Introduzione a PHP

Variabili e Vettori in PHP / Operatori

I mattoni per la costruzione di un programma php sono le variabili e gli array. Impariamo ad usare questi elementi fondamentali del linguaggio insieme agli operatori più comuni

In php esistono vari tipi di variabili. I quattro principali posso essere schematizzati nella tabella seguente

integer	Numeri interi
double	Numeri in virgola mobile
string	Stringhe
array	Array scalari o associativi

Anche in php, come in molti altri linguaggi di scripting, non è necessario dichiarare le variabili che useremo nel programma. Per inizializzare una variabile basta semplicemente scriverne il nome, preceduto dal **carattere $,** ed **assegnarle un valore usando l'operatore =** . Generalmente non serve specificare il tipo di una variabile poiché viene deciso automaticamente dall'interprete in base al contesto in cui la variabile stessa viene usata, vediamo:

```
<?php
$var = "0"; // $var e' una stringa (ASCII 48)
$var++;     // $var e' la stringa "1" (ASCII 49)
$var += 1;  // $var ora e' un intero (2)
$var = $var + 1.3; // $var ora e' un numero in virgola mobile (3.3)
$var = 5 + "10 + qualcosa"; // errore
$var = 5 + "qualcosa + 10"; // errore
```

All'occorrenza però possiamo eseguire sia una definizione che una conversione di tipo usando rispettivamente la funzione settype() oppure un'operazione di casting. I cast più comuni sono:

(int), (integer)	Conversione in intero
(double), (float) e anche (real)	Conversione in numero a doppia precisione
(string)	Conversione in stringa
(array)	Conversione in array

```
<?php
settype($b, "integer"); // $b e' un intero
$a = 10; // $a è un intero
$b = (double) $a; // $b è un numero in virgola mobile
// verifichiamo
echo gettype($b); // stampa double
echo "<br>";
$c = 1; // $c è un intero
$d = (string) $c; // $d è la stringa "1" (ASCII 49)
// verifichiamo
echo gettype($d); // stampa string
?>
```

Nell'esempio abbiamo usato anche la funzione gettype() che ci ha permesso di verificare se la variabile $b fosse davvero un double dopo averla sottoposta alla conversione di tipo.

Proviamo a completare il seguente esercizio:

```
<?php
        // Descrizione programma: Controllo tipi variabili
        // Data ult. modifica:    4 ott 2014
        // Scritto da:            F. Sacco

        $i=5;
        $d=1.33;
        $s="ciao";
        $v=array( 1,2,3);

        echo " i e' un ".gettype($i)."<br>";
        echo " d e' un ".gettype($d)."<br>";
        echo " v e' un ".gettype($v)."<br>";
        // completare
        if (gettype($i) == "integer")
                echo "i e' un intero";
        // completare

        /*
          + difficile:    definire il seguente array associativo, e
completare
                          l'esercizio senza if/else
          $tipi = array("integer" => "intero", "float" => "numero float",
                        "string" => "stringa" ... );
        */
?>
```

```
//Soluzione esercizio proposto sopra
```

```php
<?php
 $i=5;
 $d=1.33;
 $s="ciao";
 $v=array( 1,2,3);

 $tipi = array("integer" => "intero", "float" => "numero float", "string"
=> "stringa", "double" => "numero double", "array" => "vettore" );
 echo " i e' un " .$tipi[gettype($i)]."<br>";
 echo " d e' un " .$tipi[gettype($d)]."<br>";
 echo " v e' un " .$tipi[gettype($v)]."<br>";
 echo " s e' un " .$tipi[gettype($s)]."<br>";
?>
```

Introduzione agli array

E' bene ricordare che PHP non fa distinzione fra **array scalari** (l'indice dell'array e' un numero intero) e **array associativi** (l'indice e' una stringa o altro). Il modo più semplice per inizializzare un'array e' quello di assegnare un valore ai suoi elementi

```php
<?php
$a[0] = 1; // array scalare
$a[1] = 2;
$b[0] = 1;
$b[] = 2; // $b[1] = 2;
$b[] = 3; // $b[2] = 3;
$c["uno"] = 1; // array associativo
$c["due"] = 2;
?>
```

oppure possiamo usare la dichiarazione **array()**

```php
<?php
$a = array(1, 2, 3, 4, 5); // array scalare
echo $a[0]; // stampa 1
$b = array("uno" => 1,
           "due" => 2,
           "tre" => 3); // array associativo
echo $b["tre"]; // stampa 3
?>
```

