PIEMONTE
FONDI STRUTTURALI E DI INVESTIMENTO EUROPEI 2014/2020
fondo sociale europeo
ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

Funzioni in PHP

Corso Backend System Integrator

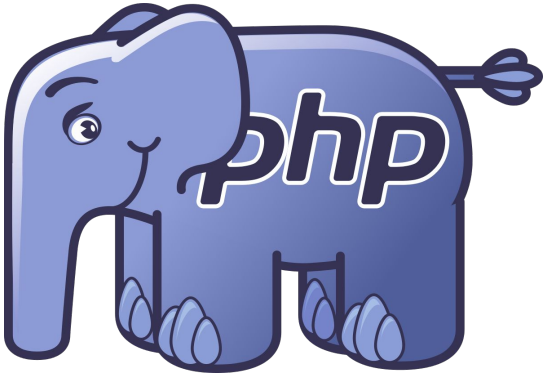
Modulo **Programmazione PHP**

Docente: Dott. Enrico Zimuel

in collaborazione con:
MINISTERO DELL'ISTRUZIONE, DELL'UNIVERSITÀ E DELLA RICERCA
UNIONE EUROPEA
REGIONE PIEMONTE
per una crescita intelligente, sostenibile ed inclusiva
www.regione.piemonte.it/europa2020
INIZIATIVA CO-FINANZIATA CON FSE

Programma

- Funzioni in PHP
- Modalità strict type
- Visibilità delle variabili
- Parametri passati per valore o riferimento
- Funzioni ricorsive
- Funzioni anonime



Funzioni

- In PHP una funzione è definita con il costrutto function:

```
function <name>(<params>): <return-type>
{
    <statement>;
    return <value>;
}
```

- *<params>*, *<return-type>* e *return <value>* sono opzionali

Esempio

```php
function hello(string $name)
{
    printf("Hello %s", $name);
}
hello('Alberto'); // prints 'Hello Alberto'
```

```php
function hi(string $name): string
{
    return sprintf("Hello %s", $name);
}
echo hi('Alberto'); // prints 'Hello Alberto'
```

Esercizio

- Scrivere una funzione **hello(string $name)** che stampi:
  - "Buon giorno $name"
  - "Buon pomeriggio $name"
  - "Buona sera $name"
  - "Buona notte $name"

  a seconda dell'ora attuale

- Suggerimento: utilizzare la funzione date() del PHP

Modalità *strict type*

- Il PHP è in grado di controllare il rispetto dei tipi delle variabili in ingresso e uscita delle funzioni attivando la modalità ***strict type***
- E' necessario specificare il codice seguente all'inizio dello script PHP e per ogni file:

```php
<?php
declare(strict_types=1);
```

Esempio

```php
<?php
function sum(int $a, int $b): int {
   return $a + $b;
};
echo sum('1', 1); // 2
```

```php
<?php
declare(strict_types=1);

function sum(int $a, int $b): int {
   return $a + $b;
};
echo sum('1', 1); // TypeError
```

PHP Fatal error: Uncaught **TypeError**: Argument 1 passed to sum() must be of the type int, string given

Suggerimento

E’ consigliabile utilizzare sempre la modalità di **strict_types=1** per evitare problemi con la conversione automatica del PHP

Return type

- int
- float
- bool
- array
- string
- object

- Class/interface name
- self/parent
- callable
- iterable
- mixed*
- void

* a partire dal PHP 8.0.0

Parametri

- I parametri di una funzione possono essere zero, uno o più (separati da virgola)
- Un parametro può avere un valore di default (parametro opzionale)
- I parametri opzionali devono essere sempre gli ultimi

Esempio

```php
function hello(string $name, string $greeting = 'Hello'): string
{
    return sprintf("%s %s", $greeting, $name);
}
hello('Alberto'); // 'Hello Alberto'
hello('Alberto', 'Buongiorno'); // 'Buongiorno Alberto'
hello('Alberto', 'Hello'); // 'Hello Alberto'
hello(); // Fatal error: Too few arguments
```

