

# Il linguaggio PHP

(Hypertext Preprocessor)

Ing. E. Tramontana

# Funzioni

- Come in altri linguaggi, le funzioni in PHP permettono di definire delle routine, utili per elaborare dei dati (per esempio passati come argomenti) e per fornire un risultato
- Nella definizione di una funzione:
  - gli argomenti non sono caratterizzati dal loro tipo
  - può essere restituito un valore di ritorno, il cui tipo non è specificato
- Non è supportata la possibilità di dare lo stesso nome a funzioni che hanno un numero diverso di argomenti e/o di tipi (cioè l'overloading)
- Non è possibile ridefinire funzioni già esistenti

# Definizione di funzioni

- La definizione di una funzione avviene secondo la sintassi:

```
<?php
    function nomefunz (arg1, arg2, ... ) {
        blocco istruzioni
        return valore;
    }
... ?>
```

Esempio:

```
function NomeCompleto ($Nome, $Cognome ) {
    return ( $Nome . " " . $Cognome );
}
```

# Argomenti di funzioni

- Per default, gli argomenti sono passati per valore, quindi i loro cambiamenti all'interno della funzione non sono visibili all'esterno

```
function inLire($imp) { // $imp è passato per valore
    // $imp viene convertito da Euro a Lire
    $imp = $imp * 1936;
    return $imp; // qui $imp è in Lire
}
$contoEuro = 135; // Euro
$contoLire = inLire($contoEuro); // invocazione
echo "Euro: " . $contoEuro . "    Lire: " . $contoLire;
```

## Output

```
Euro: 135    Lire: 261360
```

# Passaggio per riferimento

- Se un argomento di una funzione deve essere passato per riferimento, così da vedere i cambiamenti all'esterno della funzione, si deve far precedere un **&** al nome dell'argomento

```
// $tot è passato per riferimento
```

```
function incremConLire(&$tot, $val) {
```

```
    // $tot è in Euro, $val è in Lire
```

```
    $tot = $tot + $val / 1936;
```

```
}
```

```
$contoEuro = 135;
```

```
echo "<p> Valore iniziale Euro: " . $contoEuro;
```

```
incremConLire($contoEuro, 23500);
```

```
echo "<p> Valore finale Euro: " . $contoEuro;
```

# Passaggio per riferimento

- Si può scegliere se le modifiche agli argomenti passati devono essere visibili all'esterno della funzione anche al momento della chiamata.
- Se un argomento deve essere passato per riferimento si fa precedere la variabile da un **&**

```
// $tot è passato per valore
function incremConLire( $tot, $val ) {
    // $tot è in Euro, $val è in Lire
    $tot = $tot + $val / 1936;
}
$contoEuro = 135;
incremConLire ( &$contoEuro, 23500 );
echo "<p> Valore finale Euro: " . $contoEuro;
```

# Array

- Un array è una collezione di elementi. Contrariamente a quanto accade in altri linguaggi, in PHP gli elementi possono essere di tipi diversi

## Inizializzazione di un array

```
$Lista_Spesa = array("Pane", 0.5, "Latte", TRUE);
```

## Aggiunta di elementi ad un array

```
$Lista_Spesa[] = "Uova";  
$Lista_Spesa[] = 5;
```

# Array associativi

- È possibile definire array associativi, cioè array i cui elementi sono associati a chiavi di ricerca
- L'accesso ad un elemento dell'array avviene per mezzo di una chiave, anziché attraverso un indice numerico

```
$Lista_Spesa = array();  
$Lista_Spesa["Pane"] = 0.5;  
$Lista_Spesa["Latte"] = TRUE;  
echo "Pane " . $Lista_Spesa["Pane"] . " Kg";
```

