

# PHP (PHP Hypertext Preprocessor)

Classe 5ASI  
ITCG Fermi

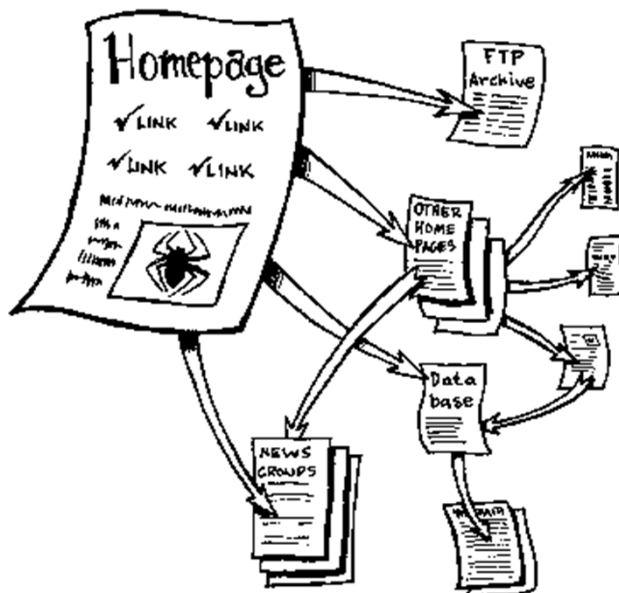
Prof. Montemurro

## Linguaggi per Realizzare Pagine Web

1. Il linguaggio **HTML** (**HyperText Markup Language**, o **linguaggio di contrassegno per gli ipertesti**) consente la creazione di pagine web attraverso **file di testo** che sono interpretati dal browser che le visualizza in **forma grafica**. È un linguaggio per la gestione di ipertesti; non è un linguaggio di programmazione in quanto non viene utilizzato per svolgere elaborazioni, ma è un **linguaggio di formattazione della pagina**.  
**Ipertesto**: testo multidimensionale (iper-), cioè a più dimensioni, infatti è costituito da un insieme di documenti che hanno tra loro un nesso logico in quanto si riferiscono ad una stessa classe di argomenti che possono essere consultati in modo non lineare mediante link a documenti esterni (**hyperlink**) per associazione di idee o di termini.  
**Navigazione ipertestuale**: modalità di lettura dei documenti che prevede il salto da uno all'altro tramite i link.

Prof. Montemurro

## Linguaggi per Realizzare Pagine Web



Prof. Montemurro

## Traduttori: Compilatori ed Interpreti

- 1. Compilatori:** programmi che prendono in ingresso l'intero programma (file sorgente/i), e che restituiscono in uscita la rappresentazione dell'intero programma in linguaggio macchina (o codice binario 01...).
- 2. Interpreti:** programmi che traducono (in linguaggio macchina) ed eseguono direttamente ciascuna istruzione del programma/script scritto nel file sorgente/i, istruzione per istruzione.

Prof. Montemurro

## Linguaggi per Realizzare Pagine Web

2. Il linguaggio **JavaScript JS** è:

- i. un linguaggio interpretato in quanto il codice non viene compilato, ma viene eseguita istruzione per istruzione direttamente dal browser;
- ii. un linguaggio di scripting (quindi è un linguaggio di programmazione) per cui è meglio parlare di script e non di programma.

**Script:** testo contenente una sequenza di istruzioni che può essere innestata *anche* nel codice di altri linguaggi, ed è eseguita da un programma interprete.

**Limite dell'HTML e di JavaScript:** non è possibile realizzare pagine web che consentano all'utente di accedere ai dati contenuti in un database che si trova su un server web.

Prof. Montemurro

## Linguaggi per Realizzare Pagine Web

**Linguaggi lato client** (HTML, JavaScript ...): interpretazione delle istruzioni viene fatta direttamente dal browser web nel computer dell'utente che svolge il ruolo di client (richiedente).

**Linguaggi lato server** (PHP ...): interpretazione delle istruzioni avviene sul server, ed il risultato dell'elaborazione viene inviato al browser; le pagine web (dinamiche) contengono quindi il risultato del codice eseguito sul server.

**Pagina web dinamica:** pagina web che viene costruita dal server quando l'utente la richiede. A differenza di una pagina statica, che è un file HTML predefinito e identico per tutti gli utenti, una pagina dinamica può variare in base a diversi fattori, quindi è interattiva e personalizzabile.

Prof. Montemurro

## Linguaggi per Realizzare Pagine Web

**Server web** (es. Apache): software per la gestione dei servizi web di un computer host o di un server di rete; tali servizi si basano sui protocolli standard delle reti e di internet, in particolare sul protocollo HTTP (HyperText Transfer Protocol) che è il principale protocollo informatico che consente il trasferimento di dati da e verso pagine web.

Il **PHP** (acronimo ricorsivo **PHP Hypertext Preprocessor**, o **preprocessore di ipertesti PHP**) è un linguaggio che estende le funzionalità del server web in quanto consente l'interpretazione di file con estensione *.php* contenenti il codice dell'applicazione lato server, oltre che naturalmente l'interpretazione dei classici marcatori (o tag) dei file con estensione *.html*.

Prof. Montemurro

## Linguaggi per Realizzare Pagine Web

Il PHP è:

1. un linguaggio interpretato in quanto il codice non viene compilato, ma viene eseguita istruzione per istruzione direttamente dal server; l'**interprete PHP** viene aggiunto al server web come modulo esterno;
2. un linguaggio di scripting (quindi è un linguaggio di programmazione) in quanto permette di innestare gli script all'interno delle pagine web dove sono presenti i marcatori del linguaggio HTML. In questo modo si possono realizzare in modo dinamico **pagine web lato server** che non dipendono solo dai marcatori statici del linguaggio HTML, ma anche dalle azioni degli script realizzati tramite il linguaggio PHP;
3. un linguaggio integrato nel linguaggio HTML;
4. un linguaggio **general** purpose, cioè multiuso, in quanto non ha un utilizzo specifico, ma **generale**, anche se oggi viene usato solo per creare siti web dinamici.

Prof. Montemurro

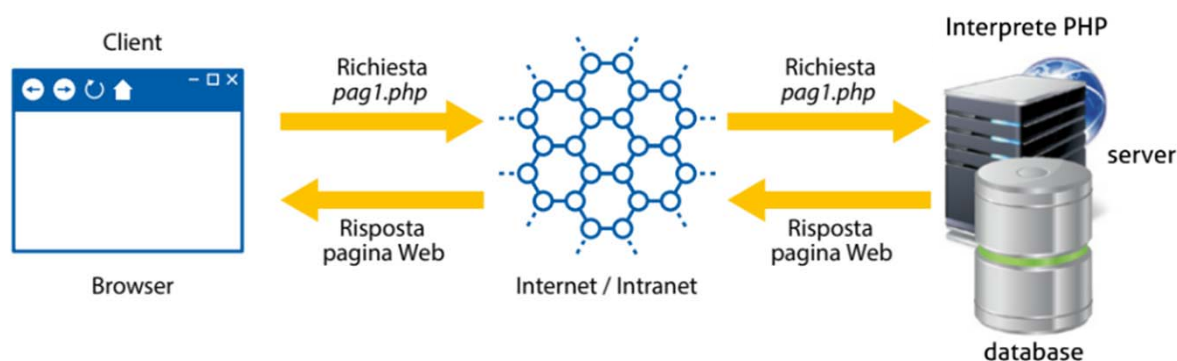
# Linguaggi per Realizzare Pagine Web

Fasi eseguite dal server web quando viene richiesta una pagina con estensione *.php*:

1. legge il file di testo *.php* riga per riga;
2. quando trova i marcatori HTML li spedisce al browser;
3. quando trova i blocchi di codice HTML:
  - i. ne esegue l'interpretazione;
  - ii. recupera gli eventuali dati richiesti prelevandoli dai file o dai database del server;
  - iii. restituisce una pagina web, creata quindi in modo dinamico, visualizzabile dal browser.

Prof. Montemurro

# Linguaggi per Realizzare Pagine Web



Prof. Montemurro

# Linguaggi per Realizzare Pagine Web

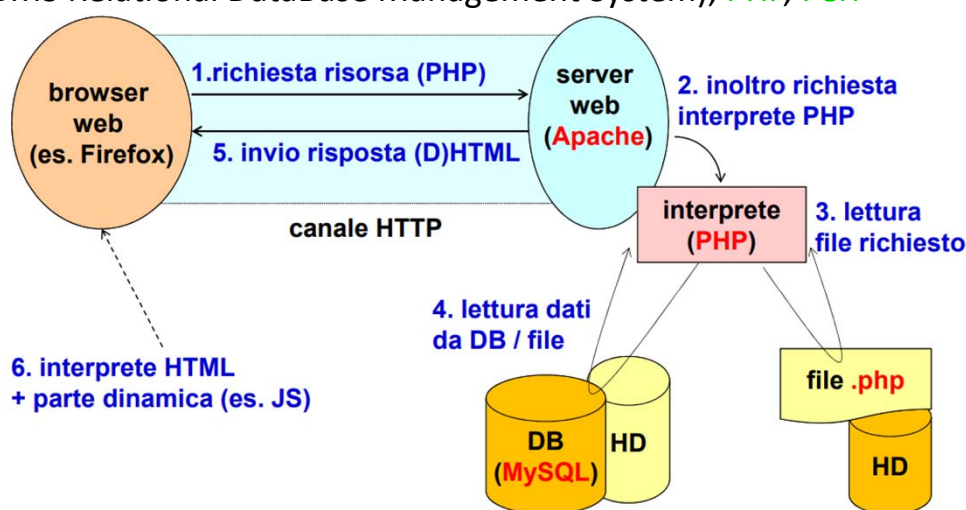
## Vantaggi del PHP:

1. rende più veloce la creazione e lo sviluppo di pagine web;
2. facilita le fasi di manutenzione e di aggiornamento delle applicazioni;
3. i suoi script sono compatibili con diverse applicazioni;
4. include la possibilità di accedere a vari tipi di database;
5. è un software libero, e viene continuamente controllato ed aggiornato; è scaricabile dal sito internet *www.php.net*.

