|  |  |  |
| --- | --- | --- |
| Bảng phân chia công viêc | | |
| Họ tên người thực hiên | Nội dung thực hiên | Kết quả |
| Đinh Cống Phong | array\_unique , array\_unshift ,  array\_values , array\_walk\_recursive | Hoàn thành |
| Thẩm Kim Dũng | array\_walk , array, arsort , asort | Hoàn thành |
| Lương Chinh | Compact, count, current, each | Hoàn thành |
| Nguyễn Thị Ngoan | End, extract , in\_array | Hoàn thành |

Nhóm 4: Tìm hiểu cú pháp, mô tả, tham số và ví dụ về 15 hàm trong array

1. **array\_unique**

*Cú pháp:*

array **array\_unique** ( array $array [, int $sort\_flags = SORT\_STRING ] )

*Mô tả:*

Lấy một mảng đầu vào và trả về một mảng mới mà không có giá trị bản sao.  
Chú ý: các từ khóa thì được tồn tại. Array\_unique () sẽ sắp xếp các giá trị điều trị như chuỗi lúc đầu, sau đó sẽ giữ từ khóa đầu tiên gặp phải cho mỗi giá trị, và bỏ qua tất cả các từ khóa sau đây. Nó không có nghĩa rằng từ khóa của các giá trị liên quan đầu tiên của mảng được phân loại sẽ được lưu giữ.

*Tham số:*

* Các mảng đầu vào.
* Sort\_flags(Cờ loại): Các sort\_flags tham số tùy chọn thứ hai có thể được sử dụng để sửa đổi hành vi phân loại sử dụng các giá trị.
* Phân loại kiểu cờ:
* SORT\_REGULAR - so sánh các mặt hàng bình thường (không thay đổi các loại)
* SORT\_NUMERIC - so sánh số lượng các mặt hang
* SORT\_STRING - so sánh các mặt hàng như các chuỗi
* SORT\_LOCALE\_STRING - so sánh các mặt hàng như các chuỗi, dựa trên ngôn ngữ hiện tại.

*Giá trị trả về:*

Trả về array filtered(mảng lọc).

*Chú ý:* Array\_unique () không có mục đích làm việc trên mảng đa chiều.

*Ví dụ:*

<html>

<head>

<meta charset="UTF-8">

<title></title>

</head>

<body>

<?php

$input = array(4, "4", "3", 4, 3, "3");

$result = array\_unique($input);

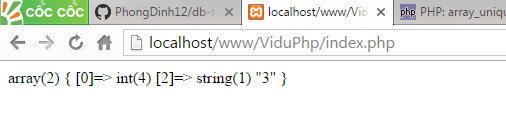
var\_dump($result);

?>

</body>

</html>

*KQ:*



1. **array\_unshift**

*Cú pháp:*

int **array\_unshift** ( array &$array , [mixed](http://php.net/manual/en/language.pseudo-types.php#language.types.mixed) $value1 [, [mixed](http://php.net/manual/en/language.pseudo-types.php#language.types.mixed) $... ] )

*Mô tả:*

Array\_unshift () thêm vào trước thông qua các yếu tố vào phía trước của mảng. Lưu ý rằng danh sách các yếu tố được thêm vào phía trước như một tổng thể, để các yếu tố được thêm vào trước ở trong cùng một trật tự. Tất cả các khóa mảng số sẽ được sửa đổi để bắt đầu đếm từ con số không trong khi các khóa bình thường sẽ không được thay đổi.

*Tham số:*

Các mảng đầu vào và giá trị đầu tiên để thêm vào trước.

*Giá trị trả về:*

Trả về số mới của các yếu tố trong mảng.

*Ví Dụ:*

<html>

<head>

<meta charset="UTF-8">

<title></title>

</head>

<body>

<?php

$queue = array("orange", "banana");

array\_unshift($queue, "apple", "raspberry");

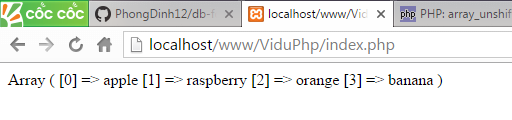
print\_r($queue);

?>

</body>

</html>

*KQ*



1. **array\_values**

*Cú pháp:*

array **array\_values** ( array $array )

*Mô tả:*

Array\_values () trả về tất cả các giá trị từ mảng và chỉ số mảng số.

