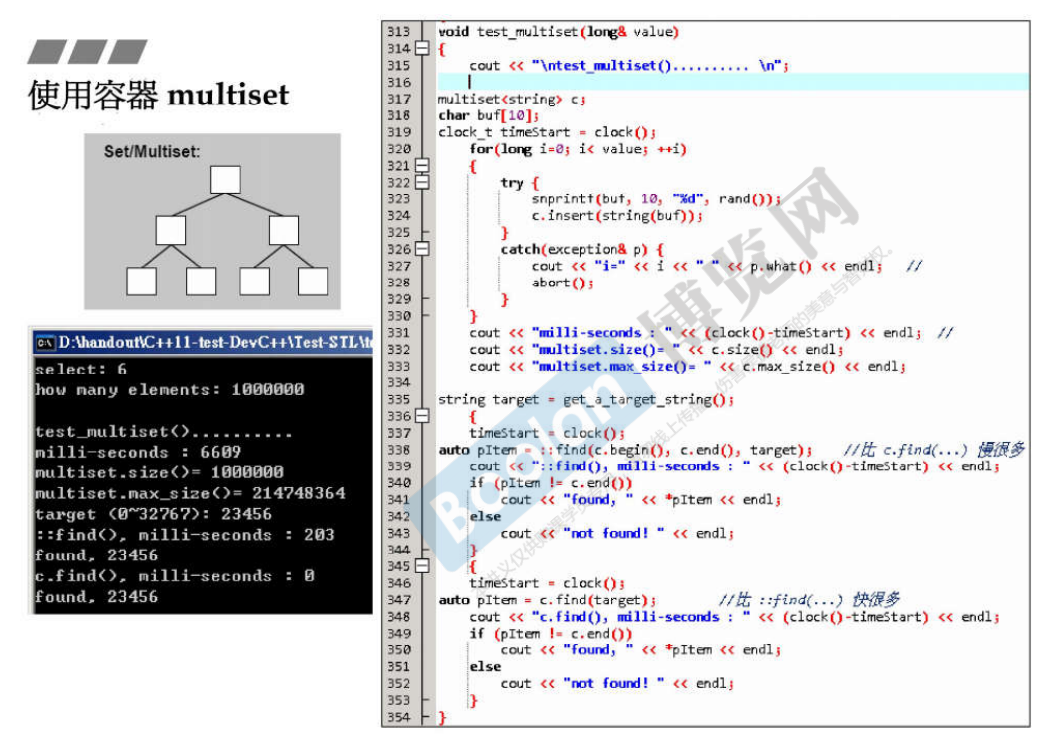
1、



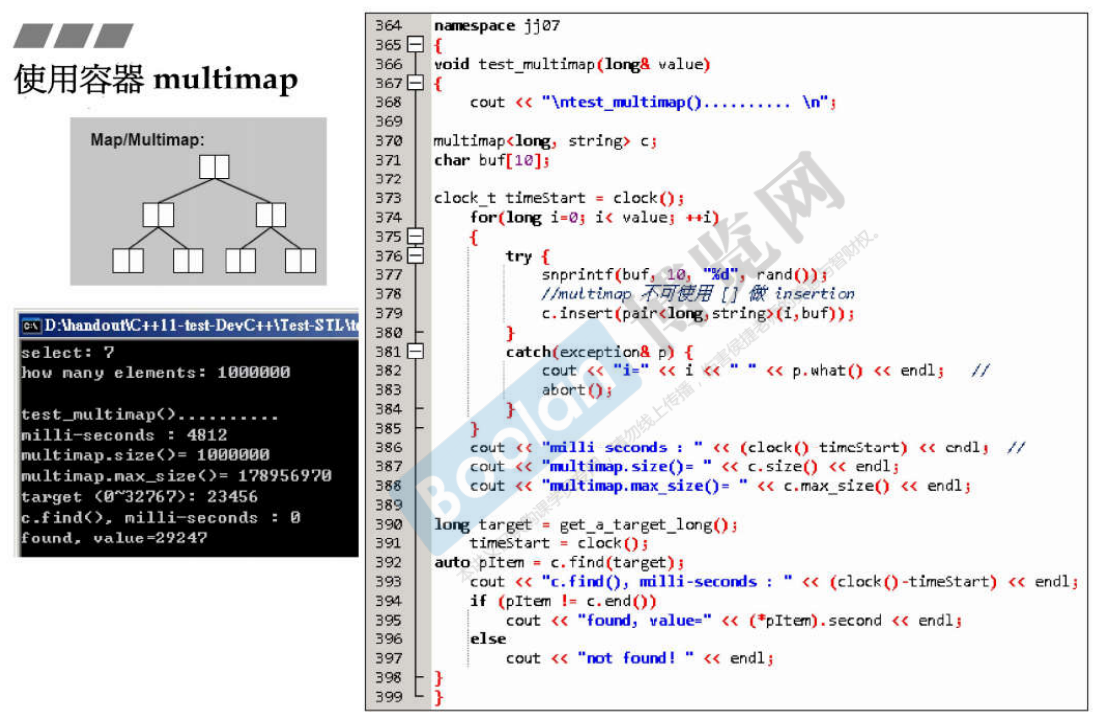
1、这里的set是没有push\_back或push\_front之类的，因为是没有顺序，因此只能插入，insert(啥)，插入到该插入的位置，是在这个红黑树中正确的位置。具体位置不用管啦。

