**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。

1. 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。

1. CP\_MapTest.h/cpp 中的void mapTest()目的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。
2. 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)

1. 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)。

1. 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 |
|  | 1. 刚声明的map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key大小升序排序 4. 若find()搜寻不到key将返回与end()相同的返回值 5. 若at()的key不存在，则抛出异常 6. 若[]的key不存在，则抛出异常，同时产生新元素，而value为0 7. 若插入新的key但同key已存在，则insiert()、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。 |

1. mapBoundTest():

|  |  |
| --- | --- |
| 运行后窗口结果 | 注释、分析、结论 |
|  | 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。 |

1. multimapTest():

|  |  |
| --- | --- |
| 运行后窗口结果 | 注释、分析、结论 |
|  | 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。 |