Gli array possono essere definiti dinamicamente:

Si provi il programma qui riportato:

```
<pre>
<?php
        // Descrizione programma: Creazione array dinamici
        // Data ult. modifica:    4 ott 2014
        // Scritto da:            F. Sacco

        $v1=array();
        $v2=array();
        $v3=array();
        $v4=array();
        $v5=array();

        // commenta il primo e lancia solo il secondo ciclo

        for ($i=0; $i < 10; $i++)
                $v1[$i]=$i+1;
        print_r($v1);

        for ($i=0; $i < 10; $i++)
                $v1[]=$i+11;
        print_r($v1);


        for ($i=0; $i < 10; $i++)
                $v2[]=rand(1,100);
        print_r($v2);

        // in C o C++ non e' possibile....
        for ($i=0; $i < 10; $i++)
                $v3[rand(1,1000)]=rand(1,10);
        print_r($v3);

        $v4=$v3;    // e' possibile!!!

        // strano ...funziona???
        $indici = array("zero","uno","due","tre","quattro");
        for($i=0; $i < 5; $i++)
                $v4[$indici[$i]]=rand(1,10);
        print_r($v4);

        // strano e difficile da capire...
        $indici = array("zero","uno","due","tre","quattro");
        for($i=0; $i < 5; $i++)
                $v5[$indici[$i]]=$indici[rand(0,4)];
        print_r($v5);

         ?>
</pre>
```

E' possibile trovare la dimensione attuale dell'array con **count()**

es:

```
$v=array(0,1,2,3);
$dimensione = count($v);

// ora $dimensione contiene 4
```

Per scorrere gli array e' anche molto comodo il ciclo foreach()

Esempio di Ciclo foreach():

Il ciclo for, anche se molto potente e versatile, può risultare scomodo nell'impiego di semplici compiti come l'attraversamento di un array. In questi casi risulta più semplice l'uso del ciclo foreach. Tale ciclo ha una sintassi del tipo:

```
foreach ($array_da_attraversare as $valore_elemento)
{
        // istruzioni da iterare
}
```

Le istruzioni raggruppate nelle parentesi graffe verranno ripetute tante volte quanti sono gli elementi presenti nell'array $array_da_attraversare. Ad ogni iterazione verrà letto un valore presente in tale array e copiato nella variabile $valore_elemento. Vediamo un esempio:

```
$colori = array('bianco','rosso','verde','blu','giallo','nero');

foreach ($colori as $colore)
{
   echo $colore , '<br>';
}
```

Oppure il foreach nella forma completa qui sotto: permette di scorrere un array e recuperarne gli elementi sotto forma di coppie chiave => valore. Ecco la struttura tipica:

```
foreach ($array_da_attraversare as $chiave_elemento => $valore_elemento)
{
   // istruzioni da iterare
}
```

Il funzionamento è lo stesso della struttura vista in precedenza. L'unica differenza è che adesso anche le chiavi di ciascun elemento vengono lette e copiate nella variabile $chiave_elemento. Vediamo un semplice esempio:

```
<?php
$film = array('titolo' => 'Via col Vento', 'anno' => 1939,'regista' => 'Victor Fleming');
```

```php
foreach ($film as $info => $valore)
{
    echo "$info: $valore <br>\n";
}
?>
```

Per riassumere il tutto si provi il seguente programma:

```
<pre>
<?php
        // Descrizione programma: Creazione array dinamici /
count() / uso di foreach
        // Data ult. modifica:    4 ott 2014
        // Scritto da:            F. Sacco

        $v=array();


        // commenta il primo e lancia solo il secondo ciclo
        echo "<hr>print_r:<br>";
        $limite = rand(1,10);
        for ($i=0; $i < $limite; $i++)
        // quanto e' lungo il vettore?
                $v[$i]=rand(1,100);
        print_r($v);

        echo "<hr>count()<br>";
        $elem = count($v);
        // numero elementi ricavato dal vettore stesso: con count()
        for ($i = 0;  $i < $elem ; $i++)
                echo "v[$i]=$v[$i]<br>\n";

        echo "<hr>foreach(semplice)<br>";
        foreach($v as $e )
        //numero elementi implicito nel ciclo foreach (senza indice)
                echo "v[]=$e<br>\n";

        echo "<hr>foreach(completo)<br>";
        foreach ($v as $i => $e) // foreach forma completa:
        // $i == indice, $e = elemento, cioe' v[$i]
                echo "v[$i]=$e<br>\n";


         ?>
</pre>
```

