高等学校计算机专业规划教材

Visual C++ 简明教程

张海林　杜忠友　姜玉波　编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http: //www.phei.com.cn

高等学校计算机专业规划教材

Visual C++ 简明教程

张海林　杜忠友　姜玉波　编著

電子工業出版社
Publishing House of Electronics Industry
北京・BEIJING

内 容 简 介

本书以 Visual C++6.0 中文版为平台，从 Windows 编程入手，系统地介绍了 Visual C++编程基础。主要内容包括：Windows 编程基础，Visual C++开发环境，消息，通用类和通用函数，鼠标、键盘、菜单、工具栏、状态栏、图形设备接口、对话框和通用控件编程，文档/视图结构，数据库编程，以及 Visual C++的程序调试方法。本书配套资源丰富，包括习题答案、程序源代码、实验指导、电子教案、视频课件等，均为免费资源。通过本书的学习，完成实验及实训，读者可以具备用 Visual C++开发小型应用系统的能力。

本书可作为大学本科、高职高专院校工科各专业的程序设计教材，也可作为 Visual C++的培训教材或读者自学的参考用书。

图书在版编目（CIP）数据

Visual C++简明教程/ 张海林，杜忠友，姜玉波编著. —北京：电子工业出版社，2007.6
高等学校计算机专业规划教材
ISBN 978-7-121-04470-0

Ⅰ. V… Ⅱ. ①张… ②杜… ③姜… Ⅲ. C 语言—程序设计—高等学校—教材 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2007）第 072673 号

责任编辑：王 纲
印 刷：北京市海淀区四季青印刷厂
装 订：涿州市桃园装订有限公司
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编 100036
开 本：787×1 092 1/16 印张：16.5 字数：422.4 千字
印 次：2007 年 6 月第 1 次印刷
印 数：4 000 册 定价：24.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

在种类繁多的可视化开发工具中，微软公司的 Visual C++6.0 以其功能强大，深受广大开发人员欢迎。本书以“简单介绍基础知识→解析实例→读者动手实践”的形式教读者如何使用 Visual C++。

Visual C++强大的功能有时会给人无从下手的感觉，这增加了学习 Visual C++的难度，本教程将尽力使这个学习过程变得相对简单。在教程内容的编排、实例的选取、章节次序的安排上，力求做到深入浅出，容易上手。

本书共 13 章，首先概要性地介绍 Windows 编程基础、Visual C++开发环境及消息映射，然后依次详细讲解通用类、鼠标、键盘、菜单、工具栏、状态栏、图形设备接口、对话框、通用控件、文档/视图结构及数据库的编程方法，最后给出了两个综合实例，可以作为综合练习、课程设计或实习实训的教学内容。

为了便于教师组织教学，本教程配有十分丰富的教学资源，具体如下：

- **习题答案、电子教案等：**本教程附课程教学大纲、实验教学大纲、实验指导书、教案、课件、教程所有实例源代码，以及全部习题答案，这些文件均可在作者的教学网站（http://zhidao.3322.org:90）或华信教育资源网（www.huaxin.edu.cn 或 www.hxedu.com.cn）上下载。
- **应用系统：**作者的教学网站上提供本教程课程设计中的综合实例源代码及开发过程讲解。
- **视频教程：**作者将本教程中所有实例的具体操作过程制作成视频课件，非常适合读者自学时使用。

本教程不仅适用于教学，也适用于 Visual C++的各类培训和读者自学。

本教程由张海林、杜忠友、姜玉波编著，全文由张海林统稿。参编人员有孙晓燕、解艳艳、李锋，其中，孙晓燕参与了第 1 章的编写，杜忠友参与了第 2，3，9 章的编写，解艳艳参与了第 4 章的编写，姜玉波参与了第 7，8，10 章的编写，李锋参与了第 12 章的编写。

本书在编写过程中，参考和引用了许多参考文献，在此向这些文献资料的作者、编者表示衷心的感谢。

由于编者水平有限，不当甚至错误之处在所难免，敬请读者批评指正。

编　者

目　录

第 1 章　Windows 编程基础

本章学习要求

1）掌握事件驱动编程机制

2）了解 Windows API 与 Windows SDK 的区别

3）了解封装、继承的概念

4）了解基于文档/视图结构的框架程序结构

过去，Windows 应用程序设计是一件很麻烦的事情，因为当时还没有像 Visual C++这样一些优秀的应用程序开发工具。现在，一个对 Windows 应用程序运行机制几乎一无所知的入门者，只需要通过短期学习，就可以使用如 Visual Basic，Delphi 之类的程序开发工具写出功能完整的 Windows 应用程序。

