冒泡排序的原理是：多次遍历待排序的元素序列，每次比较相邻两个元素，如果它们的顺序不对，则交换它们。这个过程持续进行，直到整个序列有序。

而在起泡排序过程中，某些关键字可能在某一步朝着最终排序相反的方向移动，但通过多次遍历和比较，最终序列会趋向有序。

举例如下，当我们输入一个含有十个元素的数组num[10]={5,3,8,4,2,1,7,6,9,10}时，如果我们采用标准冒泡排序将该数组排序成从小到大的形式，并且每完成一次排序后均输出一遍当时的数组形式时，我们会发现它的排序过程如下：

5 3 8 4 2 1 7 6 9 10

3 5 8 4 2 1 7 6 9 10

3 5 4 8 2 1 7 6 9 10

3 5 4 2 8 1 7 6 9 10

3 5 4 2 1 8 7 6 9 10

3 5 4 2 1 7 8 6 9 10

3 5 4 2 1 7 6 8 9 10

3 4 5 2 1 7 6 8 9 10

3 4 2 5 1 7 6 8 9 10

3 4 2 1 5 7 6 8 9 10

3 4 2 1 5 6 7 8 9 10

3 2 4 1 5 6 7 8 9 10

3 2 1 4 5 6 7 8 9 10

2 3 1 4 5 6 7 8 9 10

2 1 3 4 5 6 7 8 9 10

1 2 3 4 5 6 7 8 9 10  
  
 我们关注该排序中3和4的变化可以轻松的看出，在起泡排序过程中，有的关键字确实会在某一次起泡过程中可能朝着与最终排序相反的方向移动。

这是因为冒泡排序总是和相邻的元素进行比较，如果num[4]其实是num数组中第8大的。那么它的真实位置应该是num[7]，但是如果此时num[3]是数组中第9大的，显然num[3]>num[4]，那么冒泡排序就会暂时将num[3]和num[4]对换，这就造成了该关键字暂时向与最终排序相反的方向移动。

代码实现如下：

#include<stdio.h>

void printnum(int num[10]){

for(int i=0;i<9;i++){

printf("%d ",num[i]);

}

printf("%d\n",num[9]);

}

int main(){

int num[10]={5,3,8,4,2,1,7,6,9,10};

printnum(num);

for(int i=0;i<9;i++){

for(int j=0;j<9-i;j++){

if(num[j]>num[j+1]){

int temp=num[j];

num[j]=num[j+1];

num[j+1]=temp;

printnum(num);

}

}

}

return 0;

}