STL 算法部分主要由头文件<algorithm>,<numeric>,<functional>组成。要使用 STL 中的算法函数必须包含头文件<algorithm>，对于数值算法须包含<numeric>，<functional>中则定义了一些模板类，用来声明函数对象。

STL 中算法大致分为四类：

1、非可变序列算法：指不直接修改其所操作的容器内容的算法。

2、可变序列算法：指可以修改它们所操作的容器内容的算法。

3、排序算法：包括对序列进行排序和合并的算法、搜索算法以及有序序列上的集合操作。

4、数值算法：对容器内容进行数值计算。

以下对所有算法进行细致分类并标明功能：

<一>查找算法(13 个)：判断容器中是否包含某个值

adjacent_find: 在 iterator 对标识元素范围内，查找一对相邻重复元素，找到则返回指向这对元素的第一个元素的 ForwardIterator。否则返回 last。重载版本使用输入的二元操作符代替相等的判断。

binary_search: 在有序序列中查找 value，找到返回 true。重载的版本实用指定的比较函数对象或函数指针来判断相等。

count: 利用等于操作符，把标志范围内的元素与输入值比较，返回相等元素个数。

count_if: 利用输入的操作符，对标志范围内的元素进行操作，返回结果为 true 的个数。

equal_range: 功能类似 equal，返回一对 iterator，第一个表示 lower_bound，第二个表示 upper_bound。

find: 利用底层元素的等于操作符，对指定范围内的元素与输入值进行比较。当匹配时，结束搜索，返回该元素的一个 InputIterator。

find_end: 在指定范围内查找"由输入的另外一对 iterator 标志的第二个序列"的最后一次出现。找到则返回最后一对的第一个 ForwardIterator，否则返回输入的"另外一对"的第一个 ForwardIterator。重载版本使用用户输入的操作符代替等于操作。

find_first_of: 在指定范围内查找"由输入的另外一对 iterator 标志的第二个序列"中任意一个元素的第一次出现。重载版本中使用了用户自定义操作符。

find_if: 使用输入的函数代替等于操作符执行 find。

lower_bound: 返回一个 ForwardIterator，指向在有序序列范围内的可以插入指定值而不破坏容器顺序的第一个位置。重载函数使用自定义比较操作。

upper_bound: 返回一个 ForwardIterator，指向在有序序列范围内插入 value 而不破坏容器顺序的最后一个位置，该位置标志一个大于 value 的值。重载函数使用自定义比较操作。

search: 给出两个范围，返回一个 ForwardIterator，查找成功指向第一个范围内第一次出现子序列(第二个范围)的位置，查找失败指向 last1。重载版本使用自定义的比较操作。

search_n: 在指定范围内查找 val 出现 n 次的子序列。重载版本使用自定义的比较操作。

<二>排序和通用算法(14 个)：提供元素排序策略

inplace_merge: 合并两个有序序列，结果序列覆盖两端范围。重载版本使用输入的操作进行排序。

merge: 合并两个有序序列，存放到另一个序列。重载版本使用自定义的比较。

nth_element: 将范围内的序列重新排序，使所有小于第 n 个元素的元素都出现在它前面，而大于它的都出现在后面。重载版本使用自定义的比较操作。

partial_sort: 对序列做部分排序，被排序元素个数正好可以被放到范围内。重载版本使用自定义的比较操作。

partial_sort_copy: 与 partial_sort 类似，不过将经过排序的序列复制到另一个容器。

partition: 对指定范围内元素重新排序，使用输入的函数，把结果为 true 的元素放在结果为 false 的元素之前。

random_shuffle: 对指定范围内的元素随机调整次序。重载版本输入一个随机数产生操作。

reverse: 将指定范围内元素重新反序排序。

reverse_copy: 与 reverse 类似，不过将结果写入另一个容器。

rotate: 将指定范围内元素移到容器末尾，由 middle 指向的元素成为容器第一个元素。

rotate_copy: 与 rotate 类似，不过将结果写入另一个容器。

sort: 以升序重新排列指定范围内的元素。重载版本使用自定义的比较操作。

stable_sort: 与 sort 类似，不过保留相等元素之间的顺序关系。

stable_partition: 与 partition 类似，不过不保证保留容器中的相对顺序。

<三>删除和替换算法(15 个)