从某种角度说，利用微软的另一种编程工具 Visual Basic 开发的 Windows 应用程序不是编出来的，而是由程序员“画”出来的。但是对于具有一定基础的程序员而言，不仅要会在屏幕上“画”出应用程序的各个窗口，更重要的是要知道 Windows 应用程序的运行机制，明确自己在编写程序时要做的工作是哪些。

虽然程序员使用 Visual C++中的应用程序向导，也能做到只添加几行代码就开发出功能完整的 Windows 应用程序，但是，没有一个成功的商业软件是以这种方式开发的。只有深入理解 Windows 应用程序的运行机制，才可能写出好的应用软件。

本章主要介绍在开发 Windows 应用程序时涉及的一些概念，并对基于 MFC（Microsoft Fundation Classes，微软基础类库）的框架程序进行分析。如果一开始你对这些东西不感兴趣，可以跳过本章，阅读本书的其他部分，当遇到概念上的问题时，再回头来补充这些知识。

1.1　面向对象编程基础

近年来，面向对象技术无论在理论上还是在实践上都得到了飞速发展。面向对象技术相关概念中最重要的就是类和对象的概念。面向对象技术的主要特性有封装、继承、多态，其中，封装是对客观事物属性和方法的抽象描述，继承是关于抽象描述的演化发展，多态则是对程序流程前瞻性的刻画。

下面依次介绍类、对象、封装、继承和多态的概念，并分别举例说明。

1.1.1　类和对象

类是面向对象技术中的核心概念，它包括类名称、属性和方法 3 个构成要素，其中，类名称用来区别其他类，属性是描述客观事物静态特征的数据项，方法是描述客观事物动态特征和行为的操作。类的属性和方法是对客观事物静态特征和动态行为的抽象。类中的属性

通常被称为数据成员，类中的方法通常被称为成员函数。在 Visual C++中，类用关键字 class 来定义，下面以课程类的定义为例，说明类的定义方法。

【例 1.1】 定义一个课程类。

```
class  Course                //定义类名称为 Course
{
    int num;                 //定义整型变量，num 表示课程编号
    char name[50];           //定义字符数组，name 表示课程名称
    char teacher[8];         //定义字符数组，teacher 表示课程的任课老师
    int hour;                //定义整型变量，hour 表示课程的学时数
    GetName(    );           //定义函数 GetName，用来得到课程名称
    GetTeacher(    );        //定义函数 GetTeacher，用来得到课程的任课老师
};
```

例 1.1 中的 num, name, teacher 和 hour 是课程类的属性，是对课程静态特征的抽象。而 GetName，GetTeacher 是课程类的方法，是对课程动态行为的抽象。

对象是类的实例化，下面以课程类为例，说明对象的定义方法。

```
Course c;                    //定义类 Course 的一个实例 c
```

类与对象的关系就像数据类型与变量的关系一样。例如，整型 int 是所有整型变量共性的抽象，它本身并不能参与运算，要想在程序中使用整型数据，必须先用 int 定义一个变量，这个变量可以表示某个整型数据。类和对象的关系也是如此，作为一个自定义的数据类型，类本身不能在程序中表示任何数据，要想在程序中使用某个类的属性或方法，必须先用这个类定义一个对象，然后通过这个对象去访问类中定义的属性或方法。例如，课程类是所有课程对象的抽象，它并不表示某个课程的信息，要想表示某个课程的信息，必须用课程类定义一个对象，这个对象是课程类的一个实例，它可以表示某个具体课程的信息。

1.1.2 封装

封装是指将客观事物的属性和作用在这些属性上的操作集合在一起，成为一个类。封装可以解决类中的数据保护问题。在面向对象技术中，封装是通过引入关键字 protected 或 private 实现的。类中被这两个关键字修饰的成员是保护成员或私有成员。保护成员和私有成员都不能被类的对象访问，只有被关键字 public 修饰的成员才能被类的对象访问。

