űrlapok feldolgozása WordPress telepítése, alapbeállítások
4.Labor:	Űrlapok feldolgozása, ellenőrzése WordPress: bejegyzések, oldalak, hozzászólások, médiatár
5.Labor:	Fájlműveletek WordPress: widgetek, sablonok
6.Labor:	Sütik, munkamenetek WordPress: bővítmények
7.Labor:	Adatbázisok WordPress: menük
8.Labor:	Weboldal felépítése WordPress: konkrét példa
9.Labor:	Projektek ellenőrzése
10.Labor:	Weboldal felépítése WordPress: e-commerce
11.Labor:	Weboldal felépítése WordPress: e-commerce
12.Labor:	WordPress: e-commerce
13.Labor:	WordPress: e-commerce
14.Labor:	Projektek védése

Labor tematika- GI

1.Labor:	PHP nyelvi elemek (változó, típusok, kiíratások, műveletek)
2.Labor:	PHP nyelvi elemek (ciklusok, tömbök, függvények)
3.Labor:	Osztályok, objektumok
4.Labor:	Osztályok, objektumok
5.Labor:	Űrlap elemek kezelése
6.Labor:	Űrlapok feldolgozása, ellenőrzése
7.Labor:	Sütik használata
8.Labor:	Sessionok használata
9.Labor:	Adatbázisok kezelése (Projekt ellenőrzés)
10.Labor:	Adatbázisok kezelése
11.Labor:	Adatbázisok kezelése
12.Labor:	Internet szolgáltatások (email küldés, stb.)
13.Labor:	Fájlok és könyvtárak kezelése
14.Labor:	Projektek védelme

Könyvészet és dokumentáció

□ Könyvtári könyvek

- ▣ Virginia DeBolt: *HTML és CSS : webszerkesztés stílusosan*, Kiskapu, Budapest, 2005.
- ▣ Zandstra Matt: *Tanuljuk meg a PHP4 használatát 24 óra alatt*, Kiskapu, 2001.
- ▣ Michael Moncur: *Tanuljuk meg a JavaScript használatát 24 óra alatt*, Kiskapu Kft., 2006.
- ▣ Sági Gábor, *Webes adatbázis-kezelés MySQL és PHP használatával*, Budapest, 2005.
- ▣ Jeffrey Winesett, *Web application development with Yii and PHP*, 2012.
- ▣ Stern Hal, *Professional WordPress : design and development*, 2013.

Könyvészet és dokumentáció

□ Hasznos linkek

▣ W3Schools Online Web Tutorials:

<http://www.w3schools.com>

▣ PHP bevezető: <http://www.tizag.com/phpT/>, <http://www.quackit.com/php/>

▣ WordPress témák:

■ <http://weboldalkeszitese.org/>

■ [http://codex.wordpress.org/WordPress Lessons](http://codex.wordpress.org/WordPress_Lessons)

■ <http://www.wpbeginner.com/>

Könyvészet és dokumentáció



- Előadások, labor feladatok, egyéb dokumentációk megtalálhatók az alábbi linken:

http://www.emte.siculorum.ro/~pallaszlo/oktatas_hu.html

Vizsgakövetelmények

□ Felmérési mód: *kollokvium*

- ▣ Projekt: 70% (ellenőrzés - 8. hét, védés - 14. hét (Labor))
- ▣ Elméleti teszt: 20% (14. hét (Előadás))
- ▣ Előadás jelenlét: 10%
- ▣ Labor óra: max 3 hiányzás engedélyezett
- ▣ Részletes tantárgy követelmények:

http://www.emte.siculorum.ro/~pallaszlo/webprog/kovetelmények/Tantargy_Kovetelmenyek.pdf

