C++ 容器类〈map〉

在 C++ 中，〈map〉是标准模板库（STL）的一部分，它提供了一种关联容器，用于存储键值对（key-value pairs）。

map 容器中的元素是按照键的顺序自动排序的，这使得它非常适合需要快速查找和有序数据的场景。

定义和特性

键值对：map 存储的是键值对，其中每个键都是唯一的。

排序：map 中的元素按照键的顺序自动排序，通常是升序。

唯一性：每个键在 map 中只能出现一次。

双向迭代器：map 提供了双向迭代器，可以向前和向后遍历元素。

基本语法

包含头文件:

```
#include <map>
```

声明 map 容器:

```
std::map<key_type, value_type> myMap;
```

key_type 是键的类型。

value_type 是值的类型。

插入元素:

```
myMap[key] = value;
```

访问元素:

```
value = myMap[key];
```

遍历 map:

```
for (std::map<key_type, value_type>::iterator it = myMap.begin(); it != myMap.end();
++it) {
    std::cout << it->first << " => " << it->second << std::endl;
}
```

实例

下面是一个使用 map 的简单实例，我们将创建一个 map 来存储员工的姓名和他们的年龄，并遍历这个 map 来打印每个员工的姓名和年龄。

实例

```
#include <iostream>#include <map>
#include <string>

int main() {
    // 创建一个 map 容器，存储员工的姓名和年龄
    std::map<std::string, int> employees;

    // 插入员工信息
    employees["Alice"] = 30;
    employees["Bob"] = 25;
    employees["Charlie"] = 35;

    // 遍历 map 并打印员工信息
    for (std::map<std::string, int>::iterator it = employees.begin(); it != employee
s.end(); ++it) {
        std::cout << it->first << " is " << it->second << " years old." << std::endl;
    }

    return 0;
}
```

输出结果:

```
Alice is 30 years old.
Bob is 25 years old.
Charlie is 35 years old.
```

进阶用法

检查键是否存在:

```
if (myMap.find(key) != myMap.end()) {
    // 键存在
}
```

删除元素:

```
myMap.erase(key);
```

清空 map:

```
myMap.clear();
```

获取 map 的大小:

```
size_t size = myMap.size();
```

使用自定义比较函数:

实例

```
#include <map>
#include <string>
#include <functional>

bool myCompare(const std::string& a, const std::string& b) {
    return a < b;
}

int main() {
    std::map<std::string, int, std::function<bool(const std::string&, const std::stri
ng&)>> myMap(myCompare);

    // 其他操作...

    return 0;
}
```

map 是 C++ STL 中一个非常有用的容器，特别适合需要快速查找和有序数据的场景。