Iterator Concepts

	indirectly_readable	specifies that a type is indirectly readable by applying operator *
	indirectly_writable	specifies that a value can be written to an iterator's referenced object
	weakly_incrementable	specifies that a `semiregular` type can be incremented with pre- and post-increment operators
	incrementable	specifies that the increment operation on a `weakly_incrementable` type is equality-preserving and that the type is `equality_comparable`
	input_or_output_iterator	specifies that objects of a type can be incremented and dereferenced
	sentinel_for	specifies a type is a sentinel for an `input_or_output_iterator` type
	sized_sentinel_for	specifies that the - operator can be applied to an iterator and a sentinel to calculate their difference in constant time
	input_iterator	specifies that a type is an input iterator, that is, its referenced values can be read and it can be both pre- and post-incremented
	output_iterator	specifies that a type is an output iterator for a given value type, that is, values of that type can be written to it and it can be both pre- and post-incremented
	forward_iterator	specifies that an `input_iterator` is a forward iterator, supporting equality comparison and multi-pass
	bidirectional_iterator	specifies that a `forward_iterator` is a bidirectional iterator, supporting movement backwards
	random_access_iterator	specifies that a `bidirectional_iterator` is a random-access iterator, supporting advancement in constant time and subscripting
	contiguous_iterator	specifies that a `random_access_iterator` is a contiguous iterator, referring to elements that are contiguous in memory
	indirectly_readable	specifies that a type is indirectly readable by applying operator *
	indirectly_writable	specifies that a value can be written to an iterator's referenced object
	weakly_incrementable	specifies that a `semiregular` type can be incremented with pre- and post-increment operators
	incrementable	specifies that the increment operation on a `weakly_incrementable` type is equality-preserving and that the type is `equality_comparable`
	input_or_output_iterator	specifies that objects of a type can be incremented and dereferenced

Iterator Adaptors

	reverse_iterator	iterator adaptor for reverse-order traversal (class template)
	make_reverse_iterator	creates a std::reverse_iterator of type inferred from the argument (function template)
	move_iterator	iterator adaptor which dereferences to an rvalue reference (class template)
	move_sentinel	sentinel adaptor for use with std::move_iterator (class template)
	make_move_iterator	creates a std::move_iterator of type inferred from the argument (function template)
	common_iterator	adapts an iterator type and its sentinel into a common iterator type (class template)
	default_sentinel_t	default sentinel for use with iterators that know the bound of their range (class)
	counted_iterator	iterator adaptor that tracks the distance to the end of the range (class template)
	unreachable_sentinel_t	sentinel that always compares unequal to any `weakly_incrementable` type (class)
	back_insert_iterator	iterator adaptor for insertion at the end of a container (class template)
	back_inserter	creates a std::back_insert_iterator of type inferred from the argument (function template)
	front_insert_iterator	iterator adaptor for insertion at the front of a container (class template)
	front_inserter	creates a std::front_insert_iterator of type inferred from the argument (function template)
	insert_iterator	iterator adaptor for insertion into a container (class template)
	inserter	creates a std::insert_iterator of type inferred from the argument (function template)

Stream Iterators

	istream_iterator	input iterator that reads from std::basic_istream (class template)
	ostream_iterator	output iterator that writes to std::basic_ostream (class template)
	istreambuf_iterator	input iterator that reads from std::basic_streambuf (class template)
	ostreambuf_iterator	output iterator that writes to std::basic_streambuf (class template)
	istream_iterator	input iterator that reads from std::basic_istream (class template)

Iterator Operations

Note: A *niebloid* is a funciton object that disables Koenig lookup (*aka* Argument Dependent Lookup).

	advance	advances an iterator by given distance (function template)
	distance	returns the distance between two iterators (function template)
	next	increment an iterator (function template)
	prev	decrement an iterator (function template)
	ranges::advance	advances an iterator by given distance or to a given bound (niebloid)
	ranges::distance	returns the distance between an iterator and a sentinel, or between the beginning and end of a range (niebloid)
	ranges::next	increment an iterator by a given distance or to a bound (niebloid)
	ranges::prev	decrement an iterator by a given distance or to a bound (niebloid)

Non-member functions that provide generic interface for containers

Prefer these wherever possible.

	begincbegin	returns an iterator to the beginning of a container or array (function template)
	endcend	returns an iterator to the end of a container or array (function template)
	rbegincrbegin	returns a reverse iterator to a container or array (function template)
	rendcrend	returns a reverse end iterator for a container or array (function template)
	sizessize	returns the size of a container or array (function template)
	empty	checks whether the container is empty (function template)
	data	obtains the pointer to the underlying array (function template)