---------- 多选分类修改，数据入库增删留数组：两个数组的对比分析

```
/*
    多选分类修改，数据入库增删留数组
    说明:$k 和$v 是一一对应
*/

//旧分类
$a1=array('1'=>'杂文',
                    '2'=>'小说',
                    '3'=>'散文'
);
//新分类
$a2=array('3'=>'散文',
                    '4'=>'诗歌',
                    '6'=>'文学'
);

foreach($a1 as $k => $v)
{
    $have='';
        foreach($a2 as $ks => $vs)
        {
                echo "$v==$vs..";
                if($v==$vs)
                {
                        $have=1;
                }
                if(!array_key_exists($ks,$a1))
                {
                    $add[$ks]=$vs;
                }
        }

        if($have)
        {
                $bao[$k]=$v;
        }else
        {
            $del[$k]=$v;
        }
}
var_dump($add);//新增的记录
var_dump($del);//删除的记录
var_dump($bao);//保留的记录
```

```
//方式二：无key

//旧分类
$a1=array('杂文','小说','散文');
//新分类
$a2=array('散文','诗歌','文学');

foreach($a1 as $k => $v)
{
    $have='';
    foreach($a2 as $ks => $vs)
    {
            echo "$v==$vs..";
            if($v==$vs)
            {
                $have=1;
            }
            if(!in_array($vs, $a1))
            {
                $add[$vs]=$vs;
            }
    }

    if($have)
    {
            $bao[]=$v;
    }else
    {
        $del[]=$v;
    }
}
var_dump($add);//新增的记录
var_dump($del);//删除的记录
var_dump($bao);//保留的记录

//--------------------------- 比较全的随机数（字）
  class getRandstrClass{
    function getCode ($length = 32, $mode = 0) {
        switch ($mode) {
        case '1':
            $str = '1234567890';
            break;
        case '2':
            $str = 'abcdefghijklmnopqrstuvwxyz';
            break;
        case '3':
```

```
            $str = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
            break;
        case '4':
            $str = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
            break;
        case '5':
            $str = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890';
            break;
        case '6':
            $str = 'abcdefghijklmnopqrstuvwxyz1234567890';
            break;
        default:
        $str = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890';
        break;
        }
        $randString = '';
        $len = strlen($str)-1;
        for($i = 0;$i < $length;$i ++){
            $num = mt_rand(0, $len);
            $randString .= $str[$num];
        }
        return $randString ;
    }
}

$code = new getRandstrClass();
$length = 16;
$mode = 1;
$str = $code->getCode($length, $mode);
echo $str;
$code = NULL;

//----------------------有两个数值型变量 $a ,$b ，请在不使用用第二个变量的情况下交换它们的值
$a=3;
$b=4;


$a=$a+$b;
$b=$a-$b;
$a=$a-$b;

//list($b, $a) = array($a, $b);

/*-------------------------【选择排序（一维数组）】
