参考：<https://zh.cppreference.com/w/%E9%A6%96%E9%A1%B5>

主要基于C++ 98

算法库提供大量用途的函数（例如查找、排序、计数、操作），它们在元素范围上操作。注意范围定义为 [first, last) ，其中 last 指代要查询或修改的最后元素的*后一个*元素。

头文件<algorithm>

# 不修改序列的操作

## for\_each

template< class InputIt, class UnaryFunction >  
UnaryFunction for\_each( InputIt first, InputIt last, UnaryFunction f );

按顺序应用给定的函数对象 f 到解引用范围 [first, last) 中每个迭代器的结果。

**可能的实现:**

template<class InputIt, class UnaryFunction>

UnaryFunction for\_each(InputIt first, InputIt last, UnaryFunction f){

for (; first != last; ++first) {

f(\*first);

}

return f; // C++11 起隐式移动}

**例如:**

[std::vector](http://zh.cppreference.com/w/cpp/container/vector)<int> nums{3, 4, 2, 8, 15, 267};

std::for\_each(nums.begin(), nums.end(), [](int &n){ n++; });

这里的function使用了lambda表达式,C++98中不可用,但可以使用boost.lambda

## count/count\_if

template< class InputIt, class T >

typename iterator\_traits<InputIt>::difference\_type

    count( InputIt first, InputIt last, const T &value );

template< class InputIt, class UnaryPredicate >

constexpr typename iterator\_traits<InputIt>::difference\_type

              count\_if( InputIt first, InputIt last, UnaryPredicate p );

返回范围 [first, last) 中满足特定判别标准的元素数。

其中模板1根据值来进行判断，模板2使用一元函数进行判断是否满足条件；

例如：

int num\_items1 = std::count(v.begin(), v.end(), target1);

int num\_items3 = std::count\_if(v.begin(), v.end(), [](int i){return i % 3 == 0;});

## mismatch

## find/find\_if/find\_end/find\_first\_of/adjacend\_find

## search/search\_n

# 修改序列的操作

## copy/copy\_backward

## fill/fill\_n

## transform

**模板<一>**

template< class InputIt, class OutputIt, class UnaryOperation >

OutputIt transform( InputIt first1, InputIt last1, OutputIt d\_first,

                    UnaryOperation unary\_op );

应用一元函数 unary\_op 到 [first1, last1) 所定义的范围。

**模板<二>**

template< class InputIt1, class InputIt2, class OutputIt, class BinaryOperation >

OutputIt transform( InputIt1 first1, InputIt1 last1, InputIt2 first2,

                    OutputIt d\_first, BinaryOperation binary\_op );

应用二元函数 binary\_op 到来自二个范围的元素对：一个以 [first1, last1) 定义，而另一个始于 first2

std::transform 应用给定的函数到范围并存储结果于始于 d\_first 的另一范围。

**参数:**

first1, last1: 要变换的第一个元素范围

first2: 要变换的第二个元素的起始

d\_first: 目标范围的起始,可以等于first1或first2

unary\_op: 将要应用的一元函数的签名

binary\_op: 被使用的二元函数对象

**返回值:**

指向最后一个变换的元素的输出迭代器。

可能的实现:

|  |
| --- |
| **版本一** |
| template<class InputIt, class OutputIt, class UnaryOperation>  OutputIt transform(InputIt first1, InputIt last1, OutputIt d\_first,  UnaryOperation unary\_op){  while (first1 != last1) {  \*d\_first++ = unary\_op(\*first1++);  }  return d\_first;} |

|  |
| --- |
| **版本二** |
| template<class InputIt1, class InputIt2,  class OutputIt, class BinaryOperation>  OutputIt transform(InputIt1 first1, InputIt1 last1, InputIt2 first2,  OutputIt d\_first, BinaryOperation binary\_op){  while (first1 != last1) {  \*d\_first++ = binary\_op(\*first1++, \*first2++);  }  return d\_first;} |

**注意:**

std::transform 不保证按顺序应用 unary\_op 或 binary\_op 。为按顺序应用函数到数列，或应用修改序列元素的函数，应使用 [std::for\_each](https://zh.cppreference.com/w/cpp/algorithm/for_each" \o "cpp/algorithm/for each) 。

## generate/generate\_n

## remove/remove\_if/remove\_copy/remove\_copy\_if

## replace/replace\_if/replace\_copy/replace\_copy\_if

## swap/swap\_ranges/iter\_swap

## reverse/reverse\_copy

## rotate/rotate\_copy

## random\_shuffle

## unique

## unique\_copy

# 划分操作

## partition

## stable\_partition

# 排序操作

## sort

在很多应用中，排序都是至关重要的，而且很多 [STL](http://c.biancheng.net/stl/" \t "http://c.biancheng.net/view/_blank) 算法也只适用于有序对象序列。定义在 algorithm 头文件中的函数模板 sort<Iter>() 默认会将元素段排成升序，这也就意味着排序的对象的类型需要支持 < 运算符。

函数模板 sort<Iter>() 的类型参数 Iter 是元素段元素对应的迭代器类型，而且它们必须支持随机访问迭代器。这表明 sort() 算法只能对提供随机访问迭代器的容器中的元素进行排序，也说明 sort() 只能接受 array、vector、deque 或标准数组中的元素。

升序：std::sort(++std::begin(numbers),--std::end(numbers));

降序：std::sort(std::begin(numbers), std::end(numbers), std::greater<>());

## partial\_sort

## partial\_sort\_copy

## stable\_sort

## nth\_element

# 二分搜索操作（在已排序基础上）

lower\_bound

upper\_bound

binary\_bound

equal\_range

# 集合操作（在已排序基础上）

merge

inplace\_merge

includes

set\_difference

set\_intersection

set\_symmetric\_difference

set\_union

# 堆操作

is\_heap

make\_heap

push\_heap

pop\_heap

sort\_heap

# 最小、最大操作

max

max\_element

min

min\_element

# 比较操作

equal

lexicographical\_compare

# 排列操作

next\_permutation

prev\_permutation

# 赋值运算

accumulate

inner\_product

adjacent\_difference

partial\_sum

# 未初始化内存上的操作

uninitialized\_

## C库

定义于头文件<cstdlib>

qsort

bsearch