Output

```
Pane 0.5 Kg
```

# Integrazione tra PHP e HTML

- Un programma PHP scrive il suo output su pagine HTML. Per generare un output formattato bisogna che il programma PHP scriva anche i tag HTML necessari. Se si vuole che il codice HTML sia leggibile bisogna inserire anche gli opportuni spazi e ritorni carrello
- Una funzione PHP che scrive titolo pagina, nome e cognome, e data al centro della pagina HTML

```
function formatta( $titolo, $nome, $cognome ) {  
    echo "<center>\n"; // scrive tag per centrare testo e va a capo  
    echo " <h1>" . $titolo . "</h1> \n";  
    echo " <h2>" . $nome . " " . $cognome . "</h2> \n";  
    $d = time(); // restituisce il tempo attuale  
    // scrive data formattata  
    echo " <h3><i>" . date('F d', $d) . " </h3></i>\n";  
    echo "</center>\n";  
}
```

# Esempio di funzione

- Funzione che inserisce un array in una tabella HTML

```
function ScriviArray( $d ) {  
    // $d è l'array da scrivere  
    echo "<table border='1'>\n"; // tag per inizio tabella  
    for ($i = 0; $i < count( $d ); $i++) {  
        echo " <tr>\n"; // tag per inizio riga  
        echo " <td> Elemento " . ($i+1) . "</td>  
        echo " <td> Valore " . $d[$i] . " </td>\n";  
        echo " </tr>\n";  
    }  
    echo "</table>\n";  
}
```

# Funzione su array

- Funzione che incrementa gli elementi di un array inferiori ad un valore di soglia

```
include 'scriviarray.php';  
// inclusione di un programma php  
function IncremArray( &$dati, $soglia ) {  
    for ( $i = 0; $i < count($dati); $i++ )  
        if ( $dati[ $i ] < $soglia )    $dati[ $i ]++;  
}  
// array con dati  
$mieidati = array(11, 15, 5, 7, 3, 25, 18);  
IncremArray($mieidati, 10);  
ScriviArray($mieidati);
```

## Output

```
11, 15, 6, 8, 4, 25, 18
```

# Argomenti con valori di default

- E' possibile predefinire il valore di uno o più argomenti passati ad una funzione secondo la sintassi:

```
function nome ( arg1 = "valore" ) {  
    blocco istruzioni  
    return valore;  
}
```

- Nel caso di più argomenti, quelli con valore di default devono trovarsi a destra degli altri

# Esempio di argomento di default

Con un argomento:

```
function scriviPrefer($str = "caffè") {  
    echo "<p> La mia bevanda preferita è il " . $str;  
}  
scriviPrefer();  
scriviPrefer("tè");
```

Output

```
La mia bevanda preferita è il caffè  
La mia bevanda preferita è il tè
```

Con due argomenti:

```
function scriviPrefer($tipo, $str = "caffè") {  
    echo "<p> La mia bevanda preferita è il " . $str . " " . $tipo;  
}  
scriviPrefer("macchiato");
```

# Visibilità delle variabili

- Supponiamo di avere il programma PHP:

```
$nome = "Tizio";  
function daiNome() {  
    return $nome;  
}  
echo "Nome: " . daiNome();  
echo "Nome: $nome <br>";
```

- Una volta eseguito darà come output:

```
Nome:  
Nome: Tizio
```

- La funzione `daiNome()` non restituisce la variabile globale `$nome` ma ri-definisce la variabile al suo interno

# Visibilità delle variabili

- Per accedere alle variabili globali si può usare la parola chiave `global`

```
$nome = "Tizio";  
function daiNome() {  
    global $nome;  
    return $nome;  
}
```

- Si può anche usare l'array associativo `$GLOBALS`

```
$nome = "Tizio";  
function daiNome() {  
    return $GLOBALS["nome"];  
}
```

# Funzioni standard di PHP

- Per ottenere informazioni sul tipo di variabile:
  - `gettype(<variabile>)`: restituisce una stringa contenente la descrizione del tipo della variabile
  - `is_bool(<variabile>)`: restituisce TRUE se la variabile è boolean
  - `is_long(<variabile>)`: restituisce TRUE se la variabile è intera (le varianti utilizzabili sono `is_int`, `is_integer`)
  - `is_double(<variabile>)`: restituisce TRUE se la variabile è in virgola mobile (le varianti utilizzabili sono `is_double`, `is_real`, `is_float`)
  - `is_string(<variabile>)`: restituisce TRUE se la variabile è una stringa
  - `is_array(<variabile>)`: restituisce TRUE se la variabile è un array