【基本思想】：每一趟从待排序的数据元素中选出最小（或最大）的一个元素，顺序放在已排好序的数列的最后，直到全部待排序的数据元素排完。
【示例】：
[初始关键字] [49 38 65 97 76 13 27 49]
```

```
第一趟排序后 13 [38 65 97 76 49 27 49]
第二趟排序后 13 27 [65 97 76 49 38 49]
第三趟排序后 13 27 38 [97 76 49 65 49]
第四趟排序后 13 27 38 49 [49 97 65 76]
第五趟排序后 13 27 38 49 49 [97 97 76]
第六趟排序后 13 27 38 49 49 76 [76 97]
第七趟排序后 13 27 38 49 49 76 76 [ 97]
最后排序结果 13 27 38 49 49 76 76 97
*/
function select_sort($arr){
$count = count($arr);
for($i=0; $i<$count; $i++){
    $k = $i;
    for($j=$i+1; $j<$count; $j++){
        if ($arr[$k] > $arr[$j])
           $k = $j;
}

     //最小元素k和i调换
    if($k != $i){
        $tmp = $arr[$i];
        $arr[$i] = $arr[$k];
        $arr[$k] = $tmp;
    }
}
return $arr;
}


/*
----------------------------【插入排序（一维数组）】
【基本思想】：每次将一个待排序的数据元素，插入到前面已经排好序的数列中的适当位置，使数列依然有序；直到待排序数据元素全部插入完为止。
【示例】：
[初始关键字] [49] 38 65 97 76 13 27 49
J=2(38) [38 49] 65 97 76 13 27 49
J=3(65) [38 49 65] 97 76 13 27 49
J=4(97) [38 49 65 97] 76 13 27 49
J=5(76) [38 49 65 76 97] 13 27 49
J=6(13) [13 38 49 65 76 97] 27 49
J=7(27) [13 27 38 49 65 76 97] 49
J=8(49) [13 27 38 49 49 65 76 97]
*/
function insert_sort($arr){
$count = count($arr);
for($i=1; $i<$count; $i++){
    $tmp = $arr[$i];
    $j = $i - 1;
    while($arr[$j] > $tmp){
```

```
      $arr[$j+1] = $arr[$j];
      $arr[$j] = $tmp;
      $j--;
    }
}
return $arr;
}

/*
－－－－－－－－－－－－－冒泡排序（一维数组）

【基本思想】：两两比较待排序数据元素的大小，发现两个数据元素的次序相反时即进行交换，直到没有反序的数据元素为止。
【排序过程】：设想被排序的数组R［1..N］垂直竖立，将每个数据元素看作有重量的气泡，根据轻气泡不能在重气泡之下的原则，从下往上扫描数组R，凡扫描到违反本原则的轻气泡，就使其向上″漂浮″，如此反复进行，直至最后任何两个气泡都是轻者在上，重者在下为止。
*/


function bubble_sort($array){
$count = count($array);
if ($count <= 0) return false;

for($i=0; $i<$count; $i++){
    for($j=$count-1; $j>$i; $j--){
      if ($array[$j] < $array[$j-1]){
        $tmp = $array[$j];
        $array[$j] = $array[$j-1];
        $array[$j-1] = $tmp;
      }
    }
}
return $array;
}




/*
------------------------------汉诺塔的PHP算法

    汉诺塔（又称河内塔）问题是印度的一个古老的传说。开天辟地的神勃拉玛在一个庙里留下了三根金刚石的棒，第一根上面套着64个圆的金片，最大的一个在底下，其余一个比一个小，依次叠上去，庙里的众僧不倦地把它们一个个地从这根棒搬到另一根棒上，规定可利用中间的一根棒作为帮助，但每次只能搬一个，而且大的不能放在小的上面。解答结果请自己运行计算，程序见尾部。面对庞大的数字(移动圆片的次数)18446744073709551615，看来，众僧们耗尽毕生精力也不可能完成金片的移动。
```

后来，这个传说就演变为汉诺塔游戏:
1.有三根杆子A,B,C。A杆上有若干碟子
2.每次移动一块碟子,小的只能叠在大的上面
3.把所有碟子从A杆全部移到C杆上
经过研究发现，汉诺塔的破解很简单，就是按照移动规则向一个方向移动金片：
如3阶汉诺塔的移动：A→C,A→B,C→B,A→C,B→A,B→C,A→C
此外，汉诺塔问题也是程序设计中的经典递归问题。

```
 */

     $objMover = new Mover();
     $fromPlace = 'A';
     $toPlace = 'C';
     $assistancePlace = 'B';
     $objMover->move(3,$fromPlace,$toPlace,$assistancePlace);
     print_r($objMover->getMovedMessage());


class Mover
{
     protected $_tabMessage = array();

     public function __construct()
     {
     }
    /**
      * Enter description here...
