C++标准程序库读书笔记

STL迭代器

逆向迭代器需要专门的头文件，但是由于容器必须要定义它自己的逆向迭代器型别，所以容器本身已经包含了该头文件。

迭代器奉行一个纯抽象概念：任何东西，只要行为类似迭代器，就是一个迭代器。不同的迭代器具有不同的能力（行进和存取能力）。

迭代器分类及其能力

迭代器类型	能力	供应者
input迭代器	向前读取	istream
output迭代器	向前写入	ostream,inserter
forward迭代器	向前读取和写入	
bidirectional迭代器	向前和向后读取和写入	list set multiset map multimap
random access迭代器	随机存取，可读取也可写入	vector deque string array

关于输入迭代器的前置和后置递增操作符选用的讨论

应该尽可能优先选用前置式递增运算操作符（++iter）而不是后置式递增运算操作符（iter++），因为前者性能更好，前置式递增元素操作符不需传回旧值，所以也不必花费一个临时对象来保存旧值，因此，面对任何迭代器（以及任何抽象数据型别），应该优先使用前置式，这条原则对递减运算操作符同样适用。（input迭代器并不提供递减运算操作符）。

forward iterator不具备output迭代器的全部功能，这里会有一个约束条件：

- 面对output迭代器，无需检查是否抵达序列尾端，便可直接写入数据。
- 对于forward迭代器，必须在存取数据之前确保它有效。

以下对象和型别支持random access迭代器：

- 可随机存取的容器（vector deque）
- strings(字符串 string wstring)
- 一般array(指针)

迭代器相关辅助函数

C++标准程序库为迭代器提供了三个辅助函数：advance() distance()和iter_swap(),前两者提供个所有迭代器一些原本只有随机迭代器才有的能力：前进或后退多个元素，及处理迭代器之间的距离。第三个辅助函数允许交换两个迭代器的值。

advance()

afvance()可将迭代器的位置增加，增加到的幅度由参数决定，也就是说迭代器一次前进或后退多个元素。

void advance(InputIterator& pos,Dist n)

- 使名为pos的input迭代器步进或后退n个元素。
- 对Bidirectional迭代器和random access迭代器而言，n可为负值，表示向后退。
- Dist是个template型别，通常应该是个整数型别，因为会调用<,++,--,等操作，还要和0做比较。
- advance()从不检查迭代器是否查过序列的end()。所以，调用advance()有可能导致未定义行为----因为对着序列尾端调用operator++是一种未定义的操作行为。
- 该函数对于random access迭代器，只是简单的调用pos+=n，因此，具有常量复杂度，对于其他任何类型的迭代器调用++pos（或--pos，如果为负值）n次，因此，对于其他任何类型的迭代器，本函数具有线性复杂度。

distance()

函数distance()用来处理两个迭代器之间的距离。

Dist distance(InputIterator pos1,InputIterator pos2)

- 传回两个input迭代器pos1和pos2之间的距离。
- 两个迭代器都必须指向同一个容器。
- 如果不是random access迭代器，则从pos1开始往前走就能够到达pos2，即pos2的位置必须与pos1相同或在其后。
- 回返值Dist的型别由迭代器决定：iterator_traits::difference_type
- 面对random access迭代器，因此具备常数复杂度，对于其他迭代器类型，distance()会不断递增pos1,直到抵达pos2为止，然后传回递增次数。也就是说，distance()具备线性复杂度，因此对于non-random access迭代器而言，diatance()的性能并不好，应该尽力避免使用它。