2、multiset是有begin和end的，几乎都有，调用自身容器的find肯定是比标准库的快

3、查找是真的非常快的。

4、multiset\multimap 都用红黑树实现的

5、unordered\_multiset\unordered\_multimap都用哈希表实现的



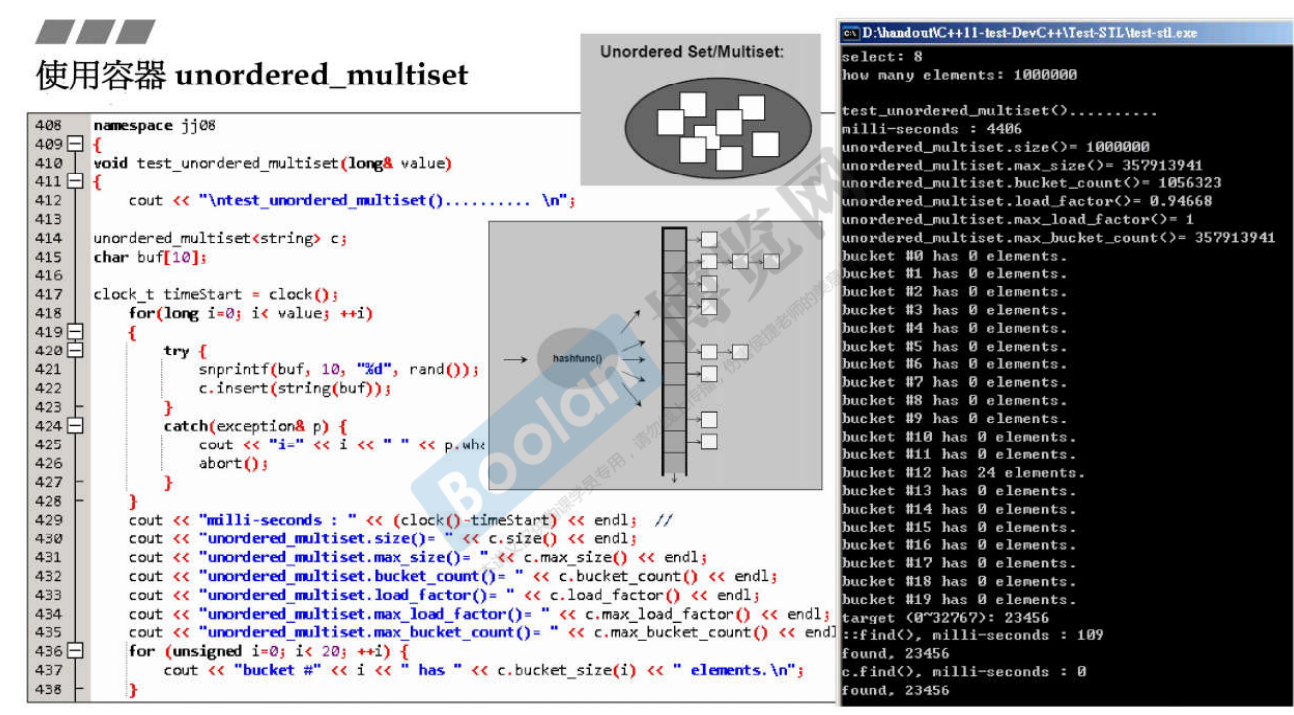
1、key可以重复

2、在379行，c.insert()，这时候要把key和value组合起来。

3、这里的key是0~999999，value是0~32767任意

4、组合起来的类型是pair<long, string>(i, buff)