# Funzioni standard di PHP

- Per manipolare stringhe:
  - `strpos(<string>, <str>)`: restituisce la posizione della prima occorrenza di `<str>` in `<string>`; se non trovato ritorna FALSE
    - `strpos("abcdef", "cd")` restituisce 3
  - `strlen(<string>)`: restituisce la lunghezza di `<string>`
    - `strlen("abcdef")` restituisce 6
  - `strrev(<str>)`: restituisce la stringa `<str>` rovesciata
  - `substr(<string>, <inizio> [, <lung> ])`: restituisce la parte di `<string>` specificata da `<inizio>` e `<lung>`
    - `substr("abcdef", 2, 1)` restituisce "c"
  - `strtolower(<str>)`: restituisce la stringa `<str>` con tutti i caratteri minuscoli
  - `strtoupper(<str>)`: restituisce la stringa `<str>` con tutti i caratteri maiuscoli

# Funzioni standard di PHP

- Per manipolare array:
  - `array_merge(<array1>, <array2>)`: restituisce un array composto dagli elementi di `<array1>` e di `<array2>`, in cui gli elementi del secondo sono accodati a quelli del primo
  - `array_pop(<array1>)`: ritorna l'ultimo elemento di `<array1>` e accorcia `<array1>` eliminando tale elemento
  - `array_push(<array1>, <elem>)`: appende `<elem>` ad `<array1>`
  - `array_search(<elem>, <array1>)`: cerca `<elem>` in `<array1>` e se lo trova ritorna la chiave (o indice) corrispondente
  - `count(<array1>)`: ritorna il numero di elementi presenti in `<array1>`

# Form HTML

- Le form HTML permettono all'utente di inviare al server alcune variabili con il loro valore
  - la richiesta di pagine dinamiche sarà più flessibile
  - le risposte del server sono molteplici e più interessanti
- Per esempio, il server potrà ricevere le variabili `$nome`, `$cognome`, `$scuola` ed il loro valore
- Gli elementi di cui si compone una form HTML sono:
  - La pagina (PHP) a cui sottoporre la richiesta (action)
  - Il metodo usato per inviare la richiesta
  - I controlli, ovvero la modalità con cui i valori delle variabili sono inseribili dall'utente

# Elementi di una form: action

- **Action** indica quale pagina sarà usata per ricevere o elaborare i dati quando la richiesta sarà inoltrata al server

Esempio di form HTML:

```
<form action="inserisci.php">
```

```
...
```

```
</form>
```

- La pagina che risponderà alla form è **inserisci.php**

# Elementi di una form: method

- **Method** indica la modalità (**GET** o **POST**) del protocollo HTTP usata per inviare i dati al server

Esempio:

```
<form method="post">
```

...

```
</form>
```

- GET invia i dati come parametri di un comando (una ricerca con Yahoo! è un tipico esempio di GET)
- POST invia i dati come stringhe successive
- Nel primo caso, il server troverà i dati nella linea di comando che lo richiama, mentre nel secondo dovrà ricevere le stringhe successivamente
  - In entrambi i casi il recupero dei dati non viene gestito dal programma PHP

# Elementi di una form: controlli

- I controlli definiscono la modalità con cui i valori delle variabili sono visibili e inseribili dall'utente
- Vari tipi di controlli sono disponibili per permettere l'inserimento di testo e numeri, selezionare da liste scelte e radio bottoni, etc.
- Sono controlli anche i bottoni che confermano la form inviando i dati inseriti al server

Esempio di variabile stringa `$cognome`:

```
<input type="text" name="cognome">
```

Esempio di bottone che invia la form:

```
<input type="submit" value="Invia">
```

# Esempio di form

Form che richiede alcuni dati anagrafici (nome, cognome) e li invia alla pagina mostra.php

