VincentCZW

STL中的set容器的一点总结

1.关于set

C++ STL 之所以得到广泛的赞誉，也被很多人使用，不只是提供了像vector, string, list等方便的容器，更重要的是STL封装了许多复杂的数据结构算法和大量常用数据结构操作。vector封装数组，list封装了链表，map和set封装了二叉树等，在封装这些数据结构的时候，STL按照程序员的使用习惯，以成员函数方式提供的常用操作，如：插入、排序、删除、查找等。让用户在STL使用过程中，并不会感到陌生。

关于set，必须说明的是set关联式容器。set作为一个容器也是用来存储同一数据类型的数据类型，并且能从一个数据集合中取出数据，在set中每个元素的值都唯一，而且系统能根据元素的值自动进行排序。应该注意的是set中数元素的值不能直接被改变。C++ STL中标准关联容器set, multiset, map, multimap内部采用的就是一种非常高效的平衡检索二叉树：红黑树，也成为RB树(Red-Black Tree)。RB树的统计性能要好于一般平衡二叉树，所以被STL选择作为了关联容器的内部结构。

关于**set**有下面几个问题：

（**1**）为何**map**和**set**的插入删除效率比用其他序列容器高？

大部分人说，很简单，因为对于关联容器来说，不需要做内存拷贝和内存移动。说对了，确实如此。set容器内所有元素都是以节点的方式来存储，其节点结构和链表差不多，指向父节点和子节点。结构图可能如下：

```
    A
   / \
  B C
 / \ / \
D E F G
```

因此插入的时候只需要稍做变换，把节点的指针指向新的节点就可以了。删除的时候类似，稍做变换后把指向删除节点的指针指向其他节点也OK了。这里的一切操作就是指针换来换去，和内存移动没有关系。

（**2**）为何每次**insert**之后，以前保存的**iterator**不会失效？

iterator这里就相当于指向节点的指针，内存没有变，指向内存的指针怎么会失效呢(当然被删除的那个元素本身已经失效了)。相对于vector来说，每一次删除和插入，指针都有可能失效，调用push_back在尾部插入也是如此。因为为了保证内部数据的连续存放，iterator指向的那块内存在删除和插入过程中可能已经被其他内存覆盖或者内存已经被释放了。即使时push_back的时候，容器内部空间可能不够，需要一块新的更大的内存，只有把以前的内存释放，申请新的更大的内存，复制已有的数据元素到新的内存，最后把需要插入的元素放到最后，那么以前的内存指针自然就不可用了。特别时在和find等算法在一起使用的时候，牢记这个原则：不要使用过期的iterator。

（**3**）当数据元素增多时，**set**的插入和搜索速度变化如何？

如果你知道log2的关系你应该就彻底了解这个答案。在set中查找是使用二分查找，也就是说，如果有16个元素，最多需要比较4次就能找到结果，有32个元素，最多比较5次。那么有10000个呢？最多比较的次数为log10000，最多为14次，如果是20000个元素呢？最多不过15次。看见了吧，当数据量增大一倍的时候，搜索次数只不过多了1次，多了1/14的搜索时间而已。你明白这个道理后，就可以安心往里面放入元素了。

2.set中常用的方法

begin()	,返回**set**容器的第一个元素
end()	,返回**set**容器的最后一个元素
clear()	,删除**set**容器中的所有的元素
empty()	,判断**set**容器是否为空
max_size()	,返回**set**容器可能包含的元素最大个数
size()	,返回当前**set**容器中的元素个数
rbegin	,返回的值和**end()**相同

程序员职场生活
发展形势(1)
移动产品相关(3)

文章档案(16)
2013年10月 (2)
2013年9月 (2)
2013年7月 (1)
2013年4月 (1)
2013年3月 (8)
2012年8月 (2)

相册(3)
2009那一年的我(3)

积分与排名
积分 - 165136
排名 - 1309

最新评论
1. Re:设计模式之建造者模式（Builder）
我觉得《大话设计模式》中的相关代码设计不严格，建造者模式中具体的建造者和产品严格上应该是一种依赖关系，但这样的代码设计已经是比较强的耦合关系了。请博主不要生搬硬套。
--alterto
2. Re:设计模式之单例模式（Singleton）
线程非安全：双重检查模式（DCL）==>可能导致部分初始化的对象（this指针溢出）
--ssslinppp
3. Re:一个即将毕业的13届毕业生校招有感
@Mr_happle这些说法不是通用的，也许别人比较聪明，在大三的时候偶尔学下习获得的东西却是你大一大二两年都学不来的呢？或者这样说吧！一种情况，你大一大二玩然后大三学下习可以找到工作，另一种情况，你.....
--huang阿贵
4. Re:设计模式之单例模式（Singleton）
java中的Synchronized 是不是相当于C#中的Lock? 感觉跟我理解的C#中的LOCK大致一个意思.初步学习,请多指教.
--酸菜sauerkraut

阅读排行榜
1. 关于C++中的友元函数的总结(101280)
2. STL中的set容器的一点总结(72766)
3. C++中四种类型转换方式(36365)
4. 设计模式之建造者模式（Builder）(34528)
5. C++中的static关键字的总结(33935)

评论排行榜
1. 大家赶快来说说C和C++到底有什么"奸情"吧(35)
2. 电梯调度算法(31)
3. 小程序员的趣味题（一）(30)
4. 数组中的趣味题（一）(27)
5. 数组循环移位中的学问(24)