5、find找出来的是一个iterator，(\*pItem)是一个pair



1、是用hash table 实现的

2、除了顺序容器外，关联容器都用insert

3、真的有bucket\_count()这个函数

4、篮子数量比元素数量多，因为有些篮子被分配了也没有用到，而有些篮子是不止一个元素的。而篮子的数量一定是比元素的个数多。

5、因此unordered\_multiset.bucket\_count() > unordered\_multiset.size() is available的

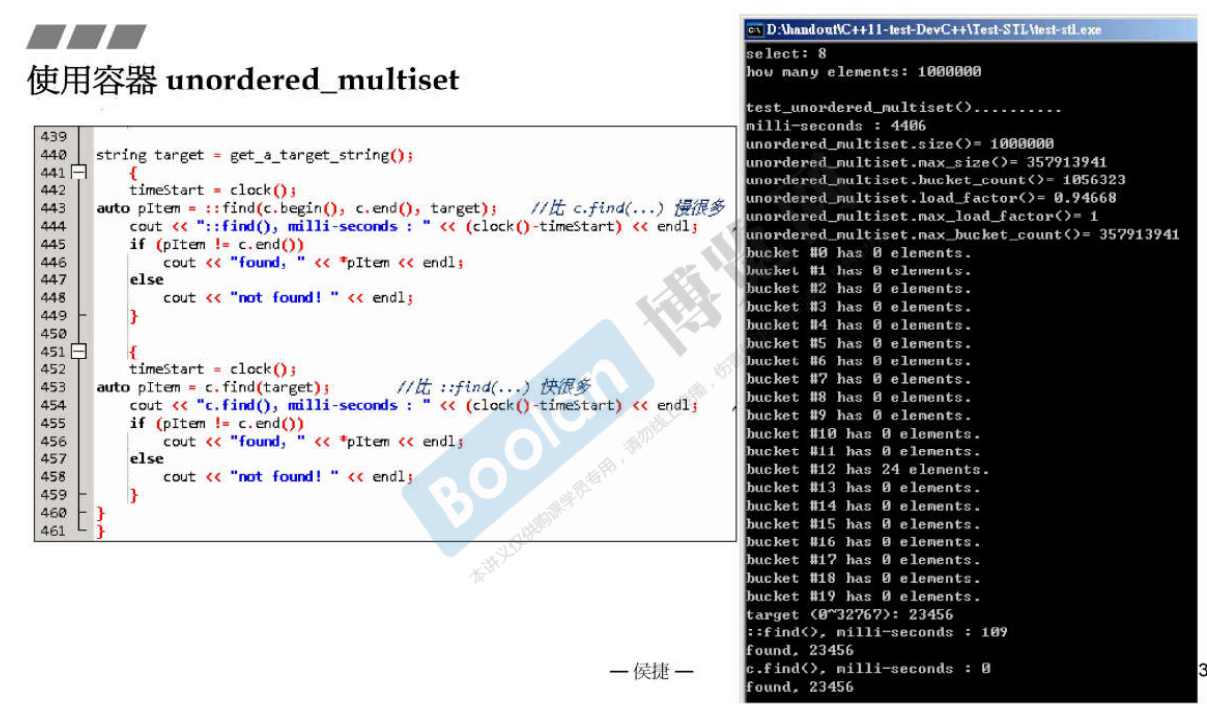
6、unordered\_multiset.load\_factor()是，没讲（盲猜是利用因子）

