Editing Algorithms library

You are not logged in. Saving will record your IP address in this page's edit history.

Preview

Remember that this is only a preview. Your changes have not yet been saved! → Go to editing area

---

The algorithms library defines functions for a variety of purposes (e.g. searching, sorting, counting, manipulating) that operate on ranges of elements. Note that a range is defined as `[first, last)` where `last` refers to the element *past* the last element to inspect or modify.

Non-modifying sequence operations

Defined in header `<algorithm>`

all_of (C++11) **any_of** (C++11) **none_of** (C++11)	checks if a predicate is `true` for all, any or none of the elements in a range (function template)
for_each	applies a function to a range of elements (function template)
for_each_n (C++17)	applies a function object to the first n elements of a sequence (function template)
count **count_if**	returns the number of elements satisfying specific criteria (function template)
mismatch	finds the first position where two ranges differ (function template)
find **find_if** **find_if_not** (C++11)	finds the first element satisfying specific criteria (function template)
find_end	finds the last sequence of elements in a certain range (function template)
find_first_of	searches for any one of a set of elements (function template)
adjacent_find	finds the first two adjacent items that are equal (or satisfy a given predicate) (function template)
search	searches for a range of elements (function template)
search_n	searches a range for a number of consecutive copies of an element (function template)

Modifying sequence operations

Defined in header `<algorithm>`

copy **copy_if** (C++11)	copies a range of elements to a new location (function template)
copy_n (C++11)	copies a number of elements to a new location (function template)
copy_backward	copies a range of elements in backwards order (function template)
move (C++11)	moves a range of elements to a new location (function template)
move_backward (C++11)	moves a range of elements to a new location in backwards order (function template)
fill	copy-assigns the given value to every element in a range (function template)
fill_n	copy-assigns the given value to N elements in a range (function template)
transform	applies a function to a range of elements, storing results in a destination range (function template)
generate	assigns the results of successive function calls to every element in a range (function template)
generate_n	assigns the results of successive function calls to N elements in a range (function template)
remove **remove_if**	removes elements satisfying specific criteria (function template)

remove_copy **remove_copy_if**	copies a range of elements omitting those that satisfy specific criteria (function template)
replace **replace_if**	replaces all values satisfying specific criteria with another value (function template)
replace_copy **replace_copy_if**	copies a range, replacing elements satisfying specific criteria with another value (function template)
swap	swaps the values of two objects (function template)
swap_ranges	swaps two ranges of elements (function template)
iter_swap	swaps the elements pointed to by two iterators (function template)
reverse	reverses the order of elements in a range (function template)
reverse_copy	creates a copy of a range that is reversed (function template)
rotate	rotates the order of elements in a range (function template)
rotate_copy	copies and rotate a range of elements (function template)
shift_left **shift_right** (C++20)	shifts elements in a range (function template)
random_shuffle (until C++17) **shuffle** (C++11)	randomly re-orders elements in a range (function template)
sample (C++17)	selects n random elements from a sequence (function template)
unique	removes consecutive duplicate elements in a range (function template)
unique_copy	creates a copy of some range of elements that contains no consecutive duplicates (function template)

Partitioning operations

Defined in header `<algorithm>`

is_partitioned (C++11)	determines if the range is partitioned by the given predicate (function template)
partition	divides a range of elements into two groups (function template)
partition_copy (C++11)	copies a range dividing the elements into two groups (function template)
stable_partition	divides elements into two groups while preserving their relative order (function template)
partition_point (C++11)	locates the partition point of a partitioned range (function template)

Sorting operations

Defined in header `<algorithm>`

is_sorted (C++11)	checks whether a range is sorted into ascending order (function template)
is_sorted_until (C++11)	finds the largest sorted subrange (function template)
sort	sorts a range into ascending order (function template)
partial_sort	sorts the first N elements of a range (function template)
partial_sort_copy	copies and partially sorts a range of elements (function template)
stable_sort	sorts a range of elements while preserving order between equal elements (function template)
nth_element	partially sorts the given range making sure that it is partitioned by the given element (function template)

Binary search operations (on sorted ranges)

Defined in header `<algorithm>`

lower_bound	returns an iterator to the first element *not less* than the given value (function template)

upper_bound	returns an iterator to the first element *greater* than a certain value (function template)
binary_search	determines if an element exists in a certain range (function template)
equal_range	returns range of elements matching a specific key (function template)

