INTRODUZIONE A PHP

Dr. Barbara Rita Barricelli

UNIVERSITÀ DEGLI STUDI DI BRESCIA

LEZIONE 2

ESERCIZI SULLA LEZIONE DI IERI

- Creare uno script che dato un valore N e un carattere C stampi a video una forma fatta con N colonne e N righe di caratteri C

Esempio:

N = 5

C = *

*****
*****
*****
*****
*****

ESERCIZI SULLA LEZIONE DI IERI

- Creare uno script che dato un valore N stampi a video tutti i numeri da 1 a N che hanno 7 come divisore.

Esempio:

N = 14

7, 14

ESERCIZI SULLA LEZIONE DI IERI

- Scrivere uno script che dato un numero N crei un array che contenga i numeri da 1 a N.

Esempio:

N = 5

Array:

1,2,3,4,5

FUNZIONI PER GESTIRE LE VARIABILI

`empty(valore)`

Una variabile è considerata vuota (empty) se non esiste o se il suo valore è impostato a false o a 0. Restituisce TRUE se è vuota, FALSE altrimenti.

FUNZIONI PER GESTIRE LE VARIABILI

```
isset(valore)
```

Restituisce TRUE se una variable è stata dichiarata, FALSE altrimenti.

FUNZIONI PER GESTIRE LE VARIABILI

```
is_int(valore)
```

Restituisce TRUE se la variabile è un intero, FALSE altrimenti.

FUNZIONI PER GESTIRE LE VARIABILI

```
is_float(valore)
```

Restituisce TRUE se la variabile è un float, FALSE altrimenti.

FUNZIONI PER GESTIRE LE VARIABILI

```
is_string(valore)
```

Restituisce TRUE se la variabile è una stringa, FALSE altrimenti.

FUNZIONI PER GESTIRE LE VARIABILI

```
is_array(valore)
```

Restituisce TRUE se la variabile è un array, FALSE altrimenti.

FUNZIONI PER GESTIRE LE VARIABILI

```
is_numeric(valore)
```

Restituisce TRUE se la variabile è di tipo numerico, FALSE altrimenti.

FUNZIONI PER GESTIRE LE VARIABILI

```
gettype(valore)
```

Restituisce il tipo della variabile.

FUNZIONI PER GESTIRE LE VARIABILI

```
print_r(valore)
```

Visualizza informazioni riguardanti una variabile in una forma leggibile da una persona.

FUNZIONI PER GESTIRE LE VARIABILI

```
unset(valore)
```

Distrugge la variabile specificata.

ESERCIZI

1. Data una variabile `$a` non ancora definita assegnarla alla variabile `$b` e verificare se `$b` è vuota
2. Data `$a = 5` assegnarla a `$b` e controllare se `$b` è settata
3. Controllare se `$b` è un float
4. Controllare se `$b` è una stringa
5. Data `$a = '5'` assegnarla a `$b` e controllare se `$b` è un intero
6. Controllare se `$b` è una stringa
7. Controllare se `$b` è numerico
8. Eliminare la variabile `$a`

FUNZIONI PER GESTIRE LE STRINGHE

`strlen(stringa)`

Restituisce la lunghezza di una stringa, 0 se la stringa è vuota.

FUNZIONI PER GESTIRE LE STRINGHE

```
substr(stringa, intero [, intero])
```

Restituisce la porzione di stringa specificata dal primo numero intero (offset) e dall'eventuale secondo intero (lunghezza).

Esempi:

```
echo substr('abcdef', 1);     // bcdef
echo substr('abcdef', 0, 4);  // abcd
echo substr('abcdef', -1, 1); // f
```

FUNZIONI PER GESTIRE LE STRINGHE

```
str_replace(stringa1, stringa2, stringa3)
```

Sostituisce tutte le occorrenze di `stringa1` dentro a `stringa3` con `stringa2`.

Esempio:

```
str_replace("world", "Peter", "Hello world!"); // Hello
Peter!
```

FUNZIONI PER GESTIRE LE STRINGHE

`strpos(stringa1, stringa2)`

Restituisce, se esiste, la posizione della prima occorrenza di `stringa2` in `stringa1`, altrimenti FALSE.