【例 1.2】 定义一个类，并演示封装的用法。

```
#include <iostream.h>
class db                              //定义类名称为 db
{
private:                              //开始定义私有成员
  double d;
protected:                            //开始定义保护成员
  SetD() {d=2.5;}
  double sqr() {return d*d;}
public:
```

```
    print() {cout<<"hello world"<<endl;}
};
void main()
{
    int i;
    double f;
    db d1;                          //定义对象 d1
    d1.SetD();                      //通过对象 d1 调用类中定义的成员函数
    f= d1.sqr();                    //通过对象 d1 调用类中定义的成员函数
    d1.print();
    cout << f << endl;              //输出数据 f
}
```

在本例中，d1.SetD();和 d1.sqr();都试图通过对象 d1 调用类 db 中的成员函数，但是这两个成员函数都被声明为保护成员，而类的保护成员不能被对象访问，所以这两条语句是错误的。由于 print 函数被声明为公有成员，因此 d1.print();是正确的。

1.1.3　继承

保持已有类的特性，构造新类的过程称为继承。继承是在一个已有类的基础上再声明一个新类的扩展机制，已有类称为基类，在基类的基础上扩展出来的类称为派生类。通常，基类是一个简单的类，描述相对简单的客观事物，派生类是一个相对复杂的类。

派生类可以调用基类中允许访问的数据成员和成员函数，其本身又可以增加新的数据成员和成员函数。继承的定义方法如下：

```
class CDerived: 继承方式 CBase
{
    CDerived 类的成员声明语句;
};
```

其中，CDerived 是新引入的派生类，CBase 是已经建立的类。

【例 1.3】 定义两个类，并演示继承的用法。

```
#include "iostream.h"
class Base                          //定义基类 Base
{
private:                            //开始定义私有成员
    int count;
public:                             //开始定义类 Base 的公有成员
    int GetCount() {return count;}
};
class Derived: public Base          //定义派生类 Derived
{
private:
```

```
    int age;
public:
    int GetAge() {return age;}
};
void main()
{
    int c1,c2;
    Derived d;                    //定义派生类的对象
    c1=d.GetCount();              //通过派生类 Derived 的对象调用基类 Base 的成员函数
    c2=d.GetAge();                //通过派生类 Derived 的对象调用派生类新定义的成员函数
    cout<<c1<<c2<<endl;
}
```

在本例中，派生类 Derived 并没有声明 GetCount()函数，所以派生类的对象 d 应该不能访问该函数，即 d.GetCount()应该是不对的。但是派生类 Derived 的基类 Base 中声明了该函数，这样派生类 Derived 的对象 d 就可以利用继承机制实现对 GetCount()的访问，即 d.GetCount()是正确的。

1.1.4 多态

多态是在类继承过程中与函数调用相关的约束应变模式。多态兼顾软件的可扩充性和软件流程的统一性，体现了变动与约束的结合。多态分为编译时的多态性和运行时的多态性两种。

编译时的多态性也称静态联编，它通过函数重载实现。函数重载是 C++支持的一种函数定义机制，即同一个函数名可以对应不同的函数体，从而实现不同的功能。

【例 1.4】 定义两个函数，并演示函数重载的用法。

```
#include "iostream.h"
double sqr(double y)
{
    return y*y;
}
int sqr(int y)
{
    return y*y;
}
void main()
{
    int i=5;
    double d=5.5;
    cout<<sqr(i)<<endl;
    cout<<sqr(d)<<endl;
}
```

本例中的函数名 sqr 就对应两个函数体，它们可以实现不同的功能，这种用法就是函数重载。

运行时的多态性也称动态联编，它是通过虚函数实现的。虚函数是用关键字 virtual 修饰的函数。简单的说，运行时的多态性是通过指向基类对象的指针动态调用派生类的虚函数，从而实现不同的功能。

【例 1.5】 定义两个类，并演示虚函数的用法。

```
#include "iostream.h"
class base
{
public:
   virtual void f1( ) {cout<<"f1 function of base \n ";}        //定义虚函数
   void f2( ){cout<<"f2 function of base \n";}                  //定义一般函数
};
class derive: public base
{
 public:
    void f1( ){cout<<"f1 function of derive \n";}               //重新定义虚函数
    void f2( ){cout<<"f2 function of derive \n";}               //重新定义一般函数
};
void main( )
{
   base ob1, *p;                                                //定义基类对象
   derive   ob2;                                                //定义派生类对象
   p=&ob1;
   p->f1( );                                                    //调用基类函数
   p->f2( );                                                    //调用基类函数
   p=&ob2;
   p->f1( );                                                    //调用派生类函数,实现多态性
   p->f2( );                                                    //由于 f2 没有定义为虚函数，所以该处调用基类函数
}
```