Other operations on sorted ranges

Defined in header <algorithm>

merge	merges two sorted ranges (function template)
inplace_merge	merges two ordered ranges in-place (function template)

Set operations (on sorted ranges)

Defined in header <algorithm>

includes	returns true if one sequence is a subsequence of another (function template)
set_difference	computes the difference between two sets (function template)
set_intersection	computes the intersection of two sets (function template)
set_symmetric_difference	computes the symmetric difference between two sets (function template)
set_union	computes the union of two sets (function template)

Heap operations

Defined in header <algorithm>

is_heap (C++11)	checks if the given range is a max heap (function template)
is_heap_until (C++11)	finds the largest subrange that is a max heap (function template)
make_heap	creates a max heap out of a range of elements (function template)
push_heap	adds an element to a max heap (function template)
pop_heap	removes the largest element from a max heap (function template)
sort_heap	turns a max heap into a range of elements sorted in ascending order (function template)

Minimum/maximum operations

Defined in header <algorithm>

max	returns the greater of the given values (function template)
max_element	returns the largest element in a range (function template)
min	returns the smaller of the given values (function template)
min_element	returns the smallest element in a range (function template)
minmax (C++11)	returns the smaller and larger of two elements (function template)
minmax_element (C++11)	returns the smallest and the largest elements in a range (function template)
clamp (C++17)	clamps a value between a pair of boundary values (function template)

Comparison operations

Defined in header <algorithm>

equal	determines if two sets of elements are the same (function template)
lexicographical_compare	returns true if one range is lexicographically less than another (function template)
lexicographical_compare_three_way (C++20)	compares two ranges using three-way comparison (function template)

Permutation operations

Defined in header `<algorithm>`

is_permutation (C++11)	determines if a sequence is a permutation of another sequence (function template)
next_permutation	generates the next greater lexicographic permutation of a range of elements (function template)
prev_permutation	generates the next smaller lexicographic permutation of a range of elements (function template)

Numeric operations

Defined in header `<numeric>`

iota (C++11)	fills a range with successive increments of the starting value (function template)
accumulate	sums up a range of elements (function template)
inner_product	computes the inner product of two ranges of elements (function template)
adjacent_difference	computes the differences between adjacent elements in a range (function template)
partial_sum	computes the partial sum of a range of elements (function template)
reduce (C++17)	similar to `std::accumulate`, except out of order (function template)
exclusive_scan (C++17)	similar to `std::partial_sum`, excludes the ith input element from the ith sum (function template)
inclusive_scan (C++17)	similar to `std::partial_sum`, includes the ith input element in the ith sum (function template)
transform_reduce (C++17)	applies an invocable, then reduces out of order (function template)
transform_exclusive_scan (C++17)	applies an invocable, then calculates exclusive scan (function template)
transform_inclusive_scan (C++17)	applies an invocable, then calculates inclusive scan (function template)

Operations on uninitialized memory

Defined in header `<memory>`

uninitialized_copy	copies a range of objects to an uninitialized area of memory (function template)
uninitialized_copy_n (C++11)	copies a number of objects to an uninitialized area of memory (function template)
uninitialized_fill	copies an object to an uninitialized area of memory, defined by a range (function template)
uninitialized_fill_n	copies an object to an uninitialized area of memory, defined by a start and a count (function template)
uninitialized_move (C++17)	moves a range of objects to an uninitialized area of memory (function template)
uninitialized_move_n (C++17)	moves a number of objects to an uninitialized area of memory (function template)
uninitialized_default_construct (C++17)	constructs objects by default-initialization in an uninitialized area of memory, defined by a range (function template)
uninitialized_default_construct_n (C++17)	constructs objects by default-initialization in an uninitialized area of memory, defined by a start and a count (function template)
uninitialized_value_construct (C++17)	constructs objects by value-initialization in an uninitialized area of memory, defined by a range (function template)
uninitialized_value_construct_n (C++17)	constructs objects by value-initialization in an uninitialized area of memory, defined by a start and a count (function template)
destroy (C++17)	destroys a range of objects

	(function template)
destroy_n (C++17)	destroys a number of objects in a range (function template)
destroy_at (C++17)	destroys an object at a given address (function template)
construct_at (C++20)	creates an object at a given address (function template)

C library

Defined in header <cstdlib>

qsort	sorts a range of elements with unspecifie (function)
bsearch	searches an array for an element of unsp (function)

unspecified type