普通高等教育“十一五”国家级规划教材配套参考书

Visual C++ 面向对象与可视化程序设计实例解析（第4版）

编著　黄维通　解　辉

高等教育出版社

普通高等教育 材配套参考书

Visual C++面向对象与可视化程序设计实例解析

Visual C++ Mianxiang Duixiang yu Keshihua Chengxu Sheji Shili Jiexi

（第4版）

编著 黄维通 解 辉

高等教育出版社·北京

内容提要

本书是《Visual C++面向对象与可视化程序设计》（第4版）的配套实例解析，全面具体地对主教材中各章的习题做了详细解答，并进行了必要的分析和代码注释，同时，还增加了部分紧扣相关知识点的典型实例，力求通过实例让读者全面掌握面向对象与可视化程序设计的思路和开发技巧。

本书不仅适合作为高等院校理工科学生学习Visual C++面向对象编程的辅助教材，还适合初步掌握Visual C++的编程人员作为习题教材，同时也可供有关科研及开发人员参考。

图书在版编目（CIP）数据

Visual C++面向对象与可视化程序设计实例解析/黄维通，解辉编著. --4版. --北京：高等教育出版社，2016.8

ISBN 978-7-04-045985-2

Ⅰ. ①V… Ⅱ. ①黄… ②解… Ⅲ. ①C语言-程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字（2016）第169104号

策划编辑 刘 茜　　责任编辑 刘 茜　　封面设计 张 志　　版式设计 张 杰
插图绘制 杜晓丹　　责任校对 刁丽丽　　责任印制 刘思涵

出版发行 高等教育出版社
社　　址 北京市西城区德外大街4号
邮政编码 100120
印　　刷 北京丰源印刷厂
开　　本 850mm×1168mm 1/16
印　　张 9.75
字　　数 240千字
购书热线 010-58581118
咨询电话 400-810-0598

网　　址 http://www.hep.edu.cn
http://www.hep.com.cn
网上订购 http://www.hepmall.com.cn
http://www.hepmall.com
http://www.hepmall.cn
版　　次 2001年6月第1版
2016年8月第4版
印　　次 2016年8月第1次印刷
定　　价 18.00元

物 料 号 45985-00

○ 前　言

随着程序设计与开发技术的不断发展，在大学计算机基础课程中，讲授面向对象技术及开发过程的可视化，已经成为计算机基础课程教学改革发展的方向之一。掌握面向对象的程序设计，是大学生信息素养和能力结构的重要组成部分，也是社会对人才的计算机应用与开发水平的要求。

学习本教材，要求先修C程序设计语言（或其他任何一种编程语言），虽然C语言已经成为高校理工科学生的必修或选修课程，是很重要的程序设计的入门基础课程，但它是面向过程的编程语言。而面向对象的编程技术已经成为当今软件开发的主流技术，因此，掌握“面向对象与可视化程序设计”的技术与方法，是大学生运用信息技术解决本学科计算问题的基本能力之一。

本教材作为《Visual C++面向对象与可视化程序设计》（第4版）的配套实例解析，在内容上力求详尽，尤其是编程开发的实例，附有详细的代码注释，同时对开发步骤也给予详细介绍，关键步骤配有过程图例。本教材的所有实例均在Visual Studio 2012环境上调试通过。

本教材所关联的主教材主要分为4个部分。第一部分讲述VC++的基础知识；第二部分介绍Windows编程构架及部分专题应用，包括Windows绘图、文本输入/输出、键盘与鼠标的应用以及资源的应用等基础知识；第三部分介绍MFC构架（包括类库的基本知识）、各种常用类在编程中的应用、常用控件的应用、利用Visual C++的资源编辑器编写资源文件及其应用、文档操作等知识点；第四部分介绍了高级编程应用，如多媒体、数据库的基本概念与方法及相关应用案例。

