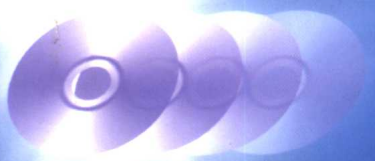

本书含光盘一张

不必艰难求索 本书最佳选择

胡峪 刘静 编著

Visual C++ 编程技巧与示例

西安电子科技大学出版社
http://www.xduph.com

VC++ 编程技巧与示例

胡峪　刘静　编著

西安电子科技大学出版社

2000

内 容 简 介

本书深入浅出地介绍了从VC++的入门知识到高级应用技术的各个主要方面，内容覆盖了简单的VC++基本操作，应用程序界面设计，复杂的进程、线程、消息和数据库，ActiveX等高级技术。为了使读者也能够编制“有个性”的应用程序，充分发挥VC++的强大功能，本书还讲解了许多高级编程技巧。因此，本书不仅适合于初学者，而且也适合于具有一定基础的读者。本书在讲解每个专题时，首先以简明的方式阐明了所涉及的技术，然后还为每个专题都提供了精巧的示例。这些示例都是由作者精心设计的，具有典型性、趣味性。

本书可供计算机程序员和计算机爱好者使用。

图书在版编目（CIP）数据

VC++编程技巧与示例/胡峪，刘静编著. —西安：西安电子科技大学出版社，2000.6

ISBN 7-5606-0846-9

Ⅰ.V… Ⅱ.① 胡… ② 刘… Ⅲ.C语言-程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字（2000）第22555号

责任编辑 毛红兵 戚文艳

出版发行 西安电子科技大学出版社（西安市太白南路2号）

电 话 （029）8227828 邮 编 710071

http://www.xduph.com E-mail: xdupfxb@pub.xaonline.com

经 销 新华书店

印 刷 西安市第三印刷厂

版 次 2000年6月第1版 2000年6月第1次印刷

开 本 787毫米×1092毫米 1/16 印张 22.75

字 数 538千字

印 数 1~4 000册

定 价 （含光盘）35.00元

ISBN 7-5606-0846-9/TP·0441

如有印装问题可调换

前　言

Visual C++（VC++）是微软公司 Visual Studio 编程开发工具包的组件之一，也是 Windows 平台上功能最强大的编程工具。C++的功能涵盖了所有的 Windows 编程领域；使用 VC++可以胜任从最简单的用户界面程序到高级、复杂的 Windows 应用程序的编程工作；使用 VC++编制的程序具有代码尺寸小和运行速度高的特点，所有这些优点都是其他编程工具（C++Builder、Visual Basic 等）所不具备的。此外，学会 VC++编程后，就会对 Windows 的各个方面有一个深刻的理解，再去学习其他编程工具就会感到十分轻松自如，有一种“一览众山小”的感觉。

但是，许多人都视 VC++为畏途，认为 VC++太难而敬而远之。实际上，VC++并不复杂。笔者已经具有两年多的 VC++编程经验，编制过大量的应用程序代码，也经历过由初学 VC++时的不知所措到熟练应用的过程，对学习 VC++的方法和技巧有许多体会。因此，我们特意编写了这本书，旨在通过自己的切身体会引导初学者由入门到精通，同时也为已经掌握了 VC++的读者提供一个有用的参考资料，以便随时查询。

本书分为三个部分：入门篇、提高篇和高级编程。入门篇首先讲解一些关于 C++的常用高级话题，引导读者灵活应用 C++的高级特性，并为学习 VC++作好准备；然后又讲解了 VC++中的通用类，这些在后面的章节中经常出现。提高篇主要讲解了 VC++编程基础，包括文档视图、对话框与控件、绘图和动态联接库。这部分以 VC++基础为主，同时在每一章的最后一两节中讲述了较为高级的编程技巧，旨在引导读者入门之后进一步提高，这些内容也有助于已经掌握了 VC++的读者作为资料查询。高级编程部分深入浅出地讲述了 VC++的高级应用技术，包括线程与进程、数据库、ActiveX 等；学会这些内容之后，读者将成为精通 VC++的程序员。根据笔者的体会，VC++的许多技巧看似复杂，实际上只要学懂了其中的几句核心代码便可以举一反三，灵活应用。因此，在讲述技巧时，本书将阐明这些技巧有何作用，并对核心代码和相关技术加以详细阐述，而不是将一大堆代码简单地罗列出来，让读者自己去领会。

