# 数组基础

含义：

数组就是一系列数据的集合体，他们按设定的顺序排列为一个“链的形状”。

注意：php中的数组单元的顺序，跟下标无关！

### 数组定义（赋值）：

$arr1 = array(3, 11, 5, 18, 2 );//这是最常见的数组，下标为“默认下标”，就是从0开始的整数；

$arr2 = array(“a”=>3, “bb”=>11, “cc123”=>5, ‘d1’=>18, ‘xyz’=>2 );关联数组，下标为字符串，常见

$arr3 = array(1=>3, 10=>11, 3=>5, 0=>18, 88=>2 );下标可以人为给定；

$arr4 = array(1=>3, ‘a1’=>11, 3=>5, ‘mn’=>18, 88=>2 );下标可以数字和字符串混合使用；

$arr5 = array(5=>3, 11, 3=>5, ‘mn’=>18, 2 ); //有指定下标，也有“自动下标”，

//此时下标为：5，6，3，”mn”, 7

//可见，自动下标为“前面最大数字下标+1”

$arr6 = array(5=>3, 7.7=>11, 3=>5, ‘mn’=>18, 2 ); //此时下标为：5，7，3，”mn”, 8

$arr7 = array(5=>3, true=>11, false=>5, ‘mn’=>18, 2 ); //此时下标为：5，1，0，”mn”, 6

$arr8 = array(1=>3, 3=>33, true=>11, ); //此时下标为：1，3,其对应值为：11, 33

//下标如果有重复，后面的值覆盖前面的值；

$arr9 = array(1=>3, -3=>33, 11, ); //此时下标为：1， -3， 2，注意：最后一个逗号“可以有”。

其他一些形式：

$arr10[] = 3;

$arr10[] = 11;

$arr10[] = 5; //该数组下标为0,1,2，常规情况

$arr11[‘a’] = 3;

$arr11[‘bb’] = 11;

$arr11[‘cc123’] = 5; //该数组下标为’a’,’bb’,’cc123’，常规情况

$arr12[1] = 3;

$arr12[] = 11; //此时下标为2

$arr13[‘cc123’] = 5; //该数组下标为1,2,’cc123’

特别注意：php中，数组单元的顺序，是由其“放入”顺序决定，而不是下标。

数组取值：

$v1=$arr1[0];

$i=3;

