INTRODUZIONE A PHP

Dr. Barbara Rita Barricelli

UNIVERSITÀ DEGLI STUDI DI BRESCIA

ORARIO, MATERIALI, CONTATTO

Giorno	Orario	Lezione
Lunedì 6	09:00 - 13:00 14:00 - 16:00	1
Martedì 7	09:00 - 13:00 14:00 - 16:00	2
Giovedì 9	09:00 - 13:00 14:00 - 16:00	3
Venerdì 10	09:00 - 13:00 14:00 - 16:00	4
Martedì 14	09:00 - 13:00 14:00 - 16:00	5

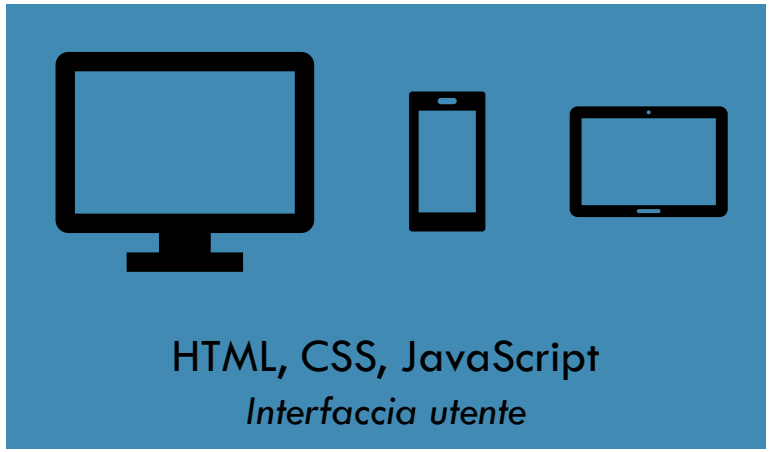
Sito web:
https://barbara-barricelli.unibs.it/

