Lambda vs. Binder

例：程序stl\_test63 对照stl\_test59

auto plus10 = [](int i) {

return i + 10;

};

*cout* << "+10: " << plus10(7) << *endl*;

auto plus10time2 = [](int i) {

return (i + 10) \* 2;

};

*cout* << "+10 \* 2" << plus10time2(7) << *endl*;

auto pow3 = [](int i) {

return i \* i \* i;

};

*cout* << "x \* x \* x: " << pow3(7) << *endl*;

auto inversDivide = [](double d1, double d2) {

return d2 / d1;

};

*cout* << "invdiv: " << inversDivide(49, 7) << *endl*;

输出为：

+10: 17

+10 \* 2: 34

x \* x \* x: 343

invdiv: 0.142857

Lambda vs 带有状态(Stateful)的function object

例：程序stl\_test63 对照stl\_test57

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

long sum = 0;

*for\_each*(coll.*begin*(), coll.*end*(), [&sum](int elem) {

sum += elem;

});

double mv = static\_cast<double>(sum)

/ static\_cast<double>(coll.*size*());

*cout* << "mean value: " << mv << *endl*;

输出为：

mean value: 4.5

带有mutable的Lambda

例：程序stl\_test63

*list*<int> coll1 = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

PrintElements(coll1, "coll1: ");

int count = 0;

// Lambda使用mutable

// count以by value形式

// 但是可以在Lambda表达式内部

// 修改其值，作用域以外没有效果

*list*<int>::*iterator* pos

= *remove\_if*(coll1.*begin*(), coll1.*end*(),

[count](int) mutable {

return ++count == 3;

});

*cout* << "count: " << count << *endl*;

coll1.*erase*(pos, coll1.*end*());

PrintElements(coll1, "3rd removed: ");

输出为：

coll1: 1 2 3 4 5 6 7 8 9 10

count: 0

3rd removed: 1 2 4 5 6 7 8 9 10