*Tham số:*

Các mảng.

*Giá trị trả về:*

Trả về một mảng đánh chỉ mục của các giá trị.

*Ví Dụ:*

<html>

<head>

<meta charset="UTF-8">

<title></title>

</head>

<body>

<?php

$array = array("size" => "XL", "color" => "gold");

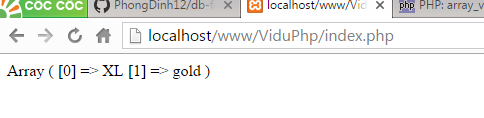
print\_r(array\_values($array));

?>

</body>

</html>

*KQ:*



1. **array\_** **walk\_recursive**

*Cú pháp:*

bool **array\_walk\_recursive** ( array &$array , [callable](http://php.net/manual/en/language.types.callable.php) $callback [, [mixed](http://php.net/manual/en/language.pseudo-types.php#language.types.mixed) $userdata = NULL ] )

*Mô tả:*

Áp dụng các hàm callback(gọi lại) người dùng định nghĩa để mỗi phần tử của mảng. Chức năng này sẽ được đệ quy vào mảng sâu hơn.

*Tham số:*

* *Array:* Các mảng đầu vào.
* *Callback:* Thông thường, gọi lại có trên hai tham số. Giá trị các tham số của mảng là vị trí đầu tiên, và các khóa / mục thứ hai.

Chú thích: Nếu gọi lại cần được làm việc với các giá trị thực tế của mảng, chỉ định các tham số đầu tiên của cuộc gọi lại như một tài liệu tham khảo. Sau đó, bất kỳ thay đổi nào trong những yếu tố sẽ được thực hiện trong các mảng ban đầu của chính nó.

* *Usedata:* Nếu tham số userdata tùy chọn được cung cấp, nó sẽ được thông qua như là tham số thứ ba để gọi lại.

**Giá trị trả về:**

Trả về TRUE nếu thành công hoặc FALSE nếu thất bại.

**Ví Dụ:**

<html>

<head>

<meta charset="UTF-8">

<title></title>

</head>

<body>

<?php

$sweet = array('a' => 'apple', 'b' => 'banana');

$fruits = array('sweet' => $sweet, 'sour' => 'lemon');

function test\_print($item, $key)

{

echo "$key holds $item\n<br>";

}

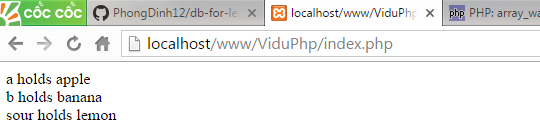
array\_walk\_recursive($fruits, 'test\_print');

?>

</body>

</html>

**KQ:**



1. **array\_walk():**

*môt tả*: áp dụng hàm do người dùng tạo tới mỗi phần tử của mảng

*cú pháp*: array\_walk ( $array, $funcname [, $parameter] );

*tham số*:

* + $array bắt buộc xác định mảng đầu vào
  + $funcname bắt buộc, đây là hàm được user tạo
  + $parameter tham số tùy ý

*Ví dụ*:

<?php

function func($value,$key){

echo "Key $key có giá trị $value <br>";

}

$arr=['one'=>1,'two'=>2,'three'=>3];

array\_walk($arr, "func");

?>

*Kết quả*:

Key one có giá trị 1  
Key two có giá trị 2  
Key three có giá trị 3

1. **arsort ()**

*Cú pháp*: arsort($array [, $sort\_flag]);

*Tham số*:

* + $array bắt buộc, là tên mảng
  + $sort\_flag tùy ý, xác định cách sắp xếp, mặc định là SORT\_REGULAR

*Ví dụ:*

<?php

$arr=['orange'=>'cam','apple'=>'táo','banana'=>'chuối'];

arsort($arr);

print\_r($arr);

?>

*Kết quả:*

Array ( [apple] => táo [banana] => chuối [orange] => cam )

Hàm này sắp xếp 1 mảng theo cách mà các chỉ mục mảng duy trì mối liên kết với các phần tử mảng mà chúng đã liên kết. Hàm được sử dụng khi sắp xếp mảng liên hợp mà thứ tự phần tử thực sự quan trọng.

