#include <algorithm>

**1.sort(a,b,c)**

void sort (RandomAccessIterator first, RandomAccessIterator last, Compare comp);

std::vector<int> myvector (myints, myints+8);

std::sort (myvector.begin(), myvector.begin()+4);

std::sort (myvector.begin()+4, myvector.end(), my function)

sort(vec.begin(),vec.end());(默认是按升序排列,即从小到大).

可以通过重写排序比较函数按照降序比较，定义排序比较函数：

bool Comp(const int &a,const int &b) return a>b;

调用时:sort(vec.begin(),vec.end(),Comp)，这样就降序排序。

降序方法1

std::sort(numbers.begin(), numbers.end(), std::greater<int>());

降序方法2

std::sort(numbers.rbegin(), numbers.rend());

**2.reverse(a,b)**

void reverse (BidirectionalIterator first, BidirectionalIterator last);

reverse(vec.begin(),vec.end());

将元素翻转(在vector中，如果一个函数中需要两个迭代器，一般后一个都不包含.)

**3.unique ( , )**

unique (myvector.begin(), myvector.end())

删除相邻重复元素，所以可以先sort，再nuique

**4.min\_element**

std::vector<int>::iterator result = std::min\_element(std::begin(v), std::end(v));

distance(first, last) 计算iterator距离 The number of elements between *first* and *last*.

可以将iterator转换为Index！

**5.swap( , )**

swap(T (&a)[N], T (&b)[N])

string s;

swap(s[1], s[2]);

**6.iter\_swap( , )**

iter\_swap(v.begin(),v.begin()+1)

**7.transform( , , )**

std::string data = "Abc";

std::transform(data.begin(), data.end(), data.begin(), ::tolower);

**8.lower\_bound**

Returns an iterator pointing to the first element in the range [first,last] which compares >= val.

auto it = lower\_bound(res.begin(), res.end(), val);

**9.upper\_bound**

Returns an iterator pointing to the first element in the range [first,last) which compares > val.