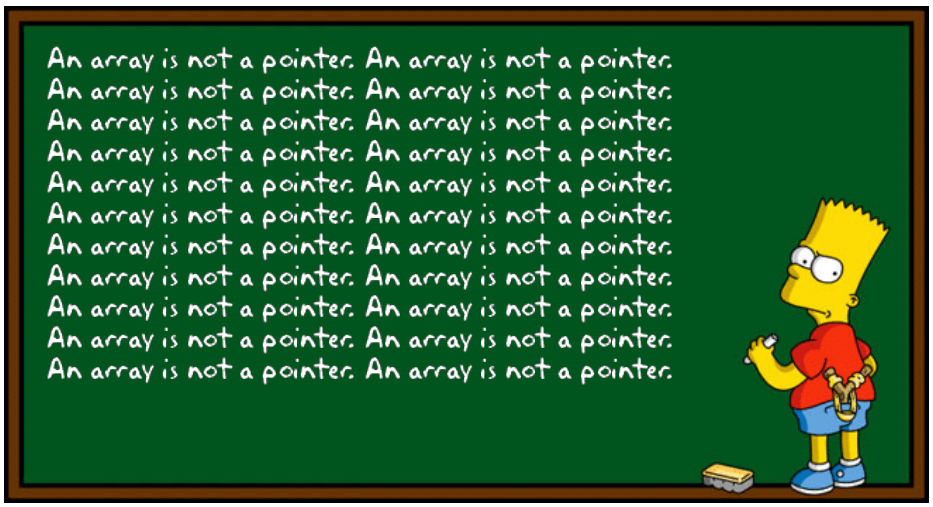
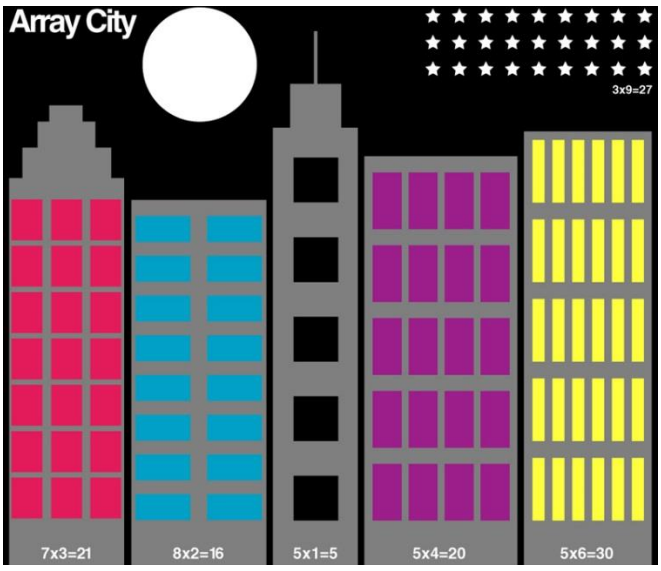
Array

