C++ sort()排序函数用法详解

C++ STL 标准库提供有很多实用的排序函数，如表 1 所示。通过调用它们，我们可以很轻松地实现对普通数组或者容器中指定范围内的元素进行排序。

C++中标准库的排序函数

函数名	用法
sort (first, last)	对容器或普通数组中 [first, last) 范围内的元素进行排序，默认进行升序排序。
stable_sort (first, last)	和 sort() 函数功能相似，不同之处在于，对于 [first, last) 范围内值相同的元素，该函数不会改变它们的相对位置。
partial_sort (first, middle, last)	从 [first,last) 范围内，筛选出 muddle-first 个最小的元素并排序存放在 [first，middle) 区间中。
partial_sort_copy (first, last, result_first, result_last)	从 [first, last) 范围内筛选出 result_last-result_first 个元素排序并存储到 [result_first, result_last) 指定的范围中。
is_sorted (first, last)	检测 [first, last) 范围内是否已经排好序，默认检测是否按升序排序。
is_sorted_until (first, last)	和 is_sorted() 函数功能类似，唯一的区别在于，如果 [first, last) 范围的元素没有排好序，则该函数会返回一个指向首个不遵循排序规则的元素的迭代器。
void nth_element (first, nth, last)	找到 [first, last) 范围内按照排序规则（默认按照升序排序）应该位于第 nth 个位置处的元素，并将其放置到此位置。同时使该位置左侧的所有元素都比其存放的元素小，该位置右侧的所有元素都比其存放的元素大。

C++ sort()排序函数

C++ STL 标准库中的 sort() 函数，本质就是一个模板函数。正如表 1 中描述的，该函数专门用来对容器或普通数组中指定范围内的元素进行排序，排序规则默认以元素值的大小做升序排序，除此之外我们也可以选择标准库提供的其它排序规则（比如std::greater 降序排序规则），甚至还可以自定义排序规则。

> sort() 函数是基于快速排序实现的，有关快速排序的具体实现过程，感兴趣的读者可阅读《快速排序（QSort，快排）算法》一文。

需要注意的是，sort() 函数受到底层实现方式的限制，它仅适用于普通数组和部分类型的容器。换句话说，只有普通数组和具备以下条件的容器，才能使用 sort() 函数：

- 容器支持的迭代器类型必须为随机访问迭代器。这意味着，sort() 只对 array、vector、deque 这 3 个容器提供支持。
- 如果对容器中指定区域的元素做默认升序排序，则元素类型必须支持<小于运算符；同样，如果选用标准库提供的其它排序规则，元素类型也必须支持该规则底层实现所用的比较运算符；
- sort() 函数在实现排序时，需要交换容器中元素的存储位置。这种情况下，如果容器中存储的是自定义的类对象，则该类的内部必须提供移动构造函数和移动赋值运算符。

sort() 函数有 2 种用法，其语法格式分别为：

```
//对 [first, last) 区域内的元素做默认的升序排序
void sort (RandomAccessIterator first, RandomAccessIterator last);
//按照指定的 comp 排序规则，对 [first, last) 区域内的元素进行排序
void sort (RandomAccessIterator first, RandomAccessIterator last, Compare comp);
```

程序示例

```
#include <iostream>     // std::cout
#include <algorithm>    // std::sort
#include <vector>       // std::vector
//以普通函数的方式实现自定义排序规则
bool mycomp(int i, int j) {
    return (i < j);
}
//以函数对象的方式实现自定义排序规则
class mycomp2 {
public:
    bool operator() (int i, int j) {
```

```
        return (i < j);
    }
};
int main() {
    std::vector<int> myvector{ 32, 71, 12, 45, 26, 80, 53, 33 };
    //调用第一种语法格式，对 32、71、12、45 进行排序
    std::sort(myvector.begin(), myvector.begin() + 4); //(12 32 45 71) 26 80 53 33
    //调用第二种语法格式，利用STL标准库提供的其它比较规则（比如 greater<T>）进行排序
    std::sort(myvector.begin(), myvector.begin() + 4, std::greater<int>()); //(71
 45 32 12) 26 80 53 33

    //调用第二种语法格式，通过自定义比较规则进行排序
    std::sort(myvector.begin(), myvector.end(), mycomp2());//12 26 32 33 45 53 71
 80
    //输出 myvector 容器中的元素
    for (std::vector<int>::iterator it = myvector.begin(); it != myvector.end(); +
+it) {
        std::cout << *it << ' ';
    }
    return 0;
}
```

程序执行结果为：

> 12 26 32 33 45 53 71 80

可以看到，程序中分别以函数和函数对象的方式自定义了具有相同功能的 mycomp 和 mycomp2 升序排序规则。需要注意的是，和为关联式容器设定排序规则不同，给 sort() 函数指定排序规则时，需要为其传入一个函数名（例如 mycomp ）或者函数对象（例如 std::greater() 或者 mycomp2() ）。

那么，sort() 函数的效率怎么样吗？该函数实现排序的平均时间复杂度为N*log2N（其中 N 为指定区域 [first, last) 中 last 和 first 的距离）。

参考http://c.biancheng.net/view/7457.html