1. Előadás - Tartalom



1. A PHP szkript nyelv
2. PHP szintaxis
3. Vezérlési szerkezetek
4. Függvények
5. Tömbök

Szerver oldali technológiák



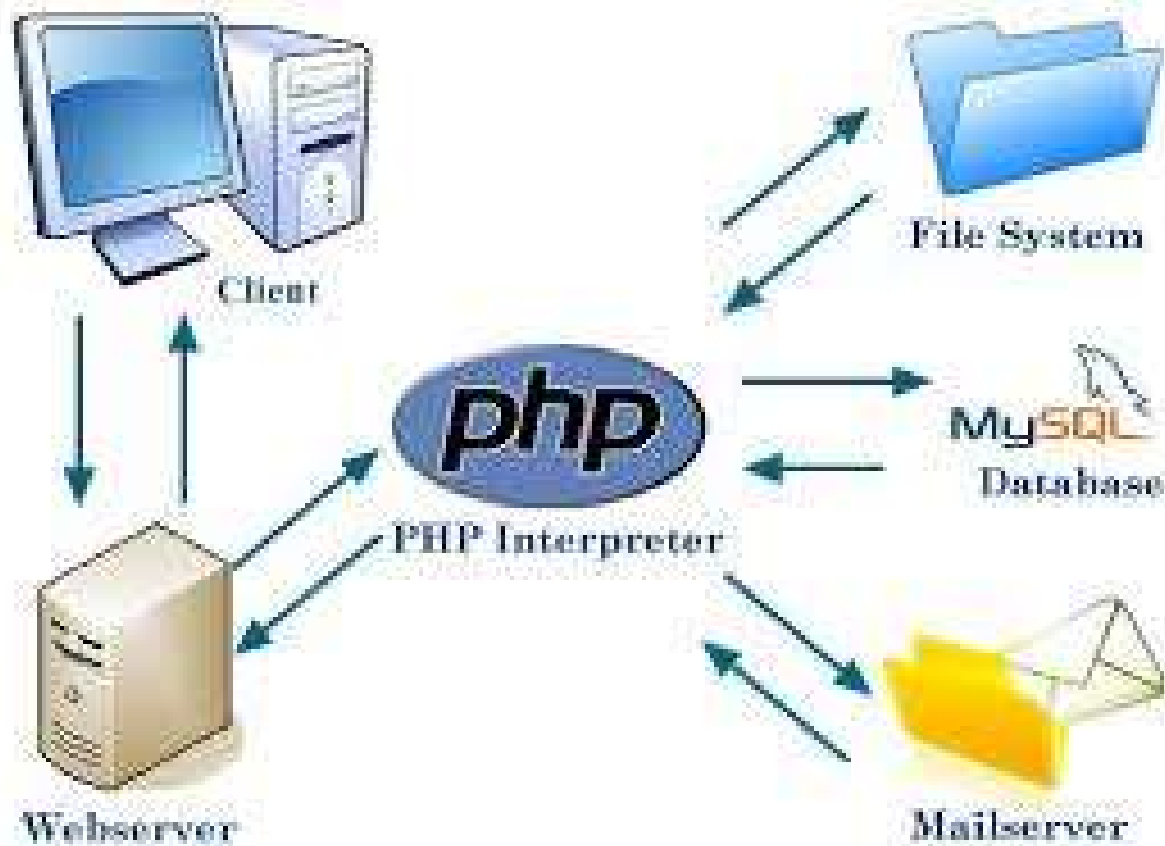
- PHP
- Servlet - Sun által 1996-ban bevezetett technológia
- Java Server Pages (JSP) - Servlet technológiára épül
- Active Server Pages (ASP) - Microsoft által támogatott
- Active Server Pages.NET (ASP.NET) - a Microsoft .NET keretrendszer része

Általános jellemzők



- a script végrehajtására a szerver gépen kerül sor
- a HTML-be vannak beágyazva
- egy script-értelmező motor dolgozza fel

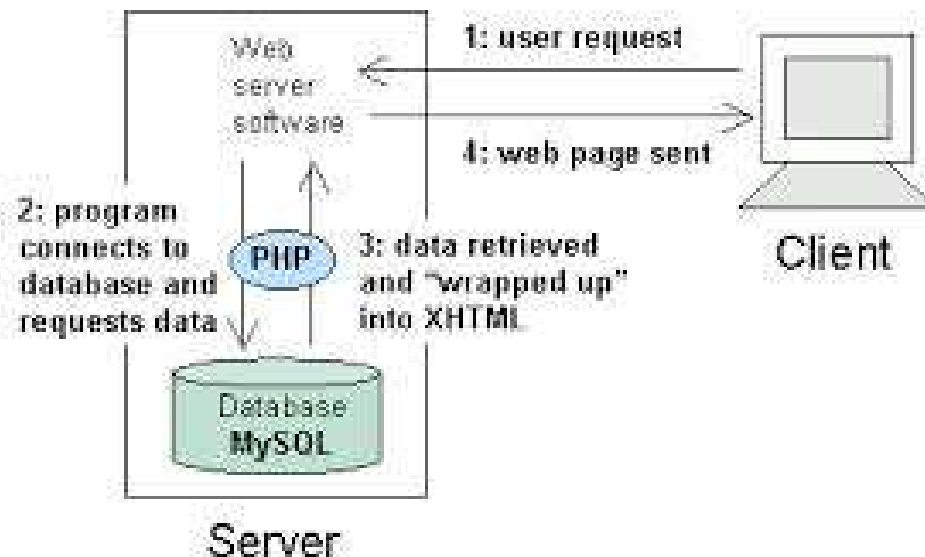
Dinamikusan létrehozott HTML oldal



Dinamikusan létrehozott HTML oldal

□ Lépések:

1. Kliens kérés
2. Webszerver továbbítja a kérést a szkript értelmezőhöz
3. További kérések (pld. Adatbázis elérés)
4. HTML visszaküldése a klienshez



A PHP script nyelv - Jellemzők

- ❑ PHP - PHP: Hypertext Preprocessor (eredetileg: Personal Home Page - Rasmus Lerdorf, 1994)
- ❑ Jelenleg a legelterjedtebb szerver-oldali script nyelv
- ❑ Nyílt forráskódú, ingyenes (<http://www.php.net>)
- ❑ Objektorientált nyelv (OOP)
- ❑ Beágyazható a HTML oldalba
- ❑ Platformfüggetlen - a legelterjedtebb op. rendszereket támogatja (Unix alapú op. rendszerek, Microsoft Windows, Mac OS X)
- ❑ Leggyakrabban az Apache Web-szerverrel együtt használják

A PHP script nyelv - Lehetőségek



- Dinamikus és interaktív web oldalak készítése
- Különböző adatbázisok támogatása (MySQL, Oracle, PostgreSQL, ODBC, stb.)
- Használható szövegfeldolgozásra, XML állományok kezelésére

PHP hivatkozások



- Főoldal (innen tölthető le):
 - ▣ <http://www.php.net/>
- Hivatalos referenciák:
 - ▣ <http://www.php.net/manual/en/>
 - ▣ Itt található: függvényreferenciák, nyelvi leírások, kódolási tanácsok, stb.

A PHP használata

□ Telepítés:

- Apache (web-szerver) - <http://www.apache.org/>
- PHP - <http://www.php.net/downloads.php>
- MySQL - <http://dev.mysql.com/downloads/>
- XAMPP (apache+php+mysql egyben):
 - <http://www.apachefriends.org/en/xampp-windows.html>
- vagy WampServer (apache+php+mysql egyben):
 - <http://www.wampserver.com/en/>

□ Szerkesztés:

- NetBeans www.netbeans.org
- Notepad++

PHP szintaxis

□ PHP kód:

```
<?php  
// put your code here  
?>
```

□ Beágyazás HTML-be:

```
<!DOCTYPE html>  
<html>  
  <head>  
    <meta charset="UTF-8">  
    <title></title>  
  </head>  
  <body>  
    <?php  
      echo "Hello World";  
    ?>  
  </body>  
</html>
```

PHP szintaxis

- Kis- és nagybetű érzékeny
- Az utasítások végén pontosvesszőt teszünk
- A fehér karaktereket (space, tab) figyelmen kívül hagyja
- Megjegyzések: egysoros vagy többsoros

```
<body>
    <?php
        // Egysoros megjegyzés
        /* Többsoros
           megjegyzés */
    ?>
</body>
```

PHP típusok



□ Négy elemi típus

- logikai
- egész
- lebegőpontos
- szöveg

□ Két összetett típus

- tömb
- osztály

□ Speciális típusok

- erőforrás
- NULL
- callbacks

Változók

- A típusok meghatározása dinamikusan történik, értékadáskor, nem kell deklarálni
- A változónév a \$ karakterrel kezdődik
- A változónév kis- és nagybetű érzékeny
- A PHP a legtöbb esetben automatikusan konvertál a típusok között, ha arra szükség van
- Példa:

```
// Itt szám  
$valtozo = 2014;  
// de itt már szöveg!  
$valtozo = "kétezetizennégy";
```

Literálok

```
//Logikai
true
false
TRUE
FALSE

//Egész
12      //decimális
-34
0123    //oktális
0x0F    //hexadecimális
0b0101  //bináris
```

```
//Lebegőpontos
3.1415
5.6e12
-7E-2

//Karakterlánc
"szöveg"
'ez is szöveg'
```

Példa:

```
$1  = true;
$i  = -23;
$d  = 23.65;
$s1 = "egysoros szöveg";
$s2 = 'ez egy másik szöveg';
```

Típusokkal kapcsolatos függvények

□ Kiíratás

- echo
- print
- print_r
- var_dump

□ Típusbeállítás

- cast
- settype

□ Típuslekérdezés

- gettype()
- is_integer()
- is_float()
- is_numeric()
- is_string()
- is_bool()
- ...