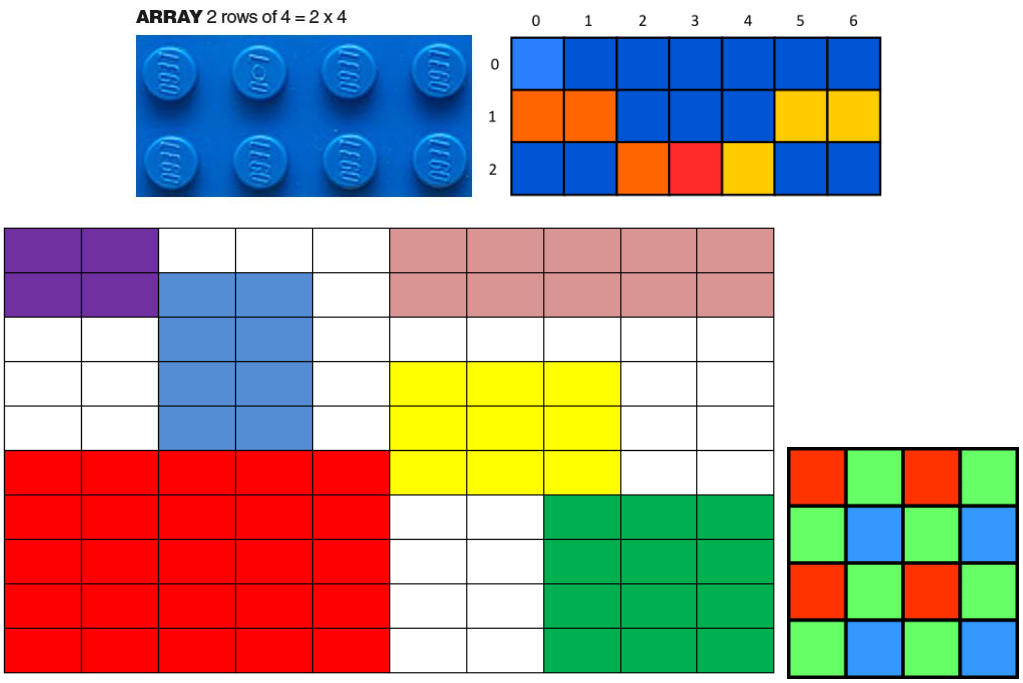


Possiamo descrivere un array come un costrutto informatico utilizzabile per memorizzare una serie di elementi. In generale un array a N elementi è paragonabile a una matrice matematica a N dimensioni.

L'array più semplice è quello a 1 dimensione. In termini più semplici può essere considerato una **collezione di elementi** ognuno identificato da un **indice**; gli indici di un array possono essere numerici o stringhe. Gli elementi dell'array possono contenere al proprio interno qualsiasi tipo di dato, compresi oggetti o altri array.

Vediamo alcuni esempi di strutture dati per rappresentare gli oggetti attraverso l'uso degli array:





Un array può essere definito in due modi equivalenti:

```
$myArray = array();
$myArray = [];
```

I due esempi si equivalgono: abbiamo definito una variabile `$myArray` assegnandole un array vuoto.

Assegnamento valori

Ad ogni elemento dell'array possiamo assegnare un valore (numerico o stringa) in due modi differenti:

- Inizializzazione: assegnamento durante la creazione dell'array:

```
$amici = array('paolo', 'luca', 'chiara');
$amici = ['paolo', 'luca', 'chiara'];
```

Anche in questo caso i due esempi si equivalgono.

- Assegnamento manuale:

```
$cars[0] = "Mercedes";
$cars[1] = "BMW";
$cars[2] = "Ferrari";
```

In ogni momento possiamo visualizzare il contenuto di un array con la funzione `print_r()`, utile in caso di debug:

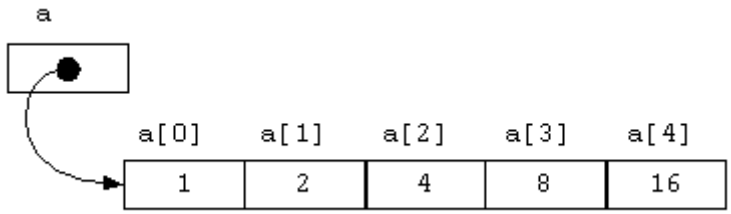
```
$amici = array('paolo', 'luca', 'chiara');
print_r($amici);
```

L'esempio restituirà un output come il seguente:

```
Array
(
    [0] => paolo
    [1] => luca
    [2] => chiara
)
```

Un array è quindi un contenitore di elementi contraddistinti da un indice. Se l'indice non viene esplicitato di default è di tipo numerico `0-based`. `0-based` sta ad indicare che l'indice dell'array inizierà da 0 anziché da 1. Come abbiamo potuto vedere nell'output dell'esempio precedente, il primo elemento dell'array (`paolo`) possedeva infatti l'indice con valore 0.

