Bài Tập Về Nhà

1. Mảng trong PHP là gì ? Có bao nhiều cách tạo mảng. Cho ví dụ.

Mảng trong Php ( array ) nó có thể hiểu đơn giản là một biến hay là biến danh sách. Nghĩa là bình thường một biến chỉ có thể lưu trữ được một giá trị còn mảng là một biến có thể lưu được nhiều giá trị trong một biến mà các phần tử không đồng nhất về mặt dữ liệu. Key là tên của mảng còn số chỉ mục có thể là kiểu chuỗi hoặc kiểu số.

**Các cách tạo mảng trong Php**

$name[key] = value;

$arr[] = value;

$arr = array($key1 => value1, $key2 => value2);

$arr = array(value1,value2);

$name = array(“value1”,“value2”,“value3”…“value-n”);

2. Cách lặp giá trị trong mảng. Cho ví dụ

1. for ($i=1;$i<100;$++)

{

Echo “value”;

}

1. Foreach ($name as $key => $value

[

Echo $value;

]

1. Các hàm hay sử dụng trong mảng php ?

**array\_change\_key\_case($array, $case) .**

Chuyển tất cả các key trong mảng $array sang chữ hoa nếu $case = 1 và sang chữ thường nếu $case = 0. Ta có thể dùng hằng số CASE\_UPPER thay cho số 1 và CASE\_LOWER thay cho số 0.

*<?php*

*$loichao = array('CHUHOA' => "xin chao");*

*$loichao = array\_change\_key\_case($loichao, CASE\_LOWER );*

*var\_dump($loichao);*

*?>*

*KQ: array(1) { ["chuhoa"]=> string(8) "xin chao" }*

array\_chunk — Split an array into chunks

Hàm array\_chunk() chia các thành phần của mảng thành nhiều mảng mới.

<?php

$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43","Harry"=>"50";

echo "<pre>";

print\_r(array\_chunk($age,3);

var\_dump($age );

?>

array\_column — Return the values from a single column in the input array

Hàm array\_column() trả về giá trị của một cột trong mảng đa chiều.

array\_combine — Creates an array by using one array for keys and another for its values

Hàm array\_combine() tạo một mảng mới từ một mảng là từ khóa, một mảng là giá trị.

+ array\_combine(keys,values);

*<?php*

*$id = array (478526,487562);*

*$user = array ("vinhtt","nhatln");*

*$capid = array\_combine ($id, $user);*

*var\_dump($capid);*

*?>*

KQ : array(2) { [478526]=> string(6) "vinhtt" [487562]=> string(6) "nhatln" }

array\_count\_values — Counts all the values of an array

Hàm array\_count\_values() trả về số lần xuất hiện của tất cả giá trị trong mảng.

+array\_count\_values(array)

*<?php*

*$ten=array("vinh","tran","tan","vinh","tran");*

*echo "<pre>";*

*print\_r(array\_count\_values($ten));*

*echo "<pre>";*

*?>*

array\_diff\_assoc — Computes the difference of arrays with additional index check

Hàm array\_diff\_assoc() so sánh khóa và giá trị của hai hay nhiều mảng và trả về một mảng chứa các phần xuất hiện ở mảng 1 và không xuất hiện ở mảng 2, mảng 3,…

+

*<?php*

*$vinh1=array("ten"=>"vinh","ho"=>"tran","chulot"=>"tan","lop"=>"39k21");*

*$vinh2=array("ten"=>"vinh","ho"=>"tran","chulot"=>"tan");*

*$khac=array\_diff\_assoc($vinh1,$vinh2);*

*print\_r($khac);*

*?>*

*KQ:* Array ( [lop] => 39k21 )  
array\_diff\_key — Computes the difference of arrays using keys for comparison

Hàm array\_diff\_key() so sánh khóa của hai mảng và trả về một mảng chứa các phần xuất hiện ở mảng 1 và không xuất hiện ở mảng 2

+array\_diff\_key(array1,array2…)

*<?php*

*$vinh1=array("ten"=>"vinh","toc"=>"tran","chulot"=>"tan");*

*$vinh2=array("ten"=>"vinh","ho"=>"tran","chulot"=>"tan");*

*$khac=array\_diff\_key($vinh1,$vinh2);*

*print\_r($khac);*

*?>*

*KQ:* Array ( [toc] => tran )

array\_diff\_uassoc — Computes the difference of arrays with additional index check which is performed by a user supplied callback function

So sánh khóa và giá trị trong mảng 2 hay nhiều mảng và trả về một mảng chứa các phần xuất hiện ở mảng 1 và không xuất hiện ở mảng 2, mảng 3,…

+array\_diff\_uassoc($array1,$array2…,"myfunction");

<?php

function myfunction($a,$b)

{

if ($a===$b)

{

return 0;

