stable_sort

```
http://www.cplusplus.com/reference/algorithm/stable_sort/
```

在头文件中

在排序中保留相等的相对次序。

1- 函数原型

```
template <class RandomAccessIterator>
  void stable_sort ( RandomAccessIterator first, RandomAccessIterator last );  /// 默认是用
operator <

template <class RandomAccessIterator, class Compare>
  void stable_sort ( RandomAccessIterator first, RandomAccessIterator last,
                     Compare comp );
```

参数：

```
RandomAccessIterator: 迭代器的范围是[first, last)

comp:Binary function that accepts two elements in the range as arguments, and returns a value
convertible to bool. The value returned indicates whether the element passed as first argument
is considered to go before the second in the specific strict weak ordering it defines.
The function shall not modify any of its arguments.
This can either be a function pointer or a function object.    /// 需要的是一个函数指针或者函数对象
```

2- 举例

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

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

int main () {
  double mydoubles[] = {3.14, 1.41, 2.72, 4.67, 1.73, 1.32, 1.62, 2.58};

  std::vector<double> myvector;
```

```
  myvector.assign(mydoubles,mydoubles+8);    /// 可以用assign赋值

  std::cout << "using default comparison:";
  std::stable_sort (myvector.begin(), myvector.end());
  for (std::vector<double>::iterator it=myvector.begin(); it!=myvector.end(); ++it)
    std::cout << ' ' << *it;
  std::cout << '\n';

  myvector.assign(mydoubles,mydoubles+8);

  std::cout << "using 'compare_as_ints' :";
  std::stable_sort (myvector.begin(), myvector.end(), compare_as_ints); // 只比较整数部分
  for (std::vector<double>::iterator it=myvector.begin(); it!=myvector.end(); ++it)
    std::cout << ' ' << *it;
  std::cout << '\n';

  return 0;
}

>>>
using default comparison: 1.32 1.41 1.62 1.73 2.58 2.72 3.14 4.67
using 'compare_as_ints' : 1.41 1.73 1.32 1.62 2.72 2.58 3.14 4.67 /// 保留了整数的相对次序
```