STL算法：用来处理一个或多个iterator区间。

for\_each()：允许以不同的方式访问、处理、修改每一个元素。

例：程序stl\_test64

*vector*<int> coll;

InsertElements(coll, 1, 9);

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

elem += 10;

});

PrintElements(coll);

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

elem += \*coll.*begin*();

});

PrintElements(coll);

输出为：

11 12 13 14 15 16 17 18 19

22 23 24 25 26 27 28 29 30

非更易型算法：不改变元素值，也不改动元素次序。

1. 元素计数count()：线性复杂度

difference\_type

count(Iterator beg, Iterator end, const T& value)

计算[beg, end)中“元素值等于value”的元素个数。

difference\_type

count\_if(Iterator beg, Iterator end, UnaryPredicate op)

计算[beg, end)中“令unary predicate op(elem)结果为true”的元素个数。

difference\_type表示iterator间距的类型

例：程序stl\_test65

// 元素计数count

*vector*<int> coll;

InsertElements(coll, 1, 9);

PrintElements(coll, "coll: ");

int number = *count*(coll.*cbegin*(), coll.*cend*(), 4);

*cout* << "number of elements equal to 4: " << number << *endl*;

number = *count\_if*(coll.*cbegin*(), coll.*cend*(), [](int elem) {

return elem % 2 == 0;

});

*cout* << "number of elements with even value: " << number << *endl*;

number = *count\_if*(coll.*cbegin*(), coll.*cend*(), [](int elem) {

return elem > 4;

});

*cout* << "number of elements greater than 4: " << number << *endl*;

输出为：

coll: 1 2 3 4 5 6 7 8 9

number of elements equal to 4: 1

number of elements with even value: 4

number of elements greater than 4: 5

1. 最小值和最大值：min\_element和max\_element，线性复杂度

Iterator min\_element(Iterator beg, Iterator end)

Iterator min\_element(Iterator beg, Iterator end, ComFunc op)

Iterator max\_element(Iterator beg, Iterator end)

Iterator max\_element(Iterator beg, Iterator end, ComFunc op)

pair<Iterator, Iterator> minmax\_element(Iterator beg, Iterator end)

pair<Iterator, Iterator> minmax\_element(Iterator beg, Iterator end, ComFunc op)

注意：如果存在多个最小值或最大值，min\_element()和max\_element()返回其所找到的第1个目标元素。minmax\_element()返回第1个最小元素和最后1个最大元素。所以此时max\_element()和minmax\_element()返回的最大元素不是同一个。

例：程序stl\_test65

bool AbsLess(int elem1, int elem2)

{

return *abs*(elem1) < *abs*(elem2);

}

// 最小值和最大值

*deque*<int> coll1;

InsertElements(coll1, 2, 6);

InsertElements(coll1, -3, 6);

*cout* << "min and max: " << *endl*;

PrintElements(coll1, "coll1: ");

*cout* << "minimum: "

<< \**min\_element*(coll1.*cbegin*(), coll1.*cend*()) << *endl*;

*cout* << "maximum: "

<< \**max\_element*(coll1.*cbegin*(), coll1.*cend*()) << *endl*;

auto mm = *minmax\_element*(coll1.*cbegin*(), coll1.*cend*());

// 第1个最小元素

*cout* << "min: " << \*(mm.*first*) << *endl*;

// 最后一个最大元素

*cout* << "max: " << \*(mm.*second*) << *endl*;

*cout* << "distance between min and max: " << *distance*(mm.*first*, mm.*second*) << *endl*;

*cout* << "minimun of absolute values: "

<< \**min\_element*(coll1.*cbegin*(), coll1.*cend*(), AbsLess) << *endl*;

*cout* << "maximum of absolute values: "

<< \**max\_element*(coll1.*cbegin*(), coll1.*cend*(), AbsLess) << *endl*;

输出为：

min and max:

coll1: 2 3 4 5 6 -3 -2 -1 0 1 2 3 4 5 6

minimum: -3

maximum: 6

min: -3

max: 6

distance between min and max: 9

minimun of absolute values: 0

maximum of absolute values: 6

1. 查找元素：find

Iterator find(Iterator beg, InputIterator end, const T& value)

Iterator find\_if(Iterator beg, Iterator end, UnaryPredicate op)

Iterator find\_if\_not(Iterator beg, Iterator end, UnaryPredicate op)

返回[beg, end)区间中第1个“造成unary predicate op(element)结果为false”的元素。

如果没有结果，返回end。

例：程序stl\_test65

// 查找元素find

*cout* << "find " << *endl*;

*list*<int> coll2;

InsertElements(coll2, 1, 9);

InsertElements(coll2, 1, 9);

PrintElements(coll2, "coll2: ");

//find first element with value 4

*list*<int>::*iterator* pos1 = *find*(coll2.*begin*(), coll2.*end*(), 4);

//find second element with value 4

*list*<int>::*iterator* pos2;

if (pos1 != coll2.*cend*())

{

pos2 = *find*(*next*(pos1), coll2.*end*(), 4);

}

// print all elements from first to second 4

if (pos1 != coll2.*end*() && pos2 != coll2.*end*())

{

*copy*(pos1, *next*(pos2), *ostream\_iterator*<int>(*cout*, " "));

*cout* << *endl*;

}

*cout* << *endl*;

*vector*<int> coll3 = {1, 2, 3, 4, 5, 6, 7, 8, 9};

PrintElements(coll3, "coll3: ");

// find first element greater than 3

*vector*<int>::*iterator* pos = *find\_if*(coll3.*begin*(), coll3.*end*(),

*bind*(*greater*<int>(), *\_1*, 3));

*cout* << "the "

<< *distance*(coll3.*begin*(), pos) + 1

<< ". element is the first greater than 3" << *endl*;

// find first element divisible by 3

pos = *find\_if*(coll3.*begin*(), coll3.*end*(), [](int elem) {

return elem % 3 == 0;

});

*cout* << "the "

<< *distance*(coll3.*begin*(), pos) + 1

<< ". element is the first divisible by 3" << *endl*;

// find first element not < 5

pos = *find\_if\_not*(coll3.*begin*(), coll3.*end*(), *bind*(*less*<int>(), *\_1*, 5));

*cout* << "first value >= 5: " << \*pos << *endl*;

输出为：

find

coll2: 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9

4 5 6 7 8 9 1 2 3 4

coll3: 1 2 3 4 5 6 7 8 9

the 4. element is the first greater than 3

the 3. element is the first divisible by 3

first value >= 5: 5

1. 查找前n个连续匹配值：search\_n

Iterator search\_n(Iterator beg, Iterator end, Size count, const T& value)

返回[beg, end)内“连续count个元素都等于value”中的第1个元素位置。

Iterator search\_n(Iterator beg, Iterator end, Size count, const T& value, BinaryPredicate op)

返回[beg, end)内“连续count个元素造成以下binary predicate结果为true”中第1个元素位置：

op(elem, value)