【C++】Day43

Class	C++
Date	@January 22, 2022
Material	
Series Number	
Summary	

【Ch10】Generic Algorithms

10.5.1 The Five Iterator Categories

Input Iterators: can read elements in a sequence. An input iterator must provide:

- Equality and inequality operators(`==`, `!=`) to compare two iterators
- Prefix and postfix increment(`++`) to advance the iterator
- Dereference operator(`*`) to read an element; dereference may appear on the right-hand side of an assignment.
- The arrow operator(`->`) as a synonym for `(*it).member`-that is, dereference the iterators and fetch a member from the underlying object.

Input iterators may be used only sequentially. We are guaranteed that `*it++` is valid, but incrementing an input iterator may invalidate all other iterators into the stream.

As a result, there is no guarantee that we can save the state of an input iterator and examine an element through that saved iterator. Input iterators, therefore, may be used only for single-pass algorithms.

The `find` and `accumulate` algorithms require input iterators; `istream_iterators` are input iterators.

Output Iterator

They can be thought of as having complementary functionality to input iterators; they write rather than read elements. Output iterators must provide

- Prefix and postfix increment(`++`) to advance the iterator
- Dereference(`*`), which may appear only as the left-hand side of an assignment. (Assigning to a dereferenced output iterator writes to the underlying element.)

The `ostream_iterator` type is an output iterator.

Forward Iterators

They can read and write a given sequence. They move in only one direction through the sequence.

Forward iterators support all the operations of both input iterators and output iterators. Moreover, they can read or write the same element multiple times.

The iterators on `forward_list` are forward iterators.

Bidirectional Iterators

They can read and write a sequence forward or backward. In addition to supporting all the operations of a forward iterator, a bidirectional iterator also supports the prefix and postfix decrement (`--`) operators.

The `reverse` algorithm requires bidirectional iterators, and aside from `forward_list`, the library containers supply iterators that meet the requirements for a bidirectional iterator.

Random-access Iterators

They provide constant-time access to any position in the sequence. These iterators support all the functionality of bidirectional iterators. In addition, random-access iterators support the following operations:

- The relational operators(`<`, `<=`, `>`, and `>=`) to compare the relative positions of two iterators.
- Addition and subtraction operators(`+`, `+=`, `-`, and `-=`) on an iterator and an integral value. The result is the iterator advanced the integral number of elements within the sequence.

- The subtraction operator (`-`) when applied to two iterators, which yields the distance between two iterators.
- The subscript operator(`iter[n]`) as a synonym for `*(iter + n)`.

The `sort` algorithms require random-access iterators. Iterators for `array`, `deque`, `string`, and `vector` are random-access iterators, as are pointers when used to access elements of a built-in array.p,