Prof. Montemurro

## Ambiente di Sviluppo in Locale: XAMPP

**XAMPP:** **C**ross-platform, **A**pache (server web locale), **M**ariaDB (MySQL che è un RDBMS Relational DataBase Management System), **P**HP, **P**erl



Prof. Montemurro

# Ambiente di Sviluppo in Locale: XAMPP

XAMPP viene usato per sviluppare e testare le pagine web sul proprio computer locale, non è adatto all'ambiente di produzione.

Ambienti necessari per mettere in piedi un nuovo servizio o per eseguire cambiamenti significativi:

1. **ambiente di sviluppo** dove il servizio prende forma grazie agli sviluppatori
2. **ambiente di collaudo** (o **ambiente di test**) dove il servizio viene testato da chi ha i permessi
3. **ambiente di produzione** dove il servizio viene usato dai clienti



Prof. Montemurro

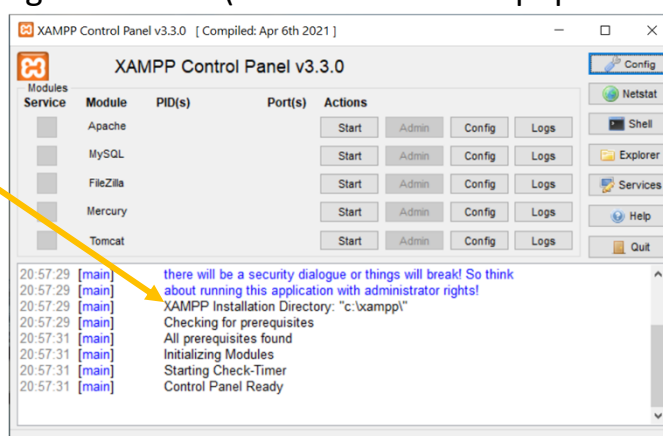
# Ambiente di Sviluppo in Locale: XAMPP

Download URL: <https://www.apachefriends.org/download.html>

Percorso assoluto in cui creare i progetti XAMPP (es cartella con file php da testare): **C:\xampp\htdocs**

Nel mio caso ho deciso di creare la cartella progetti per motivi di ordine:

**C:\xampp\htdocs\progetti**



Prof. Montemurro

## Ambiente di Sviluppo in Locale: XAMPP

Funzione **phpinfo()** restituisce in uscita le informazioni sull'interprete PHP installato nel computer (versione, caratteristiche del server web, tutte le configurazioni di PHP). Per usarla occorre creare un file di prova, chiamato ad esempio "*informazioni*" usando un editor di testi (es. blocco note) e copiare il seguente codice:

```
<?php
    phpinfo();
?>
```

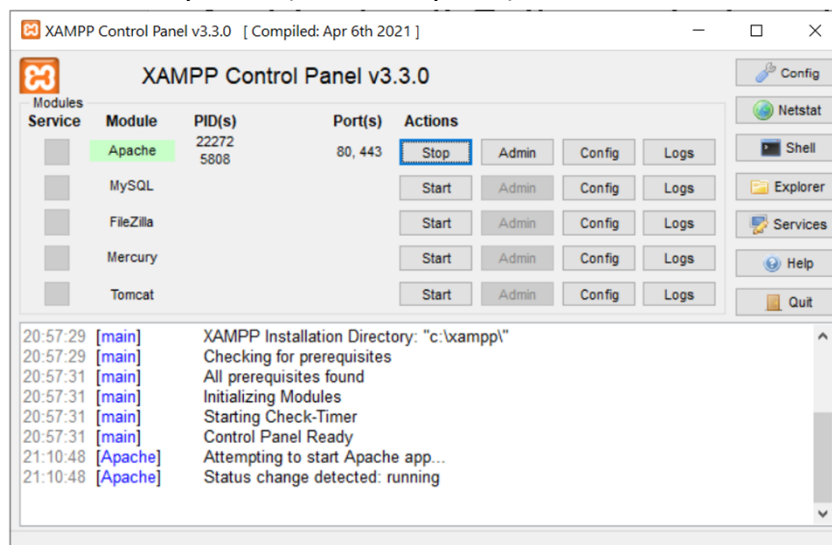
una volta salvato, occorre cambiare l'estensione del file da *informazioni.txt* a *informazioni.php*. Questo andrà copiato nella cartella

*C:\xampp\htdocs\progetti*

Prof. Montemurro

## Ambiente di Sviluppo in Locale: XAMPP

Avvio del server web Apache (80 listen port;)




Prof. Montemurro



# Ambiente di Sviluppo in Locale: XAMPP

Per richiamare il file *informazioni.php* dal browser, occorre avviare Apache in XAMPP ed andare all'indirizzo <http://localhost/progetti/>

PHP Version 8.2.12	
	
System	Windows NT PC-RAFFAELE 10.0 build 19045 (Windows 10) AMD64
Build Date	Oct 24 2023 21:10:40
Build System	Microsoft Windows Server 2019 Datacenter [10.0.17763]
Compiler	Visual C++ 2019
Architecture	x64
Configure Command	cscrip /nologo /e:jscrip configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-pdo-oci=\\.\.\.\instantclient\ sdk,shared" "--with-oci8-19=\\.\.\.\instantclient\ sdk,shared" "--enable-object-out-dir= ./obj/" "--enable-com-dotnet=shared" "--without-analyzer" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	no value
Loaded Configuration File	C:\xampp\php\php.ini

Prof. Montemurro

## Caratteristiche delle Pagine PHP

Le istruzioni del linguaggio PHP vengono scritte all'interno di pagine web, cioè in un file di testo salvato con l'estensione .php, e sono racchiuse tra il marcatore (o tag) di inizio **<?php** ed il marcatore (o tag) di fine **?>**. Questi marcatori si trovano nella sezione *body* della pagina HTML.

```
<?php
    // elenco di istruzioni in PHP
    ...;
    echo ...; // visualizza dati o messaggi nel browser
    ...;
?>
```

Prof. Montemurro

## Caratteristiche delle Pagine PHP

**Buona norma:** se un file contiene solo codice PHP, è consigliabile omettere il tag di chiusura `?>` alla fine del file stesso. In questo modo si evita che vengano aggiunti accidentalmente **spazi vuoti** o **nuove righe** dopo il tag di chiusura PHP, il che potrebbe causare effetti indesiderati perché PHP avvierà la bufferizzazione dell'uscita (\*slide seguente) anche quando il programmatore non ha alcuna intenzione di inviare alcuna uscita al buffer in quel punto dello script, perché il buffer di uscita è abilitato di default.

Prof. Montemurro

## Caratteristiche delle Pagine PHP

(\*) **Buffer PHP:** pezzo di memoria sul server per memorizzare temporaneamente contenuti che altrimenti verrebbero inviati al flusso di uscita (es. al client). Ciò fornisce un controllo sul quando l'uscita sarà presentata all'utente.

**Bufferizzazione dell'uscita** (o **controllo dell'uscita**, o **output buffering**, o **output control**): memorizzazione temporanea dell'uscita nel buffer interno prima che venga svuotato (inviato e scartato) nel browser (in un contesto web) o nella shell (sulla riga di comando). Mentre la bufferizzazione dell'uscita è attiva, nessuna uscita viene inviata dallo script sul flusso di uscita in quanto questa viene memorizzata in un buffer interno.

Prof. Montemurro

## Caratteristiche delle Pagine PHP

### Utilità della bufferizzazione dell'uscita

1. Le **reti** sono più **efficienti** quando inviano più dati in meno blocchi, piuttosto che meno dati in più blocchi.  
**Buona norma:** consegnare il contenuto al browser web del client in meno pezzi per ridurre il numero totale di richieste HTTP.  
 Ecco perché si abilita la buffering di output PHP; il buffer di uscita di PHP raccoglie fino a 4.096 byte prima di riversarne il contenuto al client.
2. Quando c'è un ritardo nel reperire qualche dato da un database; il buffer di uscita aiuta a **prevenire** (i) **caricamenti parziali di pagina** o (ii) che **dati incompleti vengano mostrati agli utenti**.

Prof. Montemurro

## Caratteristiche delle Pagine PHP

### Pagina con codice PHP

```
<HTML>
- -
<BODY>
  <?php
    //codice PHP
  ?>
</BODY>
</HTML>
```

### Interprete PHP

### Pagina ricevuta dal browser

```
<HTML>
- -
<BODY>
  soLo HTML
</BODY>
</HTML>
```

Prof. Montemurro

## Commenti

```
<?php
:
// questo è un commento su una singola riga
:
/* questo è un
commento su più righe */
:
?>
```

I commenti non vengono presi in considerazione dall'interprete PHP.