本书中每一个专题都有示例，所有示例都由笔者亲自调试通过。如果读者对本书中的示例代码感兴趣，可以与笔者联系，笔者联系地址是：

E-mail: CougarHuYu@Yahoo.com

当然，VC++是一个十分庞杂的编程工具，本书限于篇幅不可能面面俱到。同时，鉴于本人水平有限，难免有不妥之处，欢迎读者指正。

编者

2000 年 1 月

本书示例一览表

示例所演示的技术	所在章节
改变菜单状态	3.2.2
快捷菜单	3.3.1
自定义菜单	3.3.2
多文档模板	4.2.4
切分多视图	4.3.1
基于框架窗口和切分多视的多视图	4.3.2
基于 MDI 子窗口的多视图	4.3.3
全屏显示 MDI 子窗口或 SDI 框架窗口	4.4.1
对各种控件的编程	5.2.1
无模式对话框	6.2.3
属性页、属性表	6.3.3
向导（Wizard）	6.4.3
不规则形状对话框	6.5.2
进程间和进程内部通过自定义消息传递数据	7.6.3
用 GDI 进行图形绘制	8.3.6
DDB 位图和 DIB 位图	8.4.4
用鼠标绘图	8.5.2
创建和使用基于 C 语言的动态连接库	9.2.2
创建和使用 MFC 常规型动态连接库	9.3.2
创建和使用 MFC 扩展型动态连接库	9.4.2
创建和使用线程、线程间通讯	10.3.4
线程间同步	10.3.6
创建并管理子进程	10.4.2
在进程间共享数据的示例	10.4.4
ODBC 数据库应用程序	11.2.6
DAO 数据库应用程序	11.3.5
ActiveX 文档完全服务器	12.4.3
自动化服务器（基于可执行文件）	12.5.3
自动化服务器（基于动态连接库）	12.5.3
自动化客户	12.5.3
创建 ActiveX 控件	12.6.2
在 Web 页中使用 ActiveX 控件	12.6.4

目　录

第一篇　入　门　篇

第二篇 提 高 篇

第三篇 高 级 篇

第一篇 入门篇

第1章 C++应用技巧

众所周知，VC++是一个功能十分强大的应用程序开发工具，这不仅仅是因为 VC++中 MFC 类库和 Windows API 的功能强大，而且在于 C++语言本身功能就十分强大。利用 C++面向对象的编程思想，可以有效地解决程序编制中的许多问题。

本章首先讲述一些实用的 C++使用技巧。由于本书并非以讲解 C++编程为目的，本章将仅限于 C++的一些高级技术。然后还将讲述 VC++提供的几个通用类，这些类进一步方便了应用程序的设计。

1.1 运算符重载

1.1.1 概述

使用过 MATLAB 的人一定知道，MATLAB 的矩阵运算功能十分强，而且使用非常方便。实际上，方便的矩阵运算功能并不是 MATLAB 所独有的，功能强大的 C++同样可以使矩阵运算得以简化，且 C++的执行速度要比 MATLAB 快很多。

利用 C++语言，实现 MATLAB 式的矩阵运算十分简单，只要使用 C++的运算符重载就可以达到目的。

1.1.2 技术核心

C++运算符重载允许用户重新定义各种 C++运算符的功能，使自定义运算功能的实现成为可能。下面让我们来看一下什么是运算符重载，如何实现 C++运算符重载及应该注意的问题。

1. C++的运算符重载

C++的运算符重载为 C++程序员提供了自行定义运算符的功能，用户可以根据具体运算法则自行定义运算符，例如，利用 C++的运算符重载来定义复数的运算法则。C++可以重载的运算符有：

, ! != % %= &（与运算） &（求地址） && （逻辑与）
&= ( ) * * *= + （正号） + ++ += -（负号） - - -
-= -> ->* / /= < << <<= <= = == > >= >>
>>= [ ] ^ ^= | |= || ~ delete new

