通常STL容器提供的是value语义，如果希望实现Reference语义，可以考虑下面两种方式：

1. smart pointer：例如shared\_ptr

例：程序stl\_test53

class Item {

public:

Item(const *string*& name, float price = 0) : name\_(name), price\_(price)

{}

*string* name() const

{

return name\_;

}

void set\_name(const *string*& name)

{

name\_ = name;

}

float price() const {

return price\_;

}

void set\_price(float p)

{

price\_ = p;

}

private:

*string* name\_;

float price\_;

};

template<typename Coll>

void PrintItems(const *string*& msg, const Coll& coll)

{

*cout* << msg << *endl*;

for (const auto& elem : coll)

{

*cout* << ' ' << elem->name() << ":"

<< elem->price() << *endl*;

}

}

int main(int argc, char\* argv[])

{

typedef *shared\_ptr*<Item> ItemPtr;

// 使用shared\_ptr作为STL容器实现Reference语义

*set*<ItemPtr> allItems;

*deque*<ItemPtr> bestsellers;

bestsellers = {ItemPtr(new Item("Kong Yize", 20.10f)),

ItemPtr(new Item("A Midsummer Night's Dream", 14.99f)),

ItemPtr(new Item("The Maltese Falcon", 9.88f))};

allItems = {ItemPtr(new Item("Water", 0.44f)), ItemPtr(new Item("Pizza", 2.22f))};

allItems.*insert*(bestsellers.*begin*(), bestsellers.*end*());

PrintItems("bestsellers: ", bestsellers);

PrintItems("all: ", allItems);

*cout* << *endl*;

*cout* << "double price of bestsellers" << *endl*;

*for\_each*(bestsellers.*begin*(), bestsellers.*end*(), [](*shared\_ptr*<Item>& elem) {

elem->set\_price(elem->price() \* 2);

});

*cout* << "replace second bestseller by first item with name Pizza" << *endl*;

bestsellers[1] = \*(*find\_if*(allItems.*begin*(), allItems.*end*(), [](*shared\_ptr*<Item> elem) {

return elem->name() == "Pizza";

}));

bestsellers[0]->set\_price(44.77f);

PrintItems("bestsellers: ", bestsellers);

PrintItems("all: ", allItems);

return 0;

}

输出为：

bestsellers:

Kong Yize:20.1

A Midsummer Night's Dream:14.99

The Maltese Falcon:9.88

all:

A Midsummer Night's Dream:14.99

Kong Yize:20.1

Water:0.44

The Maltese Falcon:9.88

Pizza:2.22

double price of bestsellers

replace second bestseller by first item with name Pizza

bestsellers:

Kong Yize:44.77

Pizza:2.22

The Maltese Falcon:19.76

all:

A Midsummer Night's Dream:29.98

Kong Yize:44.77

Water:0.44

The Maltese Falcon:19.76

Pizza:2.22

从输出可以看出，修改了容器bestsellers中的元素，在容器allItems中对应的元素也会被修改，原因是在allItems中插入了所有bestsellers中的元素，STL容器提供是value语义，插入的其实是bestsellers元素的拷贝，但由于容器中是shared\_ptr智能指针，两个容器中的shared\_ptr指向其实同一份Item对象。

1. 使用reference wrapper：外覆器

例：程序stl\_test54

class Item {

public:

Item(const *string*& name, float price = 0) : name\_(name), price\_(price)

{}

*string* name() const

{

return name\_;

}

void set\_name(const *string*& name)

{

name\_ = name;

}

float price() const {

return price\_;

}

void set\_price(float p)

{

price\_ = p;

}

private:

*string* name\_;

float price\_;

};

int main(int argc, char\* argv[])

{

*vector*<*reference\_wrapper*<Item>> books;

Item f("Faust", 12.99f);

books.*push\_back*(f);

*cout* << "original: " << *endl*;

for (const auto& book : books)

{

*cout* << book.*get*().name() << ":" << book.*get*().price() << *endl*;

}

*cout* << *endl*;

*cout* << "change price: " << *endl*;

f.set\_price(9.99f);

for (const Item& book : books)

{

*cout* << book.name() << ":" << book.price() << *endl*;

}

return 0;

}

输出为：

original:

Faust:12.99

change price:

Faust:9.99

从输出可以看出，将Item元素f插入vector容器books后，再修改f，容器books中对应的元素也会被修改，实现了Reference语义。