# sort 排序中的less 和 greater的作用

http://www.cplusplus.com/reference/functional/less/ http://www.cplusplus.com/reference/algorithm/sort/?kw=sort

## 1-sort 函数的两种形式

```
/// 默认从小到大排序
template <class RandomAccessIterator>
  void sort (RandomAccessIterator first, RandomAccessIterator last);

/// 手动指定排序方式
template <class RandomAccessIterator, class Compare>
  void sort (RandomAccessIterator first, RandomAccessIterator last, Compare comp);
```

复杂度是`Nlog(N)`

举例

```
// sort algorithm example
#include <iostream>     // std::cout
#include <algorithm>    // std::sort
#include <vector>       // std::vector

bool myfunction (int i,int j) { return (i<j); }

struct myclass {
  bool operator() (int i,int j) { return (i<j);}
} myobject;

int main () {
  int myints[] = {32,71,12,45,26,80,53,33};
  std::vector<int> myvector (myints, myints+8);               // 32 71 12 45 26 80 53 33

  // using default comparison (operator <):
  std::sort (myvector.begin(), myvector.begin()+4);           //(12 32 45 71)26 80 53 33

  // using function as comp
  std::sort (myvector.begin()+4, myvector.end(), myfunction); // 12 32 45 71(26 33 53 80)

  // using object as comp
  std::sort (myvector.begin(), myvector.end(), myobject);     //(12 26 32 33 45 53 71 80)

  // print out content:
  std::cout << "myvector contains:";
  for (std::vector<int>::iterator it=myvector.begin(); it!=myvector.end(); ++it)
    std::cout << ' ' << *it;                 // myvector contains: 12 26 32 33 45 53 71 80

  std::cout << '\n';
```

```
  return 0;
}
```

## 2- 四种比较函数

```
less<type>()    //从小到大排序 <
grater<type>()  //从大到小排序 >
less_equal<type>()  //  <=
gtater_equal<type>()//  >=
//这四种函数
```

举例

```
// greater example
    #include <iostream>     // std::cout
    #include <functional>   // std::greater
    #include <algorithm>    // std::sort

    int main () {
      int numbers[]={20,40,50,10,30};
      std::sort (numbers, numbers+5, std::greater<int>());
      for (int i=0; i<5; i++)
        std::cout << numbers[i] << ' ';
      std::cout << '\n';
      return 0;
    }
```

## 3- 创建STL时指定排序顺序

set集合默认排序方式 从小到大即less的，我们可以通过创建set的时候指定排序方式

```
set<int,greater<int>> m_set = { 1, 1, 5, 3, 2, 9, 6, 7, 7 };        /// 创建时指定从大到小排序
for each (auto var in m_set){
    cout << var << " ";
}
```

## 4- tydef 重命名

另外如果闲创建的比较繁琐我们可以用typedef来重命名

```
typedef std::set<int,std::greater<int>> IntSet;
typedef std::set<int,std::less<int>> IntSet;
IntSet my_set
IntSet::iterator ipos;
```