Per domande o suggerimenti:
barbara.barricelli@unibs.it

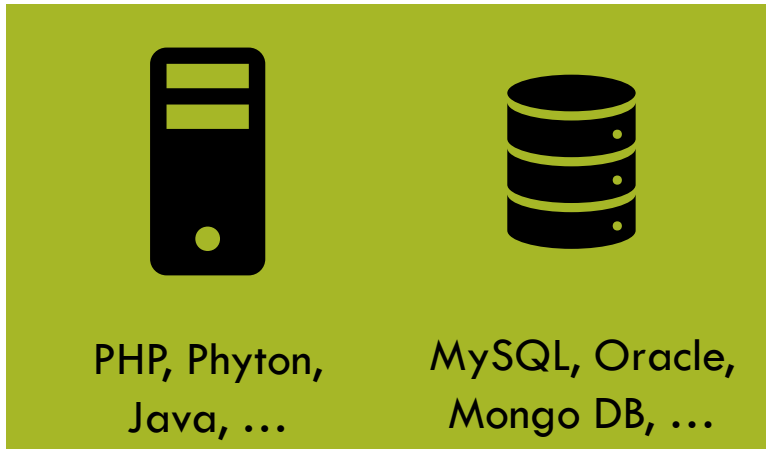
LEZIONE 1

MODELLO CLIENT/SERVER

Client Side



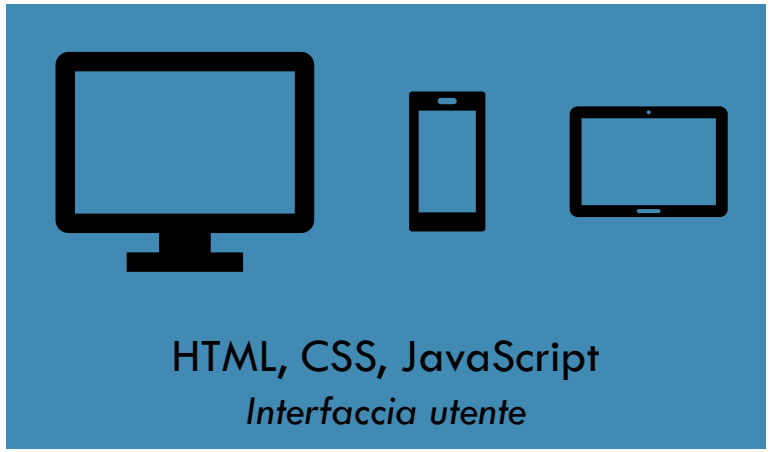
Server Side



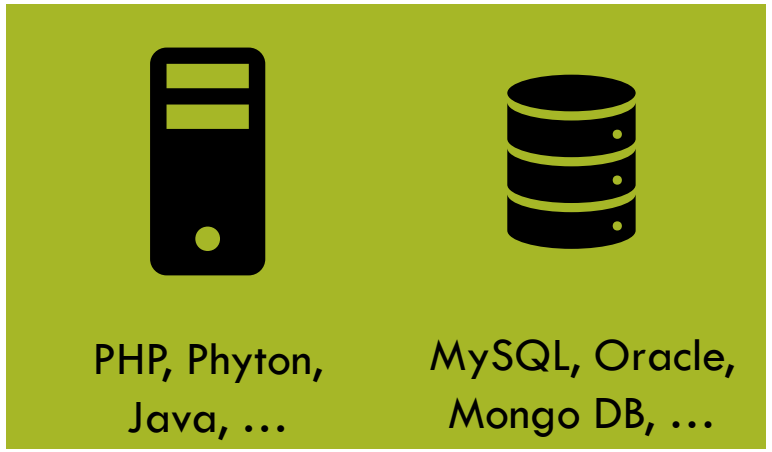
MODELLO CLIENT/SERVER

Full Stack Web Development

Front End – Client Side



Back End – Server Side



XAMPP

- Uno dei più popolari ambienti di sviluppo PHP
- Gratuito
- Server (Apache) + Database (MariaDB o SQL nelle versioni più vecchie) + PHP + …

https://www.apachefriends.org/it/index.html

PHP: HYPERTEXT PREPROCESSOR

- Linguaggio di scripting lato server:
  - gli script in linguaggio PHP sono eseguiti sul server
  - il codice eseguito non è visibile lato client
  - il risultato dell'esecuzione di uno script PHP viene restituito al browser in formato HTML
- Linguaggio interpretato
- Interprete Open Source
- Gratuito
- I file hanno estensione .php
- Possono contenere codice PHP e HTML

CREIAMO UN FILE .HTML

```
<html>
  <body>
    <p>Hello World!</p>
  </body>
</html>
```

CREIAMO UN FILE .HTML

```
<html>
  <body>
    <p>Hello World!</p>
  </body>
</html>
```

VISUALIZZIAMOLO NEL BROWSER

CREIAMO UN FILE .HTML

```
<html>
  <body>
    <p>Hello World!</p>
  </body>
</html>
```

VISUALIZZIAMO IL CODICE SORGENTE (TRAMITE IL BROWSER)

CREIAMO UN FILE .PHP

```
<html>
  <body>
    <p>
      <?php
        echo "Hello World!";
      ?>
    </p>
  </body>
</html>
```

CREIAMO UN FILE .PHP

```
<html>
  <body>
    <p>
      <?php
        echo "Hello World!";
      ?>
    </p>
  </body>
</html>
```

VISUALIZZIAMOLO NEL BROWSER

CREIAMO UN FILE .PHP

```
<html>
  <body>
    <p>
      <?php
        echo "Hello World!";
      ?>
    </p>
  </body>
</html>
```

VISUALIZZIAMO IL SORGENTE DELLA PAGINA

COMPLICHIAMO UN PO' L'ESEMPIO

```
<html>
  <body>
    <p>
      <?php
        echo "Hello World!<br />Ciao!";
      ?>
    </p>
  </body>
</html>
```

COMPLICHIAMO UN PO' L'ESEMPIO

```
<html>
  <body>
    <p>
      <?php
        echo "Hello World!<br />Ciao!";
      ?>
    </p>
  </body>
</html>
```