Inserire i seguenti dati anagrafici:

```
<form action="mostra.php" method="get">
  <table>
    <tr><td>Nome:</td>
      <td><input type="text" name="nome"></td></tr>
    <tr><td>Cognome:</td>
      <td><input type="text" name="cog"></td></tr>
  </table>
  <input type="submit" value="Invia">
</form>
```

# Ricezione dati da form

Il server riceve i dati dalla form e li fornisce al programma PHP specificato nell'action della form

Per la form con i dati anagrafici, il programma PHP (mostra.php) che visualizza i dati in una tabella è:

```
Dati anagrafici<br>
```

```
L'utente ha specificato i seguenti valori:
```

```
<table border="1">
```

```
<tr><td>Nome:</td><td><?php echo $nome; ?></td></tr>
```

```
<tr><td>Cognome:</td><td><?php echo $cog; ?></td></tr>
```

```
</table>
```

# Controllo sui dati

- Per valutare se l'utente ha inserito i dati richiesti nella form posso usare la funzione `isset()`

```
if (isset($nome)) { // true se l'utente ha inserito il nome
    blocco istruzioni
}
else { // false se l'utente non ha inserito il nome
    echo "nessun dato...";
}
```

# Form per inserire numeri

- Form che permette all'utente di inserire i valori di minimo e massimo dei numeri interi di cui si vogliono calcolare i quadrati, e
- Programma PHP che restituisce i quadrati dei numeri interi nell'intervallo inserito tramite la form

Inserire l'intervallo dei numeri su cui calcolare i quadrati:

```
<form action="calcolaquadrati.php" method="get">
  <table>
    <tr><td>Minimo:</td>
      <td><input type="text" name="min"></td></tr>
    <tr><td>Massimo:</td>
      <td><input type="text" name="max"></td></tr>
  </table>
  <input type="submit" value="Invia">
</form>
```

# Programma PHP per calcolo

- Programma che calcola i quadrati tra minimo e massimo

Calcolo dei quadrati di alcuni numeri<br>

```
<h1>Quadrati</h1>
```

```
<table>
```

```
<?php
```

```
    for ( $i = $min; $i <= $max; $i++ )
```

```
        echo "<tr><td>" . $i . "</td><td>" . $i*$i .  
                                                    "</td></tr>\n";
```

```
?>
```

```
</table>
```

# I bottoni delle form

- Se in una form compaiono più controlli di tipo `submit`, l'utente potrà (ovviamente) premerne solo uno per inviare la form ed i suoi dati
- La variabile PHP corrispondente al bottone premuto sarà uguale al campo `value` specificato nel bottone della form
- Le variabili PHP corrispondenti ai bottoni non premuti saranno uguali alla stringa vuota

# Form con tre bottoni

- Una form che permetta di confermare i dati inseriti, annullarli e ripristinare i valori di default

```
<form action="bottoni.php" method="get">
  <p>Lavoro:
    <input type="text" name="job" value="studente">
  <p>
    <input type="submit" name="ok" value="Conferma">
    <input type="submit" name="no" value="Annulla">
    <input type="reset" value="Ripristina">
</form>
```

# Form con tre bottoni

- Se l'utente invia la form premendo il bottone:
  - “Conferma”, allora le variabili saranno `$ok="Conferma"` e `$no = ""`
  - “Annulla”, allora le variabili saranno `$ok = ""` e `$no="Annulla"`
- Con un `if` è possibile capire se l'utente vuole procedere con l'invio dei dati della form, oppure se vuole annullare l'operazione

# Form con radio bottoni

- Costruiamo una form HTML con radio bottoni, che richiama un programma PHP che calcola una somma

```
<form action="calcolo.php" method="post">
  Quali sono le vostre conoscenze di HTML?<br>
  <input type="radio" name="conosci" value="1" checked>poche
  <input type="radio" name="conosci" value="2">medie
  <input type="radio" name="conosci" value="3">buone<br><br>
  Indicare la vostra esperienza di programmazione:<br>
  <input type="radio" name="esperto" value="1" checked>base
  <input type="radio" name="esperto" value="2">media
  <input type="radio" name="esperto" value="3">buona
  <p><input type="submit" value="Mostra risultati!">
</form>
```

