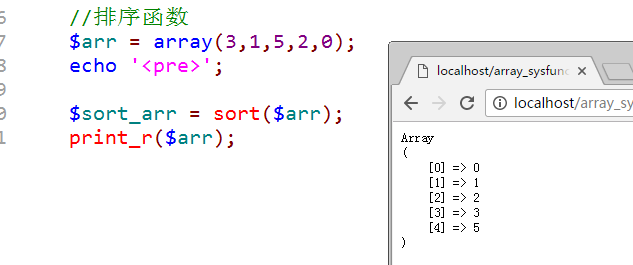
# 数组的相关函数

1）排序函数：对数组元素进行排序，都是按照ASCII码进行比较，可以进行英文比较

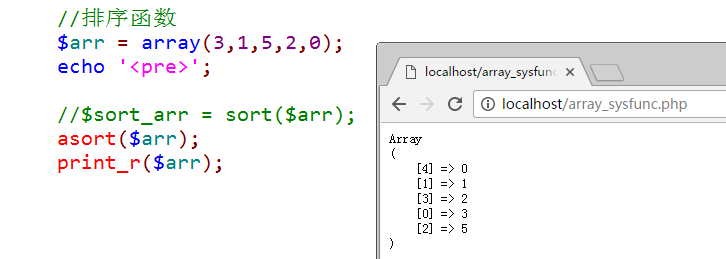
sort()：顺序排序（下标重排）

rsort()：逆序排序



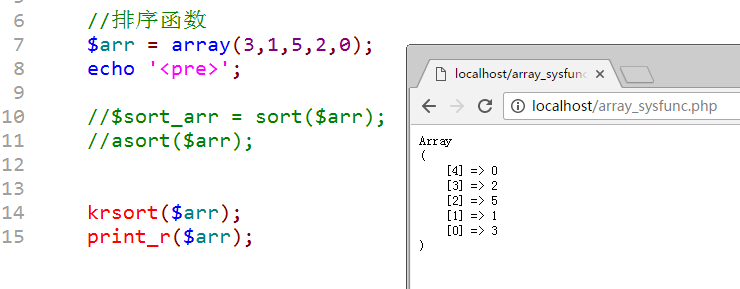
asort()：顺序排序（下标保留）

arsort()：逆序排序

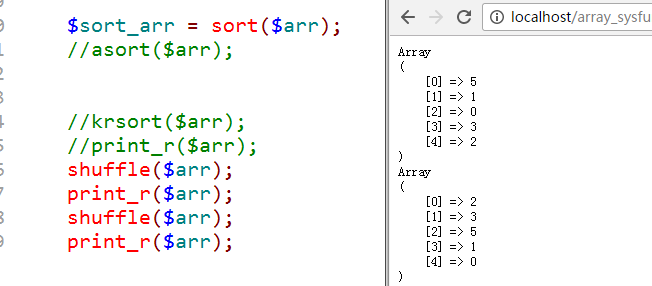


ksort()：顺序排序：按照键名（下标）

krsort()：逆序排序



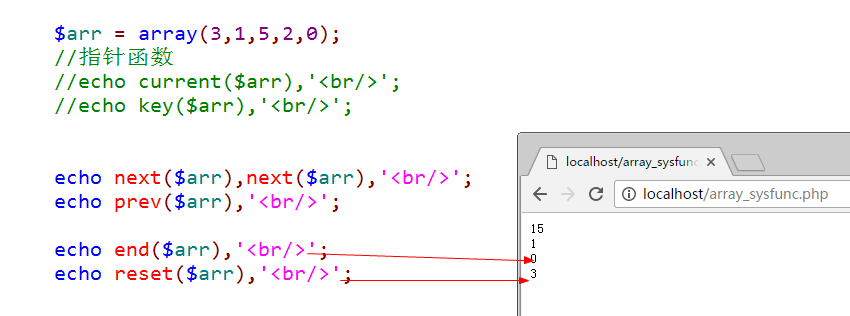
shuffle()：随机打乱数组元素，数组下标会重排



2）指针函数

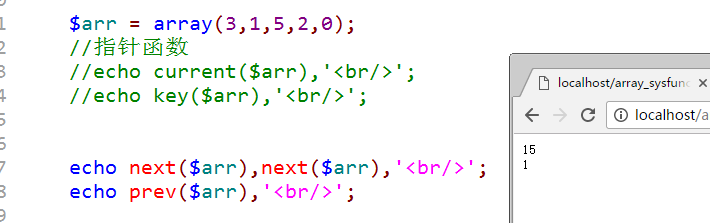
reset()：重置指针，将数组指针回到首位

end()：重置指针，将数组指针指导最后一个元素



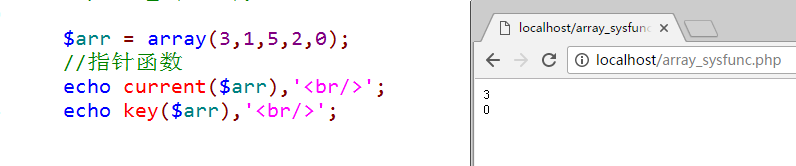
next()：指针下移，取得下一个元素的值

prev()：指针上移，取得上一个元素的值



current()：获取当前指针对应的元素值

key()：获取当前指针对应的下标值



注意事项：next和prev会移动指针，有可能导致指针移动到最前或者最后（离开数组），导致数组不能使用，通过next和prev不能回到真确的指针位置。只能通过end或者reset进行指针重置

