MFC 技术和 STL 技术链表结构应用时间效率的比较

Compare Of List's Time Efficiency in MFC and STL

陈金广　秦兰双　马丽丽　(西安工程科技学院 710048)

摘要:本文针对 MFC 技术和 STL 技术提供的链表类数据结构的运行时间效率这一问题,应用实例对其进行实际验证。对得到的实验数据进行分析,得出了两种技术提供的链表类数据结构的时间效率基本相同的结论。

关键词:MFC　STL　链表　时间效率

1　引言

在开发软件系统时,常常面临数据对象的设计,再对数据对象施加一定的算法,完成某些特殊的功能。在绘图系统中,就需要设计出合适的数据对象来保存、重现视图中用户留下的图元信息(例如:点图元的位置、颜色,直线图元的起点、终点、颜色等属性)。通常采用的数据结构是动态数组或者链表。在 VC 环境中,MFC 类库提供了 CList、CArray 等数据类,用来实现数组和链表的功能。还有一种选择是采用 C++语言的标准模板库来实现数组和链表结构。

在从两种技术中作选择时,常常困惑采用两种技术之一的运行效率是否存在差异? 很多相关技术资料上强烈推荐 C++程序员采用 STL 技术[1],那么是否是在运行效率上 STL 技术要比 MFC 技术的效率高的多? 本文针对这样的问题,分别采用这两种技术做了相关实验,对它们的运行时间效率进行统计分析(如题目所示,本文只对两种技术分别实现的链表结构进行了对比,对其他结构和算法没有进行同样验证),希望能够对读者有所裨益。

2　STL 和 MFC 中的链表类

STL(Standard Template Library),标准模板库。是 1998 年 9 月出台的 C++标准规格中的 C++标准程序库的一个重要组成部分。概括来讲,STL 主要是一些算法和容器的集合。它提供了 vector、list、queue、stack、set、map 等容器的标准模板实现,提供了 sort、search、copy 等常用算法的模板实现。这些算法和容器的实现,采用的都是经过严格数学证明的、反复在应用中得到验证的算法实现。

STL 技术有以下优点:具有非常优秀的通用性,这是得益于它在代码实现过程中普遍采用模板机制。数据结果非常简单,易于掌握,有完善的安全机制和内存管理机制。许多系统数据对象设计中的问题,都可以采用 STL 很简捷的予以解决。很多传统的 C++编写的代码实际上用几句 STL 编码就可以实现,而且绝对高效。

STL 技术对链表的实现是 list 容器。

MFC(Microsoft Foundation Class),微软基础类库。对几乎 95% 以上的 Windows API 函数进行了封装,其功能非常强大。其中的 CList 等一系列类实现链表的功能,在下列实验中,采用了其中的 CTypedPtrList 类。

3　实验方案

在实际实验过程中,设计了一个数据对象,并把它封装成一个类,类的声明头文件如下:

```
class CMonitorObj : public CObject
{
public:
    CMonitorObj(void);//构造函数
    ~CMonitorObj(void);//析构函数
public:
    CString strMachine,strZu,strGang;//机台号、机台所属组、机台所属岗
    int port,innerID;//机台端口号、机台内部 ID 号
    CPoint point;//机台在视图中显示的位置矩形的左上角坐标
```


©《计算机系统应用》编辑部 http://www.c-s-a.org.cn

```
    CSize size;//机台在视图中显示的位置矩形的大小
    int status;//机台当前的运转状态
    void Draw(CDC * pDC);//在视图中画出机台位置举行、并在矩形内显示机台号
};
```

各个数据成员和成员函数的作用在上面注释中给出了。像这样的对象在系统中有上千个，如果每一次在视图中显示机台状态布局图的时候从数据库中检索然后再显示，这样速度太慢，视图在刷新的时候画面闪烁。于是就考虑采用 MFC 或者 STL 的数据结构，在视图显示之前，就一次性的把机台所有的记录都检索出来，保存到数组或者是一个链表当中，在需要显示的时候，只需要检索链表或者数组，不再经历打开数据库、检索数据、显示的漫长过程了。

MFC 和 STL 都提供了动态数组，在 MFC 中是 CArray，在 STL 中是 vector。考虑到动态数组在增加对象或者元素时效率较低，在数组预留空间不够用时，都是经历一个分配一个大的存储单元、将原来的数组内容复制到新开辟的数组中，销毁原来的数组这样一个效率低下的过程。所以考虑选用链表结构。

下面对两种技术的运行时间作一对比，对两者实现算法所占用的存储空间未作考虑。比较的方案是在同一台计算机上运行一段程序，这段程序用两种方法实现建立 500 个数据对象的链表和检索 500 个数据对象。分别记录并显示在建立链表过程中和在对链表检索过程中所用的时间。