Prof. Montemurro

## Variabili

1. Le variabili in PHP sono identificate da un nome preceduto dal simbolo del dollaro \$.
2. I nomi delle variabili sono **case sensitive** (\$nome diverso da \$Nome).
3. I nomi delle variabili devono iniziare con una lettera.
4. I nomi delle variabili possono contenere lettere, cifre, carattere di sottolineatura \_.
5. La dichiarazione delle variabili è implicita; non è necessario dichiarare le variabili in quanto una variabile viene creata nel momento in cui le si assegna un valore.
6. Il tipo di una variabile viene attribuito dal PHP in modo autonomo (non occorre specificarlo) e dinamico (può variare all'interno dello script; es.: la stessa variabile viene considerata di tipo numerico se le si assegna un numero, oppure viene considerata una stringa se le si assegna una stringa); ecco perché PHP è un **linguaggio dinamicamente tipizzato**.

Prof. Montemurro

## Esempio Variabili

Il seguente codice assegna un valore a tre variabili; la dichiarazione delle tre variabili è implicita ed avviene quando si assegna loro un valore.

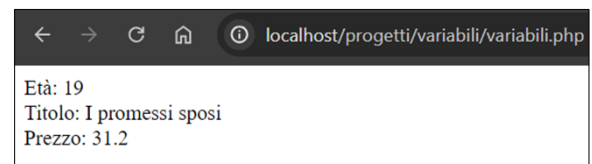
```
<?php
    $eta = 19;                // intero
    $titolo = "I promessi sposi"; // stringa
    $prezzo = 31.2;           // floating point
?>
```

Prof. Montemurro

## Esempio Variabili

Il seguente codice assegna un valore a tre variabili e le stampa a video; la dichiarazione delle tre variabili è implicita ed avviene quando si assegna loro un valore.

```
<?php
    $eta = 19;
    $titolo = "I promessi sposi";
    $prezzo = 31.2;
    echo "Età: $eta <br>";
    echo "Titolo: $titolo <br>";
    echo "Prezzo: $prezzo <br>";
?>
```



Il tag HTML `<br>` serve per andare a capo e viene interpretato dal browser.

Prof. Montemurro

## Tipi di Variabili

**Tipo di una variabile:** specifica la **dimensione** (es. 1 byte) e la **natura** (es. booleano, carattere, numero) dell'informazione che sarà memorizzata in una variabile.

**var\_dump()** : funzione predefinita PHP che stampa a video il tipo ed il valore della variabile che le viene passata come argomento.

In PHP ci sono:

1. **funzioni predefinite** (o **built-in function**), cioè già pronte all'uso;
2. **funzioni utente** (o **funzioni definite dall'utente**) (o **User-Defined Function UDF**) le quali non sono disponibili, ma vanno scritte.

Prof. Montemurro

## Tipi di Variabili

1. null
2. Numero intero
3. Numero a virgola mobile (o floating-point number)
4. Stringa
5. Valori logici (o valori booleani)
6. Array
7. Oggetto
8. callable
9. resource

Nota bene: gli ultimi tre tipi non li vedremo.

Prof. Montemurro

## Tipo null

**null**: è un tipo il cui unico valore costante è `null`; questo valore viene usato ogni volta che vogliamo rappresentare l'assenza di un valore da qualche parte (es. funzione che non ritorna alcun valore).

Una variabile è `null` nei seguenti tre casi:

1. le è stato assegnato il valore costante `null`;
2. non le è stato ancora assegnato alcun valore;
3. è stata cancellata con la funzione **`unset()`**.

Prof. Montemurro

## Tipo null

Esempio 1: variabile null perché le è stato assegnato il valore costante `null`

```
$a = null;  
var_dump($a);           // NULL (stampato a video)
```

Esempio 2: variabile null perché non le è stato ancora assegnato alcun valore

```
var_dump($b);           // NULL (stampato a video)  
echo $b;                // non stampa alcunchè a video
```

Esempio 3: variabile null perché è stata cancellata con la funzione **`unset()`**

```
$a = 1;  
var_dump($a);           // int(1) (stampato a video)  
echo "<br>";  
unset($a);              // NULL (stampato a video)  
var_dump($a);
```

Prof. Montemurro

## Tipo int

**int**: tipo numero intero

### *Esempio*

```
$a = 1;  
var_dump($a);           // int(1)
```

Prof. Montemurro

## Tipo float

**float**: tipo numero a virgola mobile

### *Esempi*

```
$a = 1.1;  
$b = 1.2e3;           // 1.2 * 103  
$c = 7E-10;           // 7 * 10-10  
var_dump($a, $b, $c); // float(1.1) float(1200)  
                      // float(7.0E-10)
```

Prof. Montemurro



## Tipo float

### Approfondimento

**Curiosità:** si parla di floating-point (punto flottante) perché, secondo la **notazione anglosassone**, la separazione tra parte intera e parte decimale è il **punto**, mentre nella **notazione italiana** è la **virgola**. I numeri a virgola mobile sono nati per rappresentare i numeri reali  $\mathbb{R}$  nei calcolatori; i numeri reali possono essere infiniti (es.  $\pi$ ), mentre i numeri a virgola mobile sono un'approssimazione finita di  $\mathbb{R}$  basata su espansioni decimali finite.

*Esempio:* spostamento del punto (notazione anglosassone)

$$0.1234 \times 10^{-2} = 0.001234$$

$$0.1234 \times 10^1 = 1.234$$

$$0.1234 \times 10^4 = 1234$$

$$0.1234 \times 10^8 = 12,340,000$$

Prof. Montemurro

## Tipo string

**Modi di assegnare una stringa ad una variabile:**

1. racchiudendo il testo tra **doppi apici** `"`; questo metodo permette di espandere (es. stampare a video) eventuali altre variabili presenti all'interno della stringa;
2. racchiudendo il testo tra **apici singoli** `'`; tale metodo non consente di espandere eventuali altre variabili presenti all'interno della stringa.

Prof. Montemurro

## Tipo string

*Esempio:* modi di assegnare una stringa ad una variabile

```
<?php
    $nome = "Giovanni";
    $saluto = "Buongiorno $nome";
    echo "$saluto <br>";
    $saluto = 'Buongiorno $nome';
    echo "$saluto <br>";
?>
```

Risultato nel browser:

Buongiorno Giovanni  
Buongiorno \$nome

Prof. Montemurro

## Tipo string

**Sequenze di escape:** sequenze di caratteri particolari che iniziano con la barra inversa (o backslash) \ e vengono interpretate, relativamente alla visualizzazione del codice PHP (non della pagina web) solo se racchiuse tra doppi apici:

- \n a capo
- \t tabulazione orizzontale
- \\ barra inversa
- \t tabulazione
- \' apice
- \" virgolette
- \r ritorno carrello
- \\$ dollaro
- \a segnale acustico

Prof. Montemurro

## Tipo string

Approfondimento

*Esempio:* differenza tra `\n` tra apici doppi, tra apici singoli, e `<br>`

```
<?php
echo "Ciao \n Pippo <br>";
echo 'Ciao \n Pluto <br>';
echo "Ciao <br> Pippo";
?>
```

Visualizzando il codice PHP con Ctrl + U, si vede che la sequenza di escape `\n` nel primo `echo` produce un ritorno a capo (cosa che non fa nella pagina web).

```
1 Ciao
2 Pippo <br>Ciao \n Pluto <br>Ciao <br> Pippo
```

Prof. Montemurro

## Tipo bool

**bool:** tipo booleano, può assumere solo due valori, **true** o **false**.

*Esempio*

```
$a = true;
var_dump($a);           // bool(true)
```

Prof. Montemurro

## Tipo array

**Array in PHP:** è una variabile che contiene un insieme di **valori** ciascuno dei quali è identificato da un **indice** (o **chiave**). Quindi un array è un insieme di elementi, e ciascun elemento è costituito da una coppia chiave-valore (o indice-valore).

In PHP gli indici dell'array possono essere:

1. interi; in questo caso si hanno **array numerici**; l'indice è un numero, quello associato al valore del primo elemento è 0;
2. stringhe; in questo caso si hanno **array associativi**; l'indice è una stringa tra apici semplici o doppi e viene associato con l'**operatore di associazione =>**;
3. misti (interi e stringhe); in questo caso si hanno **array misti**.

In genere si preferisce "indice" per 1, e "chiave" per 2 e 3.

Prof. Montemurro

## Tipo array

**Sintassi per definire il valore di un elemento dell'array:**

```
$nomeArray[indice0] = valore0;
$nomeArray[indice1] = valore1; ...
```

oppure si usa la parola chiave **array** (vedi esempi nelle slide successive).

**Sintassi per l'accesso al valore di un elemento dell'array:**

```
$pippo = $nomeArray[indice]
```

Prof. Montemurro

## Tipo array: Esempio 1 Array Numerico

*Esempio 1: array numerico*

*// primo modo di definire un array numerico*

```
$voti[0] = 8.0;
```

```
$voti[1] = 6.5;
```

```
$voti[2] = 5.5;
```

```
$voti[3] = 7.0;
```

oppure con la parola chiave **array**:

*// secondo modo di definire un array numerico*

```

           0       1       2       3
$voti = array(8.0, 6.5, 5.5, 7.0);
```

Prof. Montemurro

## Tipo array: Esempio 1 Array Numerico

*continuazione esempio 1: array numerico*

*// terzo modo di definire un array numerico*

```
$voti[] = 8.0;
```

```
$voti[] = 6.5;
```

```
var_dump($voti); // array(2) { [0]=> float(8) [1]=>
                  float(6.5) }
```

**Principio di funzionamento:** quando si assegna un valore ad un elemento di un array, se non viene specificato alcun indice (cioè se si usa `[]`), viene preso il **massimo degli indici int** esistenti ed il nuovo indice sarà **quel** valore massimo più 1. Se non esiste ancora alcun indice int, l'indice sarà 0 (zero); ciò succede in due casi:

1. quando si crea un nuovo array assegnando un valore al primo elemento (vedi esempio sopra);
2. quando l'array già esiste, ma gli indici esistenti non sono int.