3）其他函数

count()：统计数组中元素的数量

array\_push()：往数组中加入一个元素（数组后面）

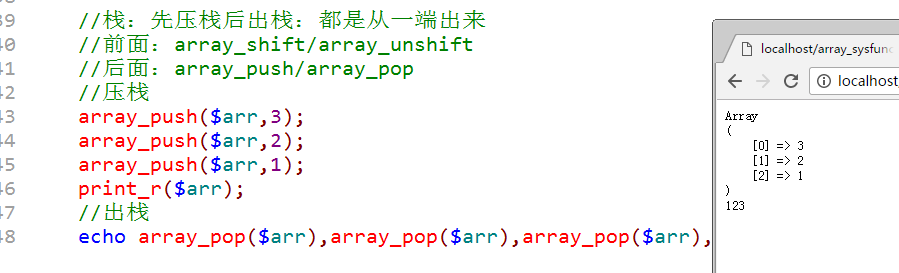
array\_pop()：从数组中取出一个元素（数组后面）

array\_shift()：从数组中取出一个元素（数组前面）

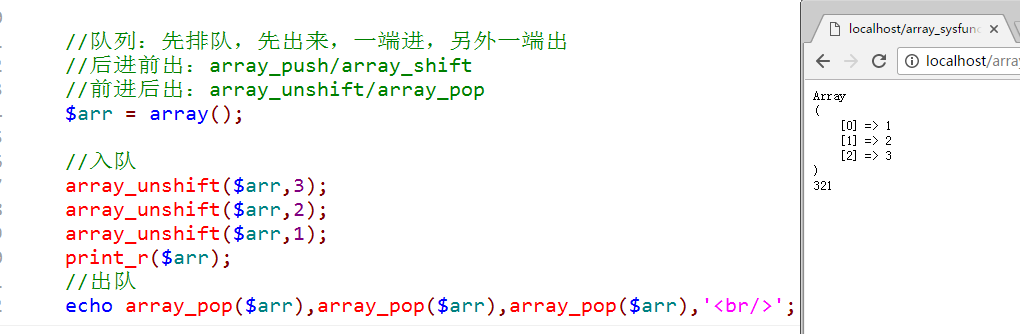
array\_unshift()：从数组中加入一个元素（数组前面）

PHP模拟数据结构：

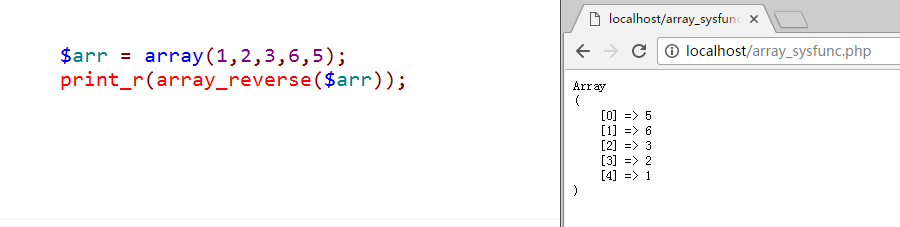
栈：压栈，先进去后出来（FILO）



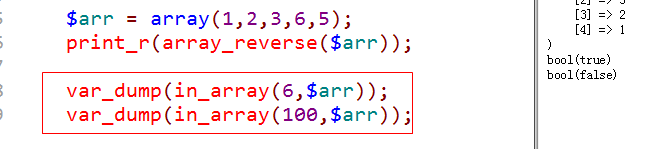
队列：排队，先进去的先出去（FIFO）



array\_reverse()：数组元素反过来

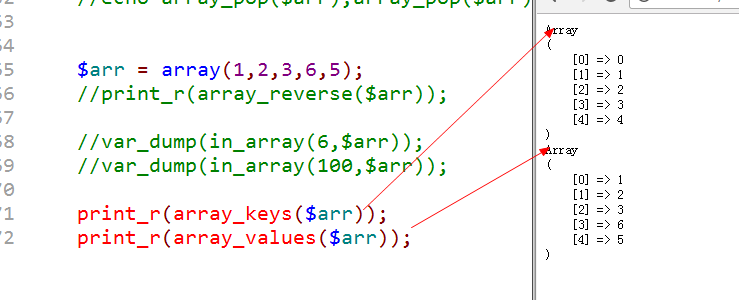


in\_array()：判断一个元素在数组中是否存在