Nota: nei prossimi esempi del corso daremo per scontato l'utilizzo di ***strict_types=1***

Esempio: invertiamo i parametri

```php
function hello(string $greeting = 'Hello', string $name): string
{
    return sprintf("%s %s", $greeting, $name);
}
hello('Alberto'); // Fatal error: Too few arguments
hello('Buongiorno', 'Alberto'); // returns 'Buongiorno Alberto'
hello(null, 'Alberto'); // Fatal error: Argument must be string
hello(); // Fatal error: Too few arguments
```

Visibilità delle variabili

- Le variabili create all'interno di una funzione PHP sono visibili soltanto all'interno della funzione stessa
- Lo ***scope*** di una variabile è il suo ambito di visibilità

Esempio

```php
function sum(array $items): int
{
    $sum = 0;
    foreach ($items as $value) {
        $sum += $value;
    }
    return $sum;
}
$sum = 10;
$numbers = [3, 12, 4];
printf("Sum: %d\n", sum($numbers)); // Sum: 19
printf("Sum: %d\n", $sum); // Sum: 10
```

Global

- Per accedere ad una variabile esterna è necessario dichiararla **global**

```php
function sum(array $items): int
{
    global $sum;
    foreach ($items as $value) {
        $sum += $value;
    }
    return $sum;
}
$sum = 10;
$numbers = [3, 12, 4];
printf("Sum: %d\n", sum($numbers)); // Sum: 29
printf("Sum: %d\n", $sum); // Sum: 29
```

Evitate di utilizzare global!!!

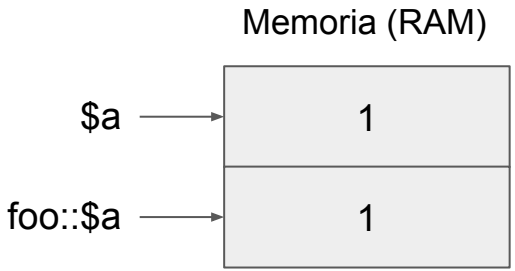
Esercizio

- Scrivere una funzione che restituisca la media aritmetica di un array di numeri

Passaggio dei parametri per valore

- I parametri di una funzione sono passati per valore in PHP

```php
function foo(int $a) {
    echo ++$a;
}
$a = 1;
foo($a); // 2
echo $a; // 1
```

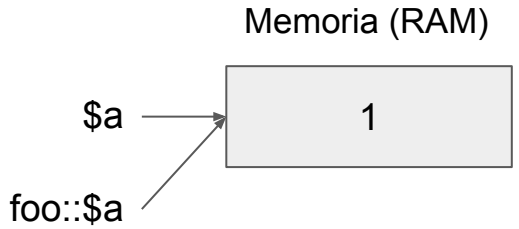


- Il valore della variabile globale $a è copiato nella variabile $a all'interno della funzione foo()

Passaggio dei parametri per riferimento

- I parametri di una funzione possono essere passati per riferimento aggiungendo il carattere **&** prima del $

```php
function foo(int &$a) {
    echo ++$a;
}
$a = 1;
foo($a); // 2
echo $a; // 2
```



- Il parametro **&$a** è un puntatore (riferimento) alla variabile $a globale

Esercizio: FizzBuzz

- **Fizz Buzz** è un gioco di gruppo per bambini, utile per insegnare loro, la divisione e la divisibilità. I giocatori si alternano per contare in modo incrementale partendo da 1 e sostituendo qualsiasi numero divisibile per 3 con la parola "Fizz" e qualsiasi numero divisibile per 5 con la parola "Buzz".
- Scrivere una funzione PHP che stampi i primi 100 risultati del gioco **Fizz Buzz** aggiungendo questa regola:
  - se il numero è divisibile per 3 e per 5 sostituire il numero con la parola “FizzBuzz”