      *
      * @param unknown_type $N, the larger the number is, the heavier it is
      * @param unknown_type $fromPlace
      * @param unknown_type $toPlace
      * @param unknown_type $assistancePlace
      */
    public function move($N,$fromPlace,$toPlace,$assistancePlace)
     {
         if($N == 1)
         {
            $this->_tabMessage[] = "Move $N from $fromPlace to $toPlace";
         }elseif($N > 1){
            $this->move(($N-1),$fromPlace,$assistancePlace,$toPlace);
            $this->_tabMessage[] = "Move $N from $fromPlace to $toPlace";
            $this->move(($N-1),$assistancePlace,$toPlace,$fromPlace);
         }

     }
     public function getMovedMessage()
     {
         return $this->_tabMessage;
     }
```

```
}
function hanoi($n,$x,$y,$z){
    if($n==1){
        move($x,1,$z);
    }else{
            hanoi($n-1,$x,$z,$y);
            move($x,$n,$z);
            hanoi($n-1,$y,$x,$z);
          }
    }
function move($x,$n,$z){
    echo 'move disk '.$n.' from '.$x.' to '.$z.'<br/>';
}
hanoi(10,'x','y','z');

/*
-----------------------------------猴子大王游戏
一群猴子排成一圈，按1，2，...，n依次编号。
然后从第1只开始数，数到第m只,把它踢出圈，从它后面再开始数，
再数到第m只，在把它踢出去...，如此不停的进行下去，
直到最后只剩下一只猴子为止，那只猴子就叫做大王。
要求编程模拟此过程，输入m、n, 输出最后那个大王的编号。
```

结果视图：

```
z=0/i=0
z=1/i=1
z=2/i=2
z=3/i=3
Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 5 [4] => 6 )
z=0/i=3
z=1/i=4
z=2/i=0
z=3/i=1
Array ( [0] => 1 [1] => 3 [2] => 5 [3] => 6 )
z=0/i=1
z=1/i=2
z=2/i=3
z=3/i=0
Array ( [0] => 3 [1] => 5 [2] => 6 )
z=0/i=0
z=1/i=1
z=2/i=2
z=3/i=0
Array ( [0] => 5 [1] => 6 )
z=0/i=0
z=1/i=1
```

```
z=2/i=0
z=3/i=1
Array ( [0] => 5 )
King is:5
*/

function monkeyKing($n,$m){
$monkeys=range(1,$n);
$i=0;//取出时候的坐标
$z=0;//数到M的时候停
while(($mnum=count($monkeys))>1){
   if($i==$mnum){
    $i=0;
   }
   echo 'z='.$z.'/i='.$i.'<br/>';
   $z++;
   $i++;
   if($z==$m){
    array_splice($monkeys,--$i,1);
    $z=0;
    print_r($monkeys);
    echo "<br/>";
   }
}
return($monkeys[0]);
}
echo 'King is:'.monkeyKing(20,4);


/*
-------------------------------------翻牌游戏
1-52张扑克牌，初始都翻开朝上
从2开始，以倍数为基础，将2翻一次，4翻一次，6翻一次。。。52翻一次
从3开始，以倍数为基础，将3翻一次，6翻一次，9翻一次。。。48翻一次
从4开始，以倍数为基础，将4翻一次，8翻一次，13翻一次。。。48翻一次
....

求最后朝上的牌有哪些？

 * Created on 2009-9-30
 *
 * To change the template for this generated file go to
 * Window - Preferences - PHPeclipse - PHP - Code Templates
 */
 class up
 {
     protected $max = 52;
     protected $min = 2;
     protected $rs = array(1);//结果集，第一张牌是朝上的
```

```
function up()
{


}

/* 循环得到2-52的整除数组
 *
 * Array
(
    [2] => Array
        (
            [0] => 2
        )

    [3] => Array
        (
            [0] => 3
        )

    [4] => Array
        (
            [0] => 2
            [1] => 4
        )

    [5] => Array
        (
            [0] => 5
        )

    [6] => Array
        (
            [0] => 2
            [1] => 3
            [2] => 6
        )

    [7] => Array
        (
            [0] => 7
        )

    [8] => Array
        (
            [0] => 2
            [1] => 4
            [2] => 8
```

```
    )

[9] => Array
    (
        [0] => 3
        [1] => 9
    )

[10] => Array
    (
        [0] => 2
        [1] => 5
        [2] => 10
    )

[11] => Array
    (
        [0] => 11
    )

[12] => Array
    (
        [0] => 2
        [1] => 3
        [2] => 4
        [3] => 6
        [4] => 12
    )

[13] => Array
    (
        [0] => 13
    )

[14] => Array
    (
        [0] => 2
        [1] => 7
        [2] => 14
    )

[15] => Array
    (
        [0] => 3
        [1] => 5
        [2] => 15
    )

[16] => Array
```

```
    (
        [0] => 2
        [1] => 4
        [2] => 8
        [3] => 16
    )

[17] => Array
    (
        [0] => 17
    )

[18] => Array
    (
        [0] => 2
        [1] => 3
        [2] => 6
        [3] => 9
        [4] => 18
    )

[19] => Array
    (
        [0] => 19
    )

[20] => Array
    (
        [0] => 2
        [1] => 4
        [2] => 5
        [3] => 10
        [4] => 20
    )

[21] => Array
    (
        [0] => 3
        [1] => 7
        [2] => 21
    )

[22] => Array
    (
        [0] => 2
        [1] => 11
        [2] => 22
    )
```

```
[23] => Array
    (
        [0] => 23
    )

[24] => Array
    (
        [0] => 2
        [1] => 3
        [2] => 4
        [3] => 6
        [4] => 8
        [5] => 12
        [6] => 24
    )

[25] => Array
    (
        [0] => 5