$v2=$arr[$i];//取得数组下标为3的单元的值

总体上，可以将取得一个数组的单元的值，看组取得一个变量的值完全一样！

# 数组分类

按键位关系来分！

索引数组：下标严格按按照从0开始的连续整数作为下标，则称为索引数组

关联数组：下标都是一个“字符串”并一定程度表名了该单元的“含义”则称为关联数组

混合数组：下标即有数字也有字符串的情况下，则称为混合数组

# 按数组的维数（复杂程度）分：

一维数组：array（1,11,111）；array（2.22.222）array（3.33.333）

二维数组：

Array（

array（1,11,111）；

array（2.22.222）

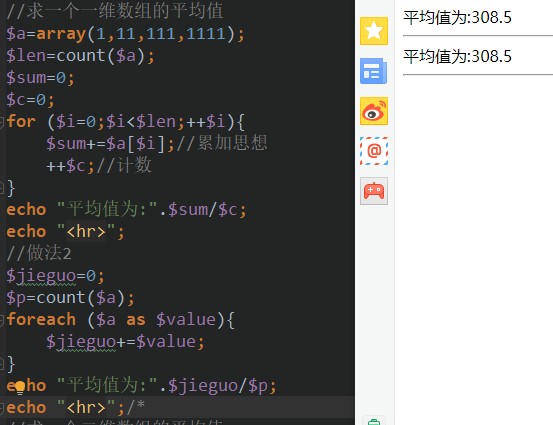
array（3.33.333）

）

多维数组：无非就是继续里面再用数组代替

# 数组的基本使用

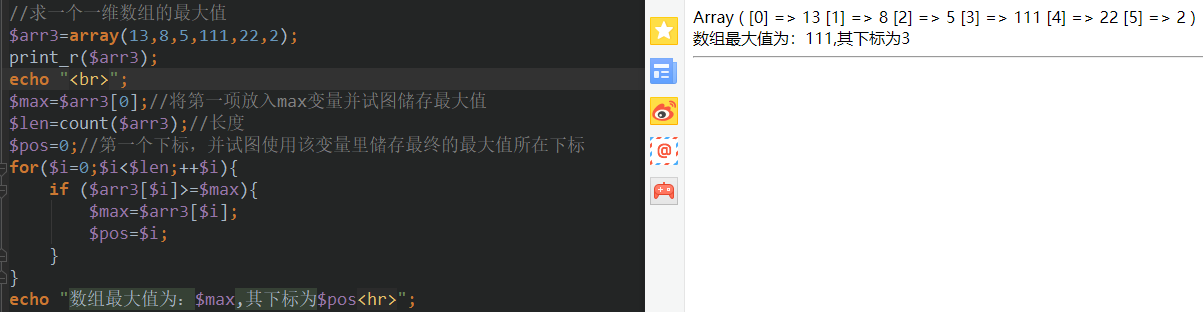
求一个一维数组的平均值



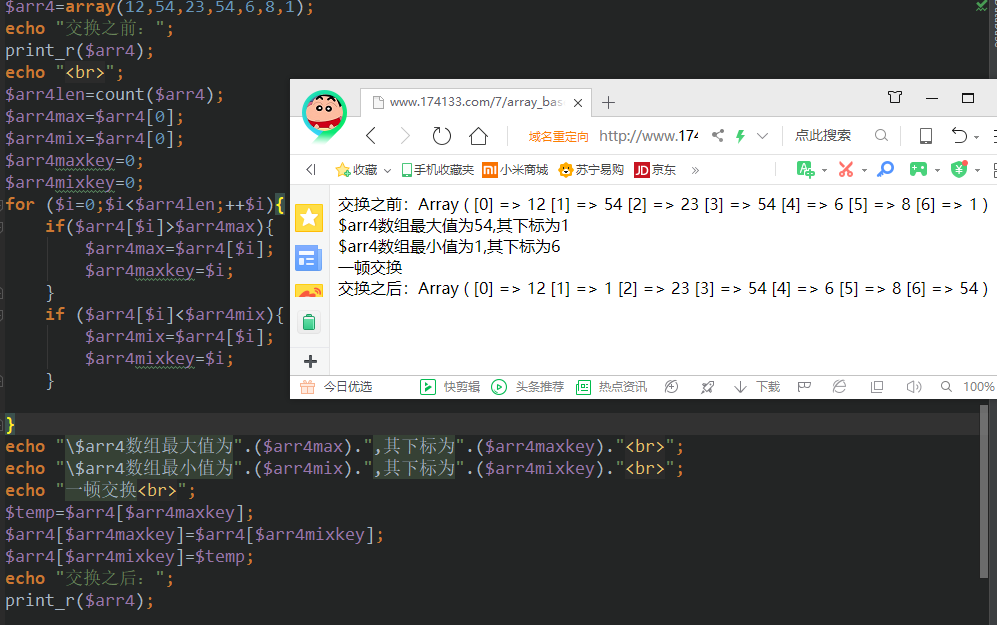
求一个二维数组的平均值



求一个一维数组的最大值



交换一个一维是数组的最大值和最小值



**数组的遍历**

Foreach基本语法

Foreach（$数组 as $key=>$value）{

//循环体 使用key和value

key和value就是该遍历语法一次次取得数组的每个单元（项）的下标和对应值

而且，它总是从数组的开头往后取数据

}

数组的指针操作及遍历原理

数组的一个形象图：

$arr4 = array(1=>3, ‘a1’=>11, 3=>5, ‘mn’=>18, 88=>2 );