Accesso agli elementi di un array



Nel caso di array con indice numerico è possibile accedere ai singoli elementi attraverso il proprio indice:

```
$amici = array('paolo', 'luca', 'chiara');
echo "Il mio amico preferito si chiama " . $amici[2];
```

L'esempio stamperà `Il mio amico preferito si chiama chiara` perché ho deciso di stampare il terzo (indice 2 partendo da 0) elemento dell'array.

indice	0	1	2
elemento	paolo	luca	chiara

La funzione echo permette di stampare a video il contenuto dell'array o del testo, come nell'esempio precedente.

Esempio:

```
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
```

Nota sugli array in php

A differenza di altri linguaggi di programmazione, in php gli elementi di un array possono essere di tipo diverso. Esempio:

```
$libro = array('Harry Potter 1', 'Fantasy', 30);
```

Array associativo

Non necessariamente si deve utilizzare un indice numerico per accedere agli elementi di un'array. È possibile infatti accedere agli elementi dell'array anche con indici alfanumerici, cioè con indici stringa anziché con valore numerico. La dichiarazione con assegnazione può essere fatta ancora in due modi. Esempio:

```
$persona1['nome'] = 'Paolo';
$persona1['cognome'] = 'Rossi';
$persona1[golfatti'] = 145;
$persona2 = array('nome' => 'Paolo', 'cognome' => 'Rossi', 'golfatti' => 145);
```

E per modificare il valore di un elemento, ancora:

```
$persona1[golfatti'] = 200;
```

Vediamo un altro esempio:

```
$annoDiNascita = [
    'Paolo' => '1990',
    'Luca' => '1993',
    'Chiara' => '1989'
];
```

Nell'array appena definito ogni elemento è caratterizzato da una coppia `chiave -> valore` in cui la chiave (o indice) è il nome di una persona e il valore è il suo anno di nascita. Supponiamo di voler stampare la data di nascita di una delle persone dell'elenco, possiamo accedere al suo valore in questo modo:

```
echo "L'anno di nascita di Paolo è " . $annoDiNascita['Paolo'];
```

indice	Paolo	Luca	Chiara
elemento	1990	1993	1989

Come per gli indici numerici è sufficiente indicare l'indice tra parentesi quadre per accedere al valore di un elemento. Usando la funzione `print_r()` sull'array definito in precedenza, eseguendo quindi il comando `print_r($annoDiNascita);` otterremo il seguente output:

```
Array
(
    [Paolo] => 1990
    [Luca] => 1993
    [Chiara] => 1989
)
```

Array annidiati

Un array può contenere ulteriori array, in questo caso potremmo avere una situazione come la seguente:

```
$amici = [
    'Paolo' => [
        'anno' => '1990',
        'sesso' => 'M',
```

```
        'email' => 'paolo2@gmail.com'
    ],
    'Luca' => [
        'anno' => '1993',
        'sesso' => 'M',
        'email' => 'luca2@yahoo.com'
    ],
    'Chiara' => [
        'anno' => '1989',
        'sesso' => 'F',
        'email' => 'chiara3@alice.it'
    ],
];
```

Nell'esempio abbiamo un array con chiavi contenenti i nomi degli amici. Il valore degli elementi dell'array è un ulteriore sotto-array contenente le proprietà degli amici: anno di nascita, sesso e email. In questo caso per accedere ad un valore specifico dell'array possiamo usare la sintassi:

```
echo 'La mail di Chiara è ' . $amici['Chiara']['email'];
```

Richiamando `print_r()` sull'array avremo in output:

```
Array
(
    [Paolo] => Array
        (
            [anno] => 1990
            [sesso] => M
            [email] => paolo2@gmail.com
        )

    [Luca] => Array
        (
            [anno] => 1993
            [sesso] => M
            [email] => luca2@yahoo.com
        )

    [Chiara] => Array
        (
            [anno] => 1989
```

```
            [sesso] => F
            [email] => chiara3@alice.it
        )
)
```