}

return ($a>$b)?-1:1;

}

$a1=array("ten"=>"vinh","chulot"=>"tan","ho"=>"tran");

$a2=array("ho"=>"vinh","chulot"=>"tan","ten"=>"tran");

$result=array\_diff\_uassoc($a1,$a2,"myfunction");

print\_r($result);

?>  
array\_diff\_ukey — Computes the difference of arrays using a callback function on the keys for comparison

Hàm array\_diff\_ukey() so sánh khóa của hai hay nhiều mảng và  trả về một mảng chứa các phần xuất hiện ở mảng 1 và không xuất hiện ở mảng 2, mảng 3,…

+array\_diff\_ukey($array1,$array2 ,…,"myfunction");

array\_diff — Computes the difference of arrays

Hàm array\_diff\_ukey() so sánh giá trị của hai mảng(hoặc nhiều hơn) và trả về một mảng chứa các thành phần xuất hiện ở mảng 1 và không xuất hiện ở mảng 2, mảng 3,…

+array\_diff($array1,$array2,$array3….);

*<?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);*

*?>*

*KQ:* Array ( [d] => yellow )

array\_fill\_keys — Fill an array with values, specifying keys

Hàm array\_fill\_keys() lấp đầy giá trị mảng khi đã chỉ định khóa.

+ array\_fill\_keys(keys,value);

<?php

$keys=array("d","e","f","g");

$a1=array\_fill\_keys($keys,"blue");

print\_r($a1);

?>

KQ: Array ( [d] => blue [e] => blue [f] => blue [g] => blue )

array\_fill — Fill an array with values

array\_filter — Filters elements of an array using a callback function

Hàm array\_filter() lọc giá trị của mảng dùng hàm do người dùng định nghĩa.

Hàm này sẽ duyệt qua từng giá trị của mảng và gọi hàm đã được định nghĩa. Nếu hàm tự định nghĩa trả về TRUE, giá trị hiện tại sẽ trả về vào mảng kết quả. Khóa của mảng được giữ nguyên.

+array\_filter(array,callbackfunction) ;

*<?php*

*function test\_odd($var)*

*{*

*return($var & 1);*

*}*

*$a1=array("a","b",2,3,4);*

*print\_r(array\_filter($a1,"test\_odd"));*

*?>*

KQ: Array ( [3] => 3 )

array\_flip — Exchanges all keys with their associated values in an array

Hàm array\_flip() sẽ chuyển đổi giá trị thành từ khóa và từ khóa thành giá trị.

+array\_flip(array);

*<?php*

*$vinh = array("namsinh" => 95 );*

*print\_r(array\_flip($vinh));*

*?>*

KQ:Array ( [95] => namsinh )

array\_intersect\_assoc — Computes the intersection of arrays with additional index check

Hàm array\_intersect\_assoc() so sánh khóa và giá trị của hai mảng và trả về mảng chứa các phần giống nhau.

+array\_intersect\_assoc(array1,array2,array3…) ;

*<?php*

*$vinh = array("namsinh" => 95 ,"ngaysinh" => 15);*

*$nhat = array ("namsinh" => 95, "ngaysinh" => 18);*

*print\_r(array\_intersect\_assoc($vinh,$nhat));*

*?>*

KQ: Array ( [namsinh] => 95 )

array\_intersect\_key — Computes the intersection of arrays using keys for comparison

Hàm array\_intersect\_key() so sánh khóa giữa hai hay nhiều bảng và trả về mảng chứa các phần giống nhau.

+ array\_intersect\_key( array1,array2,array3…) ;

*<?php*

*$vinh = array("namsinh" => 95 ,"ngaysinh" => 15);*

*$nhat = array ("namsinh" => 95, "ngaysinh" => 18);*

*print\_r(array\_intersect\_key($vinh,$nhat));*

*?>*

*KQ:* Array ( [namsinh] => 95 [ngaysinh] => 15 )

array\_intersect\_uassoc — Computes the intersection of arrays with additional index check, compares indexes by a callback function

Hàm array\_intersect\_uassoc() so sánh khóa và giá trị giữa hai hay nhiều mảng và trả vể mảng chứa các phần giống nhau.

+array\_intersect\_uassoc(array1,array2,array3,..,myfunction)

*<?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);*

*?>*

*KQ:* Array ( [b] => green )

array\_intersect\_ukey — Computes the intersection of arrays using a callback function on the keys for comparison

Hàm array\_intersect\_ukey() so sánh khóa giữa hai hay nhiều mảng và trả về phần giống nhau là các khóa chứa trong mảng 1, mảng 2, mảng 3…

+ array\_intersect\_ukey(array1,array2,array3…,myfunction)

*<?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");*

*print\_r(array\_intersect\_ukey($a1,$a2,"myfunction"));*

*?>*

KQ: Array ( [a] => red [b] => green )

array\_intersect — Computes the intersection of arrays

Hàm array\_intersect() so sánh giá trị của hai hay nhiều mảng và trả về mảng chứa các giá trị giống nhau.