Prof. Montemurro

## Tipo array: Esempio 2 Array Numerico

Nel linguaggio PHP, a differenza di altri linguaggi di programmazione, un array può essere costituito da elementi di tipo diverso tra loro, cioè non è necessario che siano tutti dello stesso tipo.

*Esempio 2: array numerico*

```
$dati[0] = 30;
$dati[1] = "Roma";
$dati[2] = 12.7;
$dati[3] = true;
$dati[4] = null;
// o equivalentemente
$dati = array(30, "Roma", 12.7, true, null);
```

Prof. Montemurro

## Tipo array: Esempio 2 Array Numerico

*...continuazione esempio 2: coppia chiave-valore per array numerico*

```

      0         1         2         3         4
$dati = array(30, "Roma", 12.7, true, null);
var_dump($dati);

/* Output var_dump($dati);
array(5) {[0]=> int(30) [1]=> string(4) "Roma"
          [2]=> float(12.7) [3]=> bool(true) [4]=> NULL}
*/

```

Numero di caratteri della stringa Roma

Numero di elementi dell'array

Prof. Montemurro

## Tipo array: Esempio Array Associativo

*Esempio:* array associativo

```
$persona["cognome"] = "Rossi";
$persona["nome"] = "Toto";
$persona["nascita"] = 2000;
$persona["professione"] = "impiegato";
```

oppure con la parola chiave **array** e l'**operatore di associazione =>**:

```
$persona = array("cognome"=>"Rossi", "nome"=>"Toto",
    "nascita"=>2000, "professione"=>"impiegato");
```

Prof. Montemurro

## Tipo array: Esempio Array Associativo

*...continuazione esempio:* coppia chiave-valore per array associativo

```
$persona = array("cognome"=>"Rossi", "nome"=>"Toto",
    "nascita"=>2000, "professione"=>"impiegato");
var_dump($persona);
```

```
/* Output var_dump($prodotta);
array(4) { ["cognome"]=> string(5) "Rossi"
           ["nome"]=> string(4) "Toto"
           ["nascita"]=> int(2000)
           ["professione"]=> string(9) "impiegato" }
*/
```

Prof. Montemurro

## Tipo array: Esempio Array Misto

*Esempio:* array misto

```
$prodotto[0] = "Gelato";
$prodotto[1] = "Torta";
$prodotto["buono"] = "Lasagne";
```

oppure con la parola chiave **array** e l'**operatore di associazione =>**:

```
$prodotto = array(0=>"Gelato", 1=>"Torta",
                  "buono"=>"Lasagne");
```

Prof. Montemurro

## Tipo array: Esempio Array Misto

*...continuazione esempio:* coppia chiave-valore per array misto

```
$prodotto = array(0=>"Gelato", 1=>"Torta",
                  "buono"=>"Lasagne");
var_dump($prodotto);

/* Output var_dump($prodotto);
array(3) { [0]=> string(6) "Gelato"
           [1]=> string(5) "Torta"
           ["buono"]=> string(7) "Lasagne" }
*/
```

Prof. Montemurro



## Conversioni Automatiche di Tipo

**Conversioni automatiche** (o **conversioni implicite**, o **cast impliciti**) di tipo riguardano le **espressioni logiche**:

1. ogni valore numerico diverso da 0 vale `true`, mentre lo 0 vale `false`;
2. una qualsiasi stringa non vuota vale `true`, mentre la stringa vuota, `" "`, vale `false`;
3. valore `null` vale `false`.

Prof. Montemurro

## Conversioni Automatiche di Tipo

*Esempio 1:* in espressioni logiche 0 vale `false`, ogni numero  $\neq 0$  vale `true`

```
$a = 0;  
var_dump(!$a); // !$a è bool(true), dunque $a è false  
echo "<br> <br>";  
$b = -123.456;  
var_dump(!$b); // !$b è bool(false), dunque $b è true
```

Prof. Montemurro

## Conversioni Automatiche di Tipo

*Esempio 2:* in espressioni logiche stringa non vuota vale `true`, stringa vuota `""` vale `false`

```
$a = "";  
var_dump(!$a); // !$a è bool(true), dunque $a è false  
echo "<br> <br>";  
$b = "ciao";  
var_dump(!$b); // !$b è bool(false), dunque $b è true
```

Prof. Montemurro

## Conversioni Automatiche di Tipo

*Esempio 3:* in espressioni logiche `null` vale `false`

```
$a = null;  
var_dump(!$a); // !$a è bool(true), dunque $a è false
```

Prof. Montemurro

## Conversioni Automatiche di Tipo

Una **stringa contenente un valore numerico**, quando compare in un'espressione aritmetica, viene automaticamente convertita in un opportuno tipo numerico.

### Esempi

```
$str = "3";  
$str = $str * 2;           // $str è intero e vale 6  
  
$val = 1;  
$val = $val + "3";        // $valore vale 4
```

Prof. Montemurro

## Conversioni Automatiche di Tipo

### Esempi

```
$val = 5;  
$val = $val + "3.4Volt"; // $valore è float e vale 8.4  
  
$val = 5;  
$val = $val + null;      // $valore vale 5
```

Prof. Montemurro

## Conversioni Esplicite di Tipo

**Conversioni esplicite** (o **cast espliciti**) di tipo si eseguono antepoendo alla variabile il tipo che si vuole ottenere racchiuso tra parentesi tonde: `(int)`, `(float)`, `(double)`, `(bool)`, `(string)`.

### Esempio

```
$a = 1;
var_dump($a);           // int(1)
echo "<br> <br>";
$b = (float) $a;
var_dump($b);           // float(1)
echo "<br> <br>";
$c = (double) $a;
var_dump($c);           // float(1), no double(1)
```

Prof. Montemurro

## Costanti

**Costanti:** il loro valore non può essere modificato dopo che sono state definite; per definire una costante si usa la funzione **define** la quale ha due ingressi separati da una virgola, il nome della costante ed il suo valore.

**Convenzione:** i nomi delle costanti sono scritti in maiuscolo.

**Nota bene:** le costanti sono utilizzate nel codice indicandone il nome senza il simbolo del dollaro \$ iniziale (es. slide successiva).

Prof. Montemurro

## Costanti

### Esempio

```
<?php
    define("PERCENTUALE", 20); // percentuale di sconto
    define("G", 9.8);           // costante di gravità
    define("REGIONE", "Marche"); // nome della regione

    // uso costante PERCENTUALE senza $
    echo PERCENTUALE;
?>
```

Prof. Montemurro

## Operatori Aritmetici

**Operatori aritmetici:** si usano coi numeri e le variabili numeriche, e sono:

- +** addizione
- sottrazione
- \*** moltiplicazione
- /** divisione
- %** resto della divisione intera
- ++** incremento unitario
- decremento unitario

Prof. Montemurro

## Operatore di Concatenazione

**Operatore di concatenazione . (punto):** per concatenare le stringhe.

*Esempio*

```
<?php
    $nome = "Giovanni";
    $saluto = "Buongiorno " . $nome;
    echo $saluto . "<br>";
    $somma = 768;
    $messaggio = "Totale " . $somma;
    echo "$messaggio <br>";
?>
```

Buongiorno Giovanni  
Totale 768

L'istruzione `echo $messaggio . "<br>";` è equivalente all'ultima.

Prof. Montemurro

## Operatori di Assegnamento

**Operatore di assegnamento =** per assegnare un valore ad una variabile (già visto).

**Operatori di assegnamento combinati:**

Operatore	Esempio	Equivale a
<b>+=</b>	<code>\$a += \$b</code>	<code>\$a = \$a + \$b</code>
<b>-=</b>	<code>\$a -= \$b</code>	<code>\$a = \$a - \$b</code>
<b>*=</b>	<code>\$a *= \$b</code>	<code>\$a = \$a * \$b</code>
<b>/=</b>	<code>\$a /= \$b</code>	<code>\$a = \$a / \$b</code>
<b>%=</b>	<code>\$a %= \$b</code>	<code>\$a = \$a % \$b</code>
<b>.=</b>	<code>\$a .= \$b</code>	<code>\$a = \$a . \$b</code>

Prof. Montemurro

# Operatori di Confronto

**Operatori di confronto** per confrontare due valori:

Operatore	Esempio	Descrizione
<code>==</code>	<code>\$a == \$b</code>	true se \$a è uguale a \$b <u>dopo manipolazione del tipo</u> (cioè <u>conversione implicita</u> )
<code>===</code>	<code>\$a === \$b</code>	true se \$a è uguale a \$b, <u>e</u> se \$a e \$b sono dello stesso tipo ( <u>no</u> manipolazione tipo)
<code>!=</code>	<code>\$a != \$b</code>	true se \$a è diverso da \$b <u>dopo manipolazione del tipo</u>
<code>!==</code>	<code>\$a !== \$b</code>	true se \$a <u>non</u> è uguale a \$b, <u>oppure</u> \$a e \$b <u>non</u> sono dello stesso tipo ( <u>no</u> manipolazione tipo)
<code>&lt;</code>	<code>\$a &lt; \$b</code>	true se \$a è minore di \$b
<code>&gt;</code>	<code>\$a &gt; \$b</code>	true se \$a è maggiore di \$b
<code>&lt;=</code>	<code>\$a &lt;= \$b</code>	true se \$a è minore o uguale a \$b
<code>&gt;=</code>	<code>\$a &gt;= \$b</code>	true se \$a è maggiore o uguale a \$b