Aggiungere o modificare il valore di un elemento

Nello stesso modo in cui possiamo accedere al contenuto di un elemento di un array possiamo anche modificarne un valore. Quindi riprendendo l'array `$amici`, supponendo di voler modificare la mail di uno degli utenti possiamo operare in questo modo:

```
$amici['Paolo']['email'] = 'paolo.rossi@gmail.com'
```

Lanciando `print_r()` dopo la modifica l'output sarà:

```
Array
(
    [Paolo] => Array
        (
            [anno] => 1990
            [sesso] => M
            [email] => paolo.rossi@gmail.com
        )

    [Luca] => Array
        (
            [anno] => 1993
            [sesso] => M
            [email] => luca2@yahoo.com
        )

    [Chiara] => Array
        (
            [anno] => 1989
            [sesso] => F
            [email] => chiara3@alice.it
        )
)
```

Nel caso in cui volessimo aggiungere un nuovo partecipante possiamo utilizzare la sintassi:

```
$amici['Daniele'] = [
    'anno' => 1968,
    'sesso' => 'M',
    'email' => 'daniele.corti@liceotosi.va.it'
];
```

E' sufficiente richiamare l'array indicandogli il nuovo indice ed il valore che conterrà.

Richiamando `print_r()` sull'array avremo:

```
Array
(
    [Paolo] => Array
        (
            [anno] => 1990
            [sesso] => M
            [email] => paolo.rossi@gmail.com
        )

    [Luca] => Array
        (
            [anno] => 1993
            [sesso] => M
            [email] => luca2@yahoo.com
        )

    [Chiara] => Array
        (
            [anno] => 1989
            [sesso] => F
            [email] => chiara3@alice.it
        )
    [Daniele] => Array
        (
            [anno] => 1968
            [sesso] => M
            [email] => daniele.corti@liceotosi.va.it
        )
)
```

Come visto esistono due alternative per creare array associativi:

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

oppure:

```
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

Esempio

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

Attraverso un ciclo possiamo estrarre i valori contenuti in un array associativo:

Esempio

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>
```

Array a due dimensioni - matrici

Un array a 2 dimensioni (matrice) è un array di array e un arrat a 3 dimensioni è un array di un array di un array.

Vediamo il seguente esempio:

Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

Possiamo memorizzare i valori rappresentati nella tabella sopra attraverso l'utilizzo delle matrici:

```
$cars = array
  (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
  );
```

ora la matrice $cars contiene 4 array e ha due indici: riga e colonna.

Vediamo un altro esempio:

```
echo "Creazione array con un elemento:<br />";
  $Biblio = array(0 => array("Autore"=>"Gianni Rodari", "Titolo"=>"il
treno azzurro", "Note"=>"Avventure"));
  print_r($Biblio);
  echo "<br /><br />aggiunta di un altro elemento<br />";
  $Biblio[1]["Autore"] = "Alberto Manzi";
  $Biblio[1]["Titolo"] = "visita allo zoo";
  $Biblio[1]["Note"] = "da colorare";
```

per accedere agli elementi della matrice dobbiamo utilizzare due indici (riga e colonna):

Esempio

```
<?php
echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
?>
```

Le funzioni PHP per gestire gli array

Ciclare gli elementi di un array

Per estrarre gli elementi di un array si possono utilizzare i costrutti for, foreach, do…while, while.

for ($i=0; $i<sizeof($myArray); $i++)

echo "<li>$myArray[$i]";

Un costrutto degno di nota è foreach:

```
1. $nomi = array("marco", "luca", "paolo");
2. foreach( $nomi as $valore)
3. {
4.     echo $valore."<br />";
5. }
```

In questo esempio si avrà che nel primo ciclo $valore sarà valorizzato dal primo elemento che compone l'array, cioè "marco"; nel secondo ciclo "luca"; nel terzo ciclo "paolo".

