【C++】Day47

Class	C++
Date	@January 27, 2022
Material	
Series Number	
Summary	

【Ch11】Associative Container

11.3.5 Accessing Elements

A Different, Iterator-Oriented Solution

Alternatively, we can solve our problem using `lower_bound` and `uppoer_bound`. Each of these operations take a key and returns an iterator.

- If the key is in the container, the iterator returned from `lower_bound` will refer to the first instance of that key and the iterator returned by `upper_bound` will refer just after the last instance of the key.
- If the element is not in the `multima`p, the `lower_bound` and `upper_bound` will return equal iterators; both will refer to the point at which the key can be inserted without disrupting the order.

Thus, calling `lower_bound` and `upper_bound` on the same key yields an iterator range that denotes all the elements with that key.

Use these operations, we can rewrite our program as follows:

```
//beg and end denote the range of elements for this author
for(auto beg = authors.lower_bound(search_item), end = authors.uppoer_bound(search_item);
  beg != end; ++beg) {
  std::cout << beg->second << std::endl; //print each title
}
```

Note: If `lower_bound` and `upper_bound` return the same iterator, then the given key is not in the container.

The `equal_range` Function

The remaining way to solve this problem is the most direct of the three approaches, we can call `equal_range`.

This function takes a key and returns a pair of iterators.

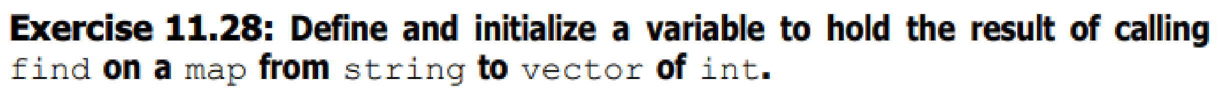
If the key is present, then the first iterator refers to the first instance of the key and the second iterator refers one past the last instance of the key.

If no matching element is found, then both the first and second iterators refer to the position where this key can be inserted.

We can use `equal_range` to modify our program once again:

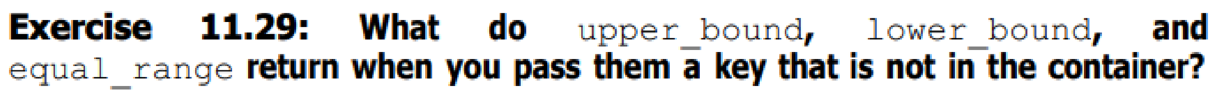
```
for(auto pos = authors.equal_range(search_item); pos.first != pos.second; ++pos.first) {
  //pos.first holds the iterator to the element
  //pos.first->second accesses the value held by the iterator pos.first
  std::cout << pos.first->second :: std::endl;
}
```

Exercise



Exercise 11.28: Define and initialize a variable to hold the result of calling `find` on a `map` from `string` to `vector` of `int`.

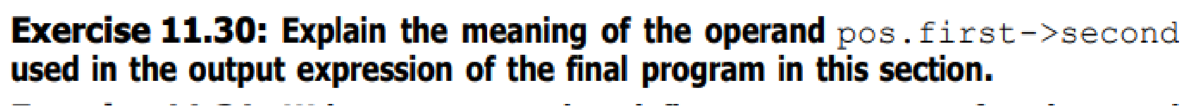
See 11_28.cpp for code



Exercise 11.29: What do `upper_bound`, `lower_bound`, and `equal_range` return when you pass them a key that is not in the container?

`upper_bound` and `lower_bound` will return the same iterator denoting where the key should be inserted.

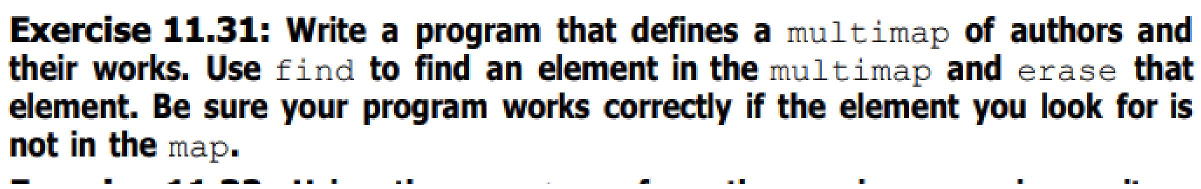
`equal_range` will return a pair, whose elements are both the iterator denoting where the key should be inserted.


Exercise 11.30: Explain the meaning of the operand `pos.first->second` **used in the output expression of the final program in this section.**

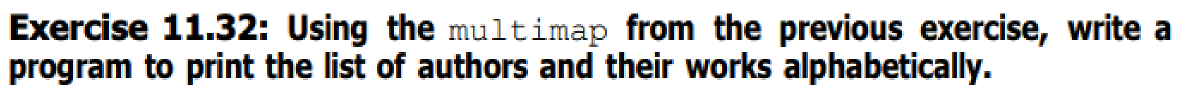
`pos` is a pair containing two iterators, the first iterator is the position where `search_item` is first found, the second iterator denotes the position one past the last instance of `search_item`.

`pos.first` gets the iterators denoting a pair in the map.

`pos.first→second` denotes the iterator, gets dereferenced to get the pair in the map, and then take the value of the pair.


Exercise 11.31: Write a program that defines a `multimap` **of authors and their works. Use** `find` **to find an element in the** `multimap` **and** `erase` **that element. Be sure your program works correctly if the element you look for is not in the** `map`**.**

See 11_31.cpp for code


Exercise 11.32: Using the `multimap` **from the previous exercise, write a program to print the list of authors and their works alphabetically.**

See 11_32.cpp for code