VISUALIZZIAMO IL SORGENTE DELLA PAGINA

CASE SENSITIVE?

```
<html>
  <body>
    <p>
      <?php
        echo "Hello World!<br />";
        ECHO "Hello World!<br />";
        EcHo "Hello World!<br />";
       ?>
    </p>
  </body>
</html>
```

CASE SENSITIVE?

```
<html>
  <body>
    <p>
      <?php
        echo "Hello World!<br />";
        ECHO "Hello World!<br />";
        EcHo "Hello World!<br />";
       ?>
    </p>
  </body>
</html>
```

VISUALIZZIAMOLO NEL BROWSER

LE VARIABILI

```
<?php
  $color = "red";
  echo "My car is " . $color . "<br />";
?>
```

DICHIARAZIONE DI VARIABILI

- Le variabili in PHP cominciano con il simbolo “$”
- Un nome di variabile deve iniziare con una lettera o con il simbolo “_” (underscore) seguiti da una serie qualsiasi di lettere, numeri o underscore.
- PHP è un linguaggio non tipizzato, quindi il tipo delle variabili è deciso dall’interprete a runtime in base al contesto. Verificate con questo esempio:

```
$number = 25;

echo $number;

$number = "questa è una stringa";

echo $number;
```

CONCATENAZIONE DI VALORI DI VARIABILI

```
$nome = "Mickey";
$cognome = "Mouse";
$nome_completo = $nome." ".$cognome;
echo $nome."<br />";
echo $cognome."<br />";
echo $nome_completo;
```

LE VARIABILI: CASE SENSITIVE?

```php
<?php
  $color = "red";
  echo "My car is " . $color . "<br />";
  echo "My house is " . $COLOR . "<br />";
  echo "My cat is " . $coLOR . "<br />";
?>
```

ALCUNI TIPI DI DATO

- Boolean

  Può assumere solo due valori: TRUE o FALSE.

- Integer

  Può assumere un valore qualsiasi appartenente all'insieme numerico $Z = \{...,-1,0,1,...\}$.

- String

  Una stringa è un insieme di caratteri a 8 bit.

- Array

  Una mappa ordinata di elementi.

BOOLEAN

Può assumere solo 2 valori: “true” e “false”

```
$var = true;
$var = false;
```

“True” e “False” sono case insensitive (verificate usando TRUE/true/True o FALSE/false/False).

Per stampare a video il valore di una variabile boolean si usa `var_dump(<nomevar>);`

INTEGER

Può assumere qualsiasi valore intero positivo o negativo.

Il valore massimo e minimo di una variabile integer dipende dall'architettura del calcolatore su cui viene eseguito lo script. Si può conoscere tramite due costanti (vi spiegherò dopo…): `PHP_INT_MAX` e `PHP_INT_MIN`

Il valore boolean "true" viene valutato 1 se convertito in Integer.

FLOATS/REAL NUMBERS/DOUBLES

Un numero con cifre decimali. Si usa il punto e non la virgola!

```
$x = 10.365;
```

STRING

Una stringa può essere specificata in due modi:

```
$string = 'esempio di stringa';
$string = "esempio di stringa";
```

1. Provate a stampare a video entrambe e verificate se ci sono differenze.
2. Provate a concatenare due stringhe con il metodo visto per la concatenazione di valori di variabili.