Ordinamento (sorting) dei vettori: sort() / asort() / ksort()

Per gestire gli array sono state create numerose funzioni. Le più usate nono quelle di ordinamento come **sort()** che ordina afabeticamente gli elementi dell'array

```
$lettere = array("a", "z", "c", "b");
sort($lettere);

/* lettere e'
lettere[0]=a
lettere[1]=b
lettere[2]=c
lettere[3]=z
*/
```

oppure **asort()** usata per ordinare gli array associativi mantenendo l'originale associazione dell'indice

```
$cognomi = array("a" => "Rossi",
                 "b" => "Bianchi",
                 "c" => "Verdi");
asort($cognomi);

/* $cognomi
cognomi[b]=Bianchi
cognomi[a]=Rossi
cognomi[c]=Verdi
*/
```

e infine ksort() ordina gli indici.

Si provi, per approfondimenti, il programma seguente:

```
<pre>
<?php
        // Descrizione programma: Sort array(): sort() / asort() / ksort()
        // Data ult. modifica:    4 ott 2014
        // Scritto da:            F. Sacco

        $v=array();
        $v2=array();
    // $v3=array(); NOTA CHE FUNZIONA ANCHE SE v3[] e' in commento!!!!!


        echo "<hr>random() e sort() - chiude i buchi e ordina <br>";

        for ($i=0; $i < 20; $i++)
                $v[rand(1,20)]=rand(1,1000);
        print_r($v);
        sort($v);
```

```
        print_r($v);

        echo "<hr>random() e asort() - mantiene indici originali
<br>";
        for ($i=0; $i < 20; $i++)
                $v2[rand(1,20)]=rand(1,1000);
        print_r($v2);
        asort($v2);
        print_r($v2);



        echo "<hr>ksort - ordina su chiavi - mantiene indice
originale<br>";
        for ($i=0; $i < 20; $i++)
                $v3[rand(1,100)]=$i;
        print_r($v3);
        ksort($v3);
        print_r($v3);

    ?>
</pre>
```

NB: le variabili dei moduli vengono passate nel **vettore** $_REQUEST[]

si veda il seguente programma php che fa il debug dei moduli HTML usando il foreach e la funzione print_r()

```
<?php
        // Descrizione programma: Scorro array $_REQUEST[]
        // Data ult. modifica:    4 ott 2014
        // Scritto da:            F. Sacco


        $v=$_REQUEST; // vettore
        echo "<hr>Debug delle variabili del modulo con foreach()<br><br>";
        foreach ($v as $i => $e) // foreach forma completa:
                                 // $i == indice, $e = elemento, cioe' v[$i]
        echo "variabile [$i]=[$e]<br>\n";

        echo "<hr>Debug delle variabili del modulo con print_r()<br>";
        echo "<pre>";
        print_r($v);
        echo "</pre>";
  ?>
```

OPERATORI ED ESPRESSIONI

In un linguaggio completo come php non potevano mancare i classici operatori aritmetici

$a + $b	Addizione
$a - $b	Sottrazione
$a * $b	Prodotto
$a / $b	Quoziente
$a % $b	Modulo (resto del quoziente)

seguiti a ruota dagli operatori logici più usati

! $a	Not	Ritorna vero se $a non è vero
$a && $b	And	Ritorna vero se $a e $b sono veri
$a || $b	Or	Ritorna vero se $a o $b è vero

Ci sono anche gli operatori di confronto

$a == $b	Uguale (**esiste anche triplo ===**)	Ritorna vero se $a è uguale a $b ("5" == 5 ritorna vero) Triplo uguale → vero solo se stesso tipo
$a != $b	Non uguale	Ritorna vero se $a non è uguale a $b
$a < $b	Minore di	Ritorna vero se $a è minore di $b
$a > $b	Maggiore di	Ritorna vero se $a è maggiore di $b
$a <= $b	Minore o uguale a	Ritorna vero se $a è minore o uguale a $b
$a >= $b	Maggiore o uguale a	Ritorna vero se $a è maggiore o uguale a $b

L'unico operatore disponibile per le stringhe è quello di concatenazione scritto usando il carattere punto.

```
$a = "concateno ";
$b = $a . "due stringhe"; // $b = "concateno due stringhe"
```

Un uso particolare che si può fare dell'operatore di assegnazione = consiste nel combinarlo con un operatore aritmetico

```
$a = 3;
$a += 2; // $a vale 5
$b = $a -=5; // $a e $b valgono 0
$c = 3; $c *= 3; // $c vale 9
```

Esistono infine anche gli operatori unari di pre/post incremento/decremento, rispettivamente ++ e --

```
$a = 3;
$b = pow($a++, 2);  // $b vale 9 (3 alla seconda)
$c = pow(++$a, 2); // $c vale 25 (5 alla seconda)// $c vale 9
```

Nell'esempio precedente è stata usata la funzione pow(), uno dei tanti comandi predefiniti di php per eseguire calcoli matematici. pow(), dunque, è l'operazione di elevamento a potenza e richiede come parametri la base e l'esponente.

