面向21世纪高等院校计算机教材系列

Visual C++ 程序设计教程习题及习题解答

孟 威 黄维通 编著

机械工业出版社
China Machine Press

面向21世纪高等院校计算机教材系列

Visual C++程序设计教程习题及习题解答

孟　威　黄维通　编著

机械工业出版社

本书是《Visual C++ 程序设计教程》的配套习题及其解答，该书对原教材中的习题进行了详细的解题编程步骤的讲解，内容涉及可视化编程过程中常用的 API 函数及 MFC 类库。希望通过对习题的讲解，帮助读者进一步加深对 Visual C++ 编程方法的理解，并提高编程开发水平。

本书既可以作为高等学校计算机软件技术课程的辅助教材，也适于有关科研及应用开发人员作为参考，同时也可供从事计算机软件开发的专业人员使用。

图书在版编目（CIP）数据

Visual C++ 程序设计教程习题及习题解答/孟威，黄维通编著. —北京：机械工业出版社，2002.3

面向 21 世纪高等院校计算机教材系列

ISBN 7-111-09914-1

Ⅰ.V... Ⅱ.①孟... ②黄... Ⅲ.C 语言-程序设计-高等学校-解题 Ⅳ.TP312-44

中国版本图书馆 CIP 数据核字（2002）第 009472 号

机械工业出版社(北京市百万庄大街 22 号 邮政编码 100037)

策　　划：胡毓坚

责任编辑：田　梅

责任印制：付方敏

北京铭成印刷有限公司印刷·新华书店北京发行所发行

2002 年 3 月第 1 版·第 1 次印刷

787mm × 1092mm 1/16·9.75 印张·234 千字

0 001—5000 册

定价：15.00 元

出版说明

随着计算机技术的飞速发展，计算机在经济与社会发展中的地位日益重要。在高等院校的培养目标中，都将计算机知识与应用能力作为其重要的组成部分。为此，国家教育部根据高等院校非计算机专业的计算机培养目标，提出了“计算机文化基础”、“计算机技术基础”和“计算机应用基础”三个层次教育的课程体系。根据计算机科学发展迅速的学科特点，计算机教育应面向社会，面向潮流，与社会接轨，与时代同行。随着计算机软硬件的不断更新换代，计算机教学内容也必须随之不断更新。

为满足高等院校计算机教材的需求，机械工业出版社聘请了清华大学、北方交通大学、北京邮电大学等院校的老师，经过反复研讨，结合当前计算机发展需要和编者长期从事计算机教学的经验精心编写出“面向21世纪高等院校计算机教材”。

本套教材理论教学和实践教学相结合，内容实用、层次分明、讲解清晰、系统全面，其中溶入了老师大量的教学经验，是各类高等院校、高等职业学校及相关院校的最佳教材，也可作为培训班教材和自学教材。

前　言

目前，面向过程的C语言已经成为高校理工科学生的必修或选修课程，但随着软件工程技术的不断发展，应用面向对象的编程技术已经成为当今软件开发的重要手段之一，尤其是Visual C++（简称 V C++）的出现，大大推进了面向对象与可视化编程技术的应用与发展。因此，掌握“面向对象与可视化程序设计”的思想与方法已经成为对大学生应用与开发能力的要求之一。

本书分API专题和MFC专题对 V C++ 编程方法及其体系结构进行介绍。在API专题中，通过具体的实例介绍消息响应的机制及其对消息的响应，内容包括读者在开发应用程序中将经常接触到的包括 Windows 绘图、文本输入/输出、资源的应用及一系列标准控件的使用等知识；在MFC专题部分主要介绍应用于MFC进行可视化编程的思想方法，包括各种类在编程中的等知识点。本书作为面向21世纪高等院校计算机技术的教材，体现了计算机软件技术课程改革的方向之一。本课程建议授课时为32小时，并要求先修C语言课程。

本书特点是从面向对象的基本概念出发，讲述可视化程序设计的思想与方法。对每一部分的知识点、概念、难点，都力求以较精炼的语言进行讲解，同时，对每一个知识点都配以必要的实例，实例中配以较为详细的步骤说明及语法说明（因此本书也适合自学），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”的思路和开发技巧与体系，本书的例子都是根据教学特点精心编排，且所有的例题都在 Windows 98 及 V C++ 6.0 的环境下调试运行通过的。

