# C++ STL map & multimap

`map`和`multimap`是C++的STL库中提供的一种容器，常用于代替hash table。本报告研究`map`和`multimap`的常用语法及其应用。

---

## map

- ### 概念

`map`提供两组储存空间，分别是`key`和`value`。`key`与`value`之间有一对一关系，可以透过`key`来查找对应的`value`。`key`跟`value`可以是任意的数据类型。`key`跟`value`之间可以是不同数据类型。`key`彼此之间不能相同。

- ### 所用函数库

`map`功能被包含在头文件`map`中。

```
#include<iostream>
#include<string>
#include<map>
using namespace std;
```

- ### 声明map

```
map<int, int> map1;
map<int, double> map2;
map<string, double> studentScore;
```

- ### 插入新元素

1. 用`insert`函数插入`pair`(若`key`已存在，则不发生改变)

```
studentScore.insert(pair<string, double>("Andy", 97.8)); //The student Andy got 97.8
studentScore.insert(pair<string, double>("Andy", 87.7)); //Andy still gets 97.8
```

2. 用`emplace`函数(若`key`已存在，则不发生改变)。
`emplace`函数写法上简洁，且可节省临时变量的构造与赋值。

```
studentScore.emplace("Tom", 100.0);
```

3. 用`array`(若`key`已存在，则改变其value)

```

studentScore["Tony"] = 95.6;  //Student Tony got 95.6
```

- ## 查找/获值

  1. 迭代器
     用`find`函数找到所求的`key`的地址，将其赋到迭代器`iterator`。
     若查无结果，则返回的值等同于`end`的返回值。

     ```
     auto iter = studentScore.find("Tony");
     int tonyValue = iter->second;  // tonyValue = 95.6
     ```

  2. 函数`at()`

     ```
     tonyValue = studentScore.at("Tony");  // tonyValue = 95.6
     ```

  3. 运算符`[]`

     ```
     tonyValue = studentScore["Toney"];  // tonyValue = 95.6
     ```

- ## 修改

  修改已存在的`key`的`value`，不能使用`insert()`,`emplace()`,`at()`。需用迭代器或运算符`[]`。

  1. 迭代器

     ```
     iter = studentScore.find("Tony");
     iter->second = 88.3;
     ```

  2. 运算符`[]`

     ```
     studentScore["Tony"] = 88.3;
     ```

- ## 遍历元素

  用迭代器遍历每个`pair`

  ```
  for(iter = studentScore.begin(); iter != studentScore.end(); iter++)
  {
      // do something with iter
  }

  // or a reversed iterator
  map<string, double>::reverse_iterator riter;
  for(riter = studentScore.rbegin(); riter != studentScore.rend(); riter++)
  {
      // do something with riter
  }
  ```

- ## 删除元素

1. 迭代器

```
iter = studentScore.find("Tony");
studentScore.erase(iter);
```

2. 直接透过 `key`

```
int n = mapStudent.erase("r123");  // succeed: n=1.  fail: n=0
```

3. 用迭代器删除特定范围--删除全部

```
studentScore.erase(studentScore.begin(), studentScore.end());
```

或是

```
studentScore.clear();
```

- **Bound**

要查找元素的话，也可以用 `lower_bound` 或 `upper_bound` 来查找。返回的是一个迭代器。

```
map<int, string>::iterator iter = studentID.lower_bound(2);  // Find the lower bound of key 2 in the map studentID
iter = studentID.upper_bound(4);  // Find the upper bound of key 4 in the map studentID
```

也可以用 `equal_range` 来查找。 `equal_range` 返回的是一个 `pair`， `pair` 的第一个元素(是个迭代器)是其 `lower_bound` 的结果，第二个元素(也是个迭代器)是其 `upper_bound` 的结果。
衍伸的意义是，如果此 `pair` 两个迭代器是相同的，代表并没有查找到对应的 `key`。

```
iterPair = studentID.equal_range(4);
if(iterPair.first != iterPair.second)
    cout<<"[ "<<iterPair.first->first<<", "<<iterPair.first->second<<" ]"<<endl;
else
    cout<<"Element not found."<<endl;
```

# multimap

- ## 概念

`multimap` 基本使用方法与 `map` 相同，差别在于可以有同样的 `key` 出现，而彼此拥有不同的 `value`。使用 `multimap` 的目的本身就是让一个 `key` 对应到多个 `value`，因此定位到特定 `value` 的 `pair` 与操作并不常见，也因此比起 `map` 少了很多单元素查找/访问功能，例如 `at()`、运算符 `[]`。

- ## 所用函数库

同 `map`，被包含在头文件 `map` 中。

- ## 声明multimap

```
multimap<int, int> map1;
multimap<int, double> map2;
multimap<string, double> studentScore;
```