本书面向高等院校本科生、研究生及从事计算机软件开发的专业人员，既适合作为高等学历教育的教材，也适合作为非学历教育的各类培训教材，同时可供计算机爱好者自学参考。

本书由黄维通、解辉编著。解辉编写了第1~2章，并审核了所有例题。黄维通编写了第3~12章。由于作者水平有限，缺点和错误在所难免，恳请读者批评指正。

谢谢喜欢阅读本书的读者！

作者联系信箱：huangwt@ tsinghua. edu. cn。

黄维通

2016年于清华园

○目　　录

第 1 章　C++基础知识 …… 001
第 2 章　Windows 应用程序基础 …… 005
第 3 章　Windows 的图形设备接口及 Windows 绘图 …… 011
第 4 章　字体及其应用 …… 033
第 5 章　Windows 应用程序对键盘与鼠标的消息响应 …… 043
第 6 章　资源在 Windows 编程中的应用 …… 053
第 7 章　MFC 基础知识 …… 075
第 8 章　控件在可视化编程中的应用 …… 079
第 9 章　单文档与多文档的应用 …… 101
第 10 章　在 MFC 中创建应用程序的资源 …… 107
第 11 章　多媒体应用程序的设计 …… 121
第 12 章　数据库应用程序的开发 …… 133

第 1 章
C++基础知识

【1-1】C++语言中函数的作用是什么?

解答：在面向对象的程序设计中，函数有非常重要的作用，它是模块划分的基本单位，是程序设计中的基本抽象单元，是对功能的抽象。一个复杂的系统往往需要划分为若干个子系统，然后分别进行开发和调试。通常将相对独立的、经常使用的功能抽象为函数，使用函数时可以只考虑函数的功能和使用方法而不必关心它的具体实现。这样有利于代码的重用，可以提高程序的开发效率、增强程序的可靠性，方便程序的维护。

【1-2】在C++语言中如何进行“类”的定义?

解答：类是C++语言的精华，是进行封装和数据隐藏的工具。通过它把逻辑上相关的实体联系起来，并具备从外部对这些实体进行访问的手段。定义一个类的语法格式如下：

```
Class 类名:基类名
{
 private:
    私有成员数据及函数;
 protected:
    保护成员数据及函数;
 public:
    公共成员数据及函数;
}类的对象声明;
```

【1-3】构造函数和析构函数的功能是什么?如何创建构造函数和析构函数?

解答：构造函数是一个特殊的成员函数，它主要用来为对象分配内存空间，对类的成员进行初始化并执行对象的其他内部管理操作。构造函数的特点是它的名字同类名相同，当定义该类的对象时，自动调用该函数完成对该对象的初始化操作，构造函数可以重载。构造函数的创建和其他的函数基本相同，但有以下特点：无返回值；可以带参数也可以不带参数。在实际的应用中如果没有给类定义构造函数，则系统自动生成一个缺省的构造函数，该函数没有参数，只是简单地把对象中的每个实例变量初始化为0。例如：

```
Class My_Class
{
      float a,b;
public:
      My_Class(…)                    //构造函数,可以带参数也可以不带参数
      {
      …                              //函数体
      }
```

```
…
}
```

析构函数也是类中的特殊成员函数，与定义它的类具有相同的名字，但要在前面加上一个波浪号（~）。析构函数没有参数也没有返回值，它不能重载，因此一个类中只能有一个析构函数。析构函数执行与构造函数相反的操作，通常用来释放分配给对象的存储空间。当程序超出对象的作用域时，或者当对一个类指针进行 delete 操作时，系统将自动调用析构函数。例如：

```
Class My_Class
{
  float a,b;
  public:
      My_Class(…)              //构造函数,可以带参数也可以不带参数
      {
        …                      //函数体
      }
      ~My_Class()              //析构函数,不可以带参数
      {
        …                      //函数体
      }
   …
}
```

【1-4】C++语言中“派生”的含义是什么？

解答：“派生”是指在保持原有类特性的基础上，进行更具体、更详细的类的定义。派生类从基类中继承所有的公共部分并可以增加数据成员和成员函数，对原有基类的属性和方法进行修改和扩充。类的继承和派生的层次结构，可以说是人们对自然界中事物进行分类分析的认识过程在程序设计中的体现。例如：一个交通工具类，其中包含了关于交通工具的共有信息，现在用它来描述汽车显然是不够的，因为汽车有属于自己的特殊的信息，如果从头来定义汽车类，那么有许多代码显然是浪费的。如果把汽车类定义为交通工具类的派生类，自动继承了交通类的信息，则只编写描述汽车特殊属性的代码即可，这样大大提高了工作效率和代码效率。

