**一. 概述**

|  |  |
| --- | --- |
| sort() | 对所有元素排序 |
| stable\_sort() | 对所有元素排序，并保持相等元素间的相对次序 |
| partial\_sort() | 排序，直到前n个元素就位 |
| partial\_sort\_copy() | 排序，直到前n个元素就位，结果复制于它处 |
| nth\_element() | 根据第n个位置进行排序 |
| partition() | 改变元素次序，使符合某准则的元素在前面 |
| stable\_partition() | 与partition()相同，但保持相对位置 |
| make\_heap() | 将一个区间转换成一个heap |
| push\_heap() | 将元素加入一个heap |
| pop\_heap() | 从heap移除一个元素 |
| sort\_heap() | 对heap进行排序，排序后不再是heap了 |

**二. 对所有元素排序**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7331856)

1. //排序，默认是从小到大顺序
2. void
3. sort (RandomAccessIterator beg, RandomAccessIterator end)
4. //排序，使用op(elem1, elem2)准则
5. void
6. sort (RandomAccessIterator beg, RandomAccessIterator end,
7. BinaryPredicate op)
8. //保证相等元素的原本相对次序在排序后保持不变
9. //换句话出，相等的元素它会按照固有的顺序，比如字典顺序排序
10. void
11. stable\_sort (RandomAccessIterator beg, RandomAccessIterator end)
12. void
13. stable\_sort (RandomAccessIterator beg, RandomAccessIterator end,
14. BinaryPredicate op)

//排序，默认是从小到大顺序

void

sort (RandomAccessIterator beg, RandomAccessIterator end)

//排序，使用op(elem1, elem2)准则

void

sort (RandomAccessIterator beg, RandomAccessIterator end,

BinaryPredicate op)

//保证相等元素的原本相对次序在排序后保持不变

//换句话出，相等的元素它会按照固有的顺序，比如字典顺序排序

void

stable\_sort (RandomAccessIterator beg, RandomAccessIterator end)

void

stable\_sort (RandomAccessIterator beg, RandomAccessIterator end,

BinaryPredicate op)

**三. 局部排序**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7331856)

1. //以 < 对区间[beg, end)内的元素进行排序，使区间[beg, sortEnd)内的元素有序
2. void
3. partial\_sort (RandomAccessIterator beg,
4. RandomAccessIterator sortEnd,
5. RandomAccessIterator end)
6. //以op(elem1, elem2)对区间内元素进行排序
7. void
8. partial\_sort (RandomAccessIterator beg,
9. RandomAccessIterator sortEnd,
10. RandomAccessIterator end,
11. BinaryPredicate op)

//以 < 对区间[beg, end)内的元素进行排序，使区间[beg, sortEnd)内的元素有序

void

partial\_sort (RandomAccessIterator beg,

RandomAccessIterator sortEnd,

RandomAccessIterator end)

//以op(elem1, elem2)对区间内元素进行排序

void

partial\_sort (RandomAccessIterator beg,

RandomAccessIterator sortEnd,

RandomAccessIterator end,

BinaryPredicate op)

**四. 根据第n个元素排序**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7331856)

1. //对区间[beg, end)内的元素进行排序，使第n个位置上的元素就位
2. //也就是n之前的元素都小于等于它，n之后的元素都大于等于它
3. //但是n前后的元素不要求有序
4. void
5. nth\_element (RandomAccessIterator beg,
6. RandomAccessIterator nth,
7. RandomAccessIterator end)
8. void
9. nth\_element (RandomAccessIterator beg,
10. RandomAccessIterator nth,
11. RandomAccessIterator end,
12. BinaryPredicate op)

//对区间[beg, end)内的元素进行排序，使第n个位置上的元素就位

//也就是n之前的元素都小于等于它，n之后的元素都大于等于它

//但是n前后的元素不要求有序

void

nth\_element (RandomAccessIterator beg,

RandomAccessIterator nth,

RandomAccessIterator end)

void

nth\_element (RandomAccessIterator beg,

RandomAccessIterator nth,

RandomAccessIterator end,

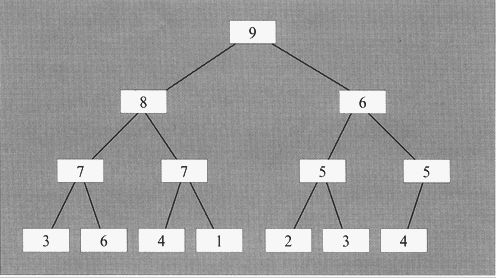
BinaryPredicate op)

**五. Heap算法**

heap可以看成一个二叉树，具有两大性质：

1. 第一个元素总是最大。

2. 总是能够在对数时间内增加或删除一个元素。



**1. 将某区间的元素转化成heap**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7331856)

1. void
2. make\_heap (RandomAccesIterator beg, RandomAccesIterator end)
3. void
4. make\_heap (RandomAccesIterator beg, RandomAccesIterator end,
5. BinaryPredicate op)

void

make\_heap (RandomAccesIterator beg, RandomAccesIterator end)

void

make\_heap (RandomAccesIterator beg, RandomAccesIterator end,

BinaryPredicate op)

**2. 对heap增加一个元素**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7331856)

1. //将end之前最后一个元素加入原本就是个heap的[beg, end-1)区间内
2. //使整个区间[beg, end)成为一个heap
3. void
4. push\_heap (RandomAccesIterator beg, RandomAccesIterator end)
5. void
6. push\_heap (RandomAccesIterator beg, RandomAccesIterator end,
7. BinaryPredicate op)

//将end之前最后一个元素加入原本就是个heap的[beg, end-1)区间内

//使整个区间[beg, end)成为一个heap

void

push\_heap (RandomAccesIterator beg, RandomAccesIterator end)

void

push\_heap (RandomAccesIterator beg, RandomAccesIterator end,

BinaryPredicate op)

**3. 对heap取出下一个元素**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7331856)

1. //将heap[beg, end)内最高元素，也就是第一个元素，移到最后位置
2. //并将剩余区间[beg, end-1)组成一个新的heap
3. void
4. pop\_heap (RandomAccesIterator beg, RandomAccesIterator end)
5. void
6. pop\_heap (RandomAccesIterator beg, RandomAccesIterator end,
7. BinaryPredicate op)

//将heap[beg, end)内最高元素，也就是第一个元素，移到最后位置

//并将剩余区间[beg, end-1)组成一个新的heap

void

pop\_heap (RandomAccesIterator beg, RandomAccesIterator end)

void

pop\_heap (RandomAccesIterator beg, RandomAccesIterator end,

BinaryPredicate op)

**4. 将heap转换成一个已序序列**

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7331856)

1. //进入函数前[beg, end)必须是一个heap, 排序后不是一个heap了
2. void
3. sort\_heap (RandomAccesIterator beg, RandomAccesIterator end)
4. void
5. sort\_heap (RandomAccesIterator beg, RandomAccesIterator end,
6. BinaryPredicate op)