C++不能重载的运算符有：

. .* :: ?: # ##

2. C++运算符重载的基本语法

C++的运算符重载有两种方法，一种是基于成员函数的运算符重载，另外一种是基于友员的运算符重载。C++运算符重载的语法分以下两类：

(1) 基于成员函数的运算符重载：

对单目运算符：

```
r_tpye operator op( )
```

对双目运算符：

```
r_tpye operator op(arg)
```

(2) 基于友员的运算符重载：

对单目运算符：

```
friend r_tpye operator op(arg)
```

对双目运算符：

```
friend r_tpye operator op(arg1，arg2)
```

其中，friend 为友员标志，r_type 为运算返回的数据类型；operator 是 C++关键字，用于声明运算符重载；op 为被重载的运算符，如+、-、*、/ 等。

C++重载的例子如下所示：

```
class CMatrix
{
public:
//基于成员函数的运算符重载：
    CMatrix operator + (CMatrix & )；//双目运算符；
    CMatrix operator ! ( )；//单目运算符；
//基于友员的运算符重载：
    friend CMatrix operator ~ ( CMatrix &)；//单目运算符；
    friend CMatrix operator - (CMatrix， CMatrix)；//双目运算符；
};
```

3. 赋值运算符

赋值运算符“=”与其他任何运算符都不同，对赋值运算符有以下规定：

(1) 赋值运算符必须被定义成非静态成员函数。

(2) 它不能被派生类继承。

(3) 如果没有定义赋值运算符，编译器将产生一个默认赋值运算符。

4. 指针丢失问题

当类中定义了一个指针时，如果赋值运算符使用不当，将出现严重问题。例如，在头文件中按如下方式定义 CMatrix 类：

```
//CMatrix.h:
class CMatrix
{
    double * data;
```

```
};
```

在 CMatrix 的构造函数中为 data 动态分配一块内存，在 CMatrix 的析构函数中删除这块内存：

```
//CMatrix.cpp
CMatrix::CMatrix( ) //构造函数：
{
    data=new double[1000];
    CMatrix & operator = (const CMatrix & mtx);
}
CMatrix::~CMatrix( )      //析构函数：
{
    delete [ ] data;
}
CMatrix::& CMatrix::operator = (const CMatrix & mtx)
{
    data=mtx.data;  //这一句代码有安全隐患；
}
```

使用时，生成了两个 CMatrix 类对象，一个全局对象和一个局部对象：

```
# include "CMatrix.h"
CMatrix m_Global;
void CalMatrix( )
{
    CMatrix m_Local;
    m_Global=m_Local;
}
    void main( )
{
    double ref;
    ref=m_Global.data[20];  //出现错误；
}
```

出错的原因是 m_Local 是一个局部变量。当 CalMatrix( )函数结束后，将调用 m_Local 的析构函数，m_Local 中 data 指向的内存被释放，而 CMatrix 类赋值运算符的定义却使 m_Global 的 data 指针同 m_Local 的 data 指针指向了同一块内存，于是出现了指针丢失问题。因此，在定义赋值运算符时，应该对 data 的各项依次进行复制，于是对 CMatrix 类的赋值运算应进行如下修改：

```
CMatrix::& CMatrix::operator = (const CMatrix & mtx)
{
        int i;
    for(i=0; i<size; I++)
    {
        data[i]=mtx.data[i];
    }
```

```
}
```

5. 拷贝构造函数

要使运算符重载后计算完全没有问题，光靠对赋值运算符进行修改是不够的，还必须修改拷贝构造函数。

C++中有 3 种构造函数：

- 默认构造函数。
- 自定义构造函数。
- 拷贝构造函数。

当进行值传递时，将产生临时的类对象，此时需要调用类的拷贝构造函数。如果没有定义拷贝构造函数，系统将使用默认拷贝构造函数进行类的复制。如果类中有动态分配的内存区，则新的类对象中的指针将指向与临时的类对象中的指针同样的一块内存区域。当临时类对象被撤消后，新的类对象的指针将仍然指向原来的内存区域，从而出现指针悬挂问题，并导致系统崩溃。因此，必须加入自定义拷贝构造函数：

```
CMatrix::CMatrix(const CMatrix& mtx)
{
    int i;
    for(i=0; i<size; i++)
    {
        data[i]=mtx.data[i];
    }
}
```