【1-5】C++语言中重载的作用是什么？

解答：重载是 C++语言的一个重要特性，它包含函数重载和操作符重载。所谓函数重载是指同一个函数体对应着多个函数的实现，即允许同一个程序中声明多个名称相同的函数，这些函数分别完成不同的功能。使用函数重载可以减轻用户的记忆负担，并使程序的结构简单、易懂。操作符重载是指将语言中已有的操作符赋予新的功能，但与该操作符的本来含义不冲突，同样也达到了减轻程序员的记忆负担和使程序结构简单易懂的效果。

第 2 章
Windows 应用程序基础

【2-1】Windows 编程中窗口的含义是什么?

解答：窗口是 Windows 应用程序中的基本操作单元，是应用程序和用户之间交互的接口环境，也是系统管理应用程序的基本单位。编写一个 Windows 应用程序首先应创建一个或多个窗口，随后应用程序的运行过程即是窗口内部、窗口与窗口之间、窗口与系统之间进行数据处理和数据交换的过程。

【2-2】事件驱动的特点是什么?

解答：Windows 操作是建立在事件驱动执行程序基础上，与利用自顶向下设计和执行的传统过程式编程方式完全不同。在使用面向过程的编程方式时，程序开发者是处于完全支配状态，程序执行的流程遵从程序员预先规定的路径。而面向对象的 Windows 程序的执行顺序则取决于事件和消息的发生顺序。所谓消息是描述事件发生的信息，当单击一个按钮时，系统就产生一条特定的消息，表示此按钮事件的发生。程序的执行顺序是由顺序产生的消息驱动的，所以面向对象的 Windows 程序的设计重点是编写事件和消息的处理程序。程序员可以根据消息类型编写程序以处理接受的消息，或者发出其他消息以驱动其他程序，但是不必预先确定消息产生的次序。事件驱动的这些特点对于编写交互式的程序很有用处。

【2-3】Windows 应用程序中的消息传递是如何进行的?

解答：Windows 为应用程序提供称为消息队列的保留区，由 Windows 以及其他应用程序向用户的应用程序发送的全部消息都存储在此队列里，等待调用。

Windows 应用程序的运行以消息为核心。Windows 将产生的消息放入应用程序的消息队列中，而应用程序的 WinMain 函数从消息循环提取队列中的消息，并将其传递给窗口函数的相应过程处理。

消息循环的常见格式如下：

```
MSG Msg;
...
while(GetMessage(&Msg,NULL,0,0))
{
    TranslateMessage(&Msg);
    DispatchMessage(&Msg);
}
```

其中 GetMessage 函数的作用是从消息队列中读取一条消息，并将消息放在一个 MSG 结构中。其形式为：

```
GetMessage
(
    lpMSG,                  //指向 MSG 结构的指针
    hwnd,
    nMsgFilterMin,          //用于消息检索的最小消息号值
```

```
    nMsgFilterMax          //用于消息检索的最大消息号值
)
```

通过设置参数 nMsgFilterMin 和 nMsgFilterMax 可实现消息的过滤，即仅处理所确定的消息号范围内的消息。如果两个参数都为 0，则不过滤消息。

TranslateMessage 函数负责将消息的虚拟键转换为字符信息，其形式为：

TranslateMessage(lpMSG)

DispatchMessage 函数将参数 lpMSG 指向的消息传送到指定窗口函数，其形式为：

DispatchMessage(lpMSG)

当 GetMessage 函数返回零值，即检索到 WM_QUIT 消息时，程序将结束循环并退出。SDK 程序以特定的循环调用 Windows 消息，称为消息循环，如下代码表示此循环一直运行直到程序接受终止执行消息时才停止。

