函数对象：行为类似函数的对象，或称为仿函数。

在C++11起，

1. 函数指针
2. 带有成员函数operator()的class所建立的object
3. 带有转换函数可将自己转换为函数指针的class所建立的object
4. lambda

都可以称为函数对象。本章所指的函数对象指的是第2)种。

函数对象相对普通函数的优点：

“行为像函数的对象”可以称为smart function。

1. 函数对象是一种带状态的函数：函数对象可以拥有成员函数和成员变量。在同一时间点，相同类型的两个不同的函数对象表述相同的机能，可具备不同的状态。
2. 函数对象都有自己的类型，通过template来实现。
3. 函数对象通常比普通函数速度快。

例：程序stl\_test40

// 函数对象

class AddValue

{

public:

AddValue(int value) : value\_(value)

{}

// 这个函数最好定义为const

void operator() (int& elem) const {

elem += value\_;

}

private:

int value\_;

};

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

{

*list*<int> coll;

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

{

coll.*push\_back*(i);

}

PrintElements(coll, "initialized: ");

// 函数对象作为算法的实参

*for\_each*(coll.*begin*(), coll.*end*(), AddValue(10));

PrintElements(coll, "after adding 10: ");

// 一个函数，两个状态

// 通过初始化不同的AddValue，呈现不同的状态

*for\_each*(coll.*begin*(), coll.*end*(), AddValue(\*coll.*begin*()));

PrintElements(coll, "after adding first element: ");

return 0;

}

输出为：

initialized: 1 2 3 4 5 6 7 8 9

after adding 10: 11 12 13 14 15 16 17 18 19

after adding first element: 22 23 24 25 26 27 28 29 30

STL库中预定义了一些函数对象，如less<>，greater<>等。

例：程序stl\_test40

*deque*<int> coll1 = {1, 2, 3, 5, 7, 11, 13, 17, 19};

PrintElements(coll1, "initialized: ");

*transform*(coll1.*cbegin*(), coll1.*cend*(), coll1.*begin*(), *negate*<int>());

PrintElements(coll1, "negated: ");

*transform*(coll1.*cbegin*(), coll1.*cend*(), coll1.*cbegin*(),

coll1.*begin*(), *multiplies*<int>());

PrintElements(coll1, "squared: ");

输出为：

initialized: 1 2 3 5 7 11 13 17 19

negated: -1 -2 -3 -5 -7 -11 -13 -17 -19

squared: 1 4 9 25 49 121 169 289 361

特殊的函数适配器：binder，将预定义的函数对象和其他数值结合为一体。

例：程序stl\_test41

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

*deque*<int> coll2;

PrintElements(coll1, "initialized: ");

// 将coll1所有元素乘以10传递到coll2中

// bind(multiplies<int>(), \_1, 10)是一个函数对象

// 由 低层的函数对象multiplies<int>()和占位符\_1合成

*transform*(coll1.*cbegin*(), coll1.*cend*(), // source：源

*back\_inserter*(coll2), // destination：目标

*bind*(*multiplies*<int>(), *\_1*, 10)); // operation：操作

PrintElements(coll2, "transformed: ");

// 替换coll2中值为70的元素为42

// bind(equal\_to<int>(), \_1, 70)是一个合成的函数对象

*replace\_if*(coll2.*begin*(), coll2.*end*(),

*bind*(*equal\_to*<int>(), *\_1*, 70),

42);

PrintElements(coll2, "replaced: ");

// 移除x>=50 && x<=80

// bind()可以用来描述复杂的判断式

coll2.*erase*(*remove\_if*(coll2.*begin*(), coll2.*end*(),

*bind*(*logical\_and*<bool>(),

*bind*(*greater\_equal*<int>(), *\_1*, 50),

*bind*(*less\_equal*<int>(), *\_1*, 80))),

coll2.*end*());

PrintElements(coll2, "removed: ");