采用 MFC 链表技术是这样实现的，先定义一个数据类型 typedef CTypedPtrList < CObList, CMonitorObj * > CMonitorObjList；然后再根据这个类型声明一个变量 CMonitorObjList m_objects；

下面列出采用 MFC 中 CTypedPtrList 类建立链表的部分程序段：

```
void CMachineView::CreateList(void)
{
    CMachineRS rs;//声明机台记录集对象
    rs.Open();//打开记录集,省略异常处理部分
    while (! rs.IsEOF())
    {
      CMonitorObj * obj = new (CMonitorObj);
      obj->strMachine = rs.m_strMachine;
      obj->point = CPoint(rs.m_leftx, rs.m_lefty);
      m_objects.AddTail(obj);//将数据对象加入链表
      rs.MoveNext();
    }
    rs.Close();//关闭记录集
}
```

在视图中显示的程序段如下：

```
POSITION pos;
    pos = m_objects.GetHeadPosition();//得到头节点位置
    while (pos != NULL)
    {
      CMonitorObj * pObj = m_objects.GetNext(pos);
      pObj->Draw(pDC); //调用机台对象函数显示机台矩形,并将机台号显示在矩形内
    }
```

采用 STL 方法时，首先声明一个链表变量 list < CMonitorObj * > m_list；链表的建立过程如下：

```
void CSTLView::CreateList(void)
{
    CMachineRS rs;//声明机台记录集对象
    rs.Open();//打开记录集,省略异常处理部分
    while (! rs.IsEOF())
    {
      CMonitorObj * obj = new (CMonitorObj);
      obj->strMachine = rs.m_strMachine;
      obj->point = CPoint(rs.m_leftx ,rs.m_lefty);
      m_list.push_back(obj);//将对象指针加入链表
      rs.MoveNext();
    }
    rs.Close();//关闭记录集
}
```

检索链表在视图上显示机台位置状态图的过程如下：

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn

```
list < CMonitorObj * > : : iterator iter ; //声明迭代器
for ( int i = 0 ; i < 100 ; i + + )
{
    for ( iter = m_list. begin( ) ; iter ! = m_list. end
( ) ; iter + + )
        ( * iter) - > Draw( pDC) ; //调用机台对象函数
显示机台矩形,并将机台号显示在矩形内
}
```

对时间的记录采用 timeGetTime 函数,该函数记录的是系统自开机到现在为止所经历的毫秒数。该函数在 Windows 2000 和 Windows NT 系统中,默认的精确值为 5 毫秒,通过调用 timeBeginPeriod 和 timeEndPeriod 两个函数可以获得更高的时间精确值[2]。

根据链表建立前后所记录的时间,可以计算出在链表建立过程中所耗费的毫秒数。反复执行链表建立程序,得到十组实验数据(单位是毫秒):

	1	2	3	4	5	6	7	8	9	10	平均
MFC 方法创建链表	313	312	313	313	312	328	312	313	329	312	315.5
STL 方法创建链表	323	313	328	328	312	312	312	312	313	312	315.3

从这组实验数据我们可以看到,不管采用 STL 或者 MFC 链表机制,建立对象链表所耗费的时间几乎是一样的。

在执行链表检索并访问链表对象数据时,发现使用上述两种技术对于链表的检索所需要的时间非常短,几乎看不出有何差别。于是在上述给出的两个链表检索显示函数中加了一个循环,让链表检索的过程循环 100 遍。增加了这 100 次循环之后,根据循环前后所记录的时间,得到以下实验数据(单位是毫秒):

	1	2	3	4	5	6	7	8	9	10	平均
MFC 方法检索链表	531	531	516	532	513	515	655	531	559	531	541.4
STL 方法检索链表	516	531	531	547	532	562	531	578	547	531	540.6

从链表检索可以看到,其效率都是非常高的。在链表中将 500 个数据对象检索出来,并且根据对象的数据成员将机台位置和机台号显示到屏幕上。这样重复 100 次才耗时 540 毫秒。而且对两种不同的链表方案它们所用的时间基本上都是一致的。

4 结论

从上面的实验方案和实验数据中可以看出,不管采用 MFC 技术还是 STL 技术,其运行所需的时间大致相同,也就是两者的时间效率是一样的。从程序实现的角度来看,采用 STL 技术,实现起来比较简单,容易理解。采用 MFC 技术实现起来也不复杂,效果也很理想。所以,在实际的应用中,操纵数据对象究竟采用 MFC 技术还是 STL 技术,从时间效率这个角度出发,两者技术都是成熟和高效率的,可以任意选择。

参考文献

1 侯捷, STL 源码剖析, 华中科技大学出版社, 2002 年 6 月。

2 MSDN Library for Visual Studio. NET 2003. Microsoft Corporation, 2003 年。

3 彭木根、王淑凌, C + + STL 程序员开发指南, 2003 年 4 月。

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn