

软件工程 2 大作业 Map Research

刘家维 2016013246

一、目标

研究整理标准库中 map 与 multimap 的用法。

二、原理

先在网上搜寻 map 与 multimap 的常见用法，整理成学习文件，此文件要适合面向任何想学习 map 与 multimap 用法的人。接着写测试代码，测试学习到的这些用法、以及有疑虑或不明确的部分。最后写下此测试报告，解释测试代码的结构与各段功能，还有写下测试结论。

三、编程

1. 工程文件中有以下文件:

CP_MapTest.h/cpp、CP_MapBoundTest.h/cpp、CP_MultimapTest.h/cpp、CP_TestMain.cpp。

2. Main 函数在 CP_TestMain.cpp 中，调用其余三对文件的对外接口。三对文件各提供一个对外接口: void mapTest()、void mapBoundTest()、void multimapTest()。

此三个函数分别独立，可以在 main 中自行调整批注，选择每次只测试其中任一项到三项。

3. CP_MapTest.h/cpp 中的 void mapTest() 旨在测试学习文件中关于 map 的用法，但除去有关 bound 与 equal_range 的用法，我将它另外放在 void mapBoundTest() 中测试。void multimapTest() 则是测试学习文件中关于 multimap 的用法，其中若有与 map 相同的用法，则略过不测。

4. void mapTest():

配合学习文件中的 map 内容，此测试可以分成多个部分，由多个函数分别完成。在此举的例子是学生与学生分数的 map，studentScore。每个 key 是一个 string，代表学生的名字，其 value 是 double 型，为该学生的分数。

众多小函数所操作的对象是相互依赖的，共享同一个 map 对象，有严格的先后关系，因此不能任意注释掉中间的函数。

插入元素的测试: mapInsertTest(studentScore)

查找元素的测试: mapFindTest(studentScore)

插入重复关键词的测试: `mapInsertRepeatedKeyTest(studentScore)`

更改关键词的测试: `mapKeyModifyTest(studentScore)`

删除元素的测试: `mapDeleteTest(studentScore)`

5. `void mapBoundTest():`

此函数独立出来，在测试 `lower_bound`、`upper_bound` 与 `equal_range` 在 `map` 中的用法与效果。举了另一个例子，`studentID`。key 为 `int` 型，指学生的 ID。Value 为 `string` 型，指学生的名字。

此函数没有分段(没有由其他小函数构成)。

6. `void multimapTest():`

配合学习文件中的 `multimap` 内容，此测试可以分成多个部分，由多个函数分别完成。在此举的例子还是 `studentScore`，不过改成是 `multimap`。

众多小函数所操作的对象是相互依赖的，共享同一个 `multimap` 对象，有严格的先后关系，因此不能任意注释掉中间的函数。

插入重复关键词的测试: `multimapInsertRepeatedKeyTest(studentScore)`

查找元素的测试: `multimapFindTest(studentScore)`

删除元素的测试: `multimapDeleteTest(studentScore)`

四、运行结果与分析

1. `mapTest():`

运行后窗口结果	注释、分析、结论
<pre>C:\Users\AlcanderLiu\Documents\Visual Studio 2012\Projects\CP. M A map has been initialized: An empty map. Insert 3 pairs by 3 different ways. The map has become: [Andy, 97.8] [Sean, 80] [Tony, 95.6] Find Tony using find(): [Tony, 95.6] Find Elbert using find(): Element not found. Find Tony using at(): [Tony, 95.6] Find Elbert using at(): Element not found. Find Tony using []: [Tony, 95.6] Find Elbert using []: [Elbert, 0]</pre>	<ol style="list-style-type: none">1. 刚声明的 <code>map</code> 不含任何元素2. 学习文件中的三种方法都能成功插入元素3. 不论插入顺序如何，会自动照 <code>key</code> 大小升序排序4. 若 <code>find()</code> 搜寻不到 <code>key</code> 将返回与 <code>end()</code> 相同的返回值5. 若 <code>at()</code> 的 <code>key</code> 不存在，则抛出异常6. 若 <code>[]</code> 的 <code>key</code> 不存在，则抛出异常，同时产生新元素，而 <code>value</code> 为 0

```
Insert Tony with value 80.0 by using insert():
The map has become:
[ Andy, 97.8 ]
[ Elbert, 0 ]
[ Sean, 80 ]
[ Tony, 95.6 ]
```

```
Insert Tony with value 80.0 by using emplace():
The map has become:
[ Andy, 97.8 ]
[ Elbert, 0 ]
[ Sean, 80 ]
[ Tony, 95.6 ]
```

```
Insert Tony with value 80.0 by using []:
The map has become:
[ Andy, 97.8 ]
[ Elbert, 0 ]
[ Sean, 80 ]
[ Tony, 80 ]
```

```
Rename Tony as Ana
The map has become:
[ Andy, 97.8 ]
[ Elbert, 0 ]
[ Sean, 80 ]
[ Tony, 80 ]
```

```
Modify Tony's value to 85.3 by using iterator:
The map has become:
[ Andy, 97.8 ]
[ Elbert, 0 ]
[ Sean, 80 ]
[ Tony, 85.3 ]
```

```
Modify Tony's value to 88.3 by using []:
The map has become:
[ Andy, 97.8 ]
[ Elbert, 0 ]
[ Sean, 80 ]
[ Tony, 88.3 ]
```

```
Erase Tony.
The map has become:
[ Andy, 97.8 ]
[ Elbert, 0 ]
[ Sean, 80 ]
```

```
Erase Tony again:
Failed erasing.
```

7. 若插入新的 key 但同 key 已存在, 则 insert()、emplace()不起作用
8. 若用运算符[]来插入新的 key 但同 key 已存在, 作用等同于修改原先的那个 value
9. 用 iterator 定位到某 pair 并尝试更改 pair 的 key, 此方法无效, 因为运算符=未定义。也就是不能更改 key。
10. 学习文件中的两种方法都能修改 value
11. 学习文件中的 erase 和 clear 效果如同预期
12. 注: 学习文件中的遍历元素方法, 已经直接体现在打印 map 的函数, printMap()中。

2. mapBoundTest():

运行后窗口结果	注释、分析、结论
The screenshot shows the output of a program in a Visual Studio 2012 console window. The text is as follows: A map has been initialized: [1, Sandy] [2, Rebecca] [3, Nancy] The lower_bound of key 2 [2, Rebecca] The upper_bound of key 4 Element not found. Try to find key 1: (using equal_range) [1, Sandy] Try to find key 4: (using equal_range) Element not found. Clear the map. The map has become: An empty map. Press any key to continue . . .	<ol style="list-style-type: none">1. lower_bound()返回所查 key 的下界，在 map 中就是该 key。(之后会知道在 multimap 就是第一个出现该 key 的 pair)2. upper_bound()返回所查 key 的上界，在 map 中就是该 key 的下一个 pair。(之后会知道在 multimap 就是最后一次出现该 key 的下一个 pair)3. 推论，若所查 key 不存在，则 lower_bound() = upper_bound() = find()，所以可以用 equal_range 一次性获取 lower_bound()和 upper_bound()，便可知道是否有查到。

3. multimapTest():

运行后窗口结果	注释、分析、结论
<pre>C:\Users\AlcanderLiu\Documents\Visual Studio 2012\Project A multimap has been initialized: An empty map. Insert some members: [Andy, 97.8] [Tony, 95.6] Insert another Tony with score 91.1 Insert the other Andy who also scored 97.8 Insert the other Tony with score 99.2 The map has become: [Andy, 97.8] [Andy, 97.8] [Tony, 95.6] [Tony, 91.1] [Tony, 99.2] Find Tony by using find(): [Tony, 95.6] Find Tony by using find() again: [Tony, 95.6] Find and print all Tony: [Tony, 95.6] [Tony, 91.1] [Tony, 99.2] Find the second pair of Tony: [Tony, 91.1] Erase all Tony: The map has become: [Andy, 97.8] [Andy, 97.8] clear(): The map has become: An empty map. Press any key to continue . . .</pre>	<ol style="list-style-type: none">1. 即使插入的 key 已存在，仍然可以插入。 Multimap 不存在运算符[]。2. 会自动依照 key 大小排序，若 key 相同， 则按照插入顺序先后排序。3. 两次 find()都得到第一个 Tony。推论 find() 皆回传排序在最前面的那个 Key。4. 可用 equal_range()得到相同 key 的区间。5. 找到第一个 key 后，可用迭代器累加得到 后续的单一 pair。6. erase()会一次删除所有相同的 key 目标。