Prof. Montemurro

# Operatori di Confronto

## Esempio

```

$a = 5;
$b = "5";
var_dump($a);           // int(5)
echo "<br> <br>";
var_dump($b);           // string(1)"5"
echo "<br> <br>";
var_dump($a == $b);     // bool(true) (*)
echo "<br> <br>";
var_dump($a === $b);    // bool(false)
//(*) conversione implicita di $a da int a bool, e
// di $b da string a bool

```

Prof. Montemurro

## Operatori Logici

**Operatori logici** per valutare le espressioni logiche:

<b>!</b>	<b>not</b>	
<b>&amp;&amp;</b>	<b>and</b>	vero solo se entrambi gli operandi sono veri
<b>  </b>	<b>or</b>	vero se almeno uno degli operandi è vero
<b>xor</b>	<b>xor</b>	vero se solo uno dei due operandi è vero

\$a	\$b	!\$a	\$a && \$b	\$a    \$b	\$a xor \$b
true	true	false	true	true	false
true	false	false	false	true	true
false	true	true	false	true	true
false	false	true	false	false	false

Prof. Montemurro

## Messaggi di Errore

Approfondimento

Se l'esecuzione dello script provoca **errori**, per scelta predefinita non vengono visualizzati messaggi di errore e compare una **pagina bianca** del browser.

Per conoscere il tipo di errore che è stato riscontrato, si deve consultare il file **err.log** (o **error.log**) nella cartella **logs** del server Web (Apache), dove gli errori vengono registrati in ordine cronologico con data, ora, tipo di errore e numero della linea dello script che ha provocato la situazione di errore

Prof. Montemurro



## Messaggi di Errore

Approfondimento

**Buona norma:** in fase di collaudo conviene inserire le seguenti due istruzioni all'inizio dello script PHP per visualizzare gli eventuali errori generati durante l'esecuzione dello script stesso.

```
error_reporting(E_ALL) ;  
ini_set("display_errors", 1) ;
```

1. La funzione **error\_reporting**(livello\_errore) serve per specificare quali tipi di errori devono essere visualizzati; passandole come argomento la costante **E\_ALL**, la funzione visualizzerà tutti i tipi di errori.
2. In generale la funzione **ini\_set**(opzione, valore) imposta un'opzione di configurazione; l'opzione **display\_errors** stampa a video gli errori se il suo valore è 1, non li stampa se il suo valore è 0.

Prof. Montemurro

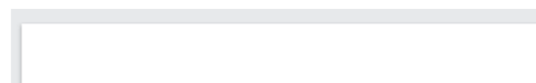
## Messaggi di Errore

Approfondimento

*Esempio*

```
<?php  
echo $a;  
?>
```

Schermata bianca



```
<?php  
error_reporting(E_ALL) ;  
ini_set("display_errors", 1) ;  
echo $a;  
?>
```

Messaggio d'errore

Notice: Undefined variable: a in /home/rxNHHh/prog.php on line 4

Prof. Montemurro

# Funzioni

## Obiettivi

1. Suddividere un problema in sottoproblemi più semplici da risolvere (ogni sottoproblema è una funzione, dunque ha senso creare una funzione anche nel caso in cui dovesse essere chiamata una sola volta)
2. Ripetere più volte determinate procedure

**Funzioni:** permettono di isolare porzioni ben definite di codice, e di assegnare loro un nome che sarà usato per richiamarle nei punti del programma in cui servono.

Prof. Montemurro

# Funzioni

In PHP, a differenza di C++, non è possibile dichiarare una funzione senza definirla (no prototipi), ma una funzione deve essere sempre dichiarata e definita. Per farlo:

1. si usa la parola chiave **function** seguita dal nome della funzione;
2. si specificano eventuali **parametri formali** (o **argomenti formali**) i quali sono racchiusi tra parentesi tonde dopo il nome della funzione, e sono separati dalla virgola;
3. le istruzioni della funzione sono racchiuse tra parentesi graffe; questo è il **corpo della funzione**;
4. se la funzione deve restituire un valore, si usa l'istruzione **return** seguita dal valore da restituire.

Prof. Montemurro

# Funzioni

Sintassi per la **definizione** di una funzione definita dall'utente:

```
function nomeFunzione (pFormale1, pFormale2,...){
    // istruzioni
    // eventuale return di qualcosa
}
```

Sintassi per la **chiamata** della funzione nomeFunzione:

```
nomeFunzione(pAttuale1, pAttuale2,...);
```

Chiaramente, se c'è un **return**, occorre salvare il valore in un contenitore:

```
nomeContenitore = nomeFunzione(pAttuale1, pAttuale2,...);
```

Prof. Montemurro

# Funzioni

*Esempio:* funzione per calcolare l'area di un rettangolo

```
<?php
function areaRettangolo($base, $altezza){
    $area = $base * $altezza;
    return $area;
}

$risultato = areaRettangolo(2, 3);
echo $risultato;
?>
```

Parametri formali (o argomenti formali)

Parametri attuali (o argomenti attuali)

// 6

Prof. Montemurro

# Funzioni

## Modalità di passaggio dei parametri alle funzioni



	Passaggio per valore	Passaggio per riferimento (&)	Passaggio per valore predefinito
Modifica parametri attuali	No	Si	No

Prof. Montemurro

# Funzioni

*Esempio:* passaggio di parametri per valore predefinito (o per default)

```
<?php
function stampa($param = "predefinito") {
    echo "Il parametro è: " . $param . "<br>";
}
```

```
stampa();  Il parametro è: predefinito
stampa("come mi pare!");  Il parametro è: come mi pare!
?>
```

Prof. Montemurro

## Funzioni

*Esempio:* passaggio di parametri per riferimento

```
<?php
function nominativo(&$nome) {
    echo $nome . "<br>";
    $nome = "Pluto";
}

$nomeOriginale = "Pippo";
nominativo($nomeOriginale);
echo $nomeOriginale;
?>
```

Pippo  
Pluto

Prof. Montemurro

## Funzioni

**Scope di una variabile** (o **ambito di una variabile**, o **visibilità di una variabile**): parte di programma in cui la variabile può essere usata.

**Variabili locali:** tutte le variabili che definiamo all'interno del corpo di una funzione.

**Variabili globali:** tutte le variabili definite fuori dalle funzioni.

**Attenzione:** i parametri formali e le variabili locali hanno senso solo all'interno della funzione stessa, cioè i parametri formali e le variabili locali sono visibili ed utilizzabili (nascono e muoiono) solo all'interno della funzione. Il loro scope è l'interno della funzione.

Prof. Montemurro

# Funzioni

*Esempio:* scope delle variabili locali

```
<?php
function f1(){
    $variabile = 1;
    var_dump($variabile);
}

var_dump($variabile);
echo "<br>";
f1();
var_dump($variabile);
?>
```

Diagram illustrating the scope of local variables:

- Inside the function `f1()`, `$variabile` is defined and its value is dumped as `NULL`.
- Outside the function, `$variabile` is not defined, so its dump results in `int(1)` (likely a typo for `int(1)` in the original image).
- After the function call, `$variabile` is still not defined in the global scope, so its dump results in `NULL`.

Prof. Montemurro

# Funzioni

*Esempio:* scope delle variabili globali

```
<?php
function f1(){
    var_dump($numero);
}

$numero = 3.14;
var_dump($numero);
echo "<br>";
f1();
?>
```

Diagram illustrating the scope of global variables:

- Inside the function `f1()`, `$numero` is not defined, so its dump results in `float(3.14)` (likely a typo for `float(3.14)` in the original image).
- Outside the function, `$numero` is defined as `3.14`, so its dump results in `NULL` (likely a typo for `float(3.14)` in the original image).

Prof. Montemurro

## Funzioni: Parametri Tipizzati Approfondimento

Nelle funzioni è possibile specificare di che tipo devono essere i suoi parametri.

### Esempio

```
<?php
function areaRettangolo(int $base, int $altezza){
    $area = $base * $altezza;
    return $area;
}

$risultato = areaRettangolo(2, "ciao");
// PHP Fatal error: Uncaught TypeError: Argument 2
// passed to areaRettangolo() must be of the type
// integer, string given...thrown...on line 2
?>
```

Prof. Montemurro

## Funzioni: Parametri Tipizzati Approfondimento

### Esempio

```
<?php
function areaRettangolo(int $base, int $altezza){
    $area = $base * $altezza;
    return $area;
}

$risultato = areaRettangolo(2, "3 ciao");
echo $risultato; // 6
// Attenzione: nonostante il secondo argomento attuale
// non sia un intero, ma una stringa "3 ciao", si ha una
// conversione implicita da stringa ad intero per cui
// $altezza non vale "3 ciao", ma vale 3
?>
```

Prof. Montemurro

## Funzioni: Parametri Tipizzati Approfondimento

Per garantire che il tipo degli argomenti attuali e formali sia esattamente lo stesso (e quindi per evitare anche conversioni implicite), dopo il tag di apertura `php` si usa **`declare(strict_types=1);`**

### Esempio

```
<?php declare(strict_types=1);
function areaRettangolo(int $base, int $altezza) {
    $area = $base * $altezza;
    return $area;
}
$risultato = areaRettangolo(2, "3 ciao");
// PHP Fatal error: Uncaught TypeError: Argument 2
passed to areaRettangolo() must be of the type integer,
string given...thrown...on line 2
?>
```

Prof. Montemurro

## Funzioni: return Tipizzato Approfondimento

**Specificare tipo dell'eventuale valore di ritorno:** (1) si scrive `: nomeTipo` tra la parentesi tonda di chiusura della lista degli argomenti e la parentesi graffa di apertura del corpo della funzione; (2) dopo il tag di apertura `php` si usa **`declare(strict_types=1);`** per evitare conversioni implicite.