在 while()循环内部，由循环的每次迭代调用 Windows API 函数 GetMessage(&Msg, NULL, 0, 0) 以得到消息队列中的下一条消息，并把它存在特定的 MSG 结构体变量 msg 中。TranslateMessage(&Msg)实现键盘上某个键的翻译。DispatchMessage(&Msg)函数向窗口的消息处理函数发送消息，调用相应的消息处理函数。完成后进入下一个循环。

例如：单击一下鼠标的左键，系统发送一条 WM_LBUTTONDOWN 消息到该程序的消息队列，在消息循环中由 GetMessage(&Msg, NULL, 0, 0)函数得到此消息并把它存储在 MSG 结构体变量 msg 中，然后调用 TranslateMessage(&Msg)函数。如果不是键盘消息，则此函数不做任何处理，接着调用 DispatchMessage(&Msg)函数将此消息发送至消息处理函数 WndProc。在消息处理函数中执行相应的消息处理程序，然后进入到下一个循环。

【2-4】句柄是什么？有何作用？

解答：句柄（HANDLE）是 Windows 编程的重要基础，一个句柄是指 Windows 使用的一个唯一的 PVOID 型的数据，是一个 4 字节长的数值，在 WinNT.h 中对 HANDLE 进行了定义“typedef PVOID HANDLE”（而 PVOID 是指向任意数据类型的指针，同样在 WinNT.h 中进行了定义：“typedef void * PVOID;”）。

句柄的作用是用于标识应用程序中不同的对象和同类对象中不同的实例，例如一个窗口、按钮、图标、滚动条、输出设备等都是不同对象，而一个应用程序的界面中可能有多个按钮，这一系列按钮就是按钮对象的不同实例。应用程序通过句柄能够访问相应的对象信息。

【2-5】Windows 应用程序最基本的构成应有哪些部分？

解答：Windows 应用程序具有相对固定的基本结构，由 WinMain(…)函数和 WndProc(…)函数构成基本框架。

WinMain(…)函数是所有 Windows 函数的入口，类似 C 语言的 main( )函数，其功能是完成一系列的定义和初始化工作，并产生消息循环。消息循环是整个程序运行的核心。

WinMain(…)函数主要由以下几个部分组成：

- 定义并注册窗口类。
- 建立窗口。
- 产生消息循环。

窗口函数 WndProc( )定义了应用程序对接收到的不同消息的响应，其中包含了应用程序对各种可能接收到的消息的处理过程，是消息处理分支控制语句的集合。通常，窗口函数由一个或多个 switch…case 语句构成，每一个 case 语句对应一种消息的代码处理模块，当应用程序接收到一个消息时，相应的 case 语句被激活并执行相应的模块。

窗口函数的一般形式如下：

```
LRESULT CALLBACK WndProc(HWND hwnd,UINT message,WPARAM wParam,
                         LPARAM lParam)
{
  …
  switch(message)              //message 为标识消息的消息步
    {
     case …
          …
          break;
     …
     case WM_DESTROY:
          PostQuitMessage(0);
     default:
          return DefWindowProc(hwnd,message,wParam,lParam);
    }
return(0);
}
```

窗口函数的主体是消息处理语句，由一系列 case 语句组成。程序员只需根据窗口可能收到的消息在 case 语句中编写相应的处理程序段即可。

在 case 语句的消息处理程序段中一般都有对消息 WM_DESTROY 的处理。当窗口消息处理函数 WindowProc 收到这条消息后，最需要做的一件事情就是调用 PostQuitMessage 发出退出消息，让消息循环结束。

PostQuitMessage 函数的作用是向应用程序发出 WM_QUIT 消息，请求退出，其退出码的值会作为 WM_QUIT 消息的 wParam 参数。除此之外，应用程序通过在消息处理程序段中加入如下语句，为未定义处理过程的消息提供默认处理：

```
default: return DefWindowProc(hwnd,message,wParam,lParam);
```

函数 DefWindowProc 是系统默认的处理过程，以保证所有发送到该窗口的消息均得以处理。

【2-6】Windows 应用程序常用的消息有哪些?

