【C++】Day44(2)

Class	C++
Date	@January 24, 2022
Material	
Series Number	
Summary	

【Ch11】Associative Container

11.2 Overview of the Associative Containers

When we initialize a `map`, we have to supply both the key and the value. We wrap each key-value pair inside curly braces:

```
{key, value}
```

to indicate that the items together form one element in the map. The key is the first element in each pair, and the value is the second.

Initializing a multimap or multiset

The keys in a `map` or a `set` must be unique; there can be only one element with a given key.

The `multimap` and `multiset` containers have no such restriction; there can be several elements with the same key.

The following example illustrates the differences between the containers with unique keys and those that have multiple keys.

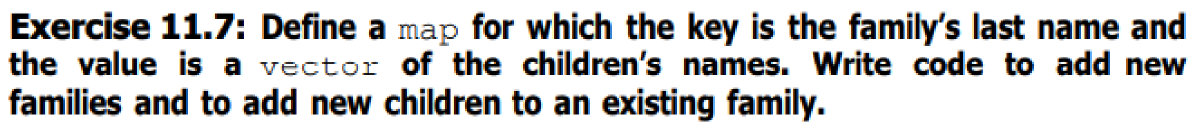
```
//define a vector with 20 elements
//holding two copies of each number from 0 to 9
vector<int> ivec;
```

```
for(int i = 0; i < 10; ++i) {
  ivec.push_back(i);
  ivec.push_back(i);
}

set<int> iset(ivec.cbegin(), ivec.cend());
multiset<int> miset(ivec.cbegin(), ivec.cend());

std::cout << ivec.size() << std::endl;
std::cout << iset.size() << std::endl;
std::cout << miset.size() << std::endl;
```

Exercise



Exercise 11.7: Define a `map` for which the key is the family's last name and the value is a `vector` of the children's names. Write code to add new families and to add new children to an existing family.

See 11_7.cpp for code

11.2.2 Requirements on Key Type

The associative containers place constraints on the type that is used as a key.

For the ordered containers-`map`, `multimap`, `set`, and `multiset`-the key type must define a way to compare the elements. Be default, the library uses the < operator for the key type to compare the keys.

Note: Callable objects passed to a sort algorithm must meet the same requirements as do the keys in an associative container.

Key Types for Ordered Containers

We can supply our own operation to use in place of the `<` operator on keys.

The specified operation must define a strict weak ordering over the key type.

The comparison function must have the following properties:

- Two keys cannot both be "less than" each other; if k1 is "less than" k2, then k2 must never be "less than" k1

- If k1 is “less than” k2 and k2 is “less than” k3, then k1 must be “less than” k3
- If there are two keys, and neither key is “less than” the other, then we’ll say that those keys are “equivalent.” If k1 is “equivalent” to k2 and k2 is “equivalent” to k3, then k1 must be “equivalent” to k3.