        [1] => 25
    )

[26] => Array
    (
        [0] => 2
        [1] => 13
        [2] => 26
    )

[27] => Array
    (
        [0] => 3
        [1] => 9
        [2] => 27
    )

[28] => Array
    (
        [0] => 2
        [1] => 4
        [2] => 7
        [3] => 14
        [4] => 28
    )

[29] => Array
    (
        [0] => 29
    )
```

```
[30] => Array
    (
        [0] => 2
        [1] => 3
        [2] => 5
        [3] => 6
        [4] => 10
        [5] => 15
        [6] => 30
    )

[31] => Array
    (
        [0] => 31
    )

[32] => Array
    (
        [0] => 2
        [1] => 4
        [2] => 8
        [3] => 16
        [4] => 32
    )

[33] => Array
    (
        [0] => 3
        [1] => 11
        [2] => 33
    )

[34] => Array
    (
        [0] => 2
        [1] => 17
        [2] => 34
    )

[35] => Array
    (
        [0] => 5
        [1] => 7
        [2] => 35
    )

[36] => Array
    (
```

```
        [0] => 2
        [1] => 3
        [2] => 4
        [3] => 6
        [4] => 9
        [5] => 12
        [6] => 18
        [7] => 36
    )

[37] => Array
    (
        [0] => 37
    )

[38] => Array
    (
        [0] => 2
        [1] => 19
        [2] => 38
    )

[39] => Array
    (
        [0] => 3
        [1] => 13
        [2] => 39
    )

[40] => Array
    (
        [0] => 2
        [1] => 4
        [2] => 5
        [3] => 8
        [4] => 10
        [5] => 20
        [6] => 40
    )

[41] => Array
    (
        [0] => 41
    )

[42] => Array
    (
        [0] => 2
        [1] => 3
```

```
        [2] => 6
        [3] => 7
        [4] => 14
        [5] => 21
        [6] => 42
    )

[43] => Array
    (
        [0] => 43
    )

[44] => Array
    (
        [0] => 2
        [1] => 4
        [2] => 11
        [3] => 22
        [4] => 44
    )

[45] => Array
    (
        [0] => 3
        [1] => 5
        [2] => 9
        [3] => 15
        [4] => 45
    )

[46] => Array
    (
        [0] => 2
        [1] => 23
        [2] => 46
    )

[47] => Array
    (
        [0] => 47
    )

[48] => Array
    (
        [0] => 2
        [1] => 3
        [2] => 4
        [3] => 6
        [4] => 8
```

```
            [5] => 12
            [6] => 16
            [7] => 24
            [8] => 48
        )

    [49] => Array
        (
            [0] => 7
            [1] => 49
        )

    [50] => Array
        (
            [0] => 2
            [1] => 5
            [2] => 10
            [3] => 25
            [4] => 50
        )

    [51] => Array
        (
            [0] => 3
            [1] => 17
            [2] => 51
        )

    [52] => Array
        (
            [0] => 2
            [1] => 4
            [2] => 13
            [3] => 26
            [4] => 52
        )

)
  *
  * */
 public function setp1()
 {
     for($i=$this->min;$i<=$this->max;$i++)
     {
         for($j=$this->min;$j<=$this->max;$j++)
         {
             if(0==$i%$j)
             {
                 $arr[$i][]=$j;
```

```
            }
        }
    }
    return $arr;
}

/* 获得整除组合为偶数的牌
 *
 * 返回值：
 *
 Array
(
    [0] => 1
    [4] => Array
        (
            [0] => Array
                (
                    [0] => 2
                    [1] => 4
                )

        )

    [9] => Array
        (
            [0] => Array
                (
                    [0] => 3
                    [1] => 9
                )

        )

    [16] => Array
        (
            [0] => Array
                (
                    [0] => 2
                    [1] => 4
                    [2] => 8
                    [3] => 16
                )

        )

    [25] => Array
        (
            [0] => Array
                (
```

```
                    [0] => 5
                    [1] => 25
                )

        )

    [36] => Array
        (
            [0] => Array
                (
                    [0] => 2
                    [1] => 3
                    [2] => 4
                    [3] => 6
                    [4] => 9
                    [5] => 12
                    [6] => 18
                    [7] => 36
                )

        )

    [49] => Array
        (
            [0] => Array
                (
                    [0] => 7
                    [1] => 49
                )

        )

)
 * */
public function execute($arr)
{
    foreach($arr as $k =>$v)
    {