Introduzione alle stringhe in PHP

Come in C/C+: sono racchiuse fra il doppio apice, es. “Ciao Mondo”
(trovi tutte le funzioni del C/C++ che forse gia' conosci, come strlen(), strcat(), sprintf(), etc.)

Inoltre ci sono tantissime altre funzioni utili, alcune vengono mostrate con l'esercizio (provalo) qui sotto riportato:

```
$a = 'IERI ERA DOMENICA';

/* $b diventa 'ieri era domenica',
 * ma $a rimane 'IERI ERA DOMENICA'
 */
$b = strtolower($a);

strlen('abcd');                      // restituisce 4
trim(' Buongiorno a tutti ');        // restituisce 'Buongiorno a tutti'
substr('Buongiorno a tutti', 4);     // 'giorno a tutti' (inizia dal quinto)
substr('Buongiorno a tutti', 4, 6);
// 'giorno'(6 caratteri a partire dal quinto)
substr('Buongiorno a tutti', -4);    // 'utti' (ultimi quattro)
substr('Buongiorno a tutti', -4, 2);
// 'ut' (2 caratteri a partire dal quartultimo)
substr('Buongiorno a tutti', 4, -2);
// 'giorno a tut' (dal quinto al terzultimo)

str_replace('Buongiorno', 'Ciao', 'Buongiorno a tutti'); // 'Ciao a tutti'
str_replace('dom', 'x', 'Domani è domenica');            // 'Domani è xenica'
str_ireplace('dom', 'x', 'Domani è domenica');           // 'xani è xenica'

strpos('Domani è domenica', 'm'); // 2 (prima 'm' trovata)
strstr('Domani è domenica', 'm');
// 'mani è domenica' (a partire dalla prima 'm')

strtoupper('Buongiorno a tutti'); // 'BUONGIORNO A TUTTI'
ucfirst('buongiorno a tutti');    // 'Buongiorno a tutti';
ucwords('buongiorno a tutti');    // 'Buongiorno A Tutti';

$v=explode(',','Alberto,Mario,Giovanni');
/* suddivide la stringa in un array, separando un elemento
 * ogni volta che trova una virgola; avremo quindi un array $V
 * di tre elementi: ('Alberto','Mario','Giovanni')
NB: Per separare ad invio usa la stringa col doppio apice: explode(“\n”,....);
 */

$v=explode(',','Alberto,Mario,Giovanni',2);
/* in questo caso l'array $v può contenere al massimo due elementi,
 * per cui nel primo elemento andrà 'Alberto' e nel secondo il
 * resto della stringa: 'Mario,Giovanni'
 */
```

```php
// Esiste anche implode()
$array = array('lastname', 'email', 'phone');

$str = implode(",", $array);

echo $str; // str-->"lastname,email,phone"
```

Funzioni in PHP

La definizione della funzione avviene attraverso la parola chiave **`function`**, seguita dal nome che abbiamo individuato per la funzione, e dalle parentesi che contengono i parametri (o argomenti) che devono essere passati alla funzione.

```php
<?php
function max($a,$b) // passaggio per valore
{
    if ($a > $b)

      $b=$a;

    return ($b); // il massimo
}
?>
```

Passaggio per riferimento:

```php
<?php

function aggiungi_qualcosa(& $stringa)
{
   $stringa.= 'e qualche altra cosa.';
}

$str= 'Questa è una stringa, ';
aggiungi_qualcosa($str);
echo $str;
// l'output sarà 'Questa è una stringa, e qualche altra cosa.'
?>
```

Visibilità, variabili **globali** da funzione:

```php
$variabile="pippo"; // esterna alla funzione

function prova()
{
   global $variabile;
// … senza global la variabile $variabile NON si vede da prova()
   echo $variabile;
}
```

Un buon programma d'esempio che mostra l'utilizzo delle funzioni, dei vettori, delle variabili globali, locali e del passaggio a funzione di variabili per valore o riferimento, è il seguente.