Da notare che a differenza del ciclo for in questo caso non viene indicato quante volte il ciclo dovrà ripetersi: infatti il ciclo si ripeterà tante volte quanti sono gli elementi dell'array. Nell'esempio riportato si avranno 3 iterazioni.

Tale ciclo potrà assumere anche un ulteriore sintassi, che può essere utile in caso di chiavi letterali (ma non solo). Ne riporto un esempio:

```
1. <?php
2. $nomi=array('direttore'=>'marco', 'vice'=>'luca', 'segretario'=>'paolo');
3.
4. foreach($nomi as $chiave => $valore)
5. {
6.    echo $chiave.": ".$valore."<br />";
7. }
8. ?>
```

Il foreach richiede come unico parametro l'array che si intende "iterare", mentre il secondo parametro sarà la variabile che dovrà, ad ogni iterazione, contenere il valore dell'elemento dell'array.

L'azione contenuta nelle parentesi graffe verrà ripetuta per quanti sono gli elementi dell'array.

Per estrarre gli elementi di un matrice occorre innestare un ciclo in un altro ciclo al fine di percorre ogni elemento di una riga iterando per tutte le righe della matrice:

Esempio

```
<?php
for ($row = 0; $row < 4; $row++) {
  echo "<p><b>Row number $row</b></p>";
  echo "<ul>";
  for ($col = 0; $col < 3; $col++) {
    echo "<li>".$cars[$row][$col]."</li>";
  }
  echo "</ul>";
}
?>
```

Contare gli elementi in un array

Supponiamo di avere un array di iscritti ad un evento e di volerne ottenere il numero, la funzione `count()` è esattamente quello di cui abbiamo bisogno:

```
$partecipanti = ['Simone', 'Gabriele', 'Renato', 'Giuseppe'];

echo "Ci saranno " . count($partecipanti) . " partecipanti all'evento";
```

L'esempio precedente restituirà:

```
Ci saranno 4 partecipanti all'evento
```

Verificare che un elemento sia contenuto in un array

Dato l'array precedente, supponiamo di voler verificare che l'utente `Simone` sia uno dei partecipanti, possiamo utilizzare la funzione `in_array()` che prende in ingresso due parametri:

1. il valore ricercato;
2. l'array in cui cercare.

   Vediamo un esempio:

```
$partecipanti = ['Simone', 'Gabriele', 'Renato', 'Giuseppe'];



if (in_array('Simone', $partecipanti)) {

    echo "Simone è uno dei partecipanti";

} else {

    echo "Simone non parteciperà all'evento";

}
```

   Nel nostro caso l'output sarà: *Simone è uno dei partecipanti*.

Mescolare gli elementi di un array

Nel caso in cui avessimo la necessità di mescolare gli elementi all'interno di un array, abbiamo a disposizione la funzione `shuffle()` che fa esattamente al caso nostro:

```
$partecipanti = ['Simone', 'Gabriele', 'Renato', 'Giuseppe'];

shuffle($partecipanti);



print_r($partecipanti);
```

L'output che avremo sarà simile al seguente, ovviamente non uguale perché il risultato è randomico:

```
Array

(

    [0] => Giuseppe

    [1] => Simone

    [2] => Renato
```

```
    [3] => Gabriele

)
```

Invertire l'ordine degli elementi

Un'altra funzione molto comune che possiamo applicare ad un array è quella di invertire l'ordine degli elementi. Vediamo un esempio con la funzione `array_reverse()`:

```
$partecipanti = ['Simone', 'Gabriele', 'Renato', 'Giuseppe'];

$ordine_inverso = array_reverse($partecipanti);



print_r($ordine_inverso);
```

Il risultato sarà:

```
Array

(

    [0] => Giuseppe

    [1] => Renato

    [2] => Gabriele

    [3] => Simone

)
```

Unire due array

Dati due o più array, attraverso la funzione `array_merge()` possiamo avere un unico array che contiene gli elementi di ognuno di essi. L'inserimento avviene in maniera da aggiungere gli elementi in coda all'array precedente. In caso di valori uguali l'ultimo valore andrà a sovrascrivere i precedenti.

