STL算法：更易型算法

1. 互换元素，swap\_ranges，线性复杂度

Iterator2 swap\_ranges(Iterator1 beg1, Iterator1 end1, Iterator2 beg2)

将区间[beg1, end1)内的元素和“从beg2开始的区间”内的对应元素互换。

返回第二区间中“最后一个被交换元素”的下一位置。

注意：两区间不得重叠。

例：程序stl\_test72

// swap\_ranges

*vector*<int> coll5;

*deque*<int> coll6;

InsertElements(coll5, 1, 9);

InsertElements(coll6, 11, 23);

PrintElements(coll5, "coll5: ");

PrintElements(coll6, "coll6: ");

*cout* << *endl*;

*cout* << "after swap:" << *endl*;

*deque*<int>::*iterator* pos

= *swap\_ranges*(coll5.*begin*(), coll5.*end*(), coll6.*begin*());

PrintElements(coll5, "coll5: ");

PrintElements(coll6, "coll6: ");

输出为：

coll5: 1 2 3 4 5 6 7 8 9

coll6: 11 12 13 14 15 16 17 18 19 20 21 22 23

after swap:

coll5: 11 12 13 14 15 16 17 18 19

coll6: 1 2 3 4 5 6 7 8 9 20 21 22 23

1. 赋值
2. 赋予相同值，fill，线性复杂度

void fill(Iterator beg, Iterator end, const T& newValue)

将区间[beg, end)内的每一个元素都赋予新值newValue

void fill\_n(Iterator beg, Size num, const T& newValue)

将“从beg开始的前num个元素”赋予新值newValue，如果num为负值则不做任何事情。

注意：从C++11起，fill\_n()返回最后被改动元素的下一个位置（beg+num），如果num为负值则返回beg。

例：程序stl\_test72

// fill

*fill\_n*(*ostream\_iterator*<float>(*cout*, " "), 10, 7.7);

*cout* << *endl*;

*list*<*string*> coll7;

*fill\_n*(*back\_inserter*(coll7), 9, "hello");

PrintElements(coll7, "coll7: ");

// overwrite all elements with "again"

*fill*(coll7.*begin*(), coll7.*end*(), "again");

PrintElements(coll7, "coll7: ");

// replace the second and up to the last

// element but one with "hmmm"

*list*<*string*>::*iterator* pos1 = coll7.*begin*();

*list*<*string*>::*iterator* pos2 = coll7.*end*();

*fill*(++pos1, --pos2, "hmmm");

PrintElements(coll7, "coll7: ");

输出为：

7.7 7.7 7.7 7.7 7.7 7.7 7.7 7.7 7.7 7.7

coll7: hello hello hello hello hello hello hello hello hello

coll7: again again again again again again again again again

coll7: again hmmm hmmm hmmm hmmm hmmm hmmm hmmm again

1. 赋予新生值，generate，线性复杂度

void generate(Iterator beg, Iterator end, Func op)

调用op()产生新值，并将它赋值给区间[beg, end)内的每个元素。

void generate\_n(Iterator beg, Size num, Func op)

调用op()产生新值，并将赋值给“以beg起始的区间”内的前num个元素。如果num为负值则不做任何事。

注意：从C++11起，generate\_n()返回最后被改动元素的下一个位置（beg+num），如果num为负值则返回beg

例：程序stl\_test72

// generate

*list*<int> coll8;

*generate\_n*(*back\_inserter*(coll8), 5, *rand*);

PrintElements(coll8, "coll8: ");

// overwrite with five new random numbers

*generate*(coll8.*begin*(), coll8.*end*(), *rand*);

PrintElements(coll8, "coll8: ");

输出为：

coll8: 41 18467 6334 26500 19169

coll8: 15724 11478 29358 26962 24464

1. 赋予一系列递增值，iota，线性复杂度

void iota(Iterator beg, Iterator end, T startValue)

依序赋值startValue、startValue + 1、startValue + 2…

例：程序stl\_test72

//iota

array<int, 10> coll9;

*iota*(coll9.*begin*(), coll9.*end*(), 42);

PrintElements(coll9, "coll9: ");

输出为：

coll9: 42 43 44 45 46 47 48 49 50 51

1. 替换元素
2. 替换序列内的元素，replace，线性复杂度

void replace(Iterator beg, Iterator end, const T& oldValue, const T& newValue)

将区间[beg, end)区间内每一个“与oldValue相等”的元素替换为newValue。

void replace\_if(Iterator beg, Iterator end, UnaryPredicate op,

const T& newValue)

将区间[beg, end)内每一个令unary predicate：

op(elem)

产生true的元素替换为newValue。

例：程序stl\_test73

*list*<int> coll;

InsertElements(coll, 2, 7);

InsertElements(coll, 4, 9);

PrintElements(coll, "coll: ");

// replace all elements with value 6 with 42

*replace*(coll.*begin*(), coll.*end*(), 6, 42);

PrintElements(coll, "coll: ");

// replace all elements with value less than 5 with 0

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

[](int elem) {

return elem < 5;

},

0);

PrintElements(coll, "coll: ");

输出为：

coll: 2 3 4 5 6 7 4 5 6 7 8 9

coll: 2 3 4 5 42 7 4 5 42 7 8 9

coll: 0 0 0 5 42 7 0 5 42 7 8 9

1. 复制并替换元素，replace\_copy，线性复杂度

Iterator2

replace\_copy(Iterator1 sourceBeg, Iterator1 sourceEnd,

Iterator2 destBeg,

const T& oldValue, const T& newValue)

将源区间[sourceBeg, sourceEnd)中的元素复制到“以destBeg为起点”的目标区，同时将其中“与oldValue相等”的所有元素替换为newValue。

Iterator2

replace\_copy\_if(Iterator1 sourceBeg, Iterator1 sourceEnd,

Iterator2 destBeg,

UnaryPredicate op, const T& newValue)

将源区间[sourceBeg, sourceEnd)中的元素复制到“以destBeg为起点”的目标区间，同时将其中“令unary predicate op(elem)为true”的所有元素替换为newValue

两个算法均返回目标区间“最后一个被复制元素”的下一位置，也就是第一个未被覆盖的元素的位置。

例：程序stl\_test73

// replace\_copy

*list*<int> coll1;

InsertElements(coll1, 2, 6);

InsertElements(coll1, 4, 9);

PrintElements(coll1, "coll1: ");

// print all elements with value 5 replaced with 55

*replace\_copy*(coll1.*cbegin*(), coll1.*cend*(), //source

*ostream\_iterator*<int>(*cout*, " "), // destination

5, // oldValue

55); // newValue

*cout* << *endl*;

// print all elements with a value less than 5 replaced with 42

*replace\_copy\_if*(coll1.*cbegin*(), coll1.*cend*(),

*ostream\_iterator*<int>(*cout*, " "),

*bind*(*less*<int>(), *\_1*, 5),

42);

输出为：

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

2 3 4 55 6 4 55 6 7 8 9

42 42 42 5 6 42 5 6 7 8 9