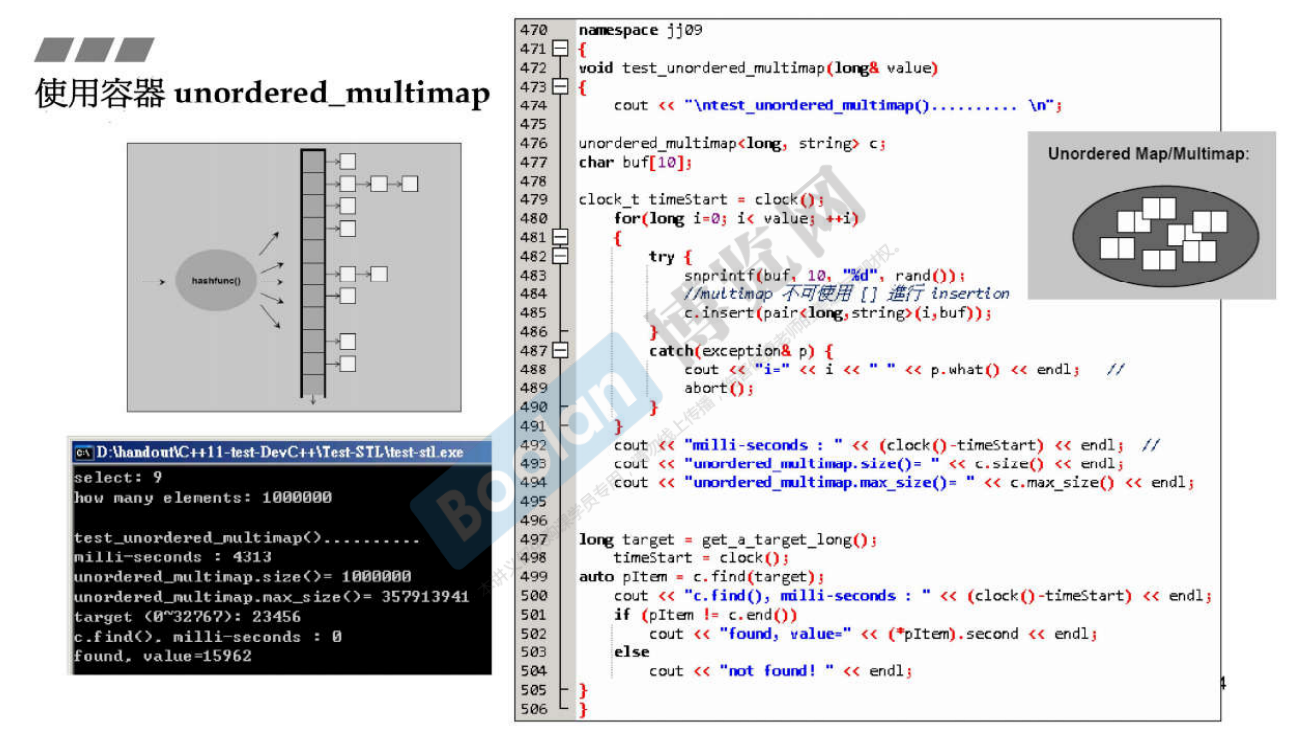
7、unordered\_multiset.max\_load\_factor()是最大的载重因子

8、unordered\_multiset.max\_bucket\_count()是

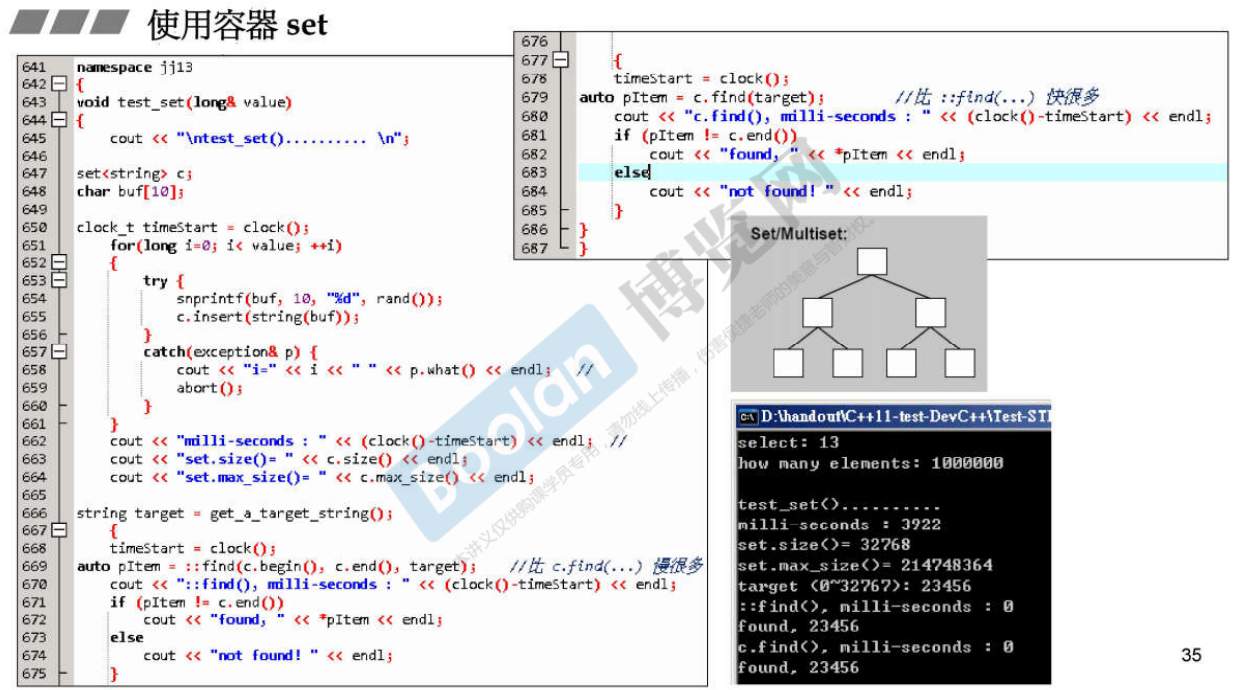
9、unordered\_multiset.bucket\_size(i)是第i个篮子里面的元素个数