Vediamo qualche esempio per chiarire i concetti:

```
$partecipanti_lista_A = ['Simone', 'Gabriele'];

$partecipanti_lista_B = ['Renato', 'Giuseppe'];

$partecipanti_lista_C = ['Maria', 'Daniela'];



$partecipanti = array_merge($partecipanti_lista_A, $partecipanti_lista_B,
$partecipanti_lista_C);



print_r($partecipanti);
```

La funzione `array_merge()` restituirà quindi il seguente risultato:

```
Array

(

    [0] => Simone

    [1] => Gabriele

    [2] => Renato

    [3] => Giuseppe

    [4] => Maria

    [5] => Daniela

)
```

Vediamo cosa succede se abbiamo array con valori uguali. Prendiamo come esempio array con chiavi alfanumeriche così da avere un esempio esplicito del funzionamento:

```
$partecipanti_lista_A = [

    'Simone' => [

        'anni' => 29,

        'citta' => 'Roma'

    ],

    'Gabriele' => [

        'anni' => 24,

        'citta' => 'Roma'
```

```
    ]
];


$partecipanti_lista_B = [
    'Renato' => [
        'anni' => 27,
        'citta' => 'Catania'
    ],
    'Simone' => [
        'anni' => 24,
        'citta' => 'Forlì'
    ]
];


$partecipanti = array_merge($partecipanti_lista_A,
$partecipanti_lista_B);


print_r($partecipanti);
```

Osservando con attenzione i due array noteremo sicuramente che ci sono due chiavi con valore `Simone`. In questo caso il risultato sarà un array di soli tre elementi e non quattro e il valore di quella chiave è l'ultimo in ordine di inserimento:

```
Array
(
    [Simone] => Array
        (
            [anni] => 24
            [citta] => Forlì
        )
```

```
    [Gabriele] => Array
        (
            [anni] => 24
            [citta] => Roma
        )

    [Renato] => Array
        (
            [anni] => 27
            [citta] => Catania
        )
)
```

Estrarre una porzione di un array

Nel caso in cui volessimo estrarre solo alcuni elementi da un array possiamo usare la funzione `array_slice()` che prende in ingresso tre parametri:

1. l'array;
2. l'offset, ovvero l'elemento da cui iniziare l'estrazione;
3. la lunghezza, ovvero il numero di elementi da estrarre a partire dall'offset.

Vediamo un esempio:

```
$array = [1, 2, 3, 4, 5, 6, 7, 8, 9];

$output = array_slice($array, 0, 3);

print_r($output);
```

Abbiamo scelto di estrarre 3 elementi a partire dal primo, il risultato sarà:

```
Array

(

    [0] => 1

    [1] => 2

    [2] => 3

)
```

Estrarre le chiavi o i valori da un array multidimensionale

Dato un array multidimensionale con coppie chiave/valori abbiamo a disposizione due funzioni, `array_keys()` e `array_values()`, che ci permettono rispettivamente di restituire un array di chiavi e di valori dell'array.

```
$array = [

    'Simone' => 29,

    'Marco' => 28,

    'Michele' => 35,

    'Luca' => 22

];



$nomi = array_keys($array);

$anni = array_values($array);



print_r($nomi);

print_r($anni);
```

L'output dei due array sarà:

```
Array
```

```
(
    [0] => Simone
    [1] => Marco
    [2] => Michele
    [3] => Luca
)


Array
(
    [0] => 29
    [1] => 28
    [2] => 35
    [3] => 22
)
```