### Esempio

```
<?php // qui non ho scritto declare(strict_types=1);
function areaRettangolo(float $base, float $altezza) : int {
    $area = $base * $altezza;
    return $area;
}
$risultato = areaRettangolo(2.2, 3.0);
echo $risultato; // 6
// Conversione implicita di $area = 2.2 * 3.0 = 6.6 da float
ad int, ecco perché il risultato è 6 e non 6.6
?>
```

Prof. Montemurro



## Funzioni: return Tipizzato Approfondimento

### Esempio

```
<?php declare(strict_types=1);
function areaRettangolo(float $base, float $altezza) : int {
    $area = $base * $altezza;
    return $area;
}
$risultato = areaRettangolo(2.2, 3.0);
echo $risultato;
// PHP Fatal error: Uncaught TypeError: Return value of
areaRettangolo() must be of the type integer, float
returned...thrown...on line 4
?>
```

Possibili soluzioni: (1) scrivo : float o (2) scrivo return (int) \$area; per convertire esplicitamente il float in int.

Prof. Montemurro

## Strutture di Controllo

**Strutture di controllo:** sono costrutti per controllare l'ordine di esecuzione delle istruzioni (i costrutti sono mattoncini da costruzione di un linguaggio di programmazione). Le due strutture di controllo che vedremo sono:

1. le **strutture di controllo di selezione** (if...else; if...elseif; switch...case);
2. le **strutture di controllo di iterazione (o di ripetizione)** (while; do...while; for; foreach).

Prof. Montemurro

## 1i Strutture di Controllo di Selezione: if...else

**Sintassi generale della struttura di controllo di selezione if...else**

```
if (condizione) {
    // istruzioni eseguite se la condizione è vera
} else {
    // istruzioni eseguite se la condizione è falsa
}
```

Se il blocco di istruzioni è composto da una sola istruzione, si possono omettere le parentesi graffe {}:

```
if (condizione) //istruz. eseguite se condizione è vera
else // istruzioni eseguite se la condizione è falsa
```

Prof. Montemurro

## 1ii Strutture di Controllo di Selezione: if...elseif

**Sintassi generale della struttura di controllo if...elseif (o if...else if) per la selezione annidata**

```
$valore = ...;
if ($valore > 0)
    echo "valore maggiore di 0";
elseif ($valore == 0)
    echo "valore uguale a 0";
else
    echo "valore minore di 0";
```

**Nota bene:** si può usare sia la sintassi **if...elseif** che **if...else if**.

Prof. Montemurro

## 1iii Strutture di Controllo di Selezione: switch...case

### Sintassi generale della struttura di controllo switch...case

```
switch (espressione) {
    case valore1:
        // istruzioni
        break;
    case valore2:
        // istruzioni
        break;
    // eventuali altri case
    default:
        // istruzioni; qui non serve break;
}
```

Prof. Montemurro

## 1iii Strutture di Controllo di Selezione: switch...case

**switch...case:** quando il programma arriva all'istruzione `switch`, viene valutata l'espressione tra parentesi tonde;

- se l'espressione è uguale al `valore1`, si esegue l'alternativa `case valore1`;
- se l'espressione è uguale al `valore2`, si esegue l'alternativa `case valore2`;
- ...in generale, si esegue l'alternativa corrispondente al `case` "che ha lo stesso valore" dell'espressione che compare accanto all'istruzione `switch`;
- l'alternativa **default** è opzionale e, se presente, viene eseguita solo se nessuna delle alternative precedenti è stata eseguita.

Se non c'è l'alternativa `default` e se il valore dell'espressione non coincide con alcun valore dei vari `case`, allora lo `switch` non viene eseguito e si passa all'istruzione successiva.

Prof. Montemurro

## 1iii Strutture di Controllo di Selezione: switch...case

*Esempio:* switch...case

```
$vocale = ...
switch($vocale) {
    case "A":
    case "a":
        echo "Hai scelto la vocale a.";
    ...
    case "U":
    case "u":
        echo "Hai scelto la vocale u.";
    default:
        echo "Non hai scelto alcuna vocale.";
}
```

Prof. Montemurro

## 2 Strutture di Controllo di Iterazione

**Ciclo** (o **struttura di controllo di iterazione**, o **struttura di controllo di ripetizione**): ripetizione di una o più istruzioni (in un algoritmo) un numero di volte definito o indefinito, specificato tramite una condizione; tale ripetizione è chiamata **iterazione**.

I cicli sono utili quando si deve operare con lunghe liste di dati, per esempio con gli array.

### Tipi di strutture di controllo di iterazione

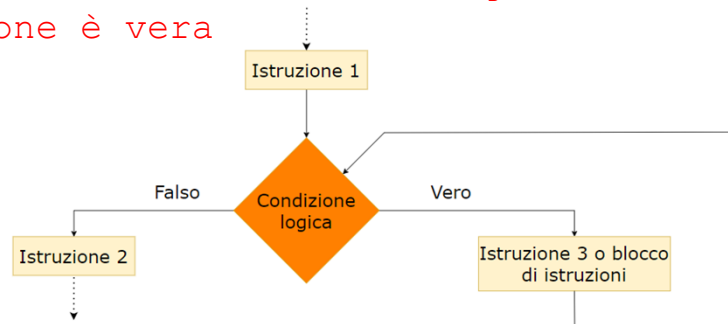
- i. ciclo `while`
- ii. ciclo `do...while`
- iii. ciclo `for`
- iv. ciclo `foreach`

Prof. Montemurro

## 2i Ciclo while

**Sintassi generale del ciclo `while`** (in italiano significa quando)

```
<?php
// istruzione 1
while (condizione) {
    // istruzione 3 o blocco di istruzioni eseguite fin
    // quando la condizione è vera
}
// istruzione 2
?>
```

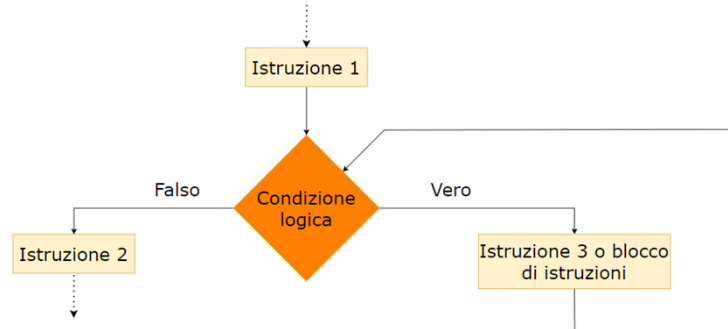


Prof. Montemurro

## 2i Ciclo while

**`while`:** quando nel programma si arriva all'istruzione `while`, viene valutata la condizione tra parentesi tonde;

- se la condizione vale `true`, viene eseguito il blocco di istruzioni tra parentesi graffe, cioè il **corpo del `while`**; dopo si ritorna alla condizione, e se vale di nuovo `true`, si ripete un'altra volta il corpo del `while`;
- se la condizione vale `false`, si esce dal ciclo `while` e si esegue l'istruzione che segue il ciclo `while`.



Prof. Montemurro

## 2i Ciclo while

### Esempio

```
// Questo ciclo while stampa i numeri da 0 a 9
$i = 0;
while ($i < 10) {
    echo "Il valore di i è: " . $i . "<br>";
    $i++;
}
```

### Output

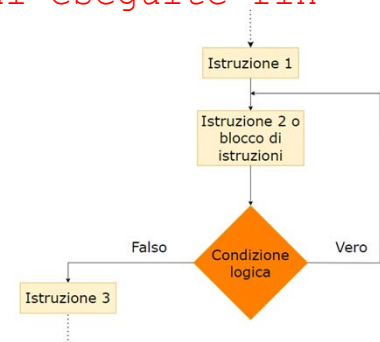
```
Il valore di i è: 0
Il valore di i è: 1
Il valore di i è: 2
Il valore di i è: 3
Il valore di i è: 4
Il valore di i è: 5
Il valore di i è: 6
Il valore di i è: 7
Il valore di i è: 8
Il valore di i è: 9
```

Prof. Montemurro

## 2ii Ciclo do...while

**Sintassi generale del ciclo do...while** (viene eseguito almeno una volta)

```
<?php
// istruzione 1
do {
    // istruzione 2 o blocco di istruzioni eseguite fin
    // quando la condizione è vera
} while (condizione);
// istruzione 3
?>
```

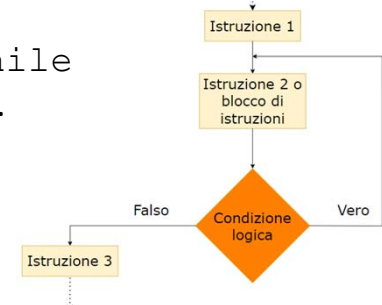


Prof. Montemurro

## 2ii Ciclo do...while

**do...while:** quando nel programma si arriva all'istruzione `do`, viene eseguito almeno una volta il blocco di istruzioni tra parentesi graffe, cioè il **corpo del do** (nel caso del ciclo `while`, il corpo del `while` viene eseguito per la prima volta solo se la condizione vale `true`). In seguito viene valutata la condizione tra parentesi tonde;

- se la condizione vale `true`, viene eseguito nuovamente il **corpo del do**; dopo si ritorna alla condizione, e se vale di nuovo `true`, si ripete un'altra volta il corpo del `do`;
- se la condizione vale `false`, si esce dal ciclo `do...while` e si esegue l'istruzione che segue il ciclo `do...while`.