1.2　Windows 应用程序

1.2.1　Windows 应用程序工作原理

Windows 应用程序提供给用户的界面中有许多可视对象。用户在界面上进行操作会触发特定的事件，这些事件会向应用程序的对象发送消息，最后由对象调用相应的消息处理函数来实现特定的功能。应用程序的运行过程就是用户的外部操作不断触发事件，这些事件被相应对象处理的过程。Windows 应用程序工作原理示意图如图 1.1 所示。

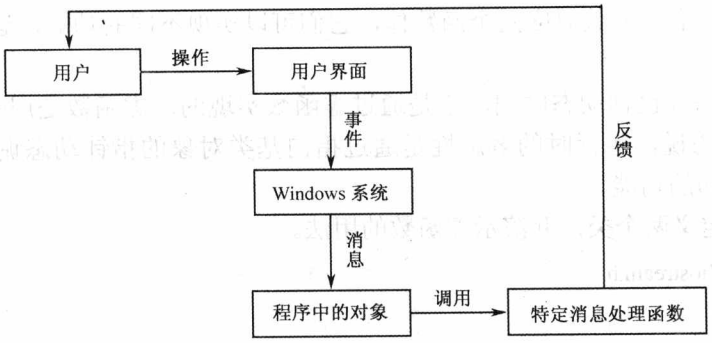

图 1.1　Windows 应用程序工作原理示意图

1.2.2　Windows 应用程序设计原理

在 Windows 平台上进行程序设计时，采用的编程方法与在 DOS 平台上不同。Windows 应用程序是事件驱动（或称消息驱动）的应用程序。Windows 为每个应用程序都设置了消息队列，Windows 应用程序的任务就是不停的从消息队列中获取消息、分析消息和处理消息，直到接到一条 WM_QUIT 消息为止，这个过程通过消息循环实现。

Windows 是一个多任务的操作系统，在 Windows 中有着多个应用程序在同时运行。在这样的操作系统中，应用程序如何获得用户的输入呢？实际上，操作系统时刻监视着用户的操作，并分析它与哪一个应用程序相关联，然后将用户的操作以消息的形式发送给该应用程序。应用程序一旦发现有未处理的消息，就分析该消息，并根据消息内容采取适当的动作，来响应用户的操作，这个过程称为消息处理。

图 1.2 给出了 Windows 应用程序的执行流程。

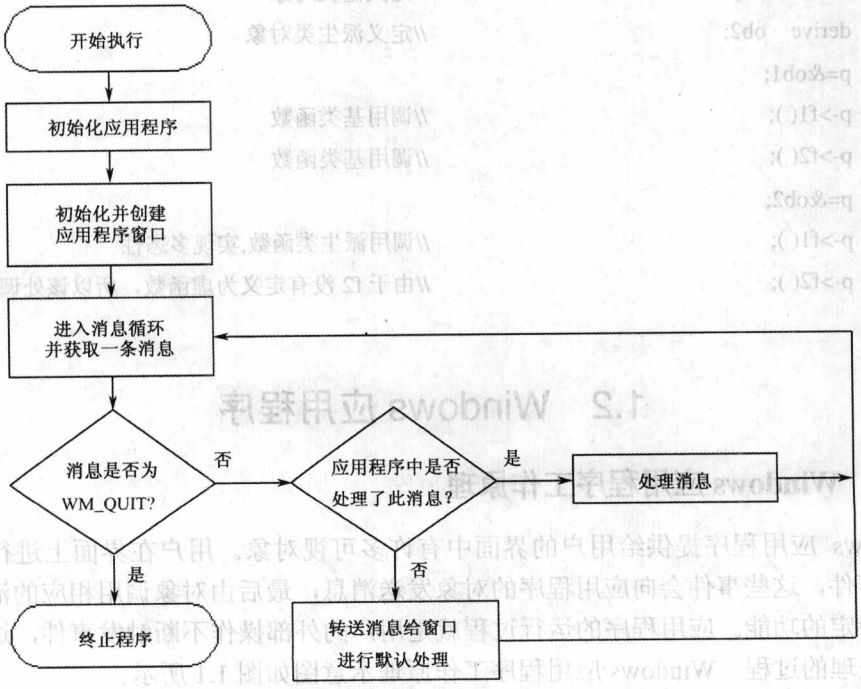

图 1.2　Windows 应用程序的执行流程示意图

如图 1.2 所示，可以看出 Windows 应用程序由一系列消息处理代码组成，编程者只能够预测用户利用应用程序进行的可能的操作，并为这些可能的操作编写消息处理代码，但不可能知道什么消息会在什么时候触发。