+ array\_intersect(array1,array2,array3…)

*<?php*

*$a1=array("a"=>"red","b"=>"green","e"=>"blue","d"=>"yellow");*

*$a2=array("e"=>"red","f"=>"green","g"=>"blue");*

*$result=array\_intersect($a1,$a2);*

*print\_r($result);*

*?>*

*KQ:*Array ( [a] => red [b] => green [e] => blue )

array\_key\_exists — Checks if the given key or index exists in the array

Hàm array\_key\_exists() kiểm tra sự tồn tại của khóa trong mảng. Hàm sẽ trả về TRUE nếu khóa tồn tại và FALSE nếu không tồn tại.

+array\_key\_exists(key,array)

*<?php*

*$vinh=array("ten"=>"vinh","ngaysinh"=>15);*

*if (array\_key\_exists("ho",$vinh))*

*{*

*echo "Khoa da ton tai !";*

*}*

*else*

*{*

*echo "Khoa khong ton tai !";*

*}*

*?>*

KQ: Khoa khong ton tai !  
array\_keys — Return all the keys or a subset of the keys of an array

array\_map — Applies the callback to the elements of the given arrays

array\_merge\_recursive — Merge two or more arrays recursively

array\_merge — Merge one or more arrays

array\_multisort — Sort multiple or multi-dimensional arrays

array\_pad — Pad array to the specified length with a value

array\_pop — Pop the element off the end of array

array\_product — Calculate the product of values in an array

array\_push — Push one or more elements onto the end of array

array\_rand — Pick one or more random entries out of an array

array\_reduce — Iteratively reduce the array to a single value using a callback function

array\_replace\_recursive — Replaces elements from passed arrays into the first array recursively

array\_replace — Replaces elements from passed arrays into the first array

array\_reverse — Return an array with elements in reverse order

array\_search — Searches the array for a given value and returns the corresponding key if successful

array\_shift — Shift an element off the beginning of array

array\_slice — Extract a slice of the array

array\_splice — Remove a portion of the array and replace it with something else

array\_sum — Calculate the sum of values in an array

array\_udiff\_assoc — Computes the difference of arrays with additional index check, compares data by a callback function

array\_udiff\_uassoc — Computes the difference of arrays with additional index check, compares data and indexes by a callback function

array\_udiff — Computes the difference of arrays by using a callback function for data comparison

array\_uintersect\_assoc — Computes the intersection of arrays with additional index check, compares data by a callback function

array\_uintersect\_uassoc — Computes the intersection of arrays with additional index check, compares data and indexes by separate callback functions

array\_uintersect — Computes the intersection of arrays, compares data by a callback function

array\_unique — Removes duplicate values from an array

array\_unshift — Prepend one or more elements to the beginning of an array

array\_values — Return all the values of an array

array\_walk\_recursive — Apply a user function recursively to every member of an array

array\_walk — Apply a user supplied function to every member of an array

array — Create an array

arsort — Sort an array in reverse order and maintain index association

asort — Sort an array and maintain index association

compact — Create array containing variables and their values

count — Count all elements in an array, or something in an object

current — Return the current element in an array

each — Return the current key and value pair from an array and advance the array cursor

end — Set the internal pointer of an array to its last element

extract — Import variables into the current symbol table from an array

in\_array — Checks if a value exists in an array

key\_exists — Alias of array\_key\_exists

key — Fetch a key from an array

krsort — Sort an array by key in reverse order

ksort — Sort an array by key

list — Assign variables as if they were an array

natcasesort — Sort an array using a case insensitive "natural order" algorithm

natsort — Sort an array using a "natural order" algorithm

next — Advance the internal array pointer of an array

pos — Alias of current

prev — Rewind the internal array pointer

range — Create an array containing a range of elements

reset — Set the internal pointer of an array to its first element

rsort — Sort an array in reverse order

shuffle — Shuffle an array

sizeof — Alias of count

sort — Sort an array

uasort — Sort an array with a user-defined comparison function and maintain index association

uksort — Sort an array by keys using a user-defined comparison function

usort — Sort an array by values using a user-defined comparison function

------------------------------------------  
Cho mảng $number = array(1,2,3,4,5,6,7,8,9,10)  
a.Tính tổng giá trị các phần tử trong mảng.  
b.Xác định phần tử nhỏ nhất trong mảng  
c.Xác định phần tử lớn nhất trong mảng  
d.Loại bỏ phần tử cuối cùng của mảng trên  
e. Loại bỏ phần tử đầu tiên của mảng trên (dùng lệnh var\_dump để kiểm tra dữ liệu sau khi loại bỏ)