# <algorithm>

Defines Standard Template Library (STL) container template functions that perform algorithms.

(see relevant links below for specific algorithm syntax)

## Remarks

The STL algorithms are generic because they can operate on a variety of data structures. The data structures that they can operate on include not only the STL container classes such as **vector** and **list**, but also program-defined data structures and arrays of elements that satisfy the requirements of a particular algorithm. STL algorithms achieve this level of generality by accessing and traversing the elements of a container indirectly through iterators.

STL algorithms process iterator ranges that are typically specified by their beginning or ending positions. The ranges referred to must be valid in the sense that all pointers in the ranges must be dereferenceable and, within the sequences of each range, the last position must be reachable from the first by incrementation.

The STL algorithms extend the actions supported by the operations and member functions of each STL container and allow working, for example, with different types of container objects at the same time. Two suffixes have been used to convey information about the purpose of the algorithms.

* The if suffix indicates that the algorithm is used with function objects operating on the values of the elements rather than on the values of the elements themselves. The **find\_if** algorithm looks for elements whose values satisfy the criterion specified by a function object, and the **find** algorithm searches for a particular value.
* The \_copy suffix indicates that the algorithm not only manipulates the values of the elements but also copies the modified values into a destination range. The **reverse** algorithm reverses the order of the elements within a range, and the **reverse\_copy** algorithm also copies the result into a destination range.

STL algorithms are often classified into groups that indicate something about their purpose or requirements. These include modifying algorithms that change the value of elements as compared with non modifying algorithms that do not. Mutating algorithms change the order of elements, but not the values of their elements. Removing algorithms can eliminate elements from a range or a copy of a range. Sorting algorithms reorder the elements in a range in various ways and sorted range algorithms only act on algorithms whose elements have been sorted in a particular way.

The STL numeric algorithms that are provided for numerical processing have their own header file <numeric>, and function objects and adaptors are defined in the header<functional> Function objects that return Boolean values are known as predicates. The default binary predicate is the comparison **operator<**. In general, the elements being ordered need to be less than comparable so that, given any two elements, it can be determined either that they are equivalent (in the sense that neither is less than the other) or that one is less than the other. This results in an ordering among the non equivalent elements.

