在php中，数组中的元素分为两部分，分别为键(Key)和值(Value)。其中“键”为元素的识别名称，也被称为数组下标，“值”为元素的内容。两者存在一种对应关系，称为映射。

索引数组： 指下标为整数的数组，“键”都是整数，可以自己指定“键”，默认从0开始，用于使用位置来标识数组元素。

关联数组：指下标为字符串的数组，使用于存储一系列具有逻辑关系的变量。

**数组的定义方式**

1. 直接给数组中的元素赋值

语法：$arrayName[Key]=mixed value

“Key”是数组的下标，类型可以是整型或字符串。

定义索引数组的示例代码：

$arr[0]=123; 如果没有指定键(即[ ]内的键名省略不写)，

$arr[1]=”hello”; 则默认从0开始。

$arr[2]=45.6;

$arr[3]=true;

$arr[4]=null;

定义关联数组示例代码：

$arr["id"]=1;

$arr["name"]="黑雪姬";

$arr["address"]="加速世界";

$arr["email"]="han@tom.com";

2.使用array()函数定义数组

如果省略key部分，则定义的数组默认为索引数组。

$arr=array( 123, ”hello” ,45.6, true );

定义关联数组

$arr=array(“id”=>1, “name”=>”杨某”, ”address”=>”清华”, “email”=>”han@tom”);

注意：

1.如果在定义数组时没有给某个元素指定下标，php会自动将目前最大的那个整数下标值加1，作为该元素的下标，并依次递增后面元素的下标值。

2.数组元素的下标只有整型和字符串两种类型，如果是其他类型，则会进行类型转换。

3.由于合法的整型值的字符串下标会被类型转换为整型下标，所以在创建数组的时候，如果转换后数组存在相同的下标时，后面出现的元素值会覆盖前面的元素值。

**数组的使用**

$arr[0]和$arr{0}是相等的。

print\_r()函数：按照一定格式显示数组中所有元素的键和值

var\_dump()函数：同时将数组元素及元素中值的数据类型打印出来。

在使用上述的函数打印输出数组时，为了方便输出格式化的数组结构形式，会将调用放在<pre>格式化文本输出之间。

unset()函数：用于删除数组中的元素，亦可删除整个数组

$arr[0]=123;

$arr[1]=456;

$arr[2]="hello";

echo "\*\*删除前\*\*<br/>";

print\_r($arr);

unset($arr[1]);

echo "\*\*删除后\*\*<br/>";

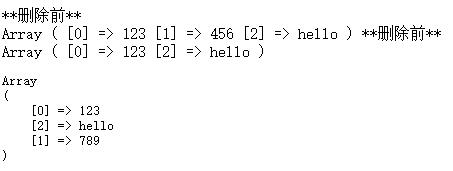
print\_r($arr)."<br>";

$arr[1]=789; //动态添加数组元素

echo "<pre>";

print\_r($arr);

echo "</pre>";



可惜键未排序，以后学到再排吧

数组运算符

|  |  |  |
| --- | --- | --- |
| 运算符 | 含义 | 示例 |
| + | 联合 | $a+$b: $a和$b的联合 |
| == | 相等 | $a==$b: 如果$a和$b具有相同的键/值对则为TRUE |
| === | 全等 | $a===$b: 如果$a和$b具有相同的键/值对并且顺序和类型都相同则为TRUE |
| != | 不等 | $a!=$b: $a不等于$b则为TRUE |
| <> | 不等 | $a<>$b: $a不等于$b则为TRUE |
| !== | 不全等 | $a!==$b: $a不全等于$b则为TRUE |

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

$b=array("a"=>"pear","b"=>"strawberry","c"=>"cherry");

$c=$a+$b; // Union of $a and $b

echo "Union of \$a and \$b: \n" ;

var\_dump($c);

$d=$b+$a ; // Union of $b and $a

echo "Union of \$b and \$a: \n" ;

var\_dump($d);

注：“+”联合运算符，用于合并数组，如果出现下标相同的元素，则保留第一个数组的元素(一般不是后入的替换前入吗…)

