1. Mảng để lưu trữ bộ các giá trị, bằng cách ấy có thể nhóm các biến vô hướng lại.

* Để khởi tạo mảng trong php ta sử dụng cú pháp sau  
  **$ten\_bien = array("gia\_tri\_1","gia\_tri\_2","gia\_tri\_3",...,"gia\_tri\_n");**
* Cú pháp
* Cú pháp cho mảng lập chỉ mục:

array(*value1,value2,value3,etc.*);

* Cú pháp cho mảng kết hợp:

array(*key=>value,key=>value,key=>value,etc.*);

ví dụ

* <?php  
  $age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");  
  echo "Peter is " . $age['Peter'] . " years old.";  
  ?
* Ví dụ 4

Tạo một mảng đa chiều:

<?php  
// A two-dimensional array:  
$cars=array  
  (  
  array("Volvo",100,96),  
  array("BMW",60,59),  
  array("Toyota",110,100)  
  );  
?>

* **Ví dụ:**  
  <?php

$sinhvien = array('Mai Thảo’, '24' );

?>

1. Cú pháp lặp giá trị trong mảng

foreach($array as $temp)

{ Hành Động }

Trong đó $array là mảng mà ta muốn thực thi việc lặp dữ liệu, và $temp là một biến trong đó ta sẽ tạm thời lưu trữ mỗi phần tử.

Ví dụ:

<?php

$name= array("Kenny", "Jiro", "Calvin", "Gillian");

foreach ($name as $test)

{ echo "$test<br>"; }

?>

* array\_change\_key\_case: chức năng thay đổi tất cả các phím trong một mảng chữ thường hay chữ hoa.

Ví dụ: Thay đổi tất cả các phím trong một mảng chữ thường:

<?php  
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");  
print\_r(array\_change\_key\_case($age,CASE\_LOWER));  
?>

* array\_chunk () chức năng chia một mảng vào khối của mảng mới.

ví dụ: Chia nhỏ một mảng vào khối của hai:

<?php  
$cars=array("Volvo","BMW","Toyota","Honda","Mercedes","Opel");  
print\_r(array\_chunk($cars,2));  
?>

* array\_column () trả về giá trị từ một cột duy nhất trong mảng đầu vào

vd:

Nhận cột tên cuối cùng từ một recordset:

<?php  
// An array that represents a possible record set returned from a database  
$a = array(  
  array(  
    'id' => 5698,  
    'first\_name' => 'Peter',  
    'last\_name' => 'Griffin',  
  ),  
  array(  
    'id' => 4767,  
    'first\_name' => 'Ben',  
    'last\_name' => 'Smith',  
  ),  
  array(  
    'id' => 3809,  
    'first\_name' => 'Joe',  
    'last\_name' => 'Doe',  
  )  
);  
  
$last\_names = array\_column($a, 'last\_name');  
print\_r($last\_names);  
?>

Đầu ra:

Array  
(  
  [0] => Griffin  
  [1] => Smith  
  [2] => Doe  
)

* array\_combine () chức năng tạo ra một mảng bằng cách sử dụng các yếu tố từ một "chìa khóa" mảng và một mảng "giá trị"

vd:

Tạo một mảng bằng cách sử dụng các yếu tố từ một "chìa khóa" mảng và một mảng "giá trị":

<?php  
$fname=array("Peter","Ben","Joe");  
$age=array("35","37","43");  
  
$c=array\_combine($fname,$age);  
print\_r($c);  
?>

* array\_count\_values ​​() chức năng đếm tất cả các giá trị của một mảng

vd:

Đếm tất cả các giá trị của mảng:

<?php  
$a=array("A","Cat","Dog","A","Dog");  
print\_r(array\_count\_values($a));  
?>

* array\_diff\_assoc ()Tính toán sự khác biệt của các mảng với chỉ số kiểm tra thêm

vd:

<?php  
$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");  
$a2=array("a"=>"red","b"=>"green","c"=>"blue");  
  
$result=array\_diff\_assoc($a1,$a2);  
print\_r($result);  
?>

* array\_diff\_key Tính toán sự khác biệt của mảng bằng cách sử dụng các phím để so sánh

vd:

<?php  
$a1=array("a"=>"red","b"=>"green","c"=>"blue");  
$a2=array("a"=>"red","c"=>"blue","d"=>"pink");  
  
$result=array\_diff\_key($a1,$a2);  
print\_r($result);  
?>

* array\_diff\_uassoc - Tính toán sự khác biệt của các mảng với thêm kiểm tra chỉ số đó được thực hiện bởi một người dùng cung cấp chức năng gọi lại

vd:

<?php  
function myfunction($a,$b)  
{  
if ($a===$b)  
  {  
  return 0;  
  }  
  return ($a>$b)?1:-1;  
}  
  
$a1=array("a"=>"red","b"=>"green","c"=>"blue");  
$a2=array("d"=>"red","b"=>"green","e"=>"blue");  
  
$result=array\_diff\_uassoc($a1,$a2,"myfunction");  
print\_r($result);  
?>

* array\_diff\_ukey - Tính toán sự khác biệt của mảng bằng cách sử dụng một chức năng gọi lại trên các phím để so sánh

vd:

<?php  
function myfunction($a,$b)  
{  
if ($a===$b)  
  {  
  return 0;  
  }  
  return ($a>$b)?1:-1;  
}  
  
$a1=array("a"=>"red","b"=>"green","c"=>"blue");  
$a2=array("a"=>"blue","b"=>"black","e"=>"blue");  
  
$result=array\_diff\_ukey($a1,$a2,"myfunction");  
print\_r($result);  
?>

* array\_diff - Tính toán sự khác biệt của các mảng

vd:

<?php  
$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");  
$a2=array("e"=>"red","f"=>"green","g"=>"blue");  
  
$result=array\_diff($a1,$a2);  
print\_r($result);  
?>

* array\_fill\_keys - Điền một mảng với giá trị quy định cụ thể các phím

vd

<?php  
$keys=array("a","b","c","d");  
$a1=array\_fill\_keys($keys,"blue");  
print\_r($a1);  
?>

* array\_fill - Điền một mảng với giá trị

vd

<?php  
$a1=array\_fill(3,4,"blue");  
print\_r($a1);  
?>

* array\_filter - Bộ lọc các yếu tố của một mảng bằng cách sử dụng một chức năng gọi lại

vd

<?php  
function test\_odd($var)  
{  
return($var & 1);  
}  
  
$a1=array("a","b",2,3,4);  
print\_r(array\_filter($a1,"test\_odd"));  
?>

* array\_flip () chức năng làm bật / trao đổi tất cả các phím với các giá trị liên quan của họ trong một mảng.

vd

<?php  
$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");  
$result=array\_flip($a1);  
print\_r($result);  
?>

* array\_intersect\_assoc - Tính toán các giao điểm của mảng có thêm kiểm tra chỉ số

vd;

<?php  
$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");  
$a2=array("a"=>"red","b"=>"green","c"=>"blue");  
  
$result=array\_intersect\_assoc($a1,$a2);  
print\_r($result);  
?>

* array\_intersect\_key - Tính toán các giao điểm của mảng bằng cách sử dụng các phím để so sánh

vd:

<?php  
$a1=array("a"=>"red","b"=>"green","c"=>"blue");  
$a2=array("a"=>"red","c"=>"blue","d"=>"pink");  
  
$result=array\_intersect\_key($a1,$a2);  
print\_r($result);  
?>

* array\_intersect\_uassoc - Tính toán các giao điểm của các mảng với chỉ số kiểm tra thêm, so sánh các chỉ số của một hàm callback

vd

<?php  
function myfunction($a,$b)  
{  
if ($a===$b)  
  {  
  return 0;  
  }  
  return ($a>$b)?1:-1;  
}  
  
$a1=array("a"=>"red","b"=>"green","c"=>"blue");  
$a2=array("d"=>"red","b"=>"green","e"=>"blue");  
  
$result=array\_intersect\_uassoc($a1,$a2,"myfunction");  
print\_r($result);  
?>

* array\_intersect\_ukey - Tính toán các giao điểm của mảng bằng cách sử dụng một chức năng gọi lại trên các phím để so sánh

vd

<?php  
function myfunction($a,$b)  
{  
if ($a===$b)  
  {  
  return 0;  
  }  
  return ($a>$b)?1:-1;  
}  
  
$a1=array("a"=>"red","b"=>"green","c"=>"blue");  
$a2=array("a"=>"blue","b"=>"black","e"=>"blue");  
  
$result=array\_intersect\_ukey($a1,$a2,"myfunction");  
print\_r($result);  
?>

* array\_intersect - Tính toán các giao điểm của mảng

vd

<?php  
$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");  
$a2=array("e"=>"red","f"=>"green","g"=>"blue");  
  