1.1.3 一个使用运算符重载的例子

正如前面所说的那样，如果能够编制一个像 MATLAB 一样使用方便的矩阵类是非常令人激动的。有了 1.1.2 的知识后，我们已经具备了编制一个使用方便并且足够可靠的矩阵类的能力。

下面将列出一个实用的 CMatrix 类的全部源代码清单，这些代码由本书作者精心调试过，并且其可靠性在实践中已被多次验证。

1. CMatrix 类的头文件

CMatrix 类的头文件清单如下：

```
//CMatrix.h :
#if !defined(AFX_CMATRIX_H_C4B3EEF5_09EA_11D3_B596_0000000038B2_INCLUDED_)
#define AFX_CMATRIX_H_C4B3EEF5_09EA_11D3_B596_0000000038B2_INCLUDED_
#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
class CMatrix
{
    public:
    //构造函数和析构函数：
```

```
CMatrix(unsigned r)；//自定义构造函数；
CMatrix( )；//默认构造函数；
CMatrix(unsigned rs，unsigned cs)；//自定义构造函数；
CMatrix(const CMatrix & mtr)；//拷贝构造函数；
CMatrix(unsigned rs，unsigned cs，double val[ ])；//自定义构造函数；
virtual ~CMatrix( )；//析构函数；
//一般成员函数：
double * getdatabuffer( )；//获取矩阵类中数据缓冲区；
void addrow(UINT r，double dat[ ])；//添加一行；
void addcol(UINT c，double dat[ ])；//添加一列；
void addrow(UINT r，double dat  )；//添加一行，用于列向量；
void addcol(UINT c，double dat  )；//添加一列，用于行向量；
UINT getsize( )；//获取数据缓冲区大小；
UINT getcols( )；//获取矩阵列数；
UINT getrows( )；//获取矩阵行数；
double mod( )；//求矩阵的模；
void deletecol(unsigned c)；//删除指定的列；
void deleterow(unsigned r)；//删除指定的行；
void MoveRow(unsigned row1，unsigned row2)；//将第 row1 行的数据移向第 row2 行；
int  calroot( )；//求方程组的根，此时 CMatrix 类对象为一个线性方程组的增广矩阵；
void setvalue(unsigned r，unsigned c)；//初始化 CMatrix 类对象，指定矩阵大小；
void setvalue(unsigned r，unsigned c，double val[ ])；//初始化 CMatrix 类对象，指定矩阵大
                                                    //小；同时为矩阵的各个元素赋值；
void setvalue(double val[ ])；//初始化 CMatrix 类对象，为矩阵的各个元素赋值；
int  savematrix(char  *fn)；//将矩阵的所有信息存入磁盘；
double det( )；//求 CMatrix 对应的行列式的值，此时矩阵应是一个方阵；
void setelem(unsigned row，unsigned col，double val)；//为矩阵的第 row 行、第 col 列赋值；
void delelem(unsigned r，unsigned c)；//以第 r 行、第 c 列为主元进行消元；
void unit( )；//将矩阵对角元素单位化（进行高斯消元）；
//运算符重载成员函数：
double operator ( ) (unsigned row，unsigned col)；//提取某一矩阵元素；
CMatrix & operator = (const CMatrix & mtx)；//赋值运算符；
//重载友员函数：
friend  CMatrix operator + (CMatrix p，CMatrix q)；//加运算符；
friend  CMatrix operator - (CMatrix p，CMatrix q)；//减运算符；
friend  CMatrix operator * (CMatrix p，CMatrix q)；//乘运算符（两个矩阵相乘）；
friend  CMatrix operator * (double  p，CMatrix q)；//乘运算符（一个实数乘一个矩阵）；
friend  CMatrix operator / (CMatrix p，double  q)；//除运算符，一个矩阵除一个实数；
friend  CMatrix operator ~ (CMatrix p)；//求原矩阵的转置矩阵；
friend  CMatrix operator ! (CMatrix p)；//求原矩阵的逆矩阵；
friend  CMatrix operator - (CMatrix p)；//求原矩阵的负矩阵；
//成员变量：
private:
```