可以将其以视觉化的方式理解为：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 数组下标： | 1 | “a1” | 3 | “mn” | 88 |
| 对应数据： | 8 | 11 | 5 | 18 | 2 |

其中，该箭头，就是数组内部的所谓“指针”——注意，不可见，不可输出，只是一种辅助理解的图形！

说明：

1，该箭头，就是数组内部的所谓“指针”

2，默认情况下，该指针指向数组的第一个单元。

3，数组的有关单元的操作，如果没有指定下标，则就是针对该指针指向的单元的操作。

4，所谓遍历，其实就是一次次取得当前单元的键和值，并放入对应的变量$key, $value,然后移动指针到下一个单元。

则，数组，作为一个 总体数据单位 ，有如下指针操作函数可以使用：

1，$v1 = current($数组) //获得数组的当做指针所在单元的“值”；

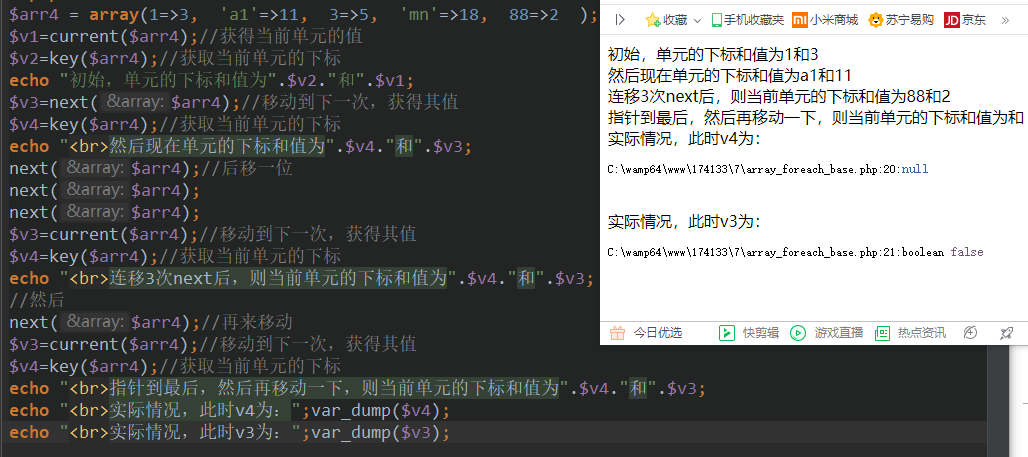
2，$v2 = key($数组); //获得数组的当做指针所在单元的“键”（下标）；