Il programma simula l'estrazione di carte da un mazzo di quaranta carte, senza reimmissione (la carta estratta non puo' essere estratta piu' di una volta!!!)

(NB: nota anche le due nuove funzioni sui vettori **in_array()** e **array_key_exist()**

```php
<?php
        // Descrizione programma: array_key_exists() e in_array() - funzioni e passaggio parametri
        // Data ult. modifica:    Ottobre 2014
        // Scritto da:            F. Sacco

$carte= array("Re","Asso","Due","Tre","Quattro","Cinque","Sei","Sette","Fante","Donna");
$semi       =   array("Cuori","Quadri","Fiori","Picche");
$mazzo      =   array();    // Mazzo di carte

$fuorimazzo =   array();   // Mazzo di carte uscite: Uso di in_array()
$uscite     =   array();
// booleani x carta uscita: USO di array_key_exist()!!!

function riempimazzo(& $mazzo) // passaggio per riferimento
{
global $carte, $semi;          // variabil globali

for($seme=0; $seme < count($semi); $seme++)
        for ($carta=0; $carta < count($carte); $carta++)
                $mazzo[]=$carte[$carta]."-di-".$semi[$seme];
                // riempio mazzo di carte da indice 0 a 39
}

// Continua
```

```php
function vedimazzo( $mazzo)  // passaggio per valore
{
   global $carte;

   echo "<table border=1>";
   for ($i=0; $i < count($carte); $i++)
        echo "<th>$carte[$i]</th>";          // titoli prima colonna
   echo "<tr>\n";
   $i=0;
   foreach($mazzo as $indice => $carta)  // Stampa mazzo di carte
   {
                echo "<td>";
                echo "$carta, posizione $indice<br>\n";
                echo "</td>";
                $i++;
                if ($i == count($carte)) // 10 carte per linea
                {
                        $i=0; echo "<tr>";
                }
   }
}

function carteuscite( $uscite)  // visualizza il numero delle carte
uscite, 0 = re di cuori,...
                                // in ordine di carta, NON di uscita
{
    global $mazzo, $fuorimazzo;

        for ($i=0; $i < count($mazzo); $i++)
              if (in_array($mazzo[$i],$fuorimazzo))
                    echo "<br>Carta uscita - posizione nel mazzo N. $i";
}
// Qui terminano le funzioni e, se fossimo in C/C++ o Java
// inizierebbe il main() !!!

riempimazzo($mazzo); // richiamo di funzione riempimazzo()

if (!isset($_REQUEST["cartedaestrarre"]))   // primo lancio di carte.php
{
 echo "<form action=carte.php>";  // FORM dinamico!!!
 echo "<input type=text value=1 name=cartedaestrarre>";
 echo "Numero carte da estrarre dal mazzo (max 40)";
 echo "<br><input type=submit value=Estrai></form>";
 echo "--- Segue mazzo di carte - Nota che decine=seme:";
 echo "0=cuori,1=quadri,... unita=carta, 0=re, 1=asso...";
 echo "[es: 27= 7 di fiori]---<br>\n";
 vedimazzo($mazzo);  // richiamo di funzione vedimazzo()
}
else // secondo lancio lancio di carte.php
{
        $cartedaestrarre=$_REQUEST["cartedaestrarre"];
        $num=1; // contatore di carte estratte
        do {  // Continua
                $c = rand(0,39);
                if (!array_key_exists($c,$uscite))
```

```
                // se nel vettore $uscite[] non esiste la chiave $c
                {
                        $uscite[$c]=true; // carta numero $c uscita
                        echo "Carta N. $num : $mazzo[$c]<br>\n";
                        $fuorimazzo[] = $mazzo[$c];
                        // carta uscita, la metto nel vettore fuorimazzo[]
                        $cartedaestrarre--;
                        $num++;
                }
        } while($cartedaestrarre != 0); // Fine DO-WHILE
          // finche' non ci sono piu' carte da estrarre
        carteuscite($uscite);  // richiamo funzione carteuscite
}
?>
// Esercizi: dato un mazzo di immagini di carte, in modo che
// il re di cuori abbia nome "0.jpg", il re di quadri "1.jpg",
// etc. ... fino alla donna di picche di nome "39.jpg"
// 1) Visualizzare le carte in immagini.
// 2) Implementare il gioco del 7 e mezzo (usa variabili hidden...)
```