1.3　基于 MFC 的框架程序分析

1.3.1　Windows API，Windows SDK 与 MFC

1．Windows API 与 Windows SDK

Windows API（Windows Application Programming Interface，Windows 应用程序编程接口）是所有 Windows 应用程序开发的根本。简单的说，Windows API 就是一系列例程，应用程序通过调用这些例程来请求操作系统完成某些功能。在 Windows 的图形用户界面中，应用程序的窗口、图标、菜单和对话框等都由 Windows API 管理。

Windows SDK（Windows Software Development Kit，Windows 软件开发工具包）是一套帮助 C 语言程序员创建 Windows 应用程序的开发工具。

面向对象的编程方法是当前流行的程序设计方法，但 Windows API 本身却是基于 C 语言的过程式编程。

传统的 C 程序以 main 函数作为程序的入口点，在 Windows 应用程序中，main 函数被 WinMain 函数取代，WinMain 函数的原型如下：

```
int WINAPI WinMain (HINSTANCE hInstance,     // 当前实例句柄
HINSTANCE hPrevInstance,                     // 前一实例句柄
LPSTR lpCmdLine,                             // 指向命令行参数的指针
int nCmdShow)                                // 窗口的显示状态
```

在上面的函数原型中，句柄是一个标识对象的变量。除此之外，还有一些奇怪的数据类型，如 HINSTANCE，LPSTR 等。实际上，这些数据类型是基本数据类型的别名，编程时常用的基本数据类型及其说明见表 1.1。

表 1.1　Windows 中常用的基本数据类型及其说明

Windows 中常用的数据类型	对应的基本数据类型	类 型 说 明
BOOL	int	布尔值
BYTE	unsigned char	8 位无符号整数
COLORREF	unsigned long	用作颜色值的 32 位值
LONG	long	32 位带符号整数
LPARAM	long	作为参数传递给窗口过程或回调函数的 32 位值
LPCSTR	const char *	指向字符串常量的 32 位指针
LPSTR	char *	指向字符串的 32 位指针
LRESULT	long	来自窗口过程或回调函数的 32 位返回值
UINT	unsigned int	32 位无符号整数
WPARAM	unsigned int	当作参数传递给窗口过程或回调函数的 32 位值

下面是用 Visual C++的应用程序向导生成的一段 WinMain 函数代码。

```
int APIENTRY WinMain(HINSTANCE hInstance,
                     HINSTANCE hPrevInstance,
                     LPSTR     lpCmdLine,
                     int       nCmdShow)
{
    // 声明变量
    MSG msg;
    HACCEL hAccelTable;
    // 初始化全局变量
    LoadString(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);
    LoadString(hInstance, IDC_TEST, szWindowClass, MAX_LOADSTRING);
    MyRegisterClass(hInstance);
    // 初始化应用程序实例
    if (!InitInstance (hInstance, nCmdShow))
    {
        return FALSE;
    }
    //调用加速键资源
    hAccelTable = LoadAccelerators(hInstance, (LPCTSTR)IDC_TEST);
    // 进入消息循环
    while (GetMessage(&msg, NULL, 0, 0))
    {
        if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
        {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
        }
    }
    return msg.wParam;
}
```

2. MFC

MFC 的全称是 Microsoft Fundation Classes，即微软基础类库，其本质是一个包含许多对象的类库。程序员在进行程序设计时，如果发现类库中某个对象能完成所需要的功能，则只要简单的调用该对象的方法，就可以实现相应的功能。基于 MFC 的编程方法充分利用了面向对象技术的优点，这使得程序员在编程时极少需要关心对象方法的实现细节，从而使应用程序的开发变得简单。

Visual C++中的应用程序向导 AppWizard 会把几个类结合起来，构成一个应用程序框架，这是一种比 Windows SDK 更为简单的编程方法。基于 MFC 的应用程序框架中定义了