        if($this->setp3(count($v)))
        {
            $this->rs[$k][] = $v;
        }
    }
    return $this->rs;
}

//判断奇偶数
public function setp3($num)
{
```

```
        if(0==$num%2)
        {
            return true;
        }else
        {
            return false;
        }
    }

 }
$arr = array();
$up = new up();
$arr = $up->setp1();
//print_r($arr);
print_r($up->execute($arr));


/*
-------------------------已知字符串 $string = "2dsjfh87HHfytasjdfldiuuidhfcjh";
找出 $string 中出现次数最多的所有字符。
*/

$string = "2dsjfh87HHfytasjdfldiuuidhfcjh";
$a = str_split($string);
$b = array_count_values ($a);
arsort($b);
print_r($b);
print_r(array_intersect($b, array(current($b))));

-----------------------
$count = 5;
echo $count++;

echo ++$count;

/*
结果是:57

---------------------------------------遍历文件夹下面的所有目录和PHP文件
*/

define('DS', DIRECTORY_SEPARATOR);
function breadth_first_files($from = '.') {
    $queue = array(rtrim($from, DS).DS);// normalize all paths
    $files = array();
    while($base = array_shift($queue )) {
  //var_dump($queue);
        if (($handle = opendir($base))) {
            while (($child = readdir($handle)) !== false) {
```

```
                if( $child == '.' || $child == '..') {
                     continue;
                  }
                 if (is_dir($base.$child)) {
                     $combined_path = $base.$child.DS;
                     array_push($queue, $combined_path);
                  } else {
                      if("php"==getextension($child))
                     {
                         $files[] = $base.$child;
                     }


                  }
              }
             closedir($handle);
         } // else unable to open directory => NEXT CHILD
     }
    return $files; // end of tree, file not found
}
// 获得文件扩展名
function getextension($filename) {
    return substr(strrchr($filename, "."), 1);
}

print_r(breadth_first_files("E:/workspace/Ozf-blog"));

/*
------------------------------------------
*/
$data=array(23,22,45,28);
//写个function,把数组的个位,十位分别拆分成如下新数组
$data=array(2,3,2,2,4,5,2,8);

print_r(str_split(join(null, $data)));
//-------------------------------------------------

//冒泡排序（数组排序）

function bubble_sort($array)

{

        $count = count($array);

        if ($count <= 0) return false;

        for($i=0; $i<$count; $i++){
```

```
                for($j=$count-1; $j>$i; $j--){
                        if ($array[$j] < $array[$j-1]){
                                $tmp = $array[$j];
                                $array[$j] = $array[$j-1];
                                $array[$j-1] = $tmp;
                        }
                }
        }
        return $array;
}
//快速排序（数组排序）
function quick_sort($array) {
        if (count($array) <= 1) return $array;
        $key = $array[0];
        $left_arr = array();
        $right_arr = array();
        for ($i=1; $i<count($array); $i++){
                if ($array[$i] <= $key)
                        $left_arr[] = $array[$i];
                else
                        $right_arr[] = $array[$i];
        }
        $left_arr = quick_sort($left_arr);
        $right_arr = quick_sort($right_arr);
        return array_merge($left_arr, array($key), $right_arr);
```

```
}
//二分查找（数组里查找某个元素）
function bin_sch($array, $low, $high, $k){
    if ($low <= $high){
        $mid = intval(($low+$high)/2);
        if ($array[$mid] == $k){
            return $mid;
        }elseif ($k < $array[$mid]){
            return bin_sch($array, $low, $mid-1, $k);
        }else{
            return bin_sch($array, $mid+1, $high, $k);
        }
    }
    return -1;
}
//顺序查找（数组里查找某个元素）
function seq_sch($array, $n, $k){
    $array[$n] = $k;
    for($i=0; $i<$n; $i++){
        if($array[$i]==$k){
            break;
        }
    }
    if ($i<$n){
```

```
        return $i;
    }else{
        return -1;
    }
}


//二维数组排序， $arr是数据，$keys是排序的健值，$order是排序规则，1是升序，0是降序
function array_sort($arr, $keys, $order=0) {
if (!is_array($arr)) {
return false;
}
$keysvalue = array();
foreach($arr as $key => $val) {
$keysvalue[$key] = $val[$keys];
}
if($order == 0){
asort($keysvalue);
}else {
arsort($keysvalue);
}
reset($keysvalue);
foreach($keysvalue as $key => $vals) {
$keysort[$key] = $key;
}
```

```
$new_array = array();

foreach($keysort as $key => $val) {

$new_array[$key] = $arr[$val];

}

return $new_array;

}
```