Operatore *variadic*

- L'operatore ***variadic*** (...) consente di specificare un numero variabile di parametri

```php
function foo(int $a, ...$options) {
    foreach ($options as $value) {
        /* ... */
    }
}
foo(1); // $options = []
foo(1, 'bar', 'baz'); // $options = ['bar', 'baz']
foo(1, 'bar', 2); // $options = ['bar', 2]
```

Sull’operatore *variadic*

- E’ possibile specificare il tipo degli elementi
- Ogni funzione può avere al massimo un operatore variadic
- Deve essere sempre l’ultimo dei parametri

Esempio

```php
function foo(int $a, string ...$options) {
    foreach ($options as $value) {
        /* ... */
    }
}
foo(1); // $options = []
foo(1, 'bar', 'baz'); // $options = ['bar', 'baz']
foo(1, 'bar', 2); // Fatal error
```

Parametri *nullable*

- UN parametro di una funzione può essere ***nullable*** ossia può accettare il valore NULL
- Per specificare un valore nullable si utilizza un punto interrogativo (**?**) prima del tipo (es. **?string $name**)
- Può essere utilizzato anche come risultato di una funzione

Esempio

```php
function hello(?string $name): ?string
{
    return $name === null ? null : 'Hello ' . $name;
}
hello('Alberto'); // Hello Alberto
hello(); // Fatal error: Too few arguments
```

Domanda

- Il valore opzionale NULL per un parametro è equivalente ad un *nullable*?

```php
function foo(int $a = null) {
    /* ... */
}
// sono equivalenti?
function foo(?int $a) {
    /* ... */
}
```

Funzioni ricorsive

- Quando una funzione richiama se stessa viene detta **ricorsiva**
- Le funzioni ricorsive sono una potente astrazione che consente di semplificare la scrittura del codice
- Vanno utilizzate con attenzione perché possono utilizzare troppe risorse (es. esplodere in un ciclo infinito o consumare troppa memoria)

Esempio: Fibonacci

```php
function fibonacci(int $num)
{
    if ($num === 0) {
        return 0;
    } elseif ($num === 1) {
        return 1;
    } else {
        return fibonacci($num-1) + fibonacci($num-2);
    }
}
$start = microtime(true);
echo fibonacci(40);
printf(" %.3f sec\n", microtime(true) - $start);
```

CPU Intel i9-8950HK 2.90GHz

ricorsione	senza
136 sec	0.04 sec

Funzioni anonime

- Il PHP consente di creare **funzioni anonime**, ossia senza un nome, conosciute anche con il nome di ***closure***.
- Le funzioni anonime sono particolarmente utilizzate quando si ha la necessità di creare una funzione da eseguire al verificarsi di un evento (programmazione asincrona)
- Vengono assegnate ad una variabile ed eseguite aggiungendo le parentesi tonde alla variabile

Esempio

```php
$avg = function (array $values) {
    return array_sum($values) / count($values);
};
echo $avg([ 1, 2, 3, 4, 5, 6]);
```

Operatore *use*

- L'operatore ***use*** consente di importare una variabile all'interno di una funzione anonima

```php
$avg = function (array $values) {
    return array_sum($values) / count($values);
};
$std = function(array $values) use ($avg) {
    $avg = $avg($values);
    $tot = 0;
    foreach ($values as $v) {
        $tot += ($v - $avg) ** 2;
    }
    return sqrt($tot/count($values));
};
echo $std([ 1, 2, 3, 4, 5, 6]);
```

Esercizio (da consegnare)

- Scrivere una funzione che ordini gli elementi di un array in senso crescente
- La funzione deve modificare direttamente l'array, senza restituire nessun valore
- Potete utilizzare qualsiasi algoritmo per l'ordinamento (es. Bubble sort)

Grazie dell’attenzione!

Per informazioni:
enrico.zimuel@its-ictpiemonte.it

POR Piemonte
FSE 2014-2020