Kiírások, típusműveletek

```
$1 = true;
$i = -23;
$d = 23.65;
$s = 'alma';

echo $1; //1
echo $i; //-23
echo $d; //23.65
echo $s; //alma

var_dump($1); //bool(true)
var_dump($i); //int(-23)
var_dump($d); //float(23.65)
var_dump($s); //string(4) "alma"
```

```
//Típuslekérdezés
echo gettype($1); //'boolean'
echo gettype($i); //'integer'
echo gettype($d); //'double'
echo gettype($s); //'string'
//
//Típusbeállítás
$sd1 = (string)$d;
$sd2 = $d;
settype($sd2, 'string');
echo gettype($sd1); //'string'
echo gettype($sd2); //'string'
```


Operátorok

□ Aritmetikai operátorok (y=5)

Operátor	Leírás	Példa	Eredmény
+	Összeadás	x=y+2	x=7
-	Kivonás	x=y-2	x=3
*	Szorzás	x=y*2	x=10
/	Osztás	x=y/2	x=2.5
%	Maradékos osztás	x=y%2	x=1
++	Növelés	x=++y	x=6
--	Csökkentés	x=--y	x=4

□ PHP kóddal:

```
$y = 5;  
$x = $y - 2; //x=3  
$x = $y + 2; //x=7  
$x = $y * 2; //x=10
```

```
$x = $y / 2; //x=2.5  
$x = $y % 2; //x=1  
$x = ++$y; //x=6  
$x = ++$y; //x=4
```

Operátorok

□ Hozzárendelő operátorok (x = 10, y=5)

Operátor	Példa	Ugyanaz	Eredmény
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
=	x=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

□ PHP kóddal:

```
$x = 10;  
$y = 5;  
$x = $y; // x = 5  
$x += $y; // x = 15
```

```
$x -= $y; // x = 5  
$x *= $y; // x = 50  
$x /= $y; // x = 2  
$x %= $y; // x = 0
```

Operátorok- Logikai

- Logikai operátorok:

- && (logikai és), || (logikai vagy), ! (logikai nem)

- Példa:

```
!true // false
true && false // false
true || false // true
true && (false || true) // true
!false && !(true || false) // false
```

Operátorok- Összehasonlító

- Összehasonlító operátorok (`==`, `!=`, `<`, `<=`, `>`, `>=`): az adott kifejezésnek mindig logikai értéke van
- Példa:

```
2 + 2 == 4      // true
2 + 2 != 4      // false
4 > 4           // false
4 >= 4          // true
5 < 5           // false
5 <= 5          // true
1 < 2 && 3 == 4  // false
```

Karakterláncok kezelése



- string literált 4 féle módon adhatunk meg:
 - aposztróffal
 - idézőjellel
 - heredoc
 - newdoc

Karakterláncok kezelése

- Aposztróf ('): a legegyszerűbb megadási mód.

- Példa:

```
$s1 = 'aposztróffal megadott karakterlánc';  
echo $s1;
```

- Az ' (aposztróf), \ (backslash) karakterek megjelenítését a \ escape karakterrel tehetjük meg

```
echo 'Ez egy aposztróf: \' '; //Ez egy aposztróf: '  
echo '<br>';  
echo 'You deleted C:\\*. *?'; //You deleted C:*. *?  
echo '<br>';  
echo 'C:\\WINDOWS\\System32'; //C:WINDOWS\\System32
```

Karakterláncok kezelése

- Idézőjel (“): az aposztrófhoz viszonyítva több escape szekvenciát tud kezelni és a változók behelyettesítődnek
- ▣ Escape karakterek: kocszi-vissza (`\n`), tab (`\t`), dollár (`$`)

```
echo "<font color=\"red\">\nszöveg\n</font>";
```

```
<font color="red">  
szöveg  
</font>
```

- ▣ Megjegyzés: a fenti kód újsor karakterrel nem a böngészőben megjelenő szövegben csinálunk sortörést, hanem csak a HTML kódban!

Karakterláncok kezelése

- ❑ *heredoc*: a „<<<” operátor vezeti be, amit követ egy azonosító, majd a tényleges szöveg. Ezt majd lezárja a korábbi azonosító
- ❑ Példa:

```
echo <<<VEGE
<!doctype html>
<html>
  <head>
    <meta charset = "utf-8">
    <title></title>
  </head>
  <body>
    <p>Szoveg<p/>
  </body>
</html >
VEGE;
```


Karakterláncok kezelése

- Behelyettesítés: változók automatikusan behelyettesítődnek, ha idézőjeles szövegben fordulnak elő, míg aposztrófos szövegben nem
- Példa:

```
$lang = "php";  
$num = 1000;  
print "A $lang nyelvben több mint $num beépített függvény van!";  
//A php nyelvben több mint 1000 beépített függvény van!  
  
print 'A $lang nyelvben több mint $num beépített függvény van!';  
//A $lang nyelvben több mint $num beépített függvény van!
```