FUNZIONI PER GESTIRE LE STRINGHE

```
strtolower(stringa)
```

Trasforma tutti i caratteri di una stringa in caratteri minuscoli.

FUNZIONI PER GESTIRE LE STRINGHE

`strtoupper(stringa)`

Trasforma tutti i caratteri di una stringa in caratteri maiuscoli.

FUNZIONI PER GESTIRE LE STRINGHE

```
ucfirst(stringa)
```

Trasforma il primo carattere di una stringa in carattere maiuscolo.

FUNZIONI PER GESTIRE LE STRINGHE

`ucwords(stringa)`

Restituisce una stringa con tutte le iniziali delle parole in maiuscolo (se il carattere è alfabetico).

FUNZIONI PER GESTIRE LE STRINGHE

```
explode(stringa1, stringa2)
```

Separa `stringa2` dividendola in sottostringhe (messe in un array) usando il carattere `stringa1` come separatore.

Esempio:

```
$pizza  = "fetta1 fetta2 fetta3 fetta4 fetta5 fetta6";
$pezzi = explode(" ", $pizza);
echo $pezzi[0]; // fetta1
echo $pezzi[1]; // fetta2
```

ESERCIZI

1. Contare il caratteri della stringa 'abcd'
2. Della stringa 'Buongiorno a tutti' creare una sottostringa a partire dal 5° carattere lunga 6 caratteri
3. Rimpiazzare la parola 'buongiorno' con la parola 'ciao' della stringa 'Buongiorno a tutti'
4. Rimpiazzare l'occorrenza 'dom' (con d minuscola) con il carattere 'x' della stringa 'Domani è domenica'
5. Individuare la posizione della prima occorrenza del carattere 'm' nella stringa 'Domani è domenica'
6. Scrivere 'Buongiorno a tutti' in maiuscolo
7. Scrivere 'buongiorno a tutti' con solo la prima lettera in maiuscolo
8. Scrivere 'buongiorno a tutti' con in maiuscolo la prima lettera di tutte le parole

FUNZIONI PER GESTIRE GLI ARRAY

```
count(array)
```

Conta gli elementi in un array.

FUNZIONI PER GESTIRE GLI ARRAY

`array_reverse(array)`

Inverte la posizione degli elementi in un array.

FUNZIONI PER GESTIRE GLI ARRAY

`sort(array)`

Ordina gli elementi di un array mettendone i valori in ordine crescente.

FUNZIONI PER GESTIRE GLI ARRAY

`rsort(array)`

Ordina gli elementi di un array mettendone i valori in ordine decrescente.

FUNZIONI PER GESTIRE GLI ARRAY

```
asort(array)
```

Ordina gli elementi di un array mettendone i valori in ordine crescente e mantenendo l'associazione degli indici.

Esempio:

```
$frutta = array("d" => "limone", "a" => "arancia", "b" =>
"banana", "c" => "mela");

asort($ffrutta); // c = mela, b = banana, d = limone, a =
arancia
```

FUNZIONI PER GESTIRE GLI ARRAY

`arsort(array)`

Ordina gli elementi di un array mettendone i valori in ordine decrescente e mantenendo l'associazione degli indici.

FUNZIONI PER GESTIRE GLI ARRAY

```
in_array(valore, array)
```

Cerca un valore all'interno di un array. Se lo trova restituisce TRUE, altrimenti FALSE.

FUNZIONI PER GESTIRE GLI ARRAY

```
array_key_exists(chiave, array)
```

Restituisce TRUE se la chiave è presente nell'array, altrimenti FALSE.

FUNZIONI PER GESTIRE GLI ARRAY

```
array_search(valore, array)
```

Cerca un valore all'interno di un array. Restituisce la sua chiave se esiste, FALSE altrimenti.

FUNZIONI PER GESTIRE GLI ARRAY

```
array_merge(array, array [, array…])
```

Unisce gli elementi di più array in modo sequenziale.