array\_keys()：获取一个数组的所有下标，返回一个索引数组

array\_values()：获取一个数组的所有值，返回一个索引数组



# 编程思想

编程思想：如何利用数学模式，来解决对应的需求问题；然后利用代码实现对应的数据模型（逻辑）。

算法：使用代码实现对应的数学模型，从而解决对应的业务问题。

## 递推算法

递推算法是一种简单的算法，即通过已知条件，利用特定关系得出中间推论，直至得到结果的算法。递推算法分为顺推和逆推两种。

顺推：通过最简单的条件（已知），然后逐步推演结果

逆推：通过结果找到规律，然后推到已知条件

斐波那契数列：1 1 2 3 5 8 13 …，通常需求：请求得指定位置N所对应的值是多少

找规律：

1、 第一个数是1

2、 第二个数也是1

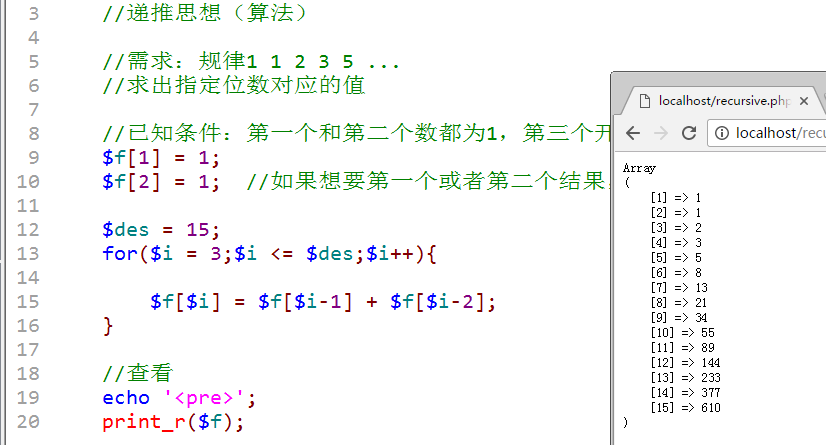
3、 从第三位开始：属于前两个数的和

代码解决思路：

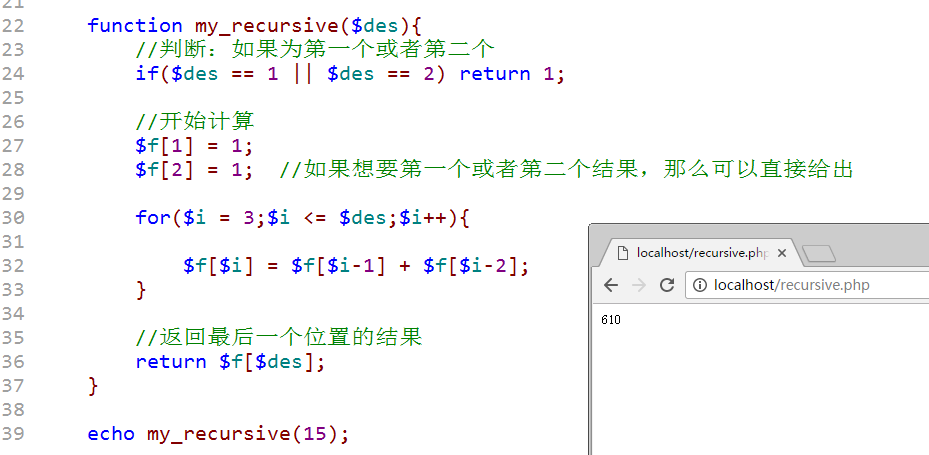
1、 如果数字位置为1和2，结果都是1