Karakterláncok kezelése

- Karakterláncok összefűzése a "." (pont) operátorral történik

```
$txt1 = "Kiss";  
$txt2 = "Janos";  
echo $txt1 . " " . $txt2; //Kiss Janos
```

Vezérlési szerkezetek



- Feltételes utasítások:
 - if utasítás
 - if ... else utasítás
 - if ... elseif ... else utasítás
 - switch utasítás

Feltételes utasítás - if

□ Szintaxis:

```
<?php
    if (feltetel){
        //utasitas
    }
?>
```

□ Példa:

```
<?php
    $nev = "Pista";
    if ($nev == "Pista") {
        echo "A neved Pista <br/>";
    }
?>
```

Feltételes utasítás - if...else

□ Szintaxis:

```
if (feltetel){  
    //utasitas1;  
}  
else{  
    //utasitas2;  
}
```

□ Példa:

```
<?php  
    $n = 3;  
    if ($n == 3) {  
        echo "true";  
    } else {  
        echo "false";  
    }  
?>
```

Feltételes utasítás - if...elseif...else

□ Példa:

```
<?php
    $d = date("D");
    if ($d == "Fri")
        echo "Have a nice weekend!";
    elseif ($d == "Sun")
        echo "Have a nice Sunday!";
    else
        echo "Have a nice day!";
?>
```

Feltételes utasítás - switch

□ Szintaxis:

```
switch (n) {  
    case label1:  
        utasitas1;  
        break;  
    case label2:  
        utasitas2;  
        break;  
    default:  
        utasitas3;  
}
```

□ Példa:

```
<?php  
switch ($x) {  
    case 1:  
        echo "Number 1";  
        break;  
    case 2:  
        echo "Number 2";  
        break;  
    default:  
        echo "No number between 1 and 2";  
}  
?>
```

Ciklus utasítások - While

□ Szintaxis:

```
while (feltétel) {  
    // utasítások  
}
```

- Amíg a while feltétele igaz, a hozzá tartozó programrész újból és újból végrehajtódik
- A programrészen belül általában megváltoztatunk valamit, ami hatással lesz a while feltételére

Ciklus utasítások - While

□ 1.Példa:

```
<?php
$szamlalo = 1;
while ($szamlalo <= 12) {
    echo "$szamlalo kétszerese " . ($szamlalo * 2) . "<br>";
    $szamlalo++;
}
?>
```

□ 2.Példa:

```
<?php
$szoveg = "";
while ($szoveg != "XXXXX") {
    echo $szoveg . "<br>";
    $szoveg = $szoveg . "X";
}
?>
```

Ciklus utasítások - While

- 3.Példa: dinamikus HTML táblázat
- Kimenet:

```
<?php
$ar = 5;
$szamlalo = 10;
echo "<table border='1' align='center'>";
echo "<tr><th>Mennyiség</th>";
echo "<th>Ár</th></tr>";
while ($szamlalo <= 100) {
    echo "<tr><td>";
    echo $szamlalo;
    echo "</td><td>";
    echo $ar * $szamlalo;
    echo "</td></tr>";
    $szamlalo = $szamlalo + 10;
}
echo "</table>";
?>
```

Mennyiség	Ár
10	50
20	100
30	150
40	200
50	250
60	300
70	350
80	400
90	450
100	500

A do..while ciklus

□ Szintaxis:

```
do{  
    // utasitasok  
} while (feltétel);  
?>
```

- Ebben a szerkezetben először hajtódik végre a kód és csak azután értékelődik ki a feltétel. Ha a feltétel hamis lesz
- Ez a ciklus akkor lehet hasznos, ha mindenképpen szeretnénk, hogy a ciklushoz tartozó programrész még akkor is legalább egyszer lefusson, ha a feltétel már az első végrehajtáskor hamis

A do..while ciklus - Példák

□ Példa1:

```
<?php
$szam = 1;
do {
    echo "Végrehajtások száma: $szam<br>";
    $szam++;
} while ($szam > 200 && $szam < 400);
?>
```

□ Példa2:

```
<?php
$i = 5;
do {
    echo "i = " . $i . "<br>";
    $i--;
} while ($i > 3);
?>
```

A for ciklus

□ Szintaxis:

```
for ( változó_hozzárendelése; feltétel; számláló_növelése) {  
    //utasítások  
}
```

- Az első kifejezés rendszerint egy számlálónak ad kezdeti értékét, a második egy feltétel, ami alapján eldől, hogy folytatódik-e a ciklus; a harmadik egy számlálót növelő utasítás