**数组指针**

数组指针用于指向数组中的某个元素，默认指向数组的第一个元素，一个数组只有一个指针。

数组指针操作函数

|  |  |
| --- | --- |
| 函数名 | 作用 |
| mixed **current**( array &$array ) | 获取数组中当前元素的值，如果内部指针超过数组的末端，则返回false |
| mixed **key**( array &$array ) | 获取当前元素的下标，即键名，默认首位 |
| mixed **next**( array &$array ) | 将数组的内部指针向前移动一位 |
| mixed **prev**( array &$array ) | 将数组的内部指针倒回一位 |
| mixed **end**( array &$array ) | 将数组的内部指针指向最后一个元素 |
| mixed **reset**( array &$array ) | 重置指针，即将数组的指针指向第一个元素 |

$arr1=array(

"os"=>"window",

"server"=>"apache",

"language"=>"php",

"database"=>"mysql"

);

while(current($arr1))

{

echo key($arr1);

echo "=>";

echo current($arr1)."<br>";

next($arr1); //将数组指针向前移动一位

}

注意：数组指针可以移到外部，一旦移到外部，其键就变为NULL，值为false，此时则不能通过prev()函数将数组中的指针向后移动一位，只能使用reset()函数重置指针。

each()函数：返回数组中当前的键／值对并将数组指针向前移动一步

返回 array 数组中当前指针位置的键／值对并向前移动数组指针。键值对被返回为四个单元的数组，键名为>，key 和 value。

单元 0 和 key 包含有数组单元的键名，1 和 value 包含有数据。

如果内部指针越过了数组的末端，则 each() 返回 FALSE 。

**关联数组：**

$arr1=array(

"os"=>"window",

"server"=>"apache",

"language"=>"php",

"database"=>"mysql"

);

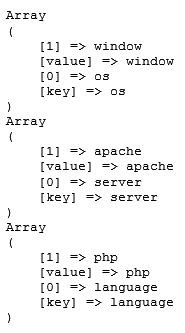
echo "<pre>";

print\_r(each($arr1));

print\_r(each($arr1));

print\_r(each($arr1));

echo "</pre>";



仔细一看[0][key]指的都是同一个…，虽然显示的有点怪，其实还是等同于其他语言的数组

**索引数组：**

$fruit=array('a','apple');

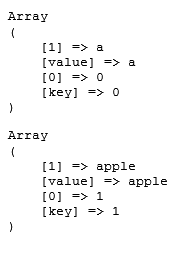
while($a=each($fruit )){

echo "<pre>";

print\_r( $a) ;

}

?>



数组遍历：依次访问数组的每一个元素

foreach语句实现数组的遍历

$arr1=array(

"os"=>"window",

"server"=>"apache",

"language"=>"php",

"database"=>"mysql"

);

foreach($arr1 as $value){ //无键名遍历

echo $value."<br>"."<br>";

}

foreach($arr1 as $key=>$value)

{

echo $key."=>".$value."<br>"; //键值对的遍历

}

注意：

(1)使用foreach遍历数组时，$key和$value只不过是一个变量名而已，任何符合语法规则的变量名均可，如$k和$v，不过还是见名知意的好。

(2)$key和$value保存的数据是通过值传递的方式赋值的，这意味着对$key和$value的修改不影响数组本身。故可以使用引用传递，在变量前加上&即可，但此方法只对$value有效。

list：是一个语言结构，作用是把数组中的值赋给一些变量

仅能用于数字索引的数组并假定索引从0开始。

案例：使用each()函数、list()语言结构和while语句遍历数组

$arr1=array(

"os"=>"window",

"server"=>"apache",

"language"=>"php",

"database"=>"mysql"

);

while(list($key,$value)=each($arr1))

{

echo "$key=>$value"."<br>";

}

array\_merge()函数：合并一个或多个数组

声明：array\_merge( $array1, $array2)

1. 如果输入的数组中有相同的字符串键名，则该键名后面的值将覆盖前一个值。
2. 如果数组包含数字键名，后面的值将不会覆盖原来的值，而是附加到数组的后面。
3. 如果是数字索引数组，则键名会以连续方式重新编排索引。