### Functions

|  |  |
| --- | --- |
| adjacent\_find | Searches for two adjacent elements that are either equal or satisfy a specified condition. |
| all\_of | Returns true when a condition is present at each element in the given range. |
| any\_of | Returns true when a condition is present at least once in the specified range of elements. |
| binary\_search | Tests whether there is an element in a sorted range that is equal to a specified value or that is equivalent to it in a sense specified by a binary predicate. |
| copy | Assigns the values of elements from a source range to a destination range, iterating through the source sequence of elements and assigning them new positions in a forward direction. |
| copy\_backward | Assigns the values of elements from a source range to a destination range, iterating through the source sequence of elements and assigning them new positions in a backward direction. |
| copy\_if | Copy all elements in a given range that test true for a specified condition |
| copy\_n | Copies a specified number of elements. |
| count | Returns the number of elements in a range whose values match a specified value. |
| count\_if | Returns the number of elements in a range whose values match a specified condition. |
| equal | Compares two ranges element by element either for equality or equivalence in a sense specified by a binary predicate. |
| equal\_range | Finds a pair of positions in an ordered range, the first less than or equivalent to the position of a specified element and the second greater than the element's position, where the sense of equivalence or ordering used to establish the positions in the sequence may be specified by a binary predicate. |
| fill | Assigns the same new value to every element in a specified range. |
| fill\_n | Assigns a new value to a specified number of elements in a range starting with a particular element. |
| find | Locates the position of the first occurrence of an element in a range that has a specified value. |
| find\_end | Looks in a range for the last subsequence that is identical to a specified sequence or that is equivalent in a sense specified by a binary predicate. |
| find\_first\_of | Searches for the first occurrence of any of several values within a target range or for the first occurrence of any of several elements that are equivalent in a sense specified by a binary predicate to a specified set of the elements. |
| find\_if | Locates the position of the first occurrence of an element in a range that satisfies a specified condition. |
| find\_if\_not | Returns the first element in the indicated range that does not satisfy a condition. |
| for\_each | Applies a specified function object to each element in a forward order within a range and returns the function object. |
| generate | Assigns the values generated by a function object to each element in a range. |
| generate\_n | Assigns the values generated by a function object to a specified number of element is a range and returns to the position one past the last assigned value. |
| includes | Tests whether one sorted range contains all the elements contained in a second sorted range, where the ordering or equivalence criterion between elements may be specified by a binary predicate. |
| inplace\_merge | Combines the elements from two consecutive sorted ranges into a single sorted range, where the ordering criterion may be specified by a binary predicate. |
| is\_heap | Returns true if the elements in the specified range form a heap. |
| is\_heap\_until | Returns true if the specified range forms a heap until the last element. |
| is\_partitioned | Returns true if all the elements in the given range that test true for a condition come before any elements that test false. |
| is\_sorted | Returns true if the elements in the specified range are in sorted order. |
| is\_sorted\_until | Returns true if the elements in the specified range are in sorted order. |
| iter\_swap | Exchanges two values referred to by a pair of specified iterators. |
| lexicographical\_compare | Compares element by element between two sequences to determine which is lesser of the two. |
| lower\_bound | Finds the position of the first element in an ordered range that has a value greater than or equivalent to a specified value, where the ordering criterion may be specified by a binary predicate. |
| make\_heap | Converts elements from a specified range into a heap in which the first element is the largest and for which a sorting criterion may be specified with a binary predicate. |
| max | Compares two objects and returns the larger of the two, where the ordering criterion may be specified by a binary predicate. |
| max\_element | Finds the first occurrence of largest element in a specified range where the ordering criterion may be specified by a binary predicate. |
| merge | Combines all the elements from two sorted source ranges into a single, sorted destination range, where the ordering criterion may be specified by a binary predicate. |
| min | Compares two objects and returns the lesser of the two, where the ordering criterion may be specified by a binary predicate. |
| min\_element | Finds the first occurrence of smallest element in a specified range where the ordering criterion may be specified by a binary predicate. |
| minmax | Compares two input parameters and returns them as a pair, in order of least to greatest. |
| minmax\_element | Performs the work performed by min\_element and max\_element in one call. |
| mismatch | Compares two ranges element by element either for equality or equivalent in a sense specified by a binary predicate and locates the first position where a difference occurs. |
| <alg> move | Move elements associated with a specified range. |
| move\_backward | Moves the elements of one iterator to another. The move starts with the last element in a specified range, and ends with the first element in that range. |
| next\_permutation | Reorders the elements in a range so that the original ordering is replaced by the lexicographically next greater permutation if it exists, where the sense of next may be specified with a binary predicate. |
| none\_of | Returns true when a condition is never present among elements in the given range. |
| nth\_element | Partitions a range of elements, correctly locating the *n*th element of the sequence in the range so that all the elements in front of it are less than or equal to it and all the elements that follow it in the sequence are greater than or equal to it. |