解答：Windows 应用程序常用的消息如下：

（1）WM_KEYDOWN：按下一个非系统键时产生的消息。

（2）WM_CHAR：按下一个非系统键时产生的消息。附加信息参数 wParam 为按键的 ASCII 码，lParam 记录了按键的重复次数、扫描码、转移代码、先前键的状态等信息。

（3）WM_CREATE：此消息由 CreateWindow 函数发出。附加信息参数 wParam 未用，lParam 包含一个指向 CREATESTRUCT 数据结构的指针。如果一个程序处理了这个消息，应当返回值为 0 时，表示窗口正常创建，如果为-1，则窗口创建失败。

（4）WM_CLOSE：关闭窗口时产生此消息。附加信息参数 wParam 和 lParam 均未用。

（5）WM_DESTROY：消除窗口时由 DestroyWindow 函数发出此消息。附加信息参数 wParam 和 lParam 均未用。

（6）WM_QUIT：退出应用程序时由 PostQuitMessage 函数发出的消息。附加信息参数 wParam 含有退出代码。

（7）WM_PAINT：当发生用户区移动或显示事件、用户窗口改变大小的事件、程序通过滚动条滚动窗口时，均产生一条 WM_PAINT 消息，即发出刷新请求。此外，当下拉式菜单关闭并需要恢复被覆盖的部分以及 Windows 清除对话框或消息框等对象，并需要恢复被覆盖的部分时，将产生 WM_PAINT 消息。

第 3 章
Windows 的图形设备接口及 Windows 绘图

【3-1】什么是图形设备接口?

解答：Windows 应用程序使用图形设备接口和 Windows 设备驱动程序来支持与设备无关的图形。图形设备接口（GDI）是 Windows 系统的重要组成部分，负责系统与用户或绘图程序之间的信息交换，并控制在输出设备上显示图形或文字。GDI 的设备无关性是 Windows 操作系统的特色之一。对于开发人员而言，所做的工作仅仅是在系统的帮助下建立一个与某个实际输出设备的关联，以要求系统加载相应的设备驱动程序，然后调用固定的 GDI 函数进行输出即可，其他的操作由系统完成。

【3-2】如何进行图形的刷新?

解答：图形刷新包括刷新的请求，系统对刷新请求的响应以及具体的刷新方法。

（1）刷新请求

当发生窗口大小的调整、窗口的移动或窗口被其他对象覆盖后，都必须刷新新窗口用户区的内容，以恢复用户区内应有的显示形态。但是 Windows 系统并不总是记录窗口中需保存的内容，系统只能在有限的几种情况下自动刷新。因此，应用程序必须具有及时处理刷新请求和刷新图形的功能。Windows 系统通常发送 WM_PAINT 消息将刷新请求传递给应用程序。

（2）系统对刷新请求的响应

刷新有 3 种可能，分别是窗口移动后的刷新、被覆盖区域的刷新以及对象穿越后的刷新。系统对这 3 种刷新提供了如下相应的处理方法。

- 窗口移动后的刷新：系统发送 WM_PAINT 消息，由消息处理函数完成刷新。
- 被覆盖区域的刷新：Windows 系统试图保存被覆盖区域的副本，以备以后刷新，如果不能有效刷新，则向应用程序发送 WM_PAINT 消息。
- 对象穿越后的刷新：此时系统自动完成刷新任务，应用程序不用考虑。

【3-3】如何获取绘图工具的句柄?

解答：常见的绘图工具主要有画笔和画刷。获取绘图工具的句柄就是获取画笔和画刷的句柄。

有两种方法能获得画笔句柄：

（1）调用函数 GetStockObject(…)获得系统定义的 4 种画笔：WHITE_PEN，BLACK_PEN，DC_PEN 和 NULL_PEN。

（2）调用函数 CreatePen(…)由用户创建画笔。

GetStockObject(…)和 CreatePen(…)这两个函数的返回值都是画笔句柄。

有 3 种方法获得画刷句柄：

（1）调用函数 GetStockObject(…)获得系统定义的 7 种画刷。

（2）调用 CreateSolidBrush(…)定义具有指定颜色的单色画刷。

（3）调用函数 CreateHatchBrush(…)定义具有指定阴影图案和颜色的画刷。

上述这 3 个函数的返回值都是画刷句柄。