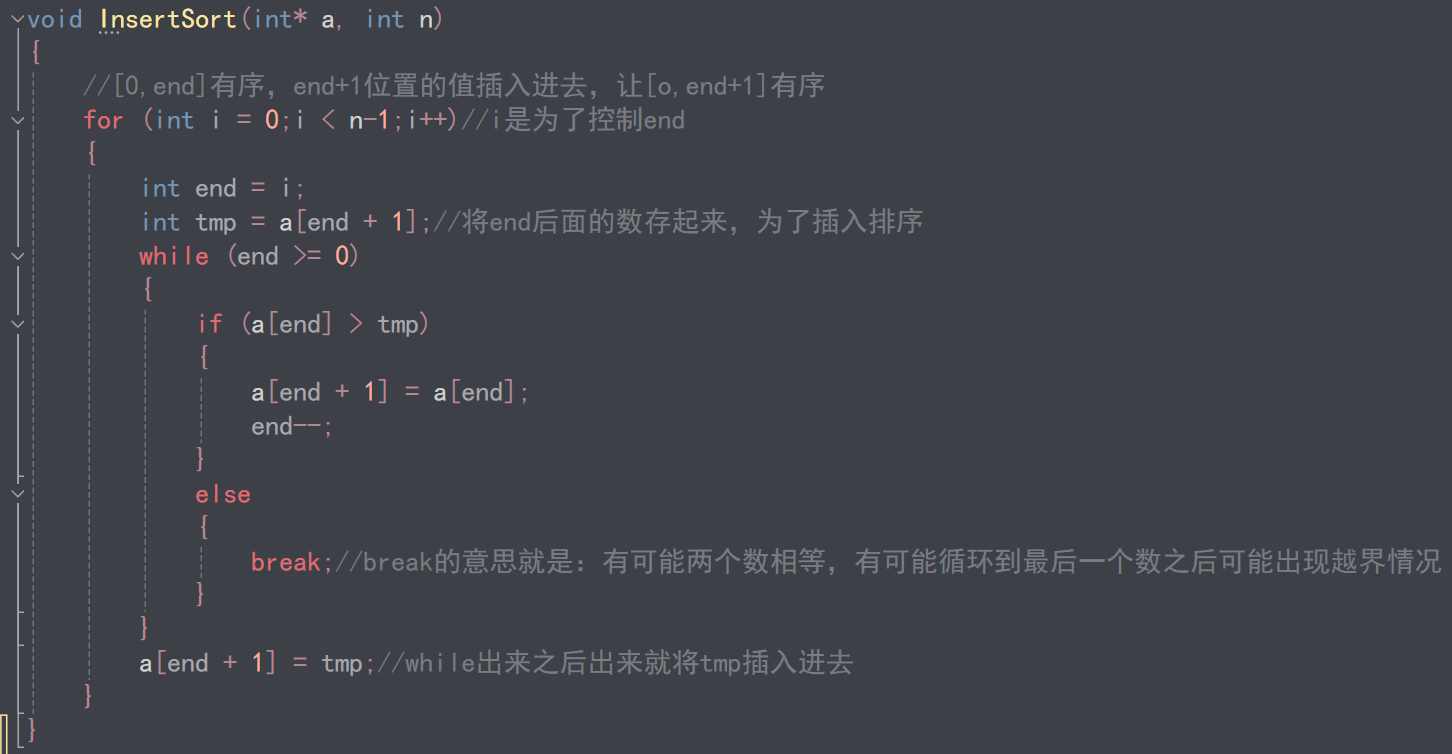
**排序**

1. **插入排序**

****

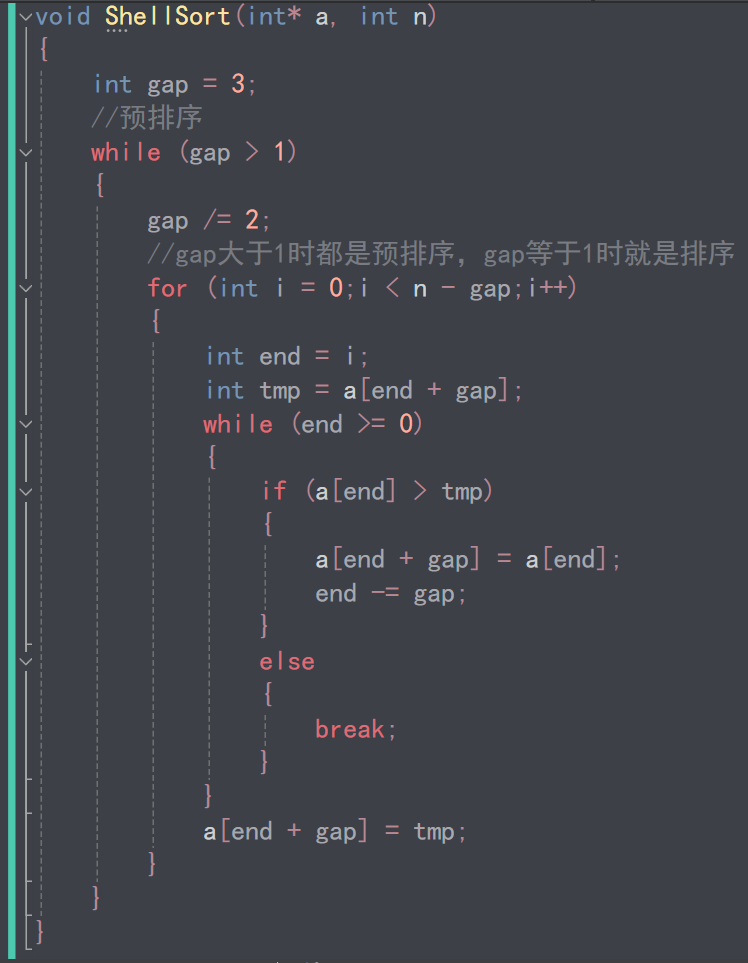
**插入排序时间复杂度是n^2**

**最坏情况：1+2+3+4+5+……+n-1（逆序）**

**最坏的情况 ：1 2 3 4 5 6 7 8 9……n（顺序）**

1. **希尔排序**

**时间复杂度是n\*logn**

****

1. **堆排序**

**什么是堆？**

**堆有两种结构，一种是二叉树，一种是数组表示**

**堆的逻辑结构是一颗完全二叉树，堆的物理结构是一个数组**

**堆的两个特性**

1. **结构性：用数组表示的完全二叉树；**
2. **有序性：任一结点的关键字是其子树所有节点的最大值（或最小值）**

**(1)、“最大堆”，也称“大顶堆”：最大值**

**(2)、“最小堆”，也称“小顶堆”：最小值**

**完全二叉树的特点：**

**第一层有一个，第二层有两个，第三层有四个，第四层有八个，以此类推**

**通过下标父子节点关系:**

**Leftchild=parent\*2+1**

**Rightchild=parent\*2+2**

**Parent=（child-1）/2**

**大堆的要求：所有的父亲都大于等于孩子（根是最大的）**

**小堆的要求：所有的父亲都小于等于孩子（根是最小的）**

**堆排序的本质是选择排序**

**建堆的方法：向下调整算法**

**前提：左右子树都是小堆**

**从根节点开始，选出左右孩子中小的那一个，跟父亲比较，如果比父亲小就交换，然后再继续往下调**

**左右子树不是小堆的 时候，就不能用向下调整算法，怎么办？**

**可以倒着从最后一颗子树开始调整，再分析倒着走，叶子不需要调整，从最后一个非叶子节点的子树开始调。**

**For(int i=(n-1-1)/2;i>=0;i++)**

**{  
 AdjustDown(a,n,i);**

**}**

**排升序建大堆。**

**Why？**

**选择排序，通过堆来选数，排升序建大堆。**

**如果建小堆，最小的数在堆顶，已经被选出来了。那么在剩下的数中再去选数，但是剩下的数结构都乱了，需要重新建堆才能选出下一个数，建堆的时间复杂度是ON，那么这样不是不可以，但是堆排序就没有效率优势。（建堆的时间复杂度是ON）**

1. **直接选择排序**