A for ciklus - Példa

□ Példa:

```
<?php
for($szamlalo=1; $szamlalo<=12;$szamlalo++)
echo "$szamlalo kétszerese: " . ($szamlalo*2) . "<br>";
?>
```

□ Kimenet:

```
1 kétszerese: 2
2 kétszerese: 4
3 kétszerese: 6
4 kétszerese: 8
5 kétszerese: 10
6 kétszerese: 12
7 kétszerese: 14
8 kétszerese: 16
9 kétszerese: 18
10 kétszerese: 20
11 kétszerese: 22
12 kétszerese: 24
```

A for ciklus - Példa

□ Példa: dinamikus HTML táblázat

```
<?php
$ar = 5;
echo "<table border='1' align='center'>";
echo "<tr><th>Mennyiség</th>";
echo "<th>Ar</th></tr>";
for ( $i = 10; $i <= 100; $i += 10) {
    echo "<tr><td>";
    echo $i;
    echo "</td><td>";
    echo $ar * $i;
    echo "</td></tr>";
}
echo "</table>";
?>
```

Ciklusvezérlő utasítások: break

- *break*:

- ▣ Lehetővé teszi, hogy más feltételektől függően megszakítsuk egy ciklus futását

- Példa:

```
<?php
$i=1;
for (;;) {
    echo "<br>3 * $i = " . 3 * $i;
    $i++;
    if ($i > 10)
        break;
}
?>
```


Ciklusvezérlő utasítások: continue

□ *continue*:

- ▣ Segítségével az éppen folyó ismétlést befejezhetjük, mégpedig úgy, hogy ez ne eredményezze az egész ciklusból való kilépést, csak a következő ismétlés kezdetét jelentse

□ Példa:

```
<?php
for($i = 10; $i>-10; $i--){
    if ($i == 0){
        continue;
    }
    echo "<br> 100 / $i=" . 100 / $i;
}
?>
```

Egymásba ágyazott ciklus

□ Példa:

```
<?php
    echo "<table border=1>"; // HTML táblázat kezdete
    for ($y = 1; $y <= 12; $y++) {
        echo "<tr>"; // sor kezdete a HTML táblázatban
        for ($x = 1; $x <= 12; $x++) {
            echo "<td>"; // cella kezdete
            echo ($x * $y);
            echo "</td>"; // cella vége
        }
        echo "</tr>"; // sor vége
    }
    echo "</table>"; // táblázat vége
?>
```

Egymásba ágyazott ciklus

□ Kimenet:

1	2	3	4	5	6	7	8	9	10	11	12
2	4	6	8	10	12	14	16	18	20	22	24
3	6	9	12	15	18	21	24	27	30	33	36
4	8	12	16	20	24	28	32	36	40	44	48
5	10	15	20	25	30	35	40	45	50	55	60
6	12	18	24	30	36	42	48	54	60	66	72
7	14	21	28	35	42	49	56	63	70	77	84
8	16	24	32	40	48	56	64	72	80	88	96
9	18	27	36	45	54	63	72	81	90	99	108
10	20	30	40	50	60	70	80	90	100	110	120
11	22	33	44	55	66	77	88	99	110	121	132
12	24	36	48	60	72	84	96	108	120	132	144

Függvények

□ Szintaxis:

```
function függvéynév(paraméterlista){  
    utasítások  
    return kifejezés;  
}
```

- A függvény valójában egy zárt, önálló kódrészlet, melyet programunkból meghívhatunk. Amikor meghívjuk, a függvény törzse lefut. A függvénynek feldolgozás céljából értékeket adhatunk át. Amikor a függvény véget ér, a hívónak egy értéket ad vissza (return).
- Az összes függvéynél kötelező a zárójel, akár kell paramétert átadnunk, akár nem

Függvények - Példák

□ 1.Példa: sima kiíratás

```
<?php
function kiir sor($sor) {
    print "$sor<br>";
}
kiir sor("Első sor");
kiir sor("Közepes sor");
kiir sor("Ez meg az utolsó");
?>
```

□ 2.Példa: paraméter átadás, visszatérítési érték

```
<?php
function atlag($a, $b) { //függvény létrehozása
    $ab_atlag = ($a + $b) / 2;
    return $ab_atlag;
}

$szam1 = 3;
$szam2 = 5; //változók létrehozása
$SZAM = atlag($szam1, $szam2); //az átlag kiszámítása
echo "Átlag: " . $SZAM;
?>
```

Függvények - Példák

□ 3.Példa: alapértelmezett érték használata

```
<?php
function meretez($szoveg, $meret = 3) {
    echo "<font size=\"\$meret\" face=\"Tahoma, Arial, Sans-Serif\">$szoveg</font>";
}

meretez("Egy címsor<br>", 5);
meretez("szöveg<br>");
meretez("újabb szöveg<br>");
meretez("még több szöveg<br>");
?>
```

