

VincentCZW

导航

[博客园](#)
[首页](#)
[新随笔](#)
[联系](#)
[管理](#)

< 2016年4月 >						
日	一	二	三	四	五	六
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

统计

随笔 - 75
文章 - 16
评论 - 368
引用 - 0

公告

昵称: VincentCZW
园龄: 3年11个月
粉丝: 232
关注: 8
[+加关注](#)

搜索

常用链接

[我的随笔](#)
[我的评论](#)
[我的参与](#)
[最新评论](#)
[我的标签](#)

随笔分类 (76)

[Android\(1\)](#)
[C#](#)
[C++\(13\)](#)
[IT小百科\(13\)](#)
[Love喜欢\(2\)](#)
[STL\(6\)](#)
[VC & MFC\(1\)](#)
[设计模式\(15\)](#)
[数据结构与算法\(24\)](#)
[自娱自乐\(1\)](#)

随笔档案 (75)

[2013年7月 \(1\)](#)
[2013年3月 \(1\)](#)
[2012年12月 \(1\)](#)
[2012年9月 \(1\)](#)
[2012年8月 \(15\)](#)
[2012年7月 \(19\)](#)
[2012年6月 \(9\)](#)

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

2012年5月 (28)

文章分类 (11)

Android(7)

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 - 151173

排名 - 1147

最新评论

1. Re:STL中的set容器的一点总结

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

--JRSmith

2. Re:设计模式之建造者模式 (Builder)

就是策略模式加一个for循环。

命令模式的有序版。

--Tekkaman

3. Re:欧几里得算法求最大公约数的局限性及解决方案

第三种算法判断条件太多效率并不高,反而是第一种算法效率最高,第一种算法求100000组8位整数的最大公约数耗时34ms
第二种算法求100组8位整数的最大公约数耗时3ms,而且有内存泄漏的问题,对于某些.....

--回忆专用小马甲

4. Re:二叉树中的那些常见的面试题

想问博主,你给出的【2.9 从根节点开始找到所有路径,使得路径上的节点值和为某一数值(路径不一定以叶子节点结束)】算法里的路径起点一定是根节点吗?是否可以除根节点以外的中间结点???

--郭可岩

阅读排行榜

1. 关于C++中的友元函数的总结(61373)

2. STL中的set容器的一点总结

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:

```

1 #include <iostream>
2 #include <set>
3
4 using namespace std;
5
6 int main()
7 {
8     set<int> s;
9     s.insert(1);
10    s.insert(2);
11    s.insert(3);
12    s.insert(1);
13    cout<<"set 的 size 值为 : "<<s.size()<<endl;
14    cout<<"set 的 maxsize的值为 : "<<s.max_size()<<endl;
15    cout<<"set 中的第一个元素是 : "<<*s.begin()<<endl;
16    cout<<"set 中的最后一个元素是:"<<*s.end()<<endl;
17    s.clear();
18    if(s.empty())
19    {
20        cout<<"set 为空 !!! "<<endl;
21    }
22    cout<<"set 的 size 值为 : "<<s.size()<<endl;
23    cout<<"set 的 maxsize的值为 : "<<s.max_size()<<endl;
24    return 0;
25 }

```

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:

```

1 #include <iostream>
2 #include <set>
3
4 using namespace std;
5
6 int main()
7 {
8     set<int> s;
9     s.insert(1);
10    s.insert(2);

```

(31017)

3. 关于C++中的虚拟继承的一些总结(23164)

4. 二叉树中的那些常见的面试题(20175)

5. C++中四种类型转换方式(18775)

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. 关于C++中的虚拟继承的一些总结(13)

2. 一个即将毕业的13届毕业生校招有感(9)

3. C++中的操作符重载(7)

4. C++中的static关键字的总结(7)

5. 二叉树中的那些常见的面试题(7)

```

11     s.insert(3);
12     s.insert(1);
13     cout<<"set 中 1 出现的次数是 : "<<s.count(1)<<endl;
14     cout<<"set 中 4 出现的次数是 : "<<s.count(4)<<endl;
15     return 0;
16 }

