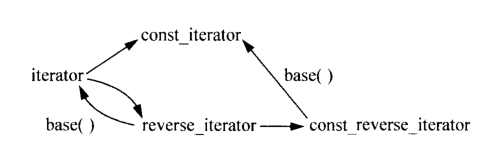
第25条：

C++11 STL中标准无序容器unordered\_set、unordered\_multiset、unordered\_map和unordered\_multimap都实现为哈希容器。

第26条：

迭代器iterator优先于const\_iterator、reverse\_iterator和const\_reverse\_iterator。



容器的insert和erase几乎只接受iterator迭代器。

第27条：（不做研究）

const\_iterator转换为iterator。

第28条：

reverse\_iterator的base()函数产生的iterator。

例：程序effective\_stl\_test11

*vector*<int> v{ 1, 2, 3, 4, 5, 6, 7, 8, 9 };

*vector*<int>::*reverse\_iterator* r\_iter = *find*(v.*rbegin*(), v.*rend*(), 3);

// 反向迭代器的插入

// 容器的insert()不接受reverse\_iterator，

// 所以需要通过base()转换。

// insert()会把元素插入到迭代器指向的元素的前面

// 在反向遍历时，99会出现在3的前面，正向遍历时

// 它出现在4的前面，即：插入时，r\_iter与r\_iter.base()等价

v.*insert*(r\_iter.*base*(), 99);

PrintElements(v, "Insert by reverse\_iterator: ");

*cout* << *endl*;

// 反向迭代器的删除

// erase()同样不接受reverse\_iterator

// 删除时r\_iter指向的元素时，r\_iter与r\_iter.base()不再等价

// r\_iter与(++r\_iter).base()等价

r\_iter = *find*(v.*rbegin*(), v.*rend*(), 3); // 之前的r\_iter因为插入操作失效

v.*erase*((++r\_iter).*base*());

PrintElements(v, "Erase by reverse\_iterator: ");

输出为：

Insert by reverse\_iterator: 1 2 3 99 4 5 6 7 8 9

Erase by reverse\_iterator: 1 2 99 4 5 6 7 8 9

第29条：

对于逐个字符的输入，可以使用istreambuf\_iterator。

ifstream input\_file(“data.txt”);

string file\_data((istreambuf\_iterator<char>(input\_file)),

istreambuf\_iterator<char>());

第30条：

使用STL算法时，要确保目标容器有足够大的空间。