# Prelevare dati dai form

- I dati provenienti da form HTML sono memorizzate su variabili speciali di PHP su array chiamati 'super global'
- Secondo il metodo di trasmissione del form (POST o GET), si usa le variabili `$HTTP_POST_VARS` o `$HTTP_GET_VARS` per accedere ai dati
- I valori sono indicati su array indicizzati dai nomi
- Nel precedente esempio, per recuperare i valori inviati con POST si possono usare:

```
$val1 = $HTTP_POST_VARS[conosci];  
$val2 = $HTTP_POST_VARS[esperto];
```

# Il codice PHP richiamato dal form

- Nell'esempio, il trattamento dei dati consiste nel sommare il punteggio delle risposte e fornire il risultato

```
<?php
    // Recupero variabili dal form
    $val1 = $_HTTP_POST_VARS[conosci];
    $val2 = $_HTTP_POST_VARS[esperto];
    $score = $val1 + $val2;    // Calcolo del punteggio
    // Mostra risultato
    echo "<h3>Il vostro punteggio è " . $score . "</h3>";
    if ( $score < 3 ) {
        echo "<p>Siete un debuttante</p>";
    } elseif ( $score < 5 ) {
        echo "<p>Voi avete una formazione media</p>";
    } else {
        echo "<p>Siete un esperto !</p>";
    }
?>
```

# Classe Persona

- Definiamo la classe Persona
  - Per definire la classe si usa la parola chiave `class`
  - Per i campi della classe si usa la parola chiave `var`

```
class Persona { // il nuovo tipo
    var $nome, $cognome; // variabili
    function set( $n, $c ) { // operazione
        $this->nome = "Nome: " . $n;
        $this->cognome = "Cognome: " . $c;
    }
    function toString() { // operazione
        $str = "<p>Si tratta di ...<br> $this->nome
                $this->cognome";
        return $str;
    }
}
```

# Classi ed oggetti

- Gli oggetti sono delle istanze di una certa classe
- Il costrutto `new`
  - permette di creare un nuovo oggetto
  - restituisce un riferimento all'oggetto creato
- Nel codice di una classe, l'operatore (variabile) `$this` rappresenta il riferimento all'oggetto corrente
- L'accesso ad operazioni e campi degli oggetti avviene tramite l'operatore `->`

```
$p = new Persona();
```

- Esecuzione dell'operazione `set` per l'oggetto `p` della classe `Persona`

```
$p->set("John", "Smith");
```

- Lettura del campo `nome` dell'oggetto `p`

```
echo $p->nome;
```

# Programma che usa le classi

- Costruire una classe Punto che:
  - rappresenta le coordinate x e y di un punto nel piano
  - fornisce le operazioni di settaggio e lettura delle coordinate
  - fornisce l'operazione che calcola la distanza del punto dall'origine [  $d = \sqrt{x^2 + y^2}$  ]
- In un programma si creano 3 istanze della classe Punto, si inseriscono dei valori nei campi e si calcola la distanza di ciascun punto dall'origine

# Classe Punto

```
class Punto {
    var $x, $y;
    function settaCoordinate( $xx, $yy ) {
        $this->x = xx;
        $this->y = yy;
    }
    function leggiPunto() {
        echo "<p>Le coordinate del punto sono: ($x, $y)";
    }
    function daiDistanza() {
        $d = sqrt($this->x*$this->x + $this->y*$this->y);
        echo "<p>La distanza dal centro &egrave; $d";
    }
}
$p1 = new Punto();
$p1->settaCoordinate(2, 3);
$p2 = new Punto();
$p2->settaCoordinate(5, 7);
$p1->leggiPunto();
$p1->daiDistanza();
$p2->leggiPunto();
$p2->daiDistanza();
```

# Ereditarietà

- Esiste anche l'ereditarietà (parola-chiave “extends”)

```
class Studente extends Persona {  
    var $scuola, $anno;  
    function settaScuola($nomeS) {  
        $this->scuola = $nomeS;  
    }  
}  
  
$s = new Studente();  
$s->set("Mario", "Rossi");
```