3，$v3 = next($数组); //先将数组的指针移向下（后）一个单元，然后取得该新单元的值；

4，$v4 = prev($数组); //先将数组的指针移向上（前）一个单元，然后取得该新单元的值；

5，$v5 = end($数组); //先将数组的指针直接移向最后一个单元，然后取得该新单元的值；

6，$v6 = reset($数组); //先将数组的指针直接移向第一个单元，然后取得该新单元的值；



# Foreach遍历流程原理图

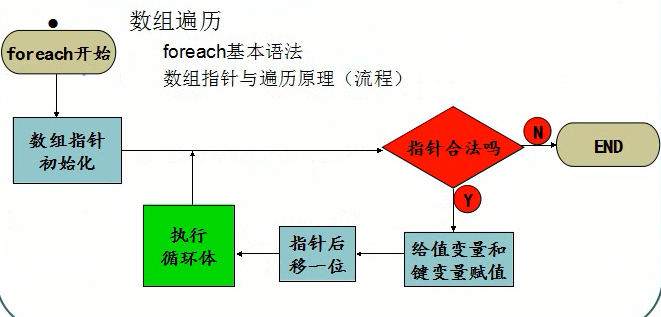
Foreach（$数组 as $key=>$value）{

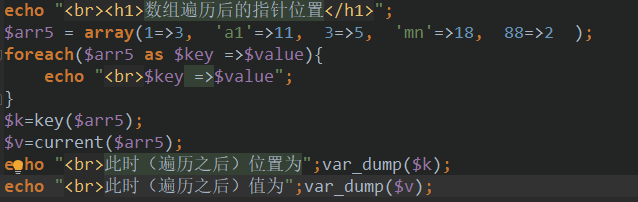
//循环体 使用key和value

key和value就是该遍历语法一次次取得数组的每个单元（项）的下标和对应值

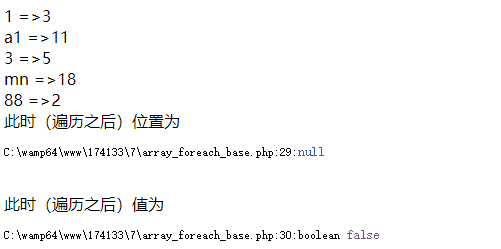
而且，它总是从数组的开头往后取数据

}





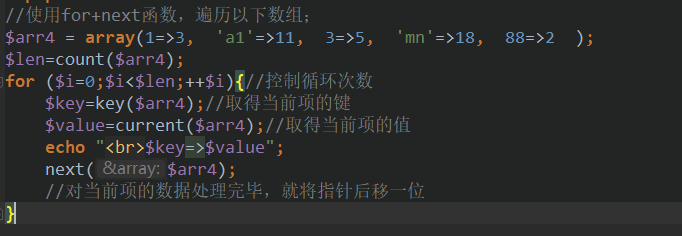
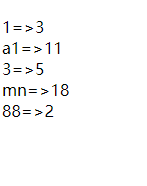
输出结果为：



# For和next遍历数组

注意：对PHP数组，不能单纯使用for循环进行遍历。

或者说：PHP中，使用for循环只能循环“下标为连续的纯整数数组”；

**Each（）函数的使用：（易取）**

**Eache（）函数的作用：先取得一个数组的当前单元的下标和值（并放到数组），然后将指针移到下一个单元；**

使用形式：

$a = each($数组名); //此时$a就是一个数组了



**取得arr4的某一项数据（键和值），但存储时做了双份存储**

**这种方式获取数据，即可以使用数字下标进行取用，也可以使用字符串下标进行使用**

**List()“函数”的使用（它不是函数）**

list()函数用于一次性取得一个数组中从0开始的数字下标的多个单元的值！

形式：

list($变量1，$变量2，$变量3， .. ） = $数组；

作用：

上述一行代码相当于如下代码：

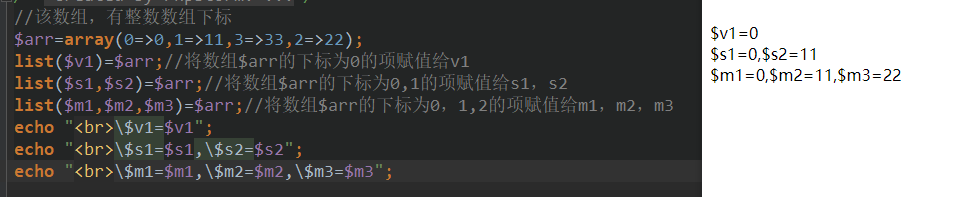
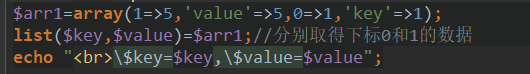
$变量1 = $数组[0];

$变量2 = $数组[1];

$变量3 = $数组[2];

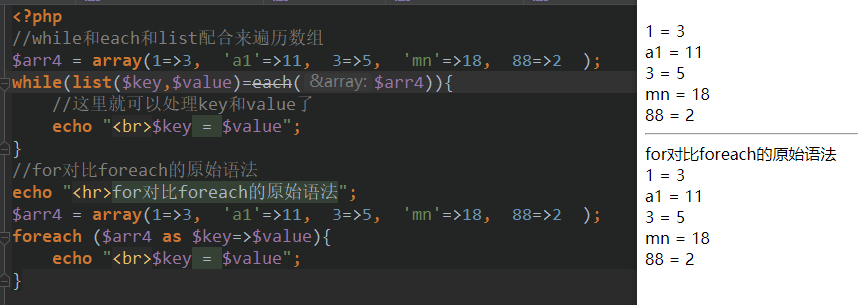
........