| partial\_sort | Arranges a specified number of the smaller elements in a range into a nondescending order or according to an ordering criterion specified by a binary predicate. |
| partial\_sort\_copy | Copies elements from a source range into a destination range where the source elements are ordered by either less than or another specified binary predicate. |
| partition | Classifies elements in a range into two disjoint sets, with those elements satisfying a unary predicate preceding those that fail to satisfy it. |
| partition\_copy | Copies elements for which a condition is true to one destination, and for which the condition is false to another. The elements must come from a specified range. |
| partition\_point | Returns the first element in the given range that does not satisfy the condition. The elements are sorted so that those that satisfy the condition come before those that do not. |
| pop\_heap | Removes the largest element from the front of a heap to the next-to-last position in the range and then forms a new heap from the remaining elements. |
| prev\_permutation | Reorders the elements in a range so that the original ordering is replaced by the lexicographically next greater permutation if it exists, where the sense of next may be specified with a binary predicate. |
| push\_heap | Adds an element that is at the end of a range to an existing heap consisting of the prior elements in the range. |
| random\_shuffle | Rearranges a sequence of *N* elements in a range into one of *N*! possible arrangements selected at random. |
| remove | Eliminates a specified value from a given range without disturbing the order of the remaining elements and returning the end of a new range free of the specified value. |
| remove\_copy | Copies elements from a source range to a destination range, except that elements of a specified value are not copied, without disturbing the order of the remaining elements and returning the end of a new destination range. |
| remove\_copy\_if | Copies elements from a source range to a destination range, except that satisfying a predicate are not copied, without disturbing the order of the remaining elements and returning the end of a new destination range. |
| remove\_if | Eliminates elements that satisfy a predicate from a given range without disturbing the order of the remaining elements and returning the end of a new range free of the specified value. |
| replace | Examines each element in a range and replaces it if it matches a specified value. |
| replace\_copy | Examines each element in a source range and replaces it if it matches a specified value while copying the result into a new destination range. |
| replace\_copy\_if | Examines each element in a source range and replaces it if it satisfies a specified predicate while copying the result into a new destination range. |
| replace\_if | Examines each element in a range and replaces it if it satisfies a specified predicate. |
| reverse | Reverses the order of the elements within a range. |
| reverse\_copy | Reverses the order of the elements within a source range while copying them into a destination range |
| rotate | Exchanges the elements in two adjacent ranges. |
| rotate\_copy | Exchanges the elements in two adjacent ranges within a source range and copies the result to a destination range. |
| search | Searches for the first occurrence of a sequence within a target range whose elements are equal to those in a given sequence of elements or whose elements are equivalent in a sense specified by a binary predicate to the elements in the given sequence. |
| search\_n | Searches for the first subsequence in a range that of a specified number of elements having a particular value or a relation to that value as specified by a binary predicate. |
| set\_difference | Unites all of the elements that belong to one sorted source range, but not to a second sorted source range, into a single, sorted destination range, where the ordering criterion may be specified by a binary predicate. |
| set\_intersection | Unites all of the elements that belong to both sorted source ranges into a single, sorted destination range, where the ordering criterion may be specified by a binary predicate. |
| set\_symmetric\_difference | Unites all of the elements that belong to one, but not both, of the sorted source ranges into a single, sorted destination range, where the ordering criterion may be specified by a binary predicate. |
| set\_union | Unites all of the elements that belong to at least one of two sorted source ranges into a single, sorted destination range, where the ordering criterion may be specified by a binary predicate. |
| sort | Arranges the elements in a specified range into a nondescending order or according to an ordering criterion specified by a binary predicate. |
| shuffle | Shuffles (rearranges) elements for a given range using a random number generator. |
| sort\_heap | Converts a heap into a sorted range. |
| stable\_partition | Classifies elements in a range into two disjoint sets, with those elements satisfying a unary predicate preceding those that fail to satisfy it, preserving the relative order of equivalent elements. |
| stable\_sort | Arranges the elements in a specified range into a nondescending order or according to an ordering criterion specified by a binary predicate and preserves the relative ordering of equivalent elements. |
| swap | Exchanges the values of the elements between two types of objects, assigning the contents of the first object to the second object and the contents of the second to the first. |
| swap\_ranges | Exchanges the elements of one range with the elements of another, equal sized range. |
| transform | Applies a specified function object to each element in a source range or to a pair of elements from two source ranges and copies the return values of the function object into a destination range. |
| unique | Removes duplicate elements that are adjacent to each other in a specified range. |
| unique\_copy | Copies elements from a source range into a destination range except for the duplicate elements that are adjacent to each other. |
| upper\_bound | Finds the position of the first element in an ordered range that has a value that is greater than a specified value, where the ordering criterion may be specified by a binary predicate. |