```

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~~~

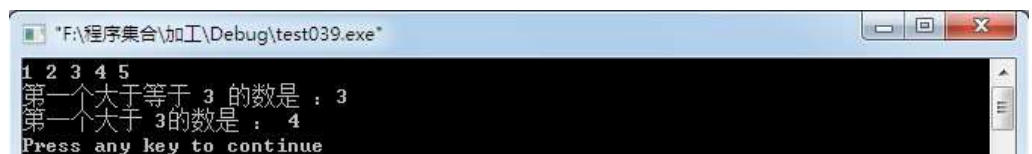
示例代码:

```

1 #include <iostream>
2 #include <set>
3
4 using namespace std;
5
6 int main()
7 {
8     set<int> s;
9     set<int>::iterator iter;
10    for(int i = 1 ; i <= 5; ++i)
11    {
12        s.insert(i);
13    }
14    for(iter = s.begin() ; iter != s.end() ; ++iter)
15    {
16        cout<<*iter<<" ";
17    }
18    cout<<endl;
19    pair<set<int>::const_iterator, set<int>::const_iterator> pr;
20    pr = s.equal_range(3);
21    cout<<"第一个大于等于 3 的数是 : "<<*pr.first<<endl;
22    cout<<"第一个大于 3的数是 : " <<*pr.second<<endl;
23    return 0;
24 }

```

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：

```
1 #include <iostream>
2 #include <set>
3
4 using namespace std;
5
6 int main()
7 {
8     set<int> s;
9     set<int>::const_iterator iter;
10    set<int>::iterator first;
11    set<int>::iterator second;
12    for(int i = 1 ; i <= 10 ; ++i)
13    {
14        s.insert(i);
15    }
16    //第一种删除
17    s.erase(s.begin());
18    //第二种删除
19    first = s.begin();
20    second = s.begin();
21    second++;
22    second++;
23    s.erase(first, second);
24    //第三种删除
25    s.erase(8);
26    cout<<"删除后 set 中元素是 : ";
27    for(iter = s.begin() ; iter != s.end() ; ++iter)
28    {
29        cout<<*iter<<" ";
30    }
31    cout<<endl;
32    return 0;
33 }
```

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() ， 返回给定值得定位器，如果没找到则返回end()。

示例代码：

```
1 #include <iostream>
2 #include <set>
3
4 using namespace std;
5
6 int main()
7 {
8     int a[] = {1,2,3};
```

```

9     set<int> s(a,a+3);
10    set<int>::iterator iter;
11    if((iter = s.find(2)) != s.end())
12    {
13        cout<<*iter<<endl;
14    }
15    return 0;
16 }

```

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：

```

1 #include <iostream>
2 #include <set>
3
4 using namespace std;
5
6 int main()
7 {
8     int a[] = {1,2,3};
9     set<int> s;
10    set<int>::iterator iter;
11    s.insert(a,a+3);
12    for(iter = s.begin() ; iter != s.end() ; ++iter)
13    {
14        cout<<*iter<<" ";
15    }
16    cout<<endl;
17    pair<set<int>::iterator,bool> pr;
18    pr = s.insert(5);
19    if(pr.second)
20    {
21        cout<<*pr.first<<endl;
22    }
23    return 0;
24 }

```

运行结果：



```

1 2 3
5
Press any key to continue.

```

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：



```
1 #include <iostream>
2 #include <set>
3
4 using namespace std;
5
6 int main()
7 {
8     set<int> s;
9     s.insert(1);
10    s.insert(3);
11    s.insert(4);
12    cout<<*s.lower_bound(2)<<endl;
13    cout<<*s.lower_bound(3)<<endl;
14    cout<<*s.upper_bound(3)<<endl;
15    return 0;
16 }
```



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 - 232

[+加关注](#)

7

0

(请您对文章做出
评价)

« 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(31017) 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？得看你要敲钉子还是拔钉子

支持(1) 反对(0)