2、 从第三个开始，想办法得到前两个的结果，就可以得到

终极解决办法：想办法把要求的位置之前的所有的值都列出来，那么要求的数就可以通过前两个之和计算出来：使用数组存储所有结果即可。



递推算法求斐波那契数列：



# 编程思想

## 递归算法

递归算法是把问题转化为规模缩小了的同类问题的子问题。然后递归调用函数（或过程）来表示问题的解。

1、 简化问题：找到最优子问题（不能再小）

2、 函数自己调用自己

斐波那契数列：1 1 2 3 5 8 13 …

需求：求指定位置的数列的值

规律：第一个和第二个为1，从第三个开始为前两个之后

F(N) = F(N-1) + F(N-2);

F(N-1) = F(N-2) + F(N - 3);

…

F(2) = F(1) = 1;

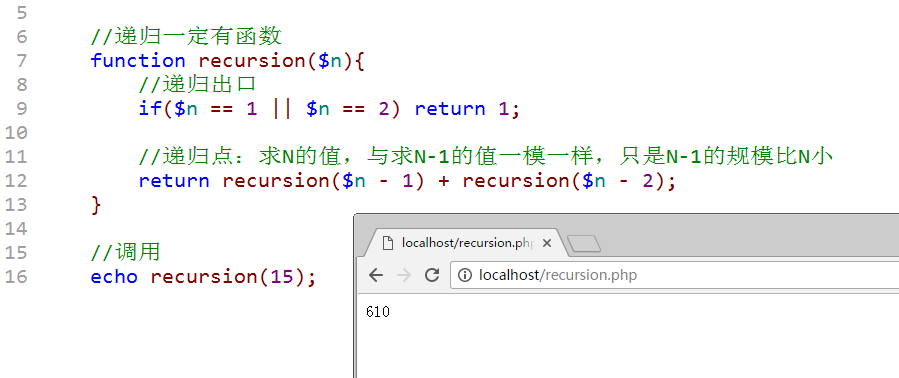
递归思想中：有两个非常重要的点

递归点：发现当前问题可以有解决当期问题的函数，去解决规模比当前小一点的问题来解决  
F(N) = F(N - 1) + F(N - 2);

