

w729926317的专栏

☰ 目录视图

☰ 摘要视图

RSS 订阅

个人资料



格格巫不想了

访问：10841次

积分：222分

排名：千里之外

原创：9篇

转载：15篇

译文：0篇

评论：9条

文章搜索

文章分类

C/C++ (3)

UNP (1)

网络 (2)

Android (5)

QT (3)

MySQL (3)

其他 (7)

文章存档

2012年04月 (1)

2012年01月 (5)

2011年12月 (5)

2011年11月 (8)

2011年10月 (5)

阅读排行

QT中不同界面传递值 (1542)

STL中rotate算法之_rotate (805)

PING通信详细过程 (749)

投票赢好礼，周周有惊喜！

2014年4月微软MVP申请开始了！

消灭0回答，赢下载分

“我的2013”年度征文活动火爆进行中！

专访Kinect手语翻译系统团队

STL中rotate算法之_rotate_cycle函数详解

2012-01-24 15:34

805人阅读

评论(1)

收藏

举报

算法

distance

class

iterator

random

access

这里的rotate操作，也就是指循环移位。比如将串“ABCDEFGG”以D为中心旋转，就相当将该串向左循环移位，直到第一个元素为D为止，最后 得到新串“DEFGABC”。

要想方便的完成rotate操作，一个常见的技巧是这样的：先将前半部分反转，再将后半部分反转，最后再将整个串反转即可（这里的前半部分与后半部分是以旋转中心来划分的）。还是以串“ABCDEFGG”以D为中心旋转为例，以D为分割点，将先半部分与后半部分分别反转后，得“CBAGFED”，最后将整个串反转即得“DEFGABC”。

这个算法在很多书上都提到过，相信大家一定都很熟悉了。那么其效率如何呢，假设串的长度为t，那么完成整个旋转过程需要约t次swap操作，也即是说需要大概3t次赋值，同时只需要常量的临时空间。因为其实现简单，所以还是差强人意。所以 SGI STL在处理双向迭代器容器时也正是使用了该算法。

但是进一步观察STL源码可以发现，在处理存储在拥有随机访问能力的容器中的串时，SGI STL却是采用了另外一种算法，而这个算法的原理在《STL源码剖析》中恰恰被hjj无视了，所以我在这里再简单地梳理一下。
首先，先帖出这个算法的源代码：

```
[cpp]

01. template < class RandomAccessIterator, class Distance >
02. void __rotate(RandomAccessIterator first,
03. RandomAccessIterator middle,
04. RandomAccessIterator last, Distance *,
05. random_access_iterator_tag)
06. {
07. Distance n = __gcd(last - first, middle - first);
08. while (n -- )
09.     __rotate_cycle(first, last, frist + n, middle - first, value_type(first));
10. }
11.
12. template < class EuclideanRingElement >
13. EuclideanRingElement __gcd(EuclideanRingElement m, EuclideanRingElement n)
14. {
15.     while (n != 0 )
16.     {
17.         EuclideanRingElement t = m % n;
18.         m = n;
19.         n = t;
20.     }
21.     return m;
22. }
23.
24. template < class _RandomAccessIterator, class Distance, class T >
25. void __rotate_cycle(RandomAccessIterator first, RandomAccessIterator last,
```

Android中Service绑定的	(732)
Android程序访问 本地tor	(730)
对listview的一些操作	(672)
Ubuntu的备份	(588)
左值与右值	(552)
Ubuntu 安装MySQL后登	(361)
字节排序函数，地址转换	(340)

评论排行	
QT中不同界面传递值	(4)
STL中rotate算法之__rotai	(1)
PING通信详细过程	(1)
Android程序访问 本地tor	(1)
Ubuntu下安装MySQL获得	(1)
C/C++相关网站	(1)
Ubuntu 安装MySQL后登	(0)
关于QString转char *	(0)
Android中Service绑定的	(0)
对listview的一些操作	(0)

推荐文章	
* 2D游戏效果之六：平安夜特别版——星光四射(粒子系统高级应用)	
* Android UI 优化——使用HierarchyViewer工具	
* 史上最全设计模式导学目录（完整版）	
* 刷一个基于html5开发的网页圣诞游戏	
* 系统架构师-基础到企业应用架构-分层[上篇]	
* android中图片的三级cache策略（内存、文件、网络）之二：内存缓存策略	

最新评论	
PING通信详细过程 chenquangobeijing: 这写的层对吗？网际层、网络接口层？ISO 7层有这个吗？	
QT中不同界面传递值 才才: 父窗体怎么给予窗体传值啊？	
Android程序访问 本地tomcat服务 hao1202: 这一集视频是哪个啊 正好在学这一块	
STL中rotate算法之__rotate_cycle yuanweihuayan: 楼主看懂了吗？数论定理？不是很清楚唉。	
Ubuntu下安装MySQL获得 mysql.h wzy9854: well done	
C/C++相关网站 buyicn: http://www.codeguru.com/ 这个也不错	
QT中不同界面传递值 hn3e21: 准确的说 dialog.exec()==QDialog::Accepted是dialog接收到一个关闭...	
QT中不同界面传递值 small_windmill: 那 dialog.exec()==QDialog::Accepted之后的窗口怎么重新调出来？	
QT中不同界面传递值 Morris: 原来如此~	