STRING

Verificate questi esempi (ci sono differenze/problemi?):

```
echo 'Torniamo un\'altra volta';
echo "Torniamo un'altra volta";
echo "Torniamo un\'altra volta";
echo 'Torniamo un'altra volta';
echo 'Anna disse "Ciao" e se ne andò';
echo "Anna disse \"Ciao\" e se ne andò";
echo 'Anna disse \"Ciao\" e se ne andò';
```

STRING

Sostituzione delle variabili nelle stringhe con apici doppi " "

```
$var = 'Paolo ';

echo "Lo studente $var è promosso";
```

STRING

Sostituzione delle variabili nelle stringhe con apici doppi " "

```
$var = 'Paolo ';

echo "Lo studente $var è promosso";
/* stampa: Lo studente Paolo  è promosso*/
```

STRING

Sostituzione delle variabili nelle stringhe con apici doppi " "

```
$var = 'Paolo ';

echo "Lo studente $var è promosso";
/* stampa: Lo studente Paolo  è promosso*/

echo "Lo studente $varRossi è promosso";
```

STRING

Sostituzione delle variabili nelle stringhe con apici doppi " "

```
$var = 'Paolo ';

echo "Lo studente $var è promosso";
/* stampa: Lo studente Paolo  è promosso*/

echo "Lo studente $varRossi è promosso";
/* stampa: Lo studente è promosso*/
```

STRING

Sostituzione delle variabili nelle stringhe con apici doppi " "

```
$var = 'Paolo ';

echo "Lo studente $var è promosso";
/* stampa: Lo studente Paolo  è promosso*/

echo "Lo studente $varRossi è promosso";
/* stampa: Lo studente è promosso*/

echo "Lo studente ${var}Rossi";
```

STRING

Sostituzione delle variabili nelle stringhe con apici doppi " "

```
$var = 'Paolo ';

echo "Lo studente $var è promosso";
/* stampa: Lo studente Paolo  è promosso*/

echo "Lo studente $varRossi è promosso";
/* stampa: Lo studente è promosso*/

echo "Lo studente ${var}Rossi";
/* stampa: Lo studente Paolo Rossi*/
```

COSTANTI

E' possibile definire delle costanti da utilizzare in seguito:

```
define('MIN_VALUE', '18');
define('MAX_VALUE', '30');
```

1. Perché è utile poter definire delle costanti?
2. In che posizione è meglio definirle all'interno di uno script PHP?
3. Vi ricordate `PHP_INT_MAX` e `PHP_INT_MIN` ?

OPERATORI ARITMETICI

Esempio	Nome	Risultato
`$a + $b`	Addizione	Somma di $a e $b
`$a - $b`	Sottrazione	Differenza tra $a e $b
`$a * $b`	Moltiplicazione	Prodotto di $a e $b
`$a / $b`	Divisione	Quoziente di $a e $b
`$a % $b`	Modulo	Resto di $a diviso per $b
`$a ** $b`	Esponente	Risultato di $a elevato alla $b

PRECEDENZE TRA OPERATORI ARITMETICI

Provate ad inserire queste istruzioni in uno script e a stamparne i risultati:

```
$addizione = 5 + 5;
$sottrazione = 5 - 5;
$moltiplicazione = 5 * 5;
$divisione = 5 / 5;
$modulo = 7 % 5;
echo 3 + 5 / 2;
```

PEMDAS

1. Parentesi
2. Esponenti
3. Moltiplicazioni
4. Divisioni

hanno la stessa priorità ma vanno eseguite in ordine da sinistra verso destra

5. Addizioni
6. Sottrazioni

hanno la stessa priorità ma vanno eseguite in ordine da sinistra verso destra

PEMDAS

1. Parentesi
2. Esponenti
3. Moltiplicazioni
4. Divisioni
5. Addizioni
6. Sottrazioni

$10 \div (3 + 2) \times 4 + 5^2 + 6 - 9$

PEMDAS

1. Parentesi
2. Esponenti
3. Moltiplicazioni
4. Divisioni
5. Addizioni
6. Sottrazioni

$10 \div (3 + 2) \times 4 + 5^2 + 6 - 9$	P
$10 \div 5 \times 4 + 5^2 + 6 - 9$	E
$10 \div 5 \times 4 + 25 + 6 - 9$	D
$2 \times 4 + 25 + 6 - 9$	M
$8 + 25 + 6 - 9$	A
$33 + 6 - 9$	A
$39 - 9$	S
30	