□ Kimenet:

```
Egy címsor
szöveg
újabb szöveg
még több szöveg
```

A változók láthatósága

- A függvényben használt változók az adott függvényre nézve helyiek maradnak, azaz a függvényen kívülről nem elérhetőek
- Egy függvényben szintén nem használhatunk olyan változónevet, amit nem ott hoztunk létre
- Ahhoz hogy egy függvénybe látható legyen egy változó, a változót láthatóvá kell tenni: paraméterátadással, *global* utasítás használata.
- Példa:

```
$x = 10;  
$y = 20;  
function kiir() {  
    global $x;  
    echo "$x<br>";  
    echo "$y<br>";  
}  
kiir();
```

Beépített függvények használata

- karakterlanc kezelő függvények:
 - ▣ echo, print, print_r (kiíratások), strlen (karakterlanc hossza), strcmp (összehasonlítás), substr (részlanc megtalálása), explode (sztring felbontása), stb.
- matematikai függvények:
 - ▣ abs, sqrt, exp, rand, fmod, oot, min, max, stb.
- tömb függvények:
 - ▣ sort, count, array_sum, array_merge, stb.
- fájlkezelő függvények:
 - ▣ fread, fputs, ock, fclose, stb.
- dátumkezelő függvények
 - ▣ time, date

Tömbök

- Indexelt tömbök létrehozás az `array()` függvény segítségével:
 - ▣ Példa: `$prim = array()` **vagy** `$prim = array(2, 3, 5, 7)`
 - ▣ Elemek elérése: `$prim[0]`, `$prim[1]`, `$prim[2]`, `$prim[3]`
- Létrehozás szögletes zárójel segítségével:
 - ▣ Példa:
 - `$prim[]=2;`
 - `$prim[]=3;`
 - `$prim[]=5;`
 - `$prim[]=7;`

Tömbök

□ Példa:

```
$prim1 = array(2, 3, 5, 7);  
echo "<br>";  
print_r($prim1) ;  
$prim2[] = 2;  
$prim2[] = 3;  
$prim2[] = 5;  
$prim2[] = 7;  
echo "<br>";  
print_r($prim2) ;  
$prim3[0] = 2;  
$prim3[1] = 3;  
$prim3[2] = 5;  
$prim3[3] = 7;  
echo "<br>";  
print_r($prim2) ;
```

□ Kimenet:

```
Array ( [0] => 2 [1] => 3 [2] => 5 [3] => 7 )  
Array ( [0] => 2 [1] => 3 [2] => 5 [3] => 7 )  
Array ( [0] => 2 [1] => 3 [2] => 5 [3] => 7 )
```

Asszociatív tömbök

- A karakterlánccal indexelt tömböket asszociatív tömböknek nevezzük
- Létrehozás:
 - ▣ Ezeket is az `array()` függvény vagy a szögletes zárójelek segítségével hozzuk létre.
 - ▣ Az első esetben a paramétereknek kulcs => érték alakú kifejezéseknek kell lenniük
 - ▣ A kulcs típusa lehet integer vagy string, az érték bármilyen típusú lehet.
- Példa:
 - ▣ `$auto = array("suly" => "100kg", "ev" => "2004", "ar" => "7000");`
 - ▣ `vagy $auto["suly"] = "100kg"; $auto["ev"] = "2004"; $auto["ar"] = "7000";`

Asszociatív tömbök

- Megjegyzés: ha nem adunk meg kulcsot egy adott értékhez, akkor annak a kulcsa az egész típusú indexek maximuma +1 lesz.
- Példa:

```
// Ez a tömb ugyanaz mint...  
array(5 => 43, 32, 56, "b" => 12);  
  
// ...ez a tömb  
array(5 => 43, 6 => 32, 7 => 56, "b" => 12);
```

Tömbök bejárása

□ Egész indexű tömbök:

▣ Példa:

```
$gyumolcs = array("alma", "körte", "málna",  
    "barack", "dinnye");  
for ($n = 0; $n < count($gyumolcs); ++$n) {  
    echo $gyumolcs[$n] . ", ";  
}
```

□ Asszociatív tömbök:

```
foreach (tömbnév as kulcs => érték) {  
    utasítások  
}
```

□ Példa:

```
$auto = array("suly" => "100kg", "ev" => "2004", "ar" => "7000");  
foreach ($auto as $kulcs => $ertek) {  
    print $kulcs . ": " . $ertek . "<br>";  
}
```

