C++标准库提供了特殊迭代器，称为迭代器适配器。

1. Insert iterator（插入型迭代器）

可以使算法以insert方式而非overwrite方式运作。

例：程序stl\_test33

*list*<int> coll1 = {1, 2, 3, 4, 5, 6, 7, 8, 9};

*vector*<int> coll2;

// 插入容器末端

// 在内部调用push\_back执行插入

*copy*(coll1.*cbegin*(), coll1.*cend*(), *back\_inserter*(coll2));

for (const auto& elem : coll2)

{

*cout* << elem << ' ';

}

*cout* << *endl*;

//vector<int> coll3; // error，vector不能执行push\_front操作

*deque*<int> coll3;

// 插入容器前端

// 在内部执行push\_front操作

*copy*(coll1.*cbegin*(), coll1.*cend*(), *front\_inserter*(coll3));

for (const auto& elem : coll3)

{

*cout* << elem << ' ';

}

*cout* << *endl*;

*set*<int> coll4;

// 一般性插入，内部调用insert()插入

// 可以用于关联式容器和无序容器

*copy*(coll1.*cbegin*(), coll1.*cend*(), *inserter*(coll4, coll4.*begin*()));

for (const auto& elem : coll4)

{

*cout* << elem << ' ';

}

*cout* << *endl*;

输出为：

1 2 3 4 5 6 7 8 9

9 8 7 6 5 4 3 2 1

1 2 3 4 5 6 7 8 9

使用这种插入时迭代器适配器，被拷贝的容器不需要提前分配空间了。

1. Stream Iterator（串流迭代器）

使的输入像一个集合，能够从中读取内容。

例：程序stl\_test34

*vector*<int> coll;

// 串流输入迭代器istream\_iterator

// 第1个istream\_iterator产生从“标准输入串流cin”

// 读取数据的iterator, 通过>>读取int型数据, cin >> int

// 第2个istream\_iterator产生一个串流结束符，同cin读取

// 结束操作类似

*copy*(*istream\_iterator*<int>(*cin*), *istream\_iterator*<int>(), *back\_inserter*(coll));

*sort*(coll.*begin*(), coll.*end*());

// 串流输出迭代器ostream\_iterator

// unique\_copy消除毗邻的重复值并复制到cout

// "\n"为元素间的分隔符

*unique\_copy*(coll.*cbegin*(), coll.*cend*(), *ostream\_iterator*<int>(*cout*, "\n"));

输入：

23 45 32 76 12 w

输出：

12

23

32

45

76

从上述程序中可以看出，串流适配器可以使输入/输出的元素像集合一样。

1. 反向迭代器（Reverse Iterator）

所有提供双向或随机访问迭代器的容器（除去forward\_list之外的序列式容器和所有关联容器）都可以通过rbegin()或crbegin()和rend()或crend()产生一个反向迭代器。

例：程序stl\_test35

*vector*<int> coll;

for (int i = 1; i <= 9; ++i)

{

coll.*push\_back*(i);

}

// 反向迭代器

*copy*(coll.*crbegin*(), coll.*crend*(), *ostream\_iterator*<int>(*cout*, " "));

*cout* << *endl*;

return 0;

输出：

9 8 7 6 5 4 3 2 1

1. 搬移迭代器（Move Iterator）

自C++11，用来将任何“对底层元素的访问”转换为一个move操作。允许从一个容器移动元素至另一个容器。