函数对象：或称为functors（仿函数）。

所谓function object是一个定义了operator ()的对象。

class FunctionObjectType

{

public:

void operator() () {

statements

}

}

函数对象的优点：

1. function object比一般函数更灵巧（smart），因为它可以拥有状态（state）。对于相同的function object class你可以拥有两个状态不同的实例（instance）。
2. 每个function object都有其类型。可以将function object的类型当作template参数传递，从而指定某种行为。容器类型也会因为function object的不同而不同。
3. 执行速度上，function object通常比function pointer更快。

function object拥有多个内部状态

例：程序stl\_test57

class IntSequence {

public:

IntSequence(int init\_value) : value\_(init\_value)

{}

int operator() () {

return ++value\_;

}

private:

int value\_;

};

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

{

*list*<int> coll;

// IntSequence(1)

*generate\_n*(*back\_inserter*(coll), // start

9, // number of elements

IntSequence(1)); // generates vlaues, starting with 1

PrintElements(coll, "generate 1-9: ");

*cout* << "replace second to last element but one with values starting at 42" << *endl*;

// IntSequence(42)

*generate*(*next*(coll.*begin*()), *prev*(coll.*end*()), IntSequence(42));

PrintElements(coll, "replace: ");

return 0;

}

输出为：

generate 1-9: 2 3 4 5 6 7 8 9 10

replace second to last element but one with values starting at 42

replace: 2 43 44 45 46 47 48 49 10

以reference形式传递function object

// 以refernce传递function object

*list*<int> coll1;

*list*<int> coll2;

IntSequence seq(1);

// 需要在调用算法时标示function object是个reference类型

*generate\_n*<*back\_insert\_iterator*<*list*<int>>,

int, IntSequence&>(*back\_inserter*(coll1), 5, seq);

PrintElements(coll1, "coll1: ");

*generate\_n*(*back\_inserter*(coll2), 5, seq);

PrintElements(coll2, "coll2: ");

输出为：

coll1: 2 3 4 5 6

coll2: 7 8 9 10 11

function object与for\_each()

for\_each()算法可以回传其function object，这样就不用通过reference来传递function object。

例：程序stl\_test57

class MeanValue {

public:

MeanValue() : number\_(0), sum\_(0) {

}

void operator() (int elem)

{

++number\_;

sum\_ += number\_;

}

double value() {

return static\_cast<double>(sum\_) / static\_cast<double>(number\_);

}

private:

long number\_;

long sum\_;

};

// for\_each算法回传function object

*vector*<int> coll3 = {1, 2, 3, 4, 5, 6, 7, 8};

MeanValue mv = *for\_each*(coll3.*begin*(), coll3.*end*(), MeanValue());

*cout* << "mean value: " << mv.value() << *endl*;

输出为：

mean value: 4.5

判断式（Predicate）：返回Boolean值（任何可转为bool的值）的函数或function object。

例：程序stl\_test58

class Nth {

public:

Nth(int n) : nth\_(n), count\_(0) {

}

// Predicate

bool operator() (int) {

return ++count\_ == nth\_;

}

private:

int nth\_;

int count\_;

};

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

{

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

PrintElements(coll, "original: ");

// 在Visual Studio上这个没问题

*list*<int>::*iterator* pos = *remove\_if*(coll.*begin*(), coll.*end*(), Nth(3));

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

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

return 0;

}