Prof. Montemurro

## 2ii Ciclo do...while

### Esempio

```
// Questo ciclo do...while stampa i numeri da 0 a 9
$i = 0;
do {
    echo "Il valore di i è: " . $i . "<br>";
    $i++;
} while ($i < 10);
```

### Output

```
Il valore di i è: 0
Il valore di i è: 1
Il valore di i è: 2
Il valore di i è: 3
Il valore di i è: 4
Il valore di i è: 5
Il valore di i è: 6
Il valore di i è: 7
Il valore di i è: 8
Il valore di i è: 9
```

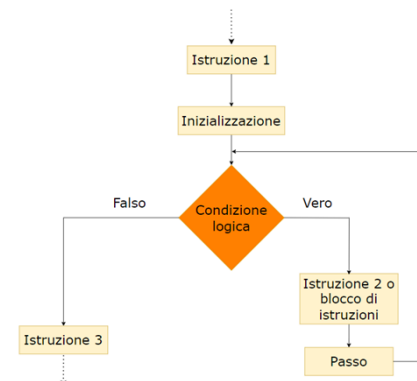
Prof. Montemurro

## 2iii Ciclo for

### Sintassi generale del ciclo for

```
<?php
for (inizializzazione; condizione; passo) {
    // istruzioni eseguite se la condizione è vera
}
?>
```

**Attenzione:** la componente `inizializzazione` viene eseguita una sola volta.



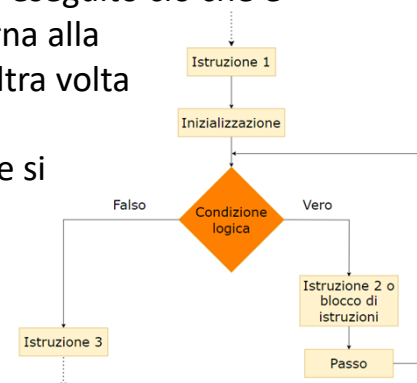
Prof. Montemurro

## 2iii Ciclo for

**for:** quando il programma arriva all'istruzione `for`, viene eseguito ciò che è specificato nella componente `inizializzazione`; in seguito viene valutata la condizione:

- se la condizione vale `true`, viene eseguito il blocco di istruzioni tra le parentesi graffe, ossia il **corpo del `for`**, e poi viene eseguito ciò che è specificato nella componente `passo`; dopo si ritorna alla condizione, e se vale di nuovo `true`, si ripete un'altra volta il corpo del `for` ed il `passo`;
- se la condizione vale `false`, si esce dal ciclo `for` e si esegue l'istruzione che segue il ciclo `for`.

**Attenzione:** la componente `inizializzazione` viene eseguita una sola volta.



Prof. Montemurro



## 2iii Ciclo for

### Esempio

```
// Questo ciclo for stampa i numeri da 0 a 9
for ($i = 0; $i < 10; $i++) {
    echo "Il valore di i è: " . $i . "<br>";
}
```

### Output

```
Il valore di i è: 0
Il valore di i è: 1
Il valore di i è: 2
Il valore di i è: 3
Il valore di i è: 4
Il valore di i è: 5
Il valore di i è: 6
Il valore di i è: 7
Il valore di i è: 8
Il valore di i è: 9
```

Prof. Montemurro

## 2iv Ciclo foreach

**Ciclo foreach:** si usa solo per gli array (\*) e serve per scorrere i suoi **elementi** (ricordare che in generale l'elemento di un array è una **coppia chiave-valore**, cioè ogni valore ha un suo indice).

Esistono due sintassi per il ciclo foreach:

1. sintassi del ciclo foreach per ottenere solo i valori degli elementi dell'array;
2. sintassi del ciclo foreach per ottenere sia i valori che i relativi indici degli elementi dell'array.

(\*) In realtà anche per gli oggetti di una classe che non vedremo (un oggetto è un'istanza di una classe).

Prof. Montemurro

## 2iv Ciclo foreach

**Sintassi del ciclo foreach per ottenere solo i valori degli elementi dell'array**

```
foreach ($nomeArray as $value) {  
    // istruzioni  
}
```

- Alla prima iterazione del ciclo `foreach`, il valore del primo elemento di `$nomeArray` viene assegnato alla variabile `$value`.
- Alla seconda iterazione del ciclo `foreach`, il valore del secondo elemento di `$nomeArray` viene assegnato alla variabile `$value` (quindi `$value` non contiene più il **valore del primo elemento** di `$nomeArray` in quanto **tale valore** è stato sovrascritto dal valore del secondo elemento di `$nomeArray`).
- ...e così via per le altre iterazioni.

Prof. Montemurro

## 2iv Ciclo foreach

*Esempio:* ciclo `foreach` per ottenere solo i valori degli elementi dell'array

```
$voti = array(8.0, 6.5, 5.5, 7.0);
```

```
foreach ($voti as $value) {  
    echo "$value <br>";  
}
```

Output

```
8  
6.5  
5.5  
7
```

Prof. Montemurro

## 2iv Ciclo foreach

**Sintassi del ciclo `foreach` per ottenere sia i valori che i relativi indici degli elementi dell'array**

```
foreach ($nomeArray as $key => $value) {  
    // istruzioni  
}
```

- Alla prima iterazione del ciclo `foreach`, il valore del primo elemento di `$nomeArray` viene assegnato alla variabile `$value` ed il suo indice viene assegnato alla variabile `$key`.
- ...e così via per le altre iterazioni.

Prof. Montemurro

## 2iv Ciclo foreach

*Esempio:* ciclo `foreach` per ottenere sia valori che i relativi indici degli elementi dell'array

```
$persona = array("cognome"=>"Rossi", "nome"=>"Toto",  
    "nascita"=>2000, "professione"=>"impiegato");
```

```
foreach ($persona as $key => $value) {  
    echo "$key: $value <br>";  
}
```

**Output**

```
cognome: Rossi  
nome: Toto  
nascita: 2000  
professione: impiegato
```

Prof. Montemurro

## Form HTML: Ripasso GET e POST

`<form action="URL" method="post">...</form>`

**action:** attributo del tag `form` che indica dove si trova il programma PHP sul server che riceverà ed elaborerà i campi del form.

**method:** attributo del tag `form` che specifica (al browser) la modalità con cui inviare i campi del form al server. Esistono due metodi, `get` e `post`, le cui differenze sono:

GET	POST
Dati inviati <u>visibili</u> nell'URL	Dati inviati <u>non</u> visibili nell'URL
Pagina generata <u>salvabile</u> nei preferiti	Pagina generata <u>non</u> salvabile nei preferiti
Massima lunghezza URL: 3000 caratteri	URL di <u>qualsiasi</u> lunghezza

**Nota bene:** `get` non adatto per l'invio di dati sensibili (es. password ecc.).

Prof. Montemurro

## Variabili superglobali

**Variabili superglobali** (o **variabili predefinite**, o **variabili d'ambiente**): variabili visibili in qualsiasi punto del codice PHP; queste variabili sono create direttamente dall'interprete PHP e sono raggruppate nei seguenti array associativi (`$_GET`, `$_POST`, `$_FILES`, `$_SERVER`, `$_COOKIE`, `$_SESSION`):

- **\$\_GET:** array associativo che contiene le variabili passate dal form HTML allo script PHP tramite il metodo get:  
`<form action="URL" method="get">`
- **\$\_POST:** array associativo che contiene le variabili passate dal form HTML allo script PHP tramite il metodo post:  
`<form action="URL" method="post">`

Prof. Montemurro

## Variabili superglobali

*Esempio 1:* invio di dati dal form (lato client) allo script PHP (lato server) con metodo post

File HTML per inviare la coppia chiave-valore (qui il valore è il testo scritto dall'utente nella casella di testo)

```
<form action="cerca.php" method="post">
  Parola: <input type="text" name="parola" />
  <input type="submit" value="Cerca" />
</form>
```

File PHP per accedere al campo parola, cioè al valore memorizzato nella casella di testo il cui name è parola; tale valore è accessibile nello script PHP tramite la seguente variabile (ipotesi: utente scrive test e poi clicca su Cerca):

```
$valoreParola = $_POST["parola"];
echo $valoreParola; // test
```

Prof. Montemurro

## Variabili superglobali

*Esempio 2:* invio di dati dal form (lato client) allo script PHP (lato server) con metodo get

File HTML per inviare la coppia chiave-valore (ipotesi: utente seleziona Visa)

```
<form action="invia.php" method="get">
  Scegli la carta di credito<br>
  <input type="radio" name="carta" value="visa"
    checked>Visa<br>
  <input type="radio" name="carta"
    value="mastercard">Mastercard<br>
```

Scegli la carta di credito

☒ Visa

☐ Mastercard

...