注意：这里变量的个数，要跟该数组的数字下标的单元对应，如果某个变量对应的该数字下标不存在，就会报错！

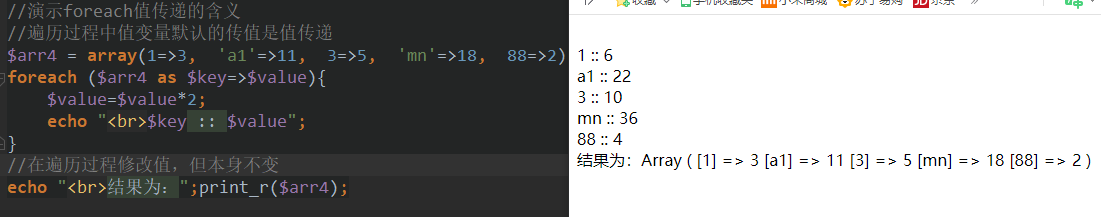
这里两个变量，跟数组arr1中的2个下标，key和value没有任何关系

# While+each+list遍历数组

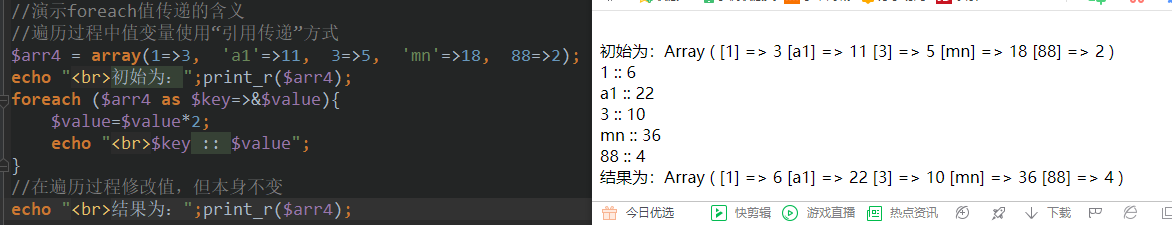


# 数组变量的一些细节

* foreach也是正常的循环语法结构，可以有break和continue等操作。



* 遍历过程中**值变量**默认的传值方式是值传递。
* 遍历过程中**值变量可以人为设定为引用传递**：foreach($arr as $key => &$value){ ... }

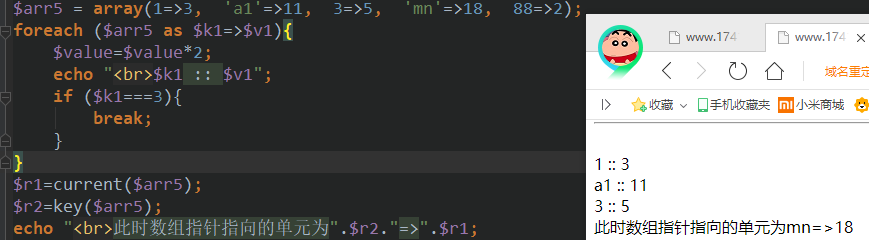


值变量$value可以使用引用传递：

注意：键（key）变量不可以

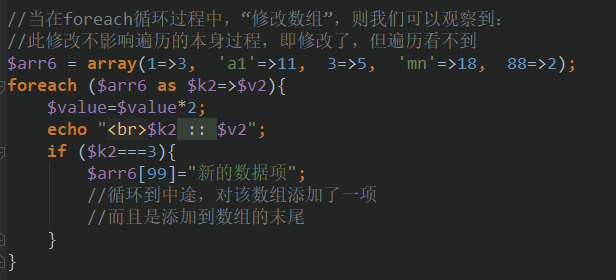
* foreach默认是原数组上进行遍历。但如果在遍历过程中对数组进行了某种修改或某种指针性操作，则会复制数组后在复制的数组上继续遍历循环（原数组保持不变）。

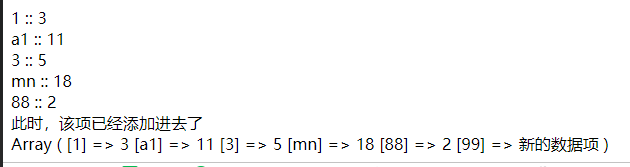
以下演示的是“foreach默认原数组上进行遍历”的效果



可见，数组的指针因为遍历中途退出，就停在该处

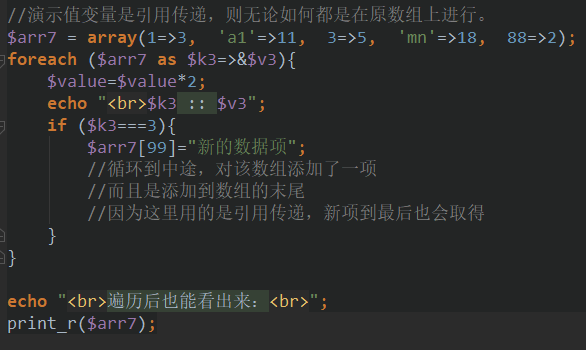
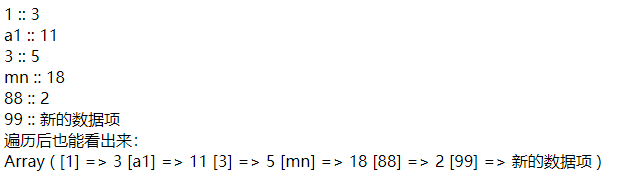
然后再演示：在某些情况下，不是在原数组身上进行遍历





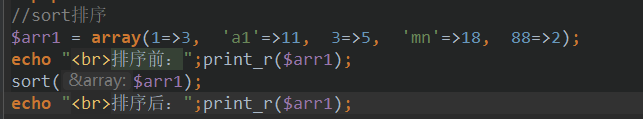
在进行添加数据项之前，会先将arr6这个数组复制一份，（假设为a6），然后foreach就在a6上进行继续遍历。此时，应该理解为foreach之前arr6是不同的变量，即foreach中arr6就是我们这里说的a6

* foreach中如果值变量是引用传递，则无论如何都是在原数组上进行。

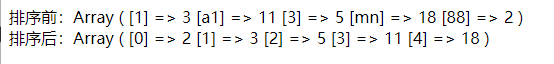
 

# 数组排序算法

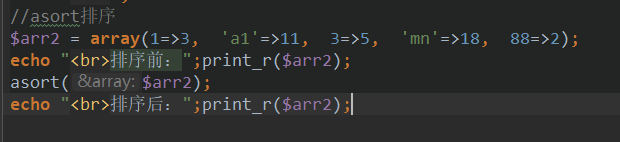
Sort排序

.