$result=array\_intersect($a1,$a2);  
print\_r($result);  
?>

* array\_key\_exists - Kiểm tra nếu các phím được hoặc chỉ tồn tại trong mảng

vd

<?php  
$a=array("Volvo"=>"XC90","BMW"=>"X5");  
if (array\_key\_exists("Volvo",$a))  
  {  
  echo "Key exists!";  
  }  
else  
  {  
  echo "Key does not exist!";  
  }  
?>

* array\_keys – trả về tất cả các phím hoặc một tập hợp con của các phím của một mảng

vd

<?php  
$a=array("Volvo"=>"XC90","BMW"=>"X5","Toyota"=>"Highlander");  
print\_r(array\_keys($a));  
?>

* array\_map - Áp dụng gọi lại để các phần tử của mảng đã cho

vd

<?php  
function myfunction($v)  
{  
  return($v\*$v);  
}  
  
$a=array(1,2,3,4,5);  
print\_r(array\_map("myfunction",$a));  
?>

* array\_merge - Hợp nhất một hoặc nhiều mảng

vd

<?php  
$a1=array("red","green");  
$a2=array("blue","yellow");  
print\_r(array\_merge($a1,$a2));  
?>

* array\_merge\_recursive - Hợp nhất hai hay nhiều mảng đệ quy

vd

<?php  
$a1=array("a"=>"red","b"=>"green");  
$a2=array("c"=>"blue","b"=>"yellow");  
print\_r(array\_merge\_recursive($a1,$a2));  
?>

* array\_multisort - Sắp xếp nhiều hoặc đa chiều mảng

vd

<?php  
$a=array("Dog","Cat","Horse","Bear","Zebra");  
array\_multisort($a);  
print\_r($a);  
?>

* array\_pad - Pad mảng với chiều dài chỉ định với một giá trị

vd

<?php  
$a=array("red","green");  
print\_r(array\_pad($a,5,"blue"));  
?

* array\_pop () chức năng xóa các phần tử cuối cùng của mảng.

vd

<?php  
$a=array("red","green","blue");  
array\_pop($a);  
print\_r($a);  
?>

* array\_product () chức năng tính toán và trả về các sản phẩm của một mảng.

Vd

<?php  
$a=array(5,5);  
echo(array\_product($a));  
?

* array\_push () chức năng chèn một hoặc nhiều phần tử vào cuối mảng.

vd

<?php  
$a=array("red","green");  
array\_push($a,"blue","yellow");  
print\_r($a);  
?>

* array\_rand - Chọn một hoặc nhiều mục ngẫu nhiên trong một mảng

vd

<?php  
$a=array("red","green","blue","yellow","brown");  
$random\_keys=array\_rand($a,3);  
echo $a[$random\_keys[0]]."<br>";  
echo $a[$random\_keys[1]]."<br>";  
echo $a[$random\_keys[2]];  
?>

* array\_reduce - lặp đi lặp lại làm giảm mảng tới một giá trị duy nhất bằng cách sử dụng một chức năng gọi lại

vd

<?php  
function myfunction($v1,$v2)  
{  
return $v1 . "-" . $v2;  
}  
$a=array("Dog","Cat","Horse");  
print\_r(array\_reduce($a,"myfunction"));  
?

* array\_replace\_recursive () chức năng thay thế các giá trị của mảng thứ nhất với các giá trị sau đây mảng đệ quy.

Vd

<?php  
$a1=array("a"=>array("red"),"b"=>array("green","blue"),);  
$a2=array("a"=>array("yellow"),"b"=>array("black"));  
print\_r(array\_replace\_recursive($a1,$a2));  
?>

* array\_replace - Thay thế các phần tử từ mảng thông qua vào mảng đầu tiên

vd

<?php  
$a1=array("red","green");  
$a2=array("blue","yellow");  
print\_r(array\_replace($a1,$a2));  
?>

* array\_reverse () trả về một mảng theo thứ tự ngược lại.

vd

<?php  
$a=array("a"=>"Volvo","b"=>"BMW","c"=>"Toyota");  
print\_r(array\_reverse($a));  
?>

* array\_search - tìm kiếm các mảng cho một giá trị nhất định và trả về các phím tương ứng nếu thành công

vd

<?php  
$a=array("a"=>"red","b"=>"green","c"=>"blue");  
echo array\_search("red",$a);  
?>

* array\_shift () chức năng loại bỏ các yếu tố đầu tiên từ một mảng,

Vd

<?php  
$a=array("a"=>"red","b"=>"green","c"=>"blue");  
echo array\_shift($a);  
print\_r ($a);  
?>