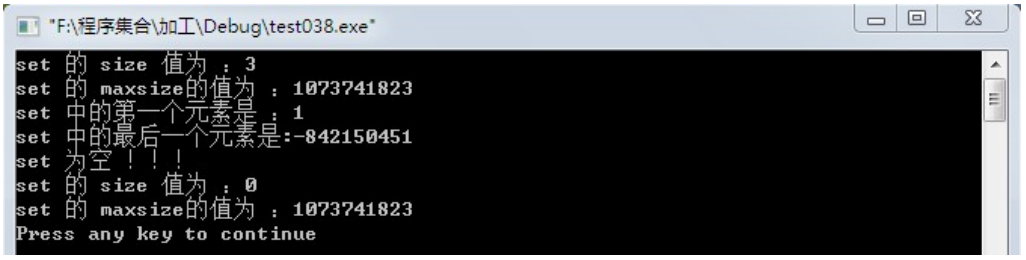
推荐排行榜
1. STL中的set容器的一点总结(15)

rend() ,返回的值和**rbegin()**相同

写一个程序练一练这几个简单操作吧：

View Code

运行结果：



小结：插入3之后虽然插入了一个1，但是我们发现set中最后一个值仍然是3哈，这就是set 。还要注意begin() 和 end()函数是不检查set是否为空的，使用前最好使用empty()检验一下set是否为空.

---

count() 用来查找set中某个某个键值出现的次数。这个函数在set并不是很实用，因为一个键值在set只可能出现0或1次，这样就变成了判断某一键值是否在set出现过了。

示例代码：

View Code

运行结果：

---

equal_range() ，返回一对定位器，分别表示第一个大于或等于给定关键值的元素和 第一个大于给定关键值的元素，这个返回值是一个pair类型，如果这一对定位器中哪个返回失败，就会等于end()的值。具体这个有什么用途我还没遇到过~~~

示例代码：

View Code

运行结果：

---

erase(iterator) ,删除定位器**iterator**指向的值

erase(first,second),删除定位器**first**和**second**之间的值

erase(key_value),删除键值**key_value**的值

看看程序吧：

View Code

运行结果：

小结：set中的删除操作是不进行任何的错误检查的，比如定位器的是否合法等等，所以用的时候自己一定要注意。

find() ，返回给定值值得定位器，如果没找到则返回end()。

示例代码：

View Code

insert(key_value); 将key_value插入到set中 ，返回值是pair<set<int>::iterator,bool>，bool标志着插入是否成功，而iterator代表插入的位置，若key_value已经在set中，则iterator表示的key_value在set中的位置。

inset(first,second);将定位器first到second之间的元素插入到set中，返回值是void.

示例代码：

View Code

运行结果：

lower_bound(key_value) ，返回第一个大于等于key_value的定位器

upper_bound(key_value)，返回最后一个大于等于key_value的定位器

示例代码：

View Code

运行结果：

学习中的一点总结，欢迎拍砖哦^^

分类: STL

标签: set容器

好文要顶 关注我 收藏该文

VincentCZW
关注 - 8
粉丝 - 301
+加关注

15 0

« 上一篇：STL中的list容器的一点总结
» 下一篇：C++中的模板那点事
posted on 2012-08-13 16:45 VincentCZW 阅读(72768) 评论(9) 编辑 收藏

Comments
#1楼
霍国峰
Posted @ 2012-08-13 17:46
我觉得还是map比较好。5151团

支持(0) 反对(1)

#2楼
lzprgmr
Posted @ 2012-08-13 21:40
@ 霍国峰
set与map完全是用在不同的场景下的，说map比set好，就好像说螺丝刀比锤子好，到底哪个好？得看你是要敲钉子还是拔钉子

支持(3) 反对(0)

#3楼
lzprgmr
Posted @ 2012-08-13 21:48
>>为何map和set的插入删除效率比用其他序列容器高？
应该改成"为何map和set的插入删除效率比用vector容器高？" 因为与list相比，map与set明显效率要低（O(1) vs O(lgn）)

>>count() 用来查找set中某个某个键值出现的次数
不理解stl为啥提供这么个积累的函数，要知道某个元素是否出现，find即可。可能是为了保持和multiset的一致性，但感觉没这个必要，又不是多态类

支持(0) 反对(0)

#4楼[楼主]
VincentCZW
Posted @ 2012-08-14 11:54
@ lzprgmr
在list中删除操作为什么是O(1)呢？不要先找到节点，再删除吗？？

支持(0) 反对(0)

#5楼[楼主]
VincentCZW
Posted @ 2012-08-14 11:55
@ 霍国峰

不可以说set就一定比map差呀，set也有自己的优势，希望你不是单纯的为了5151团发了个评论~~~

支持(0) 反对(0)

#6楼
lzprgmr
Posted @ 2012-08-15 06:39
@ BeyondAnyTime
这要看你是怎么定义这个erase函数的接口的：stl中的标准接口是直接传element的iterator(http://www.cplusplus.com/reference/stl/list/erase/)，所以是常数，当然，如果你传一个value，那么是需要先查找后删除了。

对于set的删除（http://www.cplusplus.com/reference/stl/list/erase/），我上面讲错了，删除一个元素其平摊时间还是O(1)的

支持(0) 反对(0)

#7楼
卡哇伊啊
Posted @ 2015-08-27 15:09
就是说set的查找效率并不是O(1）了？

支持(0) 反对(0)

#8楼
Interesting9301
Posted @ 2015-10-21 12:29
有的地方说的是错误的，begin和end返回的不是元素而是迭代器，取末尾元素应该是*--s.end()，*s.end()是取不到的而且会报错。而且rbegin和rend的定义也不是那样，详情参考这里：
blog.csdn.net/kjing/article/details/6936325

支持(0) 反对(0)

#9楼
JRSmith
Posted @ 2016-04-08 21:33
纠正一下定义：
lower_bound(key):返回第一个大于等于key的迭代器指针
upper_bound(key):返回第一个大于key的迭代器指针

支持(1) 反对(0)

刷新评论 刷新页面 返回顶部

（评论功能已被禁用）