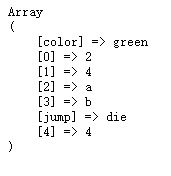
$array1=array("color"=>"red",2,4);

$array2=array("a","b","color"=>"green","jump"=>"die",4); $result=array\_merge($array1,$array2);

echo "<pre>";

print\_r($result);

echo "</pre>";



也就是数字按数字索引，关键字按关键字索引，互不干涉

array\_chunk()函数：将一个数组分割成多个数组

array\_chunk( array $input, int $size[,bool $preserve\_keys]) ;

$input表示要分割的数组，$size是分割后的每个数组中元素的个数，

$preserve\_keys是一个可选参数，默认值为false，分割后的数组中元素的索引将从零开始；参数设为true，则分割后的数组中元素保留原来的索引。

$arr=array("a","b","c");

echo "<pre>";

echo "索引将从零开始："."<br>";

print\_r(array\_chunk($arr,2))."<br>";

echo "保留原来的索引："."<br>";

print\_r(array\_chunk($arr,2,true))."<br>";

echo "</pre>";

结果：

索引将从零开始：  
Array  
(  
 [0] => Array  
 (  
 [0] => a  
 [1] => b  
 )  
  
 [1] => Array  
 (  
 [0] => c  
 [1] => d  
 )  
  
)  
保留原来的索引：  
Array  
(  
 [0] => Array  
 (  
 [0] => a  
 [1] => b  
 )  
  
 [1] => Array  
 (  
 [2] => c  
 [3] => d  
 )  
  
)

array\_rand()函数：从数组中随机取出一个或多个元素

mixed array\_rand( array $input[, int $num\_req] );

可选参数$num\_req用于指定取出元素的个数，默认为1。如果只取出一个元素，会返回一个随机元素的键名，否则返回一个包含随机键名的数组。 反正就是随机键名

$arr=array("黑雪姬","有田春雪","小瑶");

echo "<pre>";

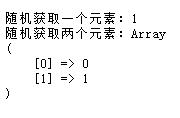
echo "随机获取一个元素：";

print\_r(array\_rand($arr));

echo "<br>"."随机获取两个元素："; //echo专门用来输出转义字符

print\_r(array\_rand($arr,2));

echo "</pre>";



array\_reverse()函数：返回一个元素顺序相反的数组

声明：array array\_reverse( array $array[, bool $preserve\_keys ]);

如果$preserve\_keys为true，则保留原来键名。

$arr=array("黑雪姬","有田春雪",array("小瑶",2035)); //把数组当做一个元素，故里面的元素顺序不变

echo "<pre>";

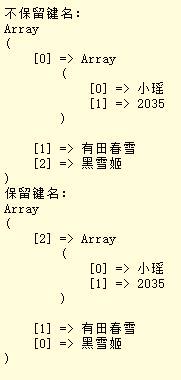
echo "不保留键名："."<br>";

print\_r(array\_reverse($arr));

echo "保留键名："."<br>";

print\_r(array\_reverse($arr,true));

echo "</pre>";



二分查找法：每次将指定元素和数组中间位置的元素进行对比，从而排除掉其中的一半元素，但此方法只适合排序后的数组

function binarySearch(&$arr,$findVal,$start,$end)

{

if($end<$start)

//当查找区间的开始索引大于结束索引时，意味着已全部对比。

{

echo "找不到该数";

return;

}

$mid = round(($end+$start)/2);

if($findVal>$arr[$mid])

{

binarySearch($arr,$findVal,$mid+1,$end);

//将查找区间的开始索引$start置为$mid+1，再次运行此函数。

}

else if($findVal<$arr[$mid])

{

binarySearch($arr,$findVal,$start,$mid-1);

//将查找区间的结束索引$end置为$mid-1，再次运行此函数。

}

else

{

echo "找到这个数，下标是$mid";

}

}

$arr = array(2,3,5,8,9);

echo binarySearch($arr,8,0,count($arr)-1);