PEMDAS

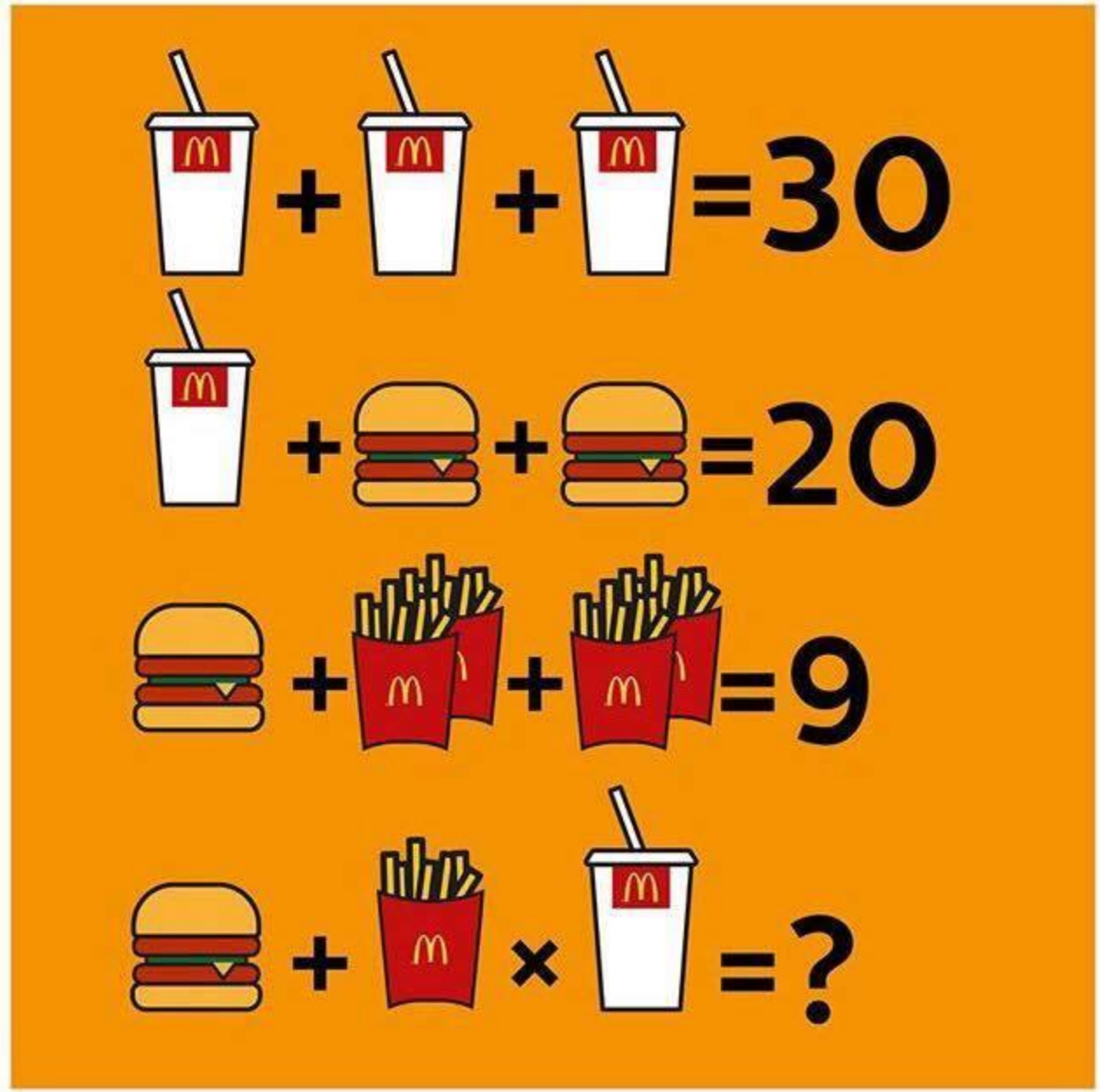
1. Parentesi
2. Esponenti
3. Moltiplicazioni
4. Divisioni
5. Addizioni
6. Sottrazioni

$4^2 - 6 \times 2 \div 4 \times 3 + 5$

PEMDAS

1. Parentesi
2. Esponenti
3. Moltiplicazioni
4. Divisioni
5. Addizioni
6. Sottrazioni

$4^2 - 6 \times 2 \div 4 \times 3 + 5$	E
$16 - 6 \times 2 \div 4 \times 3 + 5$	M
$16 - 12 \div 4 \times 3 + 5$	D
$16 - 3 \times 3 + 5$	M
$16 - 9 + 5$	S
$7 + 5$	A
12	


+ + =30
+ + =20
+ + =9
+ × =?

OPERATORI DI ASSEGNAMENTO

Verificate il risultato di queste istruzioni:

```
$a = ($b = 4) + 5;

$a += 5;

$b = "Hello ";

$b .= "There!";
```

OPERATORI DI CONFRONTO

Esempio	Nome	Risultato
`$a == $b`	Uguale	TRUE se $a è uguale a $b
`$a != $b`	Diverso	TRUE se $a è diverso da $b
`$a < $b`	Minore	TRUE se $a è minore di $b
`$a > $b`	Maggiore	TRUE se $a è maggiore di $b
`$a <= $b`	Minore o uguale	TRUE se $a è minore o uguale a $b
`$a >= $b`	Maggiore o uguale	TRUE se $a è maggiore o uguale a $b

OPERATORI DI INCREMENTO/DECREMENTO

Esempio	Nome	Risultato
++$a	Pre-incremento	Incrementa $a di 1 e poi ne restituisce il valore
$a++	Post-incremento	Restituisce il valore di $a e poi lo incrementa di 1
--$a	Pre-decremento	Decrementa $a di 1 e poi ne restituisce il valore
$a--	Post-decremento	Restituisce il valore di $a e poi lo decrementa di 1

OPERATORI DI INCREMENTO/DECREMENTO

```
$a = 5;
echo $a++ . "-" . $a . "<br />";

$a = 5;
echo ++$a . "-" . $a . "<br />";

$a = 5;
echo $a-- . "-" . $a . "<br />";

$a = 5;
echo --$a . "-" . $a . "<br />";
```

OPERATORI LOGICI

Esempio	Nome	Risultato
`$a and $b` `$a && $b`	And	TRUE se sia $a che $b sono TRUE
`$a or $b` `$a || $b`	Or	TRUE se $a è TRUE o se $b è TRUE
`$a xor $b`	Xor	TRUE se $a è TRUE o se $b è TRUE ma non entrambi
`!$a`	Not	TRUE se $a è FALSE

ARRAY

Per creare un array si usa la funzione array()

```
$giorni=array("Lu","Ma","Me","Gio","Ve");
```

Viene associato automaticamente un indice per riferirsi agli elementi dell'array. Per accedere agli elementi si usa l'operatore "[ ]" e il primo elemento è il numero 0. Verificate quale elemento viene stampato scrivendo:

```
echo $giorni[2];
```

Qual è l'indice dell'elemento "Ve"?

Qual è il valore dell'elemento in posizione [5]?

ARRAY

Per aggiungere elementi ad un array esistente: $<nome array>[]=<valore>;

```
$giorni[]="Sa";
```

Per eliminare elementi da un array o l'intero array:

```
unset($ar[2]);
unset($ar);
```

ARRAY ASSOCIATIVO

La funzione array() permette di specificare le chiavi di ricerca da associare ai valori contenuti secondo la sintassi: array(key1=>val1, ..., keyn=>valn)

```
$ar = array('nome'=>'Mickey', 'cognome'=>'Mouse');
```

La chiave dell'array può essere una stringa o un intero. Un array può avere chiavi di tipo diverso.

```
$ar = array('nome'=>'Mickey', 23=>'Minnie');
```

Per stampare il contenuto di un array:

```
print_r($ar);
```

OPERATORI PER ARRAY

Esempio	Nome	Risultato
`$a + $b`	Unione	L'unione dei due array $a e $b (solo se associativi e con chiavi diverse - chiavi uguali vengono ignorate)
`$a == $b`	Uguali	TRUE se $a e $b hanno le stesse coppie chiave/valore
`$a != $b` `$a <> $b`	Diversi	TRUE se $a e $b sono diversi
`$a === $b`	Identici	TRUE se $a e $b hanno le stesse coppie chiave/valore nello stesso ordine
`$a !== $b`	Non identici	TRUE se $a e $b non sono identici

STRUTTURE DI CONTROLLO: IF

```
if (expr)
  statement
```

Provate questo esempio:

```
$a = 15;
$b = 10;
if ($a > $b)
  echo "a è maggiore di b";
```

STRUTTURE DI CONTROLLO: IF

Ora modifichiamo l'esempio in questo modo:

```
$a = 10;
$b = 15;
if ($a > $b)
  echo "a è maggiore di b";
else
  echo "a NON è maggiore di b";
```

STRUTTURE DI CONTROLLO: IF

… e modifichiamo ulteriormente:

```
$a = 10;
$b = 10;
if ($a > $b)
  echo "a è maggiore di b";
elseif ($a == $b)
  echo "a è uguale a b";
else
  echo "a è minore di b";
```

STRUTTURE DI CONTROLLO: IF

Se le istruzioni da eseguire sono più di una usiamo le parentesi:

```
$a = 10;
$b = 10;
if ($a > $b){
  echo "a è maggiore di b";
  echo "fine";
}
elseif ($a == $b) {
  echo "a è uguale a b";
  echo $a. " " .$b;
}
else
  echo "a è minore di b";
```

STRUTTURE DI CONTROLLO: WHILE

```
while (expr)
  statement
```

Proviamo questo esempio:

```
$i = 1;
while ($i <= 10) {
    echo $i++;
}
```

STRUTTURE DI CONTROLLO: WHILE

```
while (expr)
   statement
```

Proviamo questo esempio:

```
$i = 1;
while ($i <= 10) {
     echo $i++;
}
```

```
echo $i++;
```

Può essere sostituita da due istruzioni separate.
Quali?

STRUTTURE DI CONTROLLO: FOR

```
for (expr1; expr2; expr3)
    statement
```

`expr1`: viene eseguita all'inizio dell'esecuzione del ciclo for (sempre, senza condizioni)

`expr2`: viene valutata all'inizio di ogni iterazione, se TRUE si esegue `statement` altrimenti termina il ciclo

`expr3`: viene eseguita al termine di ogni iterazione

STRUTTURE DI CONTROLLO: FOR

Proviamo questo esempio:

```
for ($i = 1; $i <= 10; $i++) {
    echo $i;
}
```

STRUTTURE DI CONTROLLO: FOREACH

```
foreach (iterable_expression as $value)
    statement
```

Proviamo questo esempio:

```
$arr = array(1, 2, 3, 4);
foreach ($arr as $value) {
    $value = $value * 2;
    echo $value." ";
}
print_r($arr);
```

STRUTTURE DI CONTROLLO: FOREACH

```
foreach (iterable_expression as $value)
    statement
```

Proviamo questo esempio:

```
$arr = array(1, 2, 3, 4);
foreach ($arr as &$value) {
    $value = $value * 2;
    echo $value." ";
}
print_r($arr);
```

Cosa è cambiato?
A cosa serve la & prima di $value?

STRUTTURE DI CONTROLLO: SWITCH

Proviamo questo esempio:

```
if ($i == 0) {
    echo "i è uguale a 0";
} elseif ($i == 1) {
    echo "i è uguale a 1";
} elseif ($i == 2) {
    echo "i è uguale a 2";
}
```

STRUTTURE DI CONTROLLO: SWITCH

Proviamo questo esempio:

```
if ($i == 0) {
    echo "i è uguale a 0";
} elseif ($i == 1) {
    echo "i è uguale a 1";
} elseif ($i == 2) {
    echo "i è uguale a 2";
}
```