结果为



Asort排序



结果为



更多排序请查手册



手册》函数参考》与变量有关的扩展》数组》Sorting Arrays

冒泡排序算法

演示数组为：

$a = array(9, 3, 5, 8, 2, 7); //下标为0,1,2,3,4,5

演算过程描述：

从数组的左边开始，依次两两比较**相邻的2个数据**的大小，如果发现左边的比右边的大，则将他们进行交换。这样进行“一趟”之后，必然可以确定最大的一个数据放在最右边。

按此方式，对“剩余的数据”继续进行下一趟，则有会确定这些剩余数据耳朵最大值放在剩余位置的最右边。

演示：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 原始数组： | 9 | 3 | 5 | 8 | 2 | 7 |
| 第1趟后： | 3 | 5 | 8 | 2 | 7 | 9 |
| 第2趟后： | 3 | 5 | 2 | 7 | 8 | 9 |
| 第3趟后： | 3 | 2 | 5 | 7 | 8 | 9 |
| 第4趟后： | 2 | 3 | 5 | 7 | 8 | 9 |
| 第5趟后： | 2 | 3 | 5 | 7 | 8 | 9 |

规律描述：

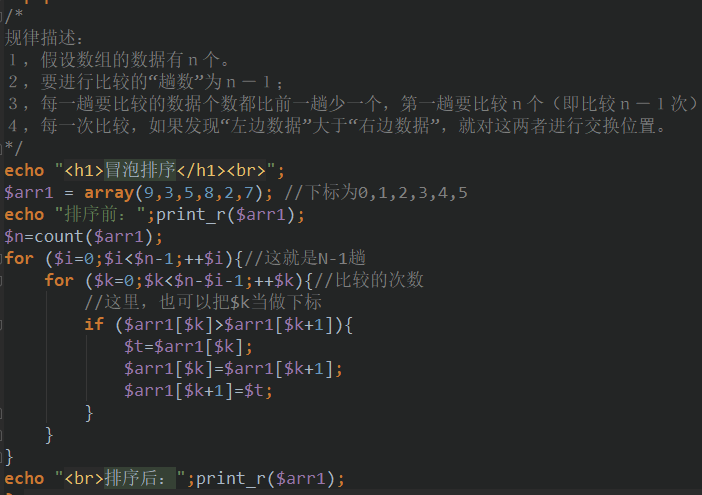
１，假设数组的数据有ｎ个。

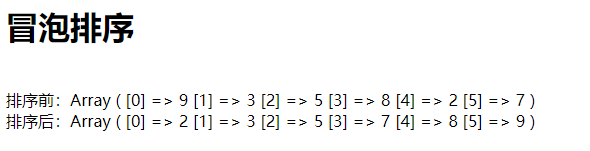
２，要进行比较的“趟数”为ｎ－１；

３，每一趟要比较的数据个数都比前一趟少一个，第一趟要比较ｎ个（即比较ｎ－１次）

４，每一次比较，如果发现“左边数据”大于“右边数据”，就对这两者进行交换位置。

代码如下：





选择排序算法

演示数组为：

$a = array(9, 3, 5, 8, 2, 7); //下标为0,1,2,3,4,5

演算过程描述：

求得一个数组的最大值的下标，并将这个最大值下标的单元跟最后一个单元进行交换；

然后，继续从剩余数据中取得最大值的下标，并将这个最大值下标的单元跟剩余的最后一个单元交换

以此类推，直到只剩下一个数据，就不用找了。

演示：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 原始数组： | 9 | 3 | 5 | 8 | 2 | 7 |
| 第1趟后： | 7 | 3 | 5 | 8 | 2 | 9 |
| 第2趟后： | 7 | 3 | 5 | 2 | 8 | 9 |
| 第3趟后： | 2 | 3 | 5 | 7 | 8 | 9 |
| 第4趟后： | 2 | 3 | 5 | 7 | 8 | 9 |
| 第5趟后： | 2 | 3 | 5 | 7 | 8 | 9 |

规律描述：

１，假设数组的数据有ｎ个。

２，要进行查找最大值单元并进行交换的“趟数”为ｎ－１；

3，每一趟都要求出“剩余数据”中的最大值单元，并且，剩余数据的数量每一趟都少1个，第一趟有n个。

4，每一趟找出最大值单元后，都要进行交换：最大值单元，跟剩余数据中的最后一个单元交换。

代码如下