File PHP per accedere al valore inviato dal form allo script PHP, tale valore è accessibile nello script PHP tramite la seguente variabile:

```
$valoreCarta = $_GET["carta"];
echo $valoreCarta; // visa
```

Prof. Montemurro

## Variabili superglobali

*Esempio 3: solito get*

**File HTML**

```
<form action="invia.php" method="get">
  Lingue conosciute<br>
  <input type="checkbox" name="cb1"
    value="ita">italiano<br>
  <input type="checkbox" name="cb2"
    value="ing">inglese<br>
  <input type="checkbox" name="cb3"
    value="fra">francese<br> ...
```

Lingue conosciute

☒ italiano

☐ inglese

☒ francese

**File PHP**

```
echo $_GET["cb3"]."<br>"; // fra
//echo $_GET["cb2"]."<br>"; // errore se non c'è ing
echo $_GET["cb1"]; // ita
```

Prof. Montemurro

## Variabili superglobali

**Funzione predefinita `isset`:** permette di verificare se l'utente ha assegnato un valore o meno ad una variabile definita nello script PHP.

*Esempio*

```
if (isset ($variabile)) {
    // istruzioni da eseguire se la variabile ha un
    // valore
} else {
    // istruzioni da eseguire se la variabile non ha un
    // valore
}
```

La funzione **`isset`** restituisce `true` se a `$variabile` è stato assegnato un valore, mentre restituisce `false` se a `$variabile` non è mai stato assegnato un valore, oppure ha se ha il valore `null`.

Prof. Montemurro

## Variabili superglobali

...continuazione esempio 3: solito get

Per evitare errori tipo quello visto 2 slide fa conviene usare **isset**.

File HTML: il solito di prima

...

File PHP

```
if (isset($_GET["cb3"])) echo $_GET["cb3"] . "<br>";
if (isset($_GET["cb2"])) echo $_GET["cb2"] . "<br>";
if (isset($_GET["cb1"])) echo $_GET["cb1"] . "<br>";
```

Prof. Montemurro

## Variabili superglobali

Approfondimento

- **\$\_FILES**: array associativo che permette di gestire l'upload di file fatto dal form; si usa l'attributo **enctype** per specificare come devono essere codificati i dati del form quando si inviano al server, questo attributo si può usare solo col metodo post e può assumere solo i seguenti tre valori:
  1. **application/x-www-form-urlencoded** questo è il valore di default e si usa solo per i campi testuali, non per l'upload di file; tutti i caratteri vengono codificati prima dell'invio, (i) gli spazi vengono convertiti in simboli "+", e (ii) i caratteri speciali vengono convertiti in valori ASCII HEX;
  2. **multipart/form-data** questo valore è necessario quando si effettua l'upload di file tramite form;
  3. **text/plain** invia dati senza alcun codifica (non raccomandato).

Prof. Montemurro

## Variabili superglobali

Approfondimento

- **`$_FILES`**

Le variabili contenute in **`$_FILES`** sono le seguenti sei:

1. **`$_FILES['file']['name']`** variabile che contiene il nome originale del file che l'utente ha scelto di caricare;
2. **`$_FILES['file']['type']`** variabile che contiene il tipo **MIME** (**Multipurpose Internet Mail Extensions**) del file;  
*Esempio: "image/gif"; questo tipo MIME non è tuttavia controllato sul lato PHP;*
3. **`$_FILES['file']['size']`** variabile che contiene la dimensione in byte del file caricato;

Prof. Montemurro

## Variabili superglobali

Approfondimento

- **`$_FILES`**

Le variabili contenute in **`$_FILES`** sono le seguenti sei:

4. **`$_FILES['file']['tmp_name']`** variabile che contiene il nome del file temporaneo in cui il file caricato è stato memorizzato sul server;
5. **`$_FILES['file']['full_path']`** variabile che contiene il percorso completo del file inviato dal browser (disponibile da PHP 8.1.0);
6. **`$_FILES['file']['error']`** variabile che contiene il codice di errore associato al caricamento del file.

Prof. Montemurro



## Variabili superglobali

Approfondimento

- **\$\_SERVER**: array associativo che contiene le variabili passate allo script PHP dal server web. Alcune variabili contenute in **\$\_SERVER** sono le seguenti:
  1. **\$\_SERVER[ 'REMOTE\_ADDR' ]** che contiene l'indirizzo IP della macchina dell'utente da cui proviene la richiesta di esecuzione dello script PHP;
  2. **\$\_SERVER[ 'SERVER\_SOFTWARE' ]** che contiene una stringa con il nome del server web che esegue lo script PHP (es. Apache/2.2.24);
  3. **\$\_SERVER[ 'HTTP\_USER\_AGENT' ]** che contiene una stringa con il nome del browser utilizzato dall'utente;
  4. **\$\_SERVER[ 'SERVER\_NAME' ]** che contiene una stringa con il nome della macchina su cui gira il server web;
  5. **\$\_SERVER[ 'PHP\_SELF' ]** che contiene una stringa con il nome dello script PHP in esecuzione.

Prof. Montemurro

## Variabili superglobali

Approfondimento

- **\$\_SERVER**: array associativo che contiene variabili che possono essere usate:
  1. per registrare gli accessi al server attraverso i **file di log** (log = registro);
  2. per produrre successivamente le statistiche relative ai visitatori (es. da dove si collegano e quale browser usano).

Gli array associativi predefiniti del linguaggio PHP devono essere scritti con i caratteri tutti maiuscoli. Gli indici delle variabili, tra parentesi quadre, sono racchiusi da apici singoli o doppi:

*Esempio: ottenere indirizzo IP dell'utente*

```
<?php echo $_SERVER[ 'REMOTE_ADDR' ] . "<br>" ?>
```

Prof. Montemurro

## Variabili superglobali

Approfondimento

- **\$\_COOKIE**: array associativo che contiene le variabili passate allo script PHP tramite i cookie.

**Cookie**: file di testo di piccole dimensioni, memorizzati sul computer dell'utente (client) su richiesta esplicita del server, questi file vengono inviati al server quando l'utente visita di nuovo il sito web. I cookie contengono alcune informazioni (es. informazioni sulle pagine web visitate sul sito sul quale si sta navigando; informazioni per riconoscere univocamente il client; informazioni sulle preferenze dell'utente quali tema chiaro/scuro, voci memorizzate nel carrello di un sito di e-commerce, username e password ecc.)

Prof. Montemurro

## Variabili superglobali

Approfondimento

- **\$\_SESSION**: array associativo che contiene le variabili utilizzate per implementare il concetto di sessione.

**Sessione** (o **sessione di lavoro**): svolgimento, in un arco temporale, di un insieme di attività tra loro correlate, e tali che una o alcune di queste che sono preliminari ad altre (es. un qualunque servizio di accesso a informazioni richiede preliminarmente l'identificazione dell'utente; fatto ciò, l'utente può accedere al servizio per accedere ai dati sfruttando gli script PHP sul server).

**Nota bene**: le variabili di uno script, di norma, hanno validità limitata al solo script PHP nel quale sono state create; terminata l'esecuzione dello script PHP, esse vengono distrutte. A questa regola fanno eccezione le variabili dell'array **\$\_SESSION** la cui validità può essere estesa a diversi script PHP. Questa caratteristica è sfruttata per realizzare sessioni di lavoro.

Prof. Montemurro

# Classi e Oggetti

aa

Prof. Montemurro

## Paradigma Imperativo di Programmazione Orientata agli Oggetti

**Paradigma orientato agli oggetti:** analizza il problema cercando di individuare gli **oggetti** che lo compongono e le associazioni tra essi. Tali oggetti possono essere oggetti reali (es. automobile) oppure concetti (o oggetti astratti) (es. figura geometrica, operazione matematica). Più oggetti comunicano tra di loro scambiandosi dei **messaggi**.

*Esempi:* linguaggio di programmazione C++, Java, PHP.

Prof. Montemurro

# Struttura di un Oggetto

La **struttura di un oggetto** è composta:

1. dagli **attributi** i quali sono elementi che descrivono le caratteristiche dell'oggetto;
2. dai **metodi** i quali sono i comportamenti e le funzionalità che l'oggetto mette a disposizione.



Prof. Montemurro

## Paradigma Imperativo di Programmazione Orientata agli Oggetti

*Esempio di oggetto: aereo*

- i. Attributi: modello, numero di posti, tipo di carburante, velocità massima ecc.
- ii. Metodi: decolla, atterra, rifornisci ecc.



Prof. Montemurro

## Accesso ai DB con MySQLi

**Obiettivo:** eseguire la connessione al DB ed eseguire le operazioni di interrogazione e manipolazione su un DB con l'**estensione MySQLi** (SQL sta per **Structured Query Language**).

**Connessione al server:** la classe **mysqli** rappresenta la connessione tra il PHP ed il **DB MySQL**. La prima operazione eseguita dallo script PHP è la connessione al **server MySQL**. Si definisce l'oggetto **\$conn** (o istanza della classe **mysqli**) si ottiene con l'operatore **new**:

```
$conn = new mysqli($host, $username, $password,  
                    $db_nome);
```

Prof. Montemurro

## Accesso ai DB con MySQLi

```
$conn = new mysqli($host, $username, $password,  
                    $db_nome);
```

Gli argomenti dell'istanza della classe **mysqli** sono i seguenti quattro:

1. **\$host** rappresenta l'indirizzo IP o il nome del server su cui è in esecuzione il server MySQL (se si utilizza un server di sviluppo su computer locale, si indica **localhost**);

Prof. Montemurro