递归出口：当问题解决的时候，已经到达（必须有）最优子问题，不能再次调用函数

如果一个函数递归调用自己而没有递归出口：就是死循环

递归的本质是函数调用函数：一个函数需要开辟一块内存空间，递归会出现同时调用N多个函数（自己）：递归的本质是利用空间换时间



# 数组排序算法

## 冒泡排序

冒泡排序（Bubble Sort），是一种计算机科学领域的较简单的排序算法。

它重复地走访过要排序的数列，一次比较两个元素，如果他们的顺序错误就把他们交换过来。走访数列的工作是重复地进行直到没有再需要交换，也就是说该数列已经排序完成。

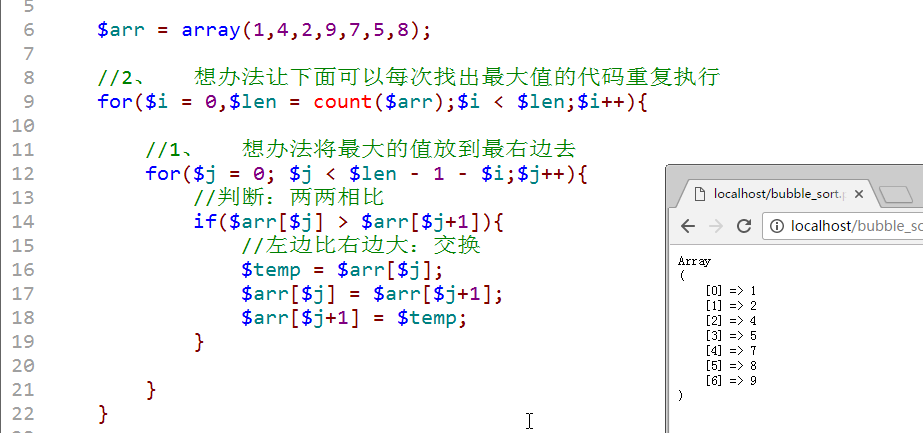
冒泡排序的算法思路：

1） 比较相邻的元素。如果第一个比第二个大，就交换他们两个。

2） 对每一对相邻元素作同样的工作，从开始第一对到结尾的最后一对。在这一点，最后的元素应该会是最大的数。

3） 针对所有的元素重复以上的步骤，除了最后一个。

4） 持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较。



# 选择排序

选择排序（Selection sort）是一种简单直观的排序算法。它的工作原理是每一次从待排序的数据元素中选出最小（或最大）的一个元素，存放在序列的起始位置，直到全部待排序的数据元素排完。 选择排序是不稳定的排序方法（比如序列[5， 5， 3]第一次就将第一个[5]与[3]交换，导致第一个5挪动到第二个5后面）。

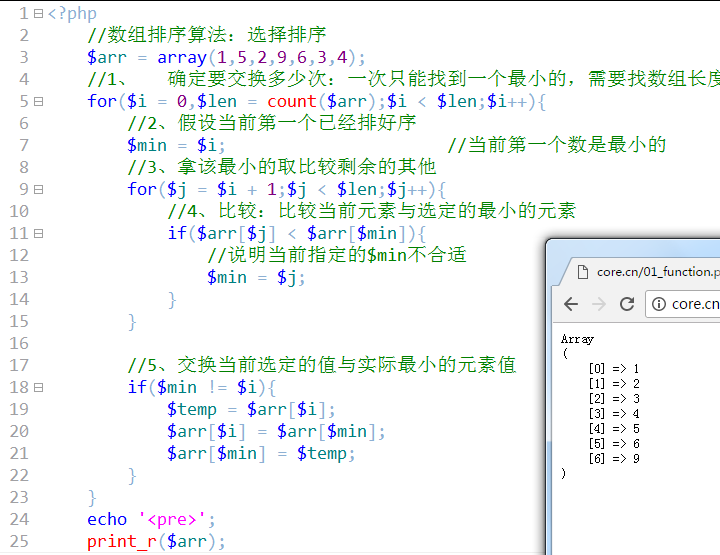
选择排序的算法思路：

1） 假设第一个元素为最小元素，记下下标。

2） 寻找右侧剩余的元素，如果有更小的，重新记下最新的下标。

3） 如果有新的最小的，交换两个元素。

4） 往右重复以上步骤，直到元素本身是最后一个。



# 插入排序

插入排序（Insert sort）,插入排序的基本操作就是将一个数据插入到已经排好序的有序数据中，从而得到一个新的、个数加一的有序数据，算法适用于少量数据的排序，是稳定的排序方法。插入算法把要排序的数组分成两部分：第一部分包含了这个数组的所有元素，但将最后一个元素除外（让数组多一个空间才有插入的位置），而第二部分就只包含这一个元素（即待插入元素）。在第一部分排序完成后，再将这个最后元素插入到已排好序的第一部分中。

插入排序的基本思想是：每步将一个待排序的纪录，按其关键码值的大小插入前面已经排序的文件中适当位置上，直到全部插入完为止。

插入排序的算法思路：

1） 设置监视哨r[0]，将待插入纪录的值赋值给r[0]；

2） 设置开始查找的位置j；

3） 在数组中进行搜索，搜索中将第j个纪录后移，直至r[0].key≥r[j].key为止；

4） 将r[0]插入r[j+1]的位置上。

1、 认定第一个元素已经排好序；

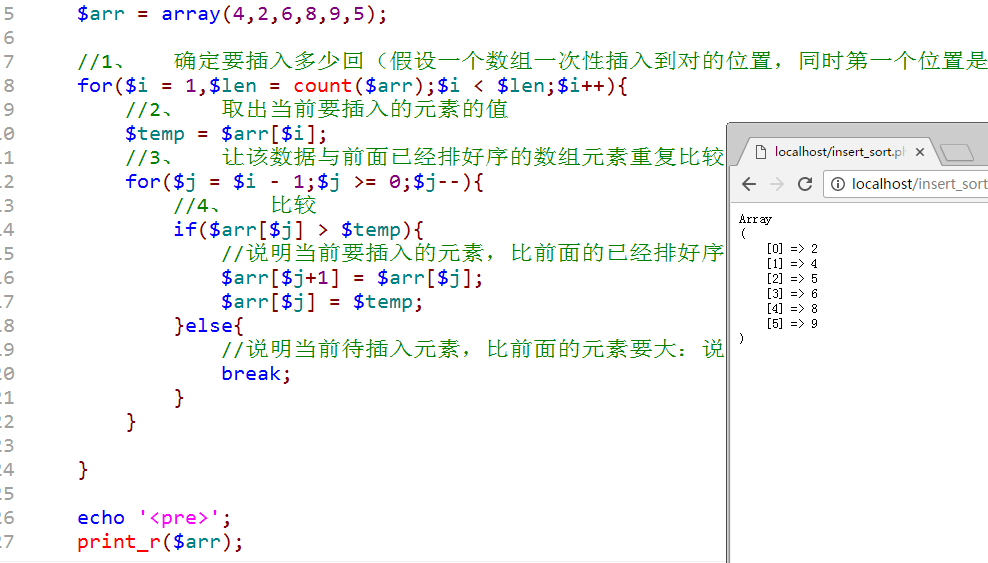
2、 取出第二个元素，作为待插入数据；

3、 与已经排好序的数组的最右侧元素开始进行比较

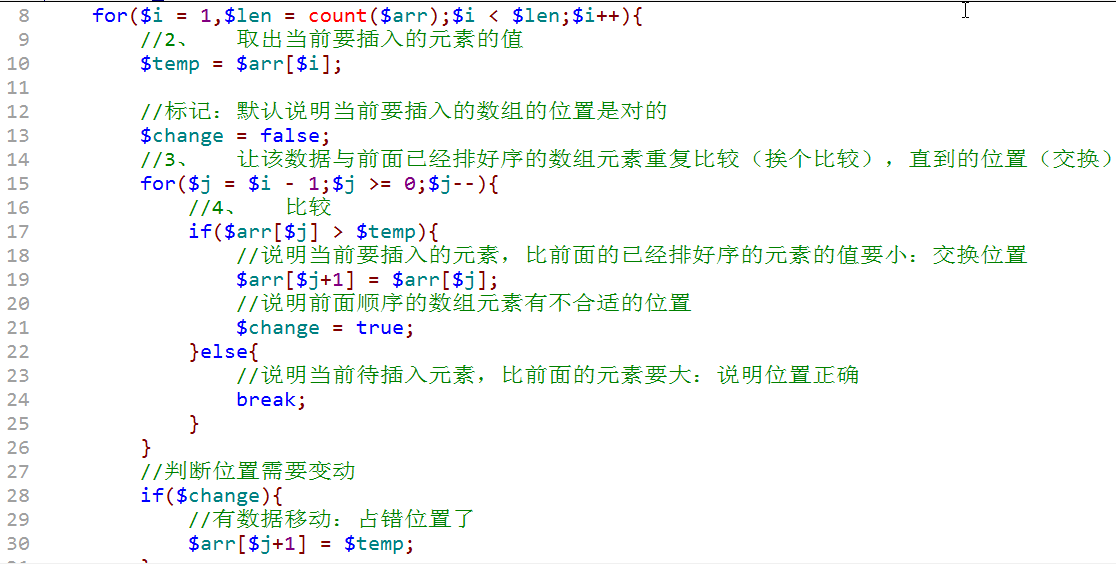
4、 如果后面的小于前面的：说明前面已经排好序的那个数组元素不在对的位置（向后移一个），然后让新的元素填充进去（继续向前比：高级）

5、 重复前面的步骤：直到当前元素插入到对位置；

6、 重复以上步骤，直到所有的数组元素都插入到对的位置。



优化代码：找到对的位置交换一次即可



# 快速排序

快速排序（Quicksort）是对冒泡排序的一种改进。通过一趟排序将要排序的数据分割成独立的两部分，其中一部分的所有数据都比另外一部分的所有数据都要小，然后再按此方法对这两部分数据分别进行快速排序，整个排序过程可以递归进行，以此达到整个数据变成有序序列。（递归）

设要排序的数组是A[0]……A[N-1]，首先任意选取一个数据（通常选用数组的第一个数）作为关键数据，然后将所有比它小的数都放到它前面，所有比它大的数都放到它后面，这个过程称为一趟快速排序。值得注意的是，快速排序不是一种稳定的排序算法，也就是说，多个相同的值的相对位置也许会在算法结束时产生变动。

快速排序的算法是：

1） 从数组中选出一个元素（通常第一个），作为参照对象。

2） 定义两个数组，将目标数组中剩余的元素与参照元素挨个比较：小的放到一个数组，大的放到另外一个数组。

3） 第二步执行完之后，前后的数组顺序不确定，但是确定了自己的位置。

4） 将得到的小数组按照第1到第3部重复操作（子问题）。

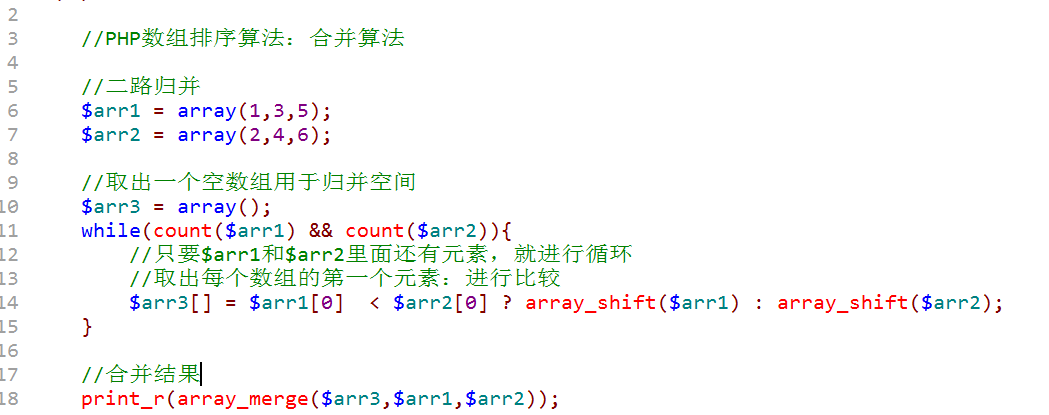
5） 回溯最小数组（一个元素）。



# 归并排序

归并排序（MERGE-SORT）是建立在归并操作上的一种有效的排序算法,该算法是采用分治法（Divide and Conquer）的一个非常典型的应用。将已有序的子序列合并，得到完全有序的序列；即先使每个子序列有序，再使子序列段间有序。若将两个有序表合并成一个有序表，称为二路归并。

二路归并实现：



归并排序的算法是：

1） 将数组拆分成两个数组。

2） 重复步骤1将数组拆分成最小单元。

3） 申请空间，使其大小为两个已经排序序列之和，该空间用来存放合并后的序列.

4） 设定两个指针，最初位置分别为两个已经排序序列的起始位置。

5） 比较两个指针所指向的元素，选择相对小的元素放入到合并空间，并移动指针到下一位置。

6） 重复步骤3直到某一指针超出序列尾。

7） 将另一序列剩下的所有元素直接复制到合并序列尾。



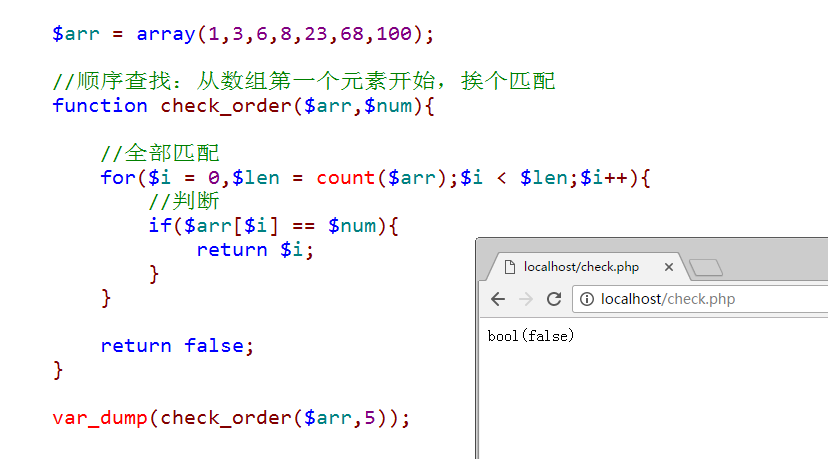
# 查找算法

## 查找算法含义

查找是在大量的信息中寻找一个特定的信息元素，在计算机应用中，查找是常用的基本运算。查找算法是指实现查找过程对应的代码结。就是中大型数组中去快速定位想要的元素。

## 顺序查找算法

顺序查找也称为线形查找，从数据结构线形表的一端开始，顺序扫描，依次将扫描到的结点关键字与给定值k相比较，若相等则表示查找成功；若扫描结束仍没有找到关键字等于k的结点，表示查找失败。



## 二分查找算法

二分查找要求线形表中的结点按关键字值升序或降序排列，用给定值k先与中间结点的关键字比较，中间结点把线形表分成两个子表，若相等则查找成功；若不相等，再根据k与该中间结点关键字的比较结果确定下一步查找哪个子表，这样递归进行，直到查找到或查找结束发现表中没有这样的结点。

折半算法思路：

1、 计算数组长度；

2、 确定左右两边的指针位置；

3、 找到中间位置；

4、 匹配

5、 然后根据大小重定边界