1. **create array:**

*Mô tả*: dùng để tạo mảng

*Cú pháp* :

các cách tạo mảng:

$arr=array(1,2,3,4);

$arr=[1,2,3,4];

mảng kết hợp $arr=['one'=>1,'two'=>2,'three'=>3];

*ví dụ:*

<?php  
  $a = ['one' => 1, 'two' => 2, 'three' => 3, 'four' => 4];  
     print\_r($a);  
?>

*Kết quả*

Array  
(  
    [one] => 1  
    [two] => 2  
    [three] => 3  
    [four] => 4  
)

1. **Asort()**

*Mô tả*: Sắp xếp mảng liên hợp theo thứ tự tăng

*Cú pháp*: tương tự arsort, asort($array [, $sort\_flag]);

SORT\_REGULAR – mặc định

SORT\_NUMERIC

SORT\_STRING

SORT\_LOCALE\_STRING

SORT\_NATURAL

*Ví dụ*:

<?php  
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");  
asort($age);  
  
foreach($age as $x=>$x\_value)  
   {  
   echo "Key=" . $x . ", Value=" . $x\_value;  
   echo "<br>";  
   }  
?>

*Kết quả*:

Key=Peter, Value=35  
Key=Ben, Value=37  
Key=Joe, Value=43

1. **Compact**

*Cú pháp*: array compact( $var1, $var2,…);

*Mô tả*: Hàm này nhận các biến làm các tham số. Mỗi tham số có thể là chuỗi chứa tên biến, hoặc một mảng các biến. Mảng này có thể chứa mảng biến khác bên trong nó, hàm compact() này xử lý mảng kiểu này một cách đệ qui.

*Tham số*: Có thể là các tên biến hoặc mảng các biến.

*Giá trị trả về*: Trả về mảng kết quả với các biến đã thêm vào.

*Ví dụ*:

<?php

$city = "San Francisco";

$state = "CA";

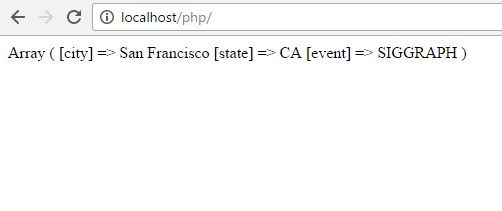
$event = "SIGGRAPH";

$result = compact("city", "state", "event");

print\_r($result);

?>

Kết quả



1. **Count**

*Cú pháp*: int count( $array\_or\_countable, $mode )

*Mô tả*: Hàm count đếm số phần tử trong mảng hoặc thuộc tính của một đối tượng. Tham số $mode có thể tùy chọn là COUNT\_RECURSIVE( hoặc 1) thì hàm sẽ đếm một cách đệ quy hoặc COUNT\_NORMAL (hoặc 0) hàm sẽ không đếm đệ quy.

*Tham số*: Tham số 1: Một mảng hoặc một đối tượng có thể đếm được. Tham số 2 là giá trị mode.

*Giá trị trả về*: Số phần tử trong mảng.

*Ví dụ*:

<?php

$a[0] = 1;

$a[1] = 3;

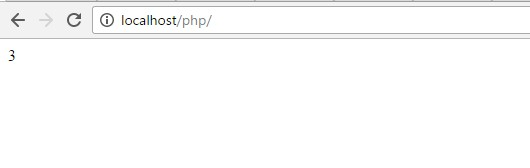
$a[2] = 5;

$result = count($a);

print($result);

?>

*Kết quả:*



1. **Current**

*Cú pháp*: current( $array );

*Mô tả*: Mỗi mảng đều có một con trỏ trỏ tới phần tử hiện tại của nó. Hàm current trả về value của phần tử mảng mà con trỏ đang trỏ đến. Hàm này không di chuyển con trỏ và sẽ trả về false nếu con trỏ đang trỏ ngoài phần tử cuối cùng trong mảng.

*Tham số*: Array- mảng đầu vào.

*Giá trị trả về*: Giá trị phần tử hiện tại trong mảng.

*Ví dụ:*

<?php

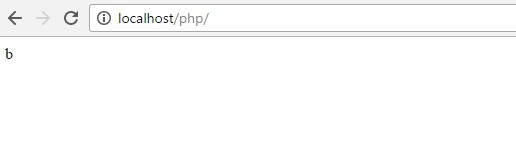
$myarray = array(0=>'a', 1=>'b', 2=>'c');

next($myarray);

print\_r(current($myarray));

?>

*Kết quả:*



1. **Each**

*Cú pháp*: array each( $array );

*Mô tả*: Hàm each() trả về cặp key và value hiện tại trong mảng và tăng con trỏ mảng. Cặp này được trả về trong mảng 4 phần tử với các key là: 0, 1, key, value. Trong đó 0 và key chứa tên key, 1 và value chứa dữ liệu.