10、每一个篮子的元素个数不能太多，就是“链子”不能太长，因为是要循序查找（可能还有其他原因）。根据这个规则，如果篮子的个数少于元素的个数，那么篮子的数量要按2倍数来扩充。（经验）扩展之后原来的元素要重新打散再放回到更大的篮子去。



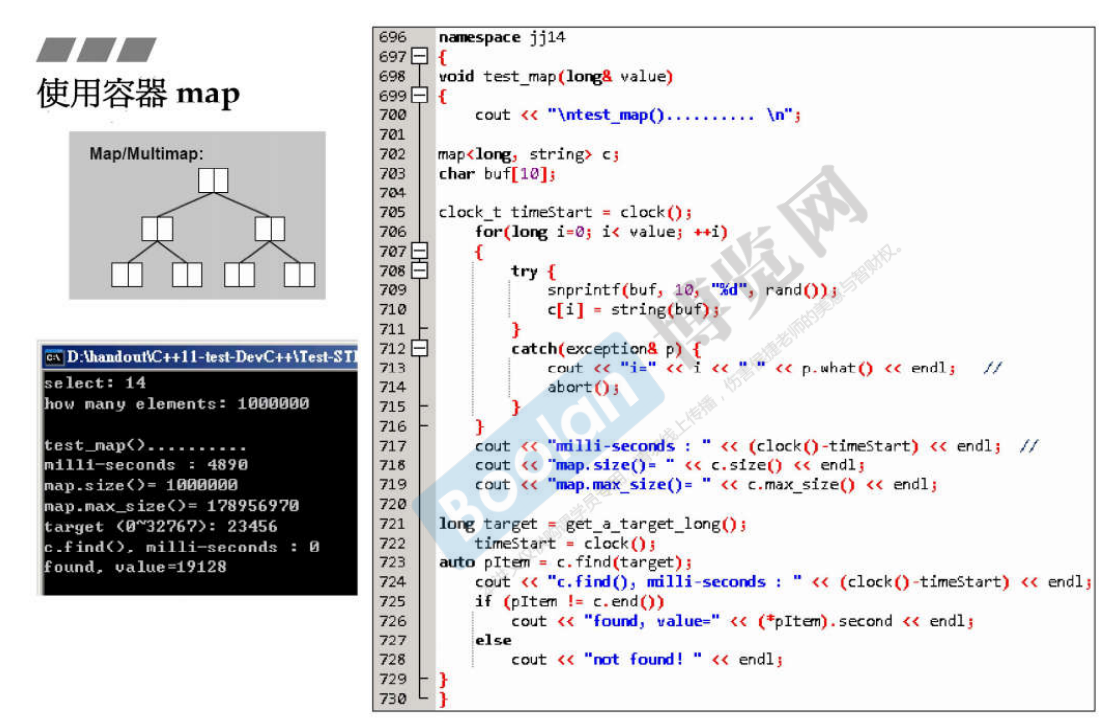


1、这是最适合使用搜寻大量元素时的容器（关联式容器均可）



1、有multi 是允许key重复的，而没有 multi是不允许重复的key的；

2、这里的32768个数字（0~32767）都存进去了



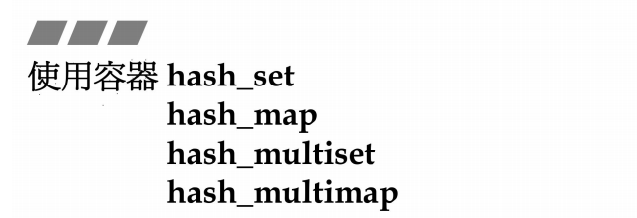
1、map是不允许重复key

2、这里的map.size()是100w的原因是key不同，而value有可以相同

3、multimap不能用 “[ ]”来进行insertion，而map可以，第710行

4、用 “[ ]”时，map内部会合成一个pair，这里看出来(\*pItem).second

5、\*pItem返回的是一个pair<int, string>变量



这些是旧版的名字，就是unordered\_......

现在还可以用，不过要包含它们的头文件时位置有点麻烦：

像在用slist时，要 #include <ext\slist>

这些都是非标准的