Abbiamo un modo per semplificare!
Possiamo usare la struttura switch

STRUTTURE DI CONTROLLO: SWITCH

Proviamo questo esempio:

```
switch ($i) {
    case 0:
        echo "i è uguale a 0";
        break;
    case 1:
        echo "i è uguale a 1";
        break;
    case 2:
        echo "i è uguale a 2";
        break;
}
```

INCLUDE / INCLUDEONCE

Creiamo due file PHP:

vars.php

```php
<?php
    $color = 'green';
    $fruit = 'apple';
?>
```

test.php

```php
<?php
    echo "A $color $fruit";
    include 'vars.php';
    echo "A $color $fruit";
?>
```

INCLUDE / INCLUDEONCE

Creiamo due file PHP:

vars.php

```php
<?php
    $color = 'green';
    $fruit = 'apple';
?>
```

test.php

```php
<?php
    echo "A $color $fruit";
    include 'vars.php';
    echo "A $color $fruit";
?>
```

`includeonce` ha un comportamento simile a quello di `include`.

La differenza è che se il codice del file è già stato incluso in precedenza, non verrà incluso di nuovo.

FUNZIONI

- Funzione: un blocco di codice composto da una sequenza di istruzioni che prende in input dei parametri, effettua delle operazioni e restituisce un output.
- PHP offre moltissime funzioni già pronte (come ad esempio `print_r()`) ma permette anche di crearne di personalizzate.

VANTAGGI:

- Evitare la ripetizione di codice
- Rendere il codice più semplice da mantenere
- Riutilizzo in altre applicazioni

FUNZIONI

```
function name(params)
{
    statement;
    return value;
}
```

Il nome di una funzione deve iniziare con una lettera o un underscore, e può continuare con lettere, numeri e underscore. Non è case sensitive.

FUNZIONI

```
function name(params)
{
    statement;
    return value;
}
```

I params sono facoltativi così come la restituzione di un valore.
Se non si restituisce un valore è sempre meglio ritornarne uno booleano (true o false) per far sapere che la funzione è stata eseguita con successo.

FUNZIONI

Una volta definita una funzione è possibile richiamarla:

```
name(param1, param2);
```

FUNZIONI

Proviamo questo esempio:

```
function perimetroRettangolo($base, $altezza)
{
    $perimetro = ($base + $altezza) * 2;
    return $perimetro;
}

$perimetro = perimetroRettangolo(4, 5);
```

VISIBILITÀ (SCOPE) DELLE VARIABILI

Le variabili definite in una funzione possono essere visibili solo nella funzione stessa (variabili locali).

Le variabili globali sono utilizzabili in qualsiasi punto del programma.

VISIBILITÀ (SCOPE) DELLE VARIABILI

```
function test()
{
    $x = 'dentro';
    echo $x;
}

$x = 'fuori';
test();
echo $x;
```

PASSAGGIO PER VALORE / PER RIFERIMENTO

I valori possono essere passati alle funzioni in due modi:

- Per valore: i valori passati vengono copiati nelle variabili definiti nei parametri della funzione. il valore originale della variabile non viene modificato dalla funzione.
- Per riferimento: non viene copiato il valore di una variabile ma la sua posizione di memoria. Il valore originale della variabile verrà modificato dalla funzione. Si antepone una & davanti al nome della variabile.

PASSAGGIO PER VALORE / PER RIFERIMENTO

```php
function myFunction($x)
{
    $x = 5;
    return $x;
}

$x = 10;
echo $x;
myFunction($x);
echo $x;
```

```php
function myFunction(&$x)
{
    $x = 5;
    return $x;
}

$x = 10;
echo $x;
myFunction($x);
echo $x;
```

ESERCIZIO 1

- Creare uno script PHP che stampi a video la tabellina del 2

ESERCIZIO 2

- Creare uno script PHP che stampi a video le tabelline dal 2 al 10

ESERCIZIO 3

- Creare uno script PHP che dato un numero intero a scelta ne visualizza i suoi divisori.