FUNZIONI PER GESTIRE GLI ARRAY

`array_pop(array)`

Estrae l'ultimo elemento di un array. La lunghezza dell'array si riduce di uno.

FUNZIONI PER GESTIRE GLI ARRAY

```
array_push(array, value [, value…])
```

Aggiunge in fondo a un array uno o più valori.

FUNZIONI PER GESTIRE GLI ARRAY

`array_shift(array)`

Estrae il primo elemento di un array. La lunghezza dell'array si riduce di uno. Tutti gli indici numerici vengono aggiornati mentre restano invariate le chiavi non numeriche.

FUNZIONI PER GESTIRE GLI ARRAY

```
array_unshift(array, value [, value, …])
```

Aggiunge uno o più elementi in testa a un array. Tutti gli indici numerici vengono aggiornati mentre restano invariate le chiavi non numeriche.

FUNZIONI PER GESTIRE GLI ARRAY

```
implode(stringa, array)
```

Restituisce tutti i valori contenuti in un array separati da una stringa.

Esempio:

```
$array = array('cognome', 'email', 'telefono');
$elenco = implode(",", $array);
echo $elenco; // cognome,email,telefono
```

ESERCIZI

1. Creare un array così composto ('Luca', 'Giovanni', 'Matteo', 'Paolo', 'Antonio', 'Marco', 'Giuseppe');
2. Contare gli elementi dell'array
3. Creare un array invertendo gli elementi del precedente array
4. Ordinare gli elementi del primo array
5. Cercare se 'Giovanni' è presente nell'array
6. Estrarre e stampare l'ultimo elemento dell'array precedentemente ordinato
7. Estrarre e stampare il primo elemento dell'array precedentemente ordinato
8. Inserire in testa a quest'ultimo array i due nominativi precedentemente estratti
9. Creare una stringa con tutti gli elementi di questo array
10. Creare un array così composto ('padre' => 'Claudio', 'madre' => 'Paola', 'figlio' => 'Marco', 'figlia' => 'Elisa');
11. Ordinare in modo decrescente l'array (comprese le chiavi)
12. Cercare se esiste un elemento di chiave 'zio'
13. Cercare se esiste l'elemento 'Claudio' e indicarne la chiave

FUNZIONI PER GESTIRE LE DATE

`time()`

Restituisce il timestamp, cioè il numero di secondi a partire dalla Unix Epoch (January 1 1970 00:00:00 GMT).

Provate e vedete che numero ottenete?

Qual è il vantaggio di usare questo formato?

FUNZIONI PER GESTIRE LE DATE

```
strtotime(stringa);
```

Ci permette di passare valori simili al linguaggio naturale e di ottenerne la conversione.

Esempi da provare:

```
echo strtotime("now");
echo strtotime("next Thursday");
echo strtotime("+1 week");
```

FUNZIONI PER GESTIRE LE DATE

```
date(formato,timestamp)
```

Permette di formattare una data in un formato specifico. Si veda il link: https://www.php.net/manual/en/datetime.format.php

Esempio:

```
echo date('d/m/Y H:i:s', $timestamp); // stamperà, ad
esempio, 17/07/2021 14:30:21
```

FUNZIONI PER GESTIRE LE DATE

```
mktime(ore, minuti, secondi, mese, giorno, anno)
```

Restituisce il timestamp di una certa data specificata.

FUNZIONI PER GESTIRE LE DATE

```
checkdate(mese, giorno, anno)
```

Verifica la validità di una data.

Esempi:

```
checkdate(12, 31, 2000); // TRUE
checkdate(2, 29, 2001); // FALSE
```

ESERCIZI

1. Calcolare il timestamp della data 24/4/2021 ore 15.56.20
2. Impostare la data con giorno in due cifre, mese in tre lettere, anno in due cifre, ora in due cifre, minuti in due cifre
3. Verificare se la data 1/5/2021 è una data valida
4. Verificare se la data 7/19/2021 è una data valida

COMPITO!

- Create una semplice pagina web
- Create un file CSS per gestirne l'aspetto grafico