copy: 复制序列

copy_backward: 与 copy 相同，不过元素是以相反顺序被拷贝。

iter_swap: 交换两个 ForwardIterator 的值。

remove: 删除指定范围内所有等于指定元素的元素。注意，该函数不是真正删除函数。内置函数不适合使用 remove 和 remove_if 函数。

remove_copy: 将所有不匹配元素复制到一个制定容器，返回 OutputIterator 指向被拷贝的末元素的下一个位置。

remove_if: 删除指定范围内输入操作结果为 true 的所有元素。

remove_copy_if: 将所有不匹配元素拷贝到一个指定容器。

replace: 将指定范围内所有等于 vold 的元素都用 vnew 代替。

replace_copy: 与 replace 类似，不过将结果写入另一个容器。

replace_if: 将指定范围内所有操作结果为 true 的元素用新值代替。

replace_copy_if: 与 replace_if，不过将结果写入另一个容器。

swap: 交换存储在两个对象中的值。

swap_range: 将指定范围内的元素与另一个序列元素值进行交换。

unique: 清除序列中重复元素，和 remove 类似，它也不能真正删除元素。重载版本使用自定义比较操作。

unique_copy: 与 unique 类似，不过把结果输出到另一个容器。

<四>排列组合算法(2 个)：提供计算给定集合按一定顺序的所有可能排列组合

next_permutation: 取出当前范围内的排列，并重新排序为下一个排列。重载版本使用自定义的比较操作。

prev_permutation: 取出指定范围内的序列并将它重新排序为上一个序列。如果不存在上一个序列则返回 false。重载版本使用自定义的比较操作。

<五>算术算法(4 个)

accumulate: iterator 对标识的序列段元素之和，加到一个由 val 指定的初始值上。重载版本不再做加法，而是传进来的二元操作符被应用到元素上。

partial_sum: 创建一个新序列，其中每个元素值代表指定范围内该位置前所有元素之和。重载版本使用自定义操作代替加法。

inner_product: 对两个序列做内积(对应元素相乘，再求和)并将内积加到一个输入的初始值上。重载版本使用用户定义的操作。

adjacent_difference: 创建一个新序列，新序列中每个新值代表当前元素与上一个元素的差。重载版本用指定二元操作计算相邻元素的差。

<六>生成和异变算法(6 个)

fill: 将输入值赋给标志范围内的所有元素。

fill_n: 将输入值赋给 first 到 first+n 范围内的所有元素。

for_each: 用指定函数依次对指定范围内所有元素进行迭代访问，返回所指定的函数类型。该函数不得修改序列中的元素。

generate: 连续调用输入的函数来填充指定的范围。

generate_n: 与 generate 函数类似，填充从指定 iterator 开始的 n 个元素。

transform: 将输入的操作作用与指定范围内的每个元素，并产生一个新的序列。重载版本将操作作用在一对元素上，另外一个元素来自输入的另外一个序列。结果输出到指定容器。

<七>关系算法(8 个)

equal: 如果两个序列在标志范围内元素都相等，返回 true。重载版本使用输入的操作符代替默认的等于操作符。

includes: 判断第一个指定范围内的所有元素是否都被第二个范围包含，使用底层元素的<操作符，成功返回 true。重载版本使用用户输入的函数。

lexicographical_compare: 比较两个序列。重载版本使用用户自定义比较操作。

max: 返回两个元素中较大一个。重载版本使用自定义比较操作。

max_element: 返回一个 ForwardIterator，指出序列中最大的元素。重载版本使用自定义比较操作。

min: 返回两个元素中较小一个。重载版本使用自定义比较操作。

min_element: 返回一个 ForwardIterator，指出序列中最小的元素。重载版本使用自定义比较操作。

mismatch: 并行比较两个序列，指出第一个不匹配的位置，返回一对 iterator，标志第一个不匹配元素位置。如果都匹配，返回每个容器的 last。重载版本使用自定义的比较操作。

<八>集合算法(4 个)

set_union: 构造一个有序序列，包含两个序列中所有的不重复元素。重载版本使用自定义的比较操作。

set_intersection: 构造一个有序序列，其中元素在两个序列中都存在。重载版本使用自定义的比较操作。

set_difference: 构造一个有序序列，该序列仅保留第一个序列中存在的而第二个中不

存在的元素。重载版本使用自定义的比较操作。

set_symmetric_difference: 构造一个有序序列，该序列取两个序列的对称差集(并集-交集)。

<九>堆算法(4 个)

make_heap: 把指定范围内的元素生成一个堆。重载版本使用自定义比较操作。

pop_heap: 并不真正把最大元素从堆中弹出，而是重新排序堆。它把 first 和 last-1 交换，然后重新生成一个堆。可使用容器的 back 来访问被"弹出"的元素或者使用 pop_back 进行真正的删除。重载版本使用自定义的比较操作。

push_heap: 假设 first 到 last-1 是一个有效堆，要被加入到堆的元素存放在位置 last-1，重新生成堆。在指向该函数前，必须先把元素插入容器后。重载版本使用指定的比较操作。

sort_heap: 对指定范围内的序列重新排序，它假设该序列是个有序堆。重载版本使用自定义比较操作。