*Tham số*: Array: mảng đầu vào.

*Giá trị trả về*: Trả về cặp key và value hiện tại trong mảng.

*Ví dụ*:

<?php

$transport = array('foot', 'bike', 'car', 'plane');

$key\_value = each($transport);

print\_r($key\_value);

print "<br />";

$key\_value = each($transport);

print\_r($key\_value);

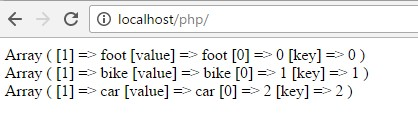
print "<br />";

$key\_value = each($transport);

print\_r($key\_value);

?>

*Kết quả:*



1. **End**

*Cú pháp:* end($array);

*Mô tả:* Trả về phần tử cuối cùng của mảng, nếu mảng rỗng trả về giá trị False

*Tham số*: array-mảng

*ví dụ*

*<?php*

$dongvat=array(‘cho’,’mèo’,’gà’, ‘vit’);

Echo end($dongvat);

?>

Kết quả:

vịt

1. **extract**

Cú pháp: extract(Sarray, $extract\_type, $prefix)

*Tham số*:

* + $array là tên mảng
  + $extract type: kiểm tra các tên biến không hợp lệ và xung đột với các tên biến hiện có. Tham số này xác định cách đối xử với các biến không hợp lệ và các biến xung đột.
  + Phân loại extract\_type:
* EXTR\_OVERWRITE − Mặc định. Nếu xung đột, biến hiện tại sẽ bị ghi đè
* EXTR\_SKIP − Nếu xung đột, biến hiện tại sẽ không bị ghi đè
* EXTR\_PREFIX\_SAME − Nếu xung đột, biến hiện tại sẽ được cung cấp một prefix
* EXTR\_PREFIX\_ALL − Tất cả tên biến sẽ được cung cấp một prefix
* EXTR\_PREFIX\_INVALID − Chỉ các tên biến không hợp lệ hoặc tên biến giá trị số sẽ được cung cấp một prefix
* EXTR\_IF\_EXISTS − Chỉ ghi đè các biên đang tồn tại trong current symbol table, nếu không thì không làm gì cả
* EXTR\_PREFIX\_IF\_EXISTS − Chỉ thêm prefix tới các biến nếu biến giống nhau tồn tại trong current symbol table
* EXTR\_REFS − Trích các biến như là tham chiếu. Các biến đã được import sẽ đang tham chiếu tới value của tham số mảng

*Ví dụ:*

<?php

$size=”large”;

$var\_array=array(“color”=>”blue”,”size”=>”medium”,”shape”=>”sphere”);

extract($var\_array,EXTR\_PREFIX\_SAME,”wddx”);

echo ”$color, $size, $shape, $wddx\_size”;

?>

ở đây $size=”large” bị xung đột với phần tử mảng “size”=medium, và ghi đè phần tử $wddx\_size

nếu chọn EXTR\_PREFIX\_ALL thì sẽ có biến $wddx\_color, $wddx\_size, $wddx\_sphere được tạo ra

*Kết quả*:

blue,large,sphere,medium

## in\_array

## *Cú pháp:* in\_array($value, $array[,strict])

*Mô tả:* kiểm tra một giá trị $value có tồn tại trong $array không.Nếu tham số thứ ba **strict** được thiết lập là TRUE, thì hàm in\_array() cũng sẽ kiểm tra kiểu của $value. Hàm này trả về TRUE nếu value được tìm thấy trong mảng, nếu không là FALSE.

*Ví dụ*:

<?php

$os = array("Mac", "NT", "Irix", "Linux");

if (in\_array("Irix", $os)) {

echo "Tìm thấy giá trị Irix";

}

if (in\_array("mac", $os)) {

echo "Tìm thấy giá trị mac";

}

?>

Chạy theo thứ tự tìm thấy kết quả đúng thì trả về kết quả và dừng:

*Kết quả:* Tìm thấy giá trị Iric