* array\_slice () trả về chức năng lựa chọn các bộ phận của một mảng.

vd

<?php  
$a=array("red","green","blue","yellow","brown");  
print\_r(array\_slice($a,2));  
?>

* array\_splice () chức năng loại bỏ các yếu tố được lựa chọn từ một mảng và thay thế nó bằng các yếu tố mới.

vd

<?php  
$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");  
$a2=array("a"=>"purple","b"=>"orange");  
array\_splice($a1,0,2,$a2);  
print\_r($a1);  
?>

* array\_sum () trả về tổng của tất cả các giá trị trong mảng.

vd

<?php  
$a=array(5,15,25);  
echo array\_sum($a);  
?>

* array\_udiff\_assoc - Tính toán sự khác biệt của các mảng với chỉ số kiểm tra thêm, so sánh dữ liệu của một chức năng gọi lại

vd

<?php  
function myfunction($a,$b)  
{  
if ($a===$b)  
  {  
  return 0;  
  }  
  return ($a>$b)?1:-1;  
}  
  
$a1=array("a"=>"red","b"=>"green","c"=>"blue");  
$a2=array("a"=>"red","b"=>"blue","c"=>"green");  
  
$result=array\_udiff\_assoc($a1,$a2,"myfunction");  
print\_r($result);  
?>

* array\_udiff\_uassoc - Tính toán sự khác biệt của các mảng với chỉ số kiểm tra thêm, so sánh dữ liệu và lập chỉ mục bởi một chức năng gọi lại

vd

<?php  
function myfunction\_key($a,$b)  
{  
if ($a===$b)  
  {  
  return 0;  
  }  
  return ($a>$b)?1:-1;  
}  
  
function myfunction\_value($a,$b)  
{  
if ($a===$b)  
  {  
  return 0;  
  }  
  return ($a>$b)?1:-1;  
}  
  
$a1=array("a"=>"red","b"=>"green","c"=>"blue");  
$a2=array("a"=>"red","b"=>"green","c"=>"green");  
  
$result=array\_udiff\_uassoc($a1,$a2,"myfunction\_key","myfunction\_value");  
print\_r($result);  
?>

* array\_udiff - Tính toán sự khác biệt của mảng bằng cách sử dụng một chức năng gọi lại để so sánh dữ liệu

vd

<?php  
function myfunction($a,$b)  
{  
if ($a===$b)  
  {  
  return 0;  
  }  
  return ($a>$b)?1:-1;  
}  
  
$a1=array("a"=>"red","b"=>"green","c"=>"blue");  
$a2=array("a"=>"blue","b"=>"black","e"=>"blue");  
  
$result=array\_udiff($a1,$a2,"myfunction");  
print\_r($result);  
?>

* array\_uintersect\_assoc - Tính toán các giao điểm của các mảng với chỉ số kiểm tra thêm, so sánh dữ liệu của một chức năng gọi lại

vd

<?php  
function myfunction($a,$b)  
{  
if ($a===$b)  
  {  
  return 0;  
  }  
  return ($a>$b)?1:-1;  
}  
  
$a1=array("a"=>"red","b"=>"green","c"=>"blue");  
$a2=array("a"=>"red","b"=>"blue","c"=>"green");  
  
$result=array\_uintersect\_assoc($a1,$a2,"myfunction");  
print\_r($result);  
?>

* array\_uintersect\_uassoc - Tính toán các giao điểm của các mảng với chỉ số kiểm tra thêm, so sánh dữ liệu và lập chỉ mục bởi các hàm callback riêng

vd

<?php  
function myfunction\_key($a,$b)  
{  
if ($a===$b)  
  {  
  return 0;  
  }  
  return ($a>$b)?1:-1;  
}  
  
function myfunction\_value($a,$b)  
{  
if ($a===$b)  
  {  
  return 0;  
  }  
  return ($a>$b)?1:-1;  
}  
  
$a1=array("a"=>"red","b"=>"green","c"=>"blue");  
$a2=array("a"=>"red","b"=>"green","c"=>"green");  
  
$result=array\_uintersect\_uassoc($a1,$a2,"myfunction\_key","myfunction\_value");  
print\_r($result);  
?>

* array\_uintersect - Tính toán các giao điểm của các mảng, so sánh dữ liệu của một chức năng gọi lại

vd

<?php  
function myfunction($a,$b)  
{  
if ($a===$b)  
  {  
  return 0;  
  }  
  return ($a>$b)?1:-1;  
}  
  
$a1=array("a"=>"red","b"=>"green","c"=>"blue");  
$a2=array("a"=>"blue","b"=>"black","e"=>"blue");  
  
$result=array\_uintersect($a1,$a2,"myfunction");  
print\_r($result);  
?>

* array\_unique - Loại bỏ các giá trị nhân bản từ một mảng

vd

<?php  
function myfunction($a,$b)  
{  
if ($a===$b)  
  {  
  return 0;  
  }  
  return ($a>$b)?1:-1;  
}  
  
$a1=array("a"=>"red","b"=>"green","c"=>"blue");  
$a2=array("a"=>"blue","b"=>"black","e"=>"blue");  
  
$result=array\_uintersect($a1,$a2,"myfunction");  
print\_r($result);  
?>

* array\_unshift - thêm vào trước một hoặc nhiều yếu tố để khởi đầu của một mảng

vd

<?php  
$a=array("a"=>"red","b"=>"green");  
array\_unshift($a,"blue");  
print\_r($a);  
?>

* array\_values ​​() trả về một mảng chứa tất cả các giá trị của một mảng.

vd

<?php  
$a=array("Name"=>"Peter","Age"=>"41","Country"=>"USA");  
print\_r(array\_values($a));  
?>

* array\_walk\_recursive - Áp dụng một chức năng sử dụng đệ quy để mọi thành viên của một mảng

vd

<?php  
function myfunction($value,$key)  
{  
echo "The key $key has the value $value<br>";  
}  
$a1=array("a"=>"red","b"=>"green");  
$a2=array($a1,"1"=>"blue","2"=>"yellow");  
array\_walk\_recursive($a2,"myfunction");  
?>

* array\_walk - Áp dụng một người sử dụng chức năng cung cấp cho mỗi thành viên của một mảng

vd

<?php  
function myfunction($value,$key)  
{  
echo "The key $key has the value $value<br>";  
}  
$a=array("a"=>"red","b"=>"green","c"=>"blue");  
array\_walk($a,"myfunction");  
?>

* array - Tạo một mảng

vd

<?php  
$cars=array("Volvo","BMW","Toyota");  
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";  
?>

* arsort () chức năng sắp xếp một mảng kết hợp trong thứ tự giảm dần, theo giá trị.

Vd

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

* asort () chức năng sắp xếp một mảng kết hợp trong thứ tự tăng dần, theo giá trị.

Vd

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

* compact () chức năng tạo ra một mảng từ biến và giá trị của họ.

vd

<?php  
$firstname = "Peter";  
$lastname = "Griffin";  
$age = "41";  
  
$result = compact("firstname", "lastname", "age");  
  
print\_r($result);  
?>

* count () trả về số phần tử trong một mảng.

vd

<?php  
$cars=array("Volvo","BMW","Toyota");  
echo count($cars);  
?>

* current Trở về các yếu tố hiện trong một mảng

vd

<?php  
$people = array("Peter", "Joe", "Glenn", "Cleveland");  
  
echo current($people) . "<br>";  
?>

* each () trả về key phần tử hiện tại và giá trị, và di chuyển con trỏ nội bộ về phía trước.

vd

<?php  
$people = array("Peter", "Joe", "Glenn", "Cleveland");  
print\_r (each($people));  
?>

* end  chức năng di chuyển con trỏ bên trong để, và kết quả đầu ra, các yếu tố cuối cùng trong mảng

vd

<?php  
$people = array("Peter", "Joe", "Glenn", "Cleveland");  
  
echo current($people) . "<br>";  
echo end($people);   
?>

* extract nhập khẩu biến thành các bảng biểu tượng hiện tại từ một mảng

vd

<?php  
$a = "Original";  
$my\_array = array("a" => "Cat","b" => "Dog", "c" => "Horse");  
extract($my\_array);  
echo "\$a = $a; \$b = $b; \$c = $c";  
?>

* in\_array - Kiểm tra một giá trị tồn tại trong một mảng

vd

<?php  
$people = array("Peter", "Joe", "Glenn", "Cleveland");  
  
if (in\_array("Glenn", $people))  
  {  
  echo "Match found";  
  }  
else  
  {  
  echo "Match not found";  
  }  
?>

* key\_exists - Bí danh của array\_key\_exists
* key : trả về key phần tử từ vị trí con trỏ nội bộ hiện hành.

Vd

<?php  
$people=array("Peter","Joe","Glenn","Cleveland");  
echo "The key from the current position is: " . key($people);  
?>

* krsort - Sắp xếp một mảng bằng phím theo thứ tự ngược

vd

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

* ksort () chức năng sắp xếp một mảng kết hợp trong thứ tự tăng dần, theo chìa khóa.

Vd

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

* list- Gán biến như thể họ là một mảng

vd

<?php  
$my\_array = array("Dog","Cat","Horse");  
  
list($a, $b, $c) = $my\_array;  
echo "I have several animals, a $a, a $b and a $c.";  
?>

* natcasesort - Sắp xếp một mảng bằng cách sử dụng một trường hợp nhạy cảm "trật tự tự nhiên" thuật toán

vd

<?php  
$temp\_files = array("temp15.txt","Temp10.txt",  
"temp1.txt","Temp22.txt","temp2.txt");  
  
natsort($temp\_files);  
echo "Natural order: ";  
print\_r($temp\_files);  
echo "<br />";  
  
natcasesort($temp\_files);  
echo "Natural order case insensitve: ";  
print\_r($temp\_files);  
?>

* natsort - Sắp xếp một mảng bằng cách sử dụng một "trật tự tự nhiên" thuật toán

vd

<?php  
$temp\_files = array("temp15.txt","temp10.txt",  
"temp1.txt","temp22.txt","temp2.txt");  
  
sort($temp\_files);  
echo "Standard sorting: ";  
print\_r($temp\_files);  
echo "<br>";  
  
natsort($temp\_files);  
echo "Natural order: ";  
print\_r($temp\_files);  
?>

* next- Thúc đẩy con trỏ mảng nội bộ của một mảng

vd

<?php  
$people = array("Peter", "Joe", "Glenn", "Cleveland");  
  
echo current($people) . "<br>";  
echo next($people);  
?>

* pos - Bí danh của hiện tạ

vd

<?php  
$people = array("Peter", "Joe", "Glenn", "Cleveland");  
  
echo pos($people) . "<br>";  
?>

* prev - Tua con trỏ mảng nội bộ

vd

<?php  
$people = array("Peter", "Joe", "Glenn", "Cleveland");  
  
echo current($people) . "<br>";  
echo next($people) . "<br>";  
echo prev($people);  
?>

* range phạm vi - Tạo một mảng có chứa một loạt các yếu tố

vd

<?php  
$number = range(0,5);  
print\_r ($number);  
?>

* reset - Đặt con trỏ bên trong của một mảng tới phần tử đầu tiên của mình

vd

<?php  
$people = array("Peter", "Joe", "Glenn", "Cleveland");  
  
echo current($people) . "<br>";  
echo next($people) . "<br>";  
  
echo reset($people);  
?>

* rsort - Sắp xếp một mảng theo thứ tự ngược

vd

<?php  
$cars=array("Volvo","BMW","Toyota");  
rsort($cars);  
?>

* Shuffle phát ngẫu nhiên một mảng

Vd

<?php  
$my\_array = array("red","green","blue","yellow","purple");  
  
shuffle($my\_array);  
print\_r($my\_array);  
?>

* sizeof () trả về số phần tử trong một mảng.

vd

<?php  
$cars=array("Volvo","BMW","Toyota");  
echo sizeof($cars);  
?>

* sắp xếp () chức năng sắp xếp một mảng chỉ mục thứ tự tăng dần.

vd

<?php  
$cars=array("Volvo","BMW","Toyota");  
sort($cars);  
?>

* uasort () chức năng sắp xếp một mảng bằng giá trị sử dụng một người dùng xác định chức năng so sánh.

Vd

<?php  
function my\_sort($a,$b)  
{  
if ($a==$b) return 0;  
return ($a<$b)?-1:1;  
}  
  
$arr=array("a"=>4,"b"=>2,"c"=>8,d=>"6");  
uasort($arr,"my\_sort");  
?>

* uksort () chức năng sắp xếp một mảng bằng cách sử dụng các phím một người dùng xác định chức năng so sánh.

Vd

<?php  
function my\_sort($a,$b)  
{  
if ($a==$b) return 0;  
return ($a<$b)?-1:1;  
}  
  
$arr=array("a"=>4,"b"=>2,"c"=>8,d=>"6");  
uksort($arr,"my\_sort");  
?>

* usort () chức năng sắp xếp một mảng bằng cách sử dụng một người dùng xác định chức năng so sánh.

Vd

<?php  
function my\_sort($a,$b)  
{  
if ($a==$b) return 0;  
return ($a<$b)?-1:1;  
}  
  
$a=array(4,2,8,6);  
usort($a,"my\_sort");  
?>