Tömbök - hasznos függvények

- *unset*: egy értéket töröl a tömbből. Figyelj arra, hogy a tömb nem lesz újraindexelve! Az újraindexelés hatását a `array_values()` függvénnyel lehet elérni.
- Példa:

```
$a = array(1 => 'egy', 2 => 'ketto', 3 => 'három');  
unset($a[2]);  
// ennek eredményeképpen $a így fog kinézni:  
// $a = array(1 => 'egy', 3 => 'három');  
// és NEM így:  
// $a = array(1 => 'egy', 2 => 'három');  
$b = array_values($a);  
// Itt a $b így néz ki: array(0 => 'egy', 1 => 'három');
```

Tömbök - hasznos függvények

- *print_r*: tömb elemeinek a kiíratása.
- Példa:

```
$a = array( 'szín' => 'piros', 'íz' => 'édes', 'alak' => 'kerek', 'név' => 'alma');  
print_r($a);
```

- Eredmény:

```
Array ( [szín] => piros [íz] => édes [alak] => kerek  
[név] => alma )
```

Többdimenziós tömbök

- Mivel a tömb egy értéke bármi lehet, értéként akár egy másik tömb is megadható. Ilyen formában többdimenziós tömböket is lehet készíteni

- Példa:

```
$rockBands = array(  
    array('Beatles', 'Love Me Do', 'Hey Jude', 'Helter Skelter'),  
    array('Rolling Stones', 'Waiting on a Friend', 'Angie', 'Yesterday\'s Papers'),  
    array('Eagles', 'Life in the Fast Lane', 'Hotel California', 'Best of My Love')  
);
```

- Kiíratás:

```
foreach($rockBands as $rockBand) {  
    echo '<tr>';  
    foreach($rockBand as $item) {  
        echo "<td>$item</td>";  
    }  
    echo '</tr>';  
}
```

Rockband	Song 1	Song 2	Song 3
Beatles	Love Me Do	Hey Jude	Helter Skelter
Rolling Stones	Waiting on a Friend	Angie	Yesterday's Papers
Eagles	Life in the Fast Lane	Hotel California	Best of My Love

Fejlesztési eszközök

□ Böngészők:



- Mozilla FireFox 
- Chrome 
- Opera 
- Microsoft Explorer 

□ Böngésző kiegészítő

- FireBug: a FireFox kiegészítője. Weboldal készítéshez, tanulmányozáshoz használjuk.



□ Szerkesztő (HTML, CSS):

- NetBeans 
- Notepad++ 

Fejlesztési eszközök

- FTP (File Transfer Protocol - állománytovábbító protokoll)
kliens programok:
 - WinSCP
 - SmartFTP
- A webes projekteket minden hallgató egy FTP kliens program segítségével tudja a szerverre másolni

A NetBeans fejlesztői környezet

- A NetBeans IDE (Integrated Development Environment) egy nyílt forráskódú integrált fejlesztői környezet, amely a Java nyelven alapul.
- A Sun Microsystems 2000 júniusában hozta létre, ma már az Oracle fejleszti tovább.
- Lehetővé teszi a programozók számára, hogy programokat írjanak, fordítsanak, teszteljenek, hibakeresést végezzenek az alkalmazásokban, programokat telepítsenek.
- 2010-től PHP támogatást is kapott, így lehetőség van PHP alapú webalkalmazások készítésére
- A www.netbeans.org oldalról tölthető le. A telepítéséhez szükség van java környezetre, amely a www.java.com/en/download/index.jsp oldalról tölthető le.

A NetBeans letölthető változatai

NetBeans IDE Download Bundles					
Supported technologies *	Java SE	Java EE	C/C++	HTML5 & PHP	All
④ NetBeans Platform SDK	•	•			•
④ Java SE	•	•			•
④ Java FX	•	•			•
④ Java EE		•			•
④ Java ME					•
④ HTML5		•		•	•
④ Java Card™ 3 Connected					•
④ C/C++			•		•
④ Groovy					•
④ PHP				•	•
Bundled servers					
④ GlassFish Server Open Source Edition 4.0		•			•
④ Apache Tomcat 7.0.41		•			•
	Download	Download	Download	Download	Download
	Free, 84 MB	Free, 185 MB	Free, 59 MB	Free, 60 MB	Free, 204 MB

A NetBeans tulajdonságai



- Szín sémák használata
- Kódkiegészítés
- Hibakeresés
- Kódkiemelés
- Kódrendezés (Jobb klikk a kódon -- *Format*)
- HTML állományok megtekintése a böngészőben (Jobb klikk a kódon -- *View (Shift+F6)*)

Könyvészet



- <http://hu.wikipedia.org>
- <http://www.w3schools.com>
- <http://webfejlesztes.inf.elte.hu>