26.	RandomAccessIterator initial, Distance shift, T *)
27.	{
28.	T value = * initial;
29.	RandomAccessIterator ptr1 = initial;
30.	RandomAccessIterator ptr2 = ptr1 + shift;
31.	while (ptr2 != initial)
32.	{
33.	* ptr1 = * ptr2;
34.	ptr1 = ptr2;
35.	if (last - ptr2 > shift)
36.	ptr2 += shift;
37.	else
38.	ptr2 = first + (shift - (last - ptr2));
39.	}
40.	* ptr1 = value;
41.	}

上面只涉及到三个函数：__rotate、__gcd、__rotate_cycle。后两个函数都比较容易理解：__gcd没什么好说的，XXX 嘛，当然是求最大公约数了。

__rotate_cycle，是从某个初始元素开始，依次将其替换成其后相隔固定距离的元素。如果后面没有足够的偏移距离了，则又返回头部继续计算（相当于求模）。直到最后形成一个置换圈为止。

现在来仔细观察函数__rotate，这个函数实际上也不复杂，才区区三行：先是求出偏移距离和串总长的最大公约数，然后循环n次，分别以串的前n个元素为起点进行__rotate_cycle操作，over。但这怎么就保证了能正确地完成对输入串的rotate操作呢？

这就涉及到数论中的一个小定理：若有两个正整数m、n，且gcd(m,n)=d，那么序列{m%n, 2m%n, 3m%n,..., nm%n}一定是{0, d, 2d,..., n-d}的某个排列并重复出现d次，其中%号代表求模操作。

比如若m=6, n=8, d=gcd(m,n)=2，那么{6%8, 12%8, 18%8,..., 48%8}即为{0, 2, 4, 6}的某个排列并重复两次，事实上也正是{6, 4, 2, 0, 6, 4, 2, 0}。特别地，若m、n互素，d=1，那么序列{m%n,2m%n,3m%n,...,(n-1)m%n}实际上就是{1, 2, 3,..., n-1}的某个排列。

这个定理的证明过程可以很多书中找到（比如具体数学4.8节），这里不再详述。了解这个引理后，就很容易看出__rotate函数的内涵了。若第一步求得的最大公约数n为1，那么只需一次__rotate_cycle就可以遍历到所有的元素，并将每个元素正确的替换为其后相距某个距离的元素，于是也就完成了循环左移操作。若n大于1，那么每一次__rotate_cycle只能将t/n的元素正确的左移，其中t为串的总长度，而这些被移动的元素是以d为等间距的，所以循环n次，并分别以串的前n个元素为起点进行__rotate_cycle操作，就能保证将所有的元素都移动到正确的位置上。

在这个新的算法中，每次__rotate_cycle需要t/n+1次赋值，n次循环，所以总共只需要t+n次赋值操作，显然是要比前面所说的三次反转的算法快上许多。

比如考虑当串a = { 1, 2, 3, 4, 5} 循环左移2位，即期望得到串{ 3, 4, 5, 1, 2}，那么该算法的赋值过程如下：
tmp = a[0] -> tmp = 1
a[0] = a[2] ->{ 3, 2, 3, 4, 5}
a[2] = a[4] ->{ 3, 2, 5, 4, 5}
a[4] = a[1] ->{ 3, 2, 5, 4, 2}
a[1] = a[3] ->{ 3, 4, 5, 4, 2}

a[3] = tmp ->{ 3, 4, 5, 1, 2}
这里因为2与5互素，所以6次赋值就已搞定，而如果是用三次翻转的算法则需要大约3*5次赋值（实际上为12次）。

<http://www.cnblogs.com/atyuwen/archive/2009/11/08/rotate.html>

更多 0

上一篇：[ubuntu下安装nvidia显卡驱动](#)
下一篇：[ubuntu文件查看器PDF乱码的解决办法](#)

美国移民		美国移民
美国移民		美国移民
技术移民		技术移民

查看评论

1楼 [yuanweihuayan](#) 2012-05-24 21:40发表



楼主看懂了吗?数论定理?不是很清楚唉.

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

专区推荐内容

- 优化、并行化、及向量化
- 如何从数据库调出数据显示到页面
- Android 游戏教程: 让人物...
- Linux 下的 UltraEd...
- 关于HTML怎样用图片做背景
- Android 游戏教程: 让人物...

<< >>

更多招聘职位

核心技术类目

- 全部主题
- JavaVPNAndroidiOSERP
- IE10EclipseCRMJavaScriptUbuntu
- NFCWAPjQuery数据库BIHTML5SpringApacheHadoop.NETAPI
- HTMLSDKIISFedoraXMLLBSUnitySplashtopUMLcomponents
- Windows MobileRailsQEMUKDECassandraCloudStackFTCcoremailOPhone
- CouchBase云计算iOS6RackspaceWeb AppSpringSideMaemoCompuware
- 大数据aptechPerlTornadoRubyHibernateThinkPHPSparkHBasePure
- SolrAngularCloud FoundryRedisScalaDjangoBootstrap