- ## 插入新元素

插入元素的方法，比起 `map`，少了运算符 `[]`。

可以重复插入同样的 `key`。 `value`可相同也可不相同。即使相同 `key`相同`value`也会产生新元素。

1. `insert()`

```
studentScore.insert(pair<string, double>("Andy", 97.8));  //A Student Andy got 97.8
studentScore.insert(pair<string, double>("Andy", 97.8));  //The other Andy also got 97.8
```

2. `emplace()`

```
studentScore.emplace("Tony", 95.6);  //A student Tony got 95.6
studentScore.emplace("Tony", 91.1);  //The other Tony got 91.1
```

- ## 查找/获值

只能使用迭代器。比起 `map`，少了 `at()`、运算符 `[]`。

1. `find()`

   `find()`只能回传一个 `iterator`，如果存在多个相同 `key`的 `pair`，则有可能得到其中任何一个。(测试报告中做测试)

```
   auto iter = studentScore.find("Tony");
   // iter->seconde = 95.6 or 91.1??
```

2. 找一整组 `key`，用 `equal_range()`

   `equal_range()`在此的原理与在`map`中相同，他返回的是一个 `pair<iterator, iterator>`，前一个迭代器是查找`key`的 `lower_bound`，后一个是 `upper_bound`。其含意就是，所有相同`key`的元素，被包含在这两个`iterator`的区间之间，而且这个区间是前闭后开的。

```
   auto iterPair = studentScore.equal_range("Tony");
   int tonyValueArray[2];
   int i = 0;
   for(auto iter = iterPair.first; iter != iterPair.second; iter++ )
   {
    tonyValueArray[i] = iter->second;
        i++;
   }
   //  tonyValueArray[0] = 95.6;
   //  tonyValueArray[1] = 91.1;
```

3. 找单个，用 `bound`，然后递增

1. 找某个 `key` 的 `lower_bound`，及该 `key` 的第一个 `pair`

```
iter = studentScore.lower_bound("Tony");
```

2. 找某个 `key` 的 `upper_bound`，及该 `key` 的最后一个 `pair`

```
iter = studentScore.upper_bound("Tony");
```

3. `lower_bound` 之后的第 `k` 个 `pair`

```
iter = studentScore.lower_bound("Tony");
for(i = 0; i < k-1; i++ )
        iter ++;
```

- ## 修改

只要用迭代器寻找到目标，就可以修改。寻找方法如上，修改方法如同 `map` 中所介绍。

- ## 遍历元素

1. 遍历 `multimap` 中所有 `pair`

```
for(iter = studentScore.begin(); iter != studentScore.end(); iter++)
{
    // do something with iter
}

// or a reversed iterator
multimap<string, double>::reverse_iterator riter;
for(riter = studentScore.rbegin(); riter != studentScore.rend(); riter++)
{
    // do something with riter
}
```

2. 遍历某组相同 `key` 的所有 `pair`

```
iterPair = studentScore.equal_range("Tony");
for(iter = iterPair->first; iter != iterPair->second; iter++ )
{
    // do something to iter
}
```

- ## 删除元素

1. 透过单一迭代器删除删除 `pair`，先用上述的查找方法定位到要删除的 `pair`，之后删除方法如同 `map` 中所介绍

```
studentScore.erase(iter);
```

2. 移除某组相同`key`的所有`pair`

```
int n = studentScore.erase("Tony");
// n = number of elements removed
```

3. 与`map`相同地有:
   删除特定范围--删除全部

```
studentScore.erase(studentScore.begin(), studentScore.end());
```

   或

```
studentScore.clear();
```

---

# 参考资料

资料来源-以下网页：


1. C/C++ - Map (STL) 用法與心得完全攻略 http://mropengate.blogspot.tw/2015/12/cc-map-stl.html
2. cplusplus.com http://www.cplusplus.com/reference/map/multimap/find/
3. STL中的常用的vector, map, set, Sort用法 http://www.360doc.com/content/10/0814/11/2595782_45943931.shtml用法
4. cppreference.com-std::map http://en.cppreference.com/w/cpp/container/map
5. How to pass an iterator of a container of a template class to a function? https://stackoverflow.com/questions/11437375/how-to-pass-an-iterator-of-a-container-of-a-template-class-to-a-function
6. How to remove a specific pair from a C++ multimap? https://stackoverflow.com/questions/3952476/how-to-remove-a-specific-pair-from-a-c-multimap
7. cppreference.com-std::multimap http://en.cppreference.com/w/cpp/container/multimap
8. What's the difference between std::multimap and std::map > https://stackoverflow.com/questions/8602068/whats-the-difference-between-stdmultimapkey-value-and-stdmapkey-stds
9. Find specific element of multimap https://bytes.com/topic/c/answers/168190-find-specific-element-multimap