算法（Algorithm）：并非面向对象的编程思维，而是泛型函数编程思维。STL容器的算法有一定的缺陷。

例：程序stl\_test32

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

auto min\_pos = *min\_element*(coll.*cbegin*(), coll.*cend*());

*cout* << "min: " << \*min\_pos << *endl*;

auto max\_pos = *max\_element*(coll.*cbegin*(), coll.*cend*());

*cout* << "max: " << \*max\_pos << *endl*;

// 排序

*sort*(coll.*begin*(), coll.*end*());

for (const auto& elem : coll)

{

*cout* << elem << ' ';

}

*cout* << *endl*;

// 找到第一个值为3的元素

auto pos3 = *find*(coll.*begin*(), coll.*end*(), 3);

// 反序

// 将查找到的值为3的元素及之后的元素反序

*reverse*(pos3, coll.*end*());

for (const auto& elem : coll)

{

*cout* << elem << ' ';

}

*cout* << *endl*;

输出为：

min: 1

max: 6

1 2 3 4 5 6

1 2 6 5 4 3

区间：所有算法处理都是半开区间，包含起始元素，但不包含结束元素。[begin, end)。

程序员必须保证提供给算法的区间是有效的。

例：程序stl\_test32

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

*vector*<int> coll2;

coll2.*resize*(coll1.*size*());

*copy*(coll1.*cbegin*(), coll1.*cend*(), coll2.*begin*());

for (const auto& elem : coll2)

{

*cout* << elem << ' ';

}

*cout* << *endl*;