参加本书编写、程序调试工作的同志还有刘嘉、程楠、张伟浩、马骏锋、李阳、胡定涵、张国华、王增宏、杜春青、李静等，在此表示衷心感谢！

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。

目　录

第 1 章　Windows 应用程序

1.1　Win32 应用程序框架的基本组成。

解答：Win32 应用程序框架主要由“初始化窗口类”、“窗口类的注册”、“窗口的创建”以及“窗口消息响应函数”等组成。典型代码如下：

```
#include <windows.h>
LRESULT CALLBACK WndProc(HWND,UINT,WPARAM,LPARAM);//窗口函数说明
//-------------------以下初始化窗口类 --------------------------
int WINAPI WinMain(     HINSTANCE   hInstance,          //WinMain 函数说明
                        HINSTANCE   hPrevInst,
                        LPSTR       lpszCmdLine,
                        int         nCmdShow)
{ HWND hwnd;
  MSG Msg;
  WNDCLASS wndclass;
  char lpszClassName[] = "窗口";                        //窗口类名
  char lpszTitle[] = "窗口示例程序";                    //窗口标题名
  //窗口类的定义
  wndclass.style = 0;                                   //窗口类型为缺省类型
  wndclass.lpfnWndProc = WndProc;                       //窗口处理函数为 WndProc
  wndclass.cbClsExtra = 0;                              //窗口类无扩展
  wndclass.cbWndExtra = 0;                              //窗口实例无扩展
  wndclass.hInstance = hInstance;                       //当前实例句柄
  wndclass.hIcon = LoadIcon(NULL,IDI_APPLICATION);      //使用缺省图标
  wndclass.hCursor = LoadCursor(NULL,IDC_ARROW);        //窗口采用箭头光标
  wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);//窗口背景为白色
  wndclass.lpszMenuName = NULL;                         //窗口中无菜单
  wndclass.lpszClassName = lpszClassName;               //窗口类名为"窗口示例"
  //---------------以下进行窗口类的注册 ----------------
  if(!RegisterClass(&wndclass))                         //若注册失败则发出警告声音
  { MessageBeep(0);
    return FALSE;
  }
  //创建窗口操作
  hwnd = CreateWindow(lpszClassName,                    //窗口类名
                      lpszTitle,                        //窗口实例的标题名
                      WS_OVERLAPPEDWINDOW,              //窗口的风格
                      CW_USEDEFAULT,CW_USEDEFAULT,      //窗口左上角坐标为缺省值
```

```
                    CW_USEDEFAULT,CW_USEDEFAULT,  //窗口的高和宽为缺省值
                    NULL,                         //此窗口无父窗口
                    NULL,                         //此窗口无主菜单
                    hInstance,                    //应用程序的当前句柄
                    NULL) ;                       //不使用该值
  ShowWindow( hwnd, nCmdShow) ;                   //显示窗口
  UpdateWindow(hwnd);                             //绘制用户区
  while( GetMessage(&Msg, NULL, 0, 0))            //消息循环
  { TranslateMessage( &Msg) ;
    DispatchMessage( &Msg) ;
  }
  return Msg.wParam;                              //程序终止时将信息返回系统
}

//窗口函数
LRESULT CALLBACK WndProc(HWND hwnd,UINT message,WPARAM wParam,LPARAM lParam)
{ switch(message)
  {
    case WM_DESTROY:
        PostQuitMessage(0);                       //调用该函数发出 WM_QUIT 消息
    default:                                      //缺省消息处理函数
        return DefWindowProc(hwnd,message,wParam,lParam);
    }
     return (0);
}
```

1.2 编写 Win32 应用程序的基本步骤。

解答:

1. 编写 WinMain 函数

WinMain 函数是所有 Windows 应用程序的入口,类似 C 语言中的 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 函数实现以下功能:

- 注册窗口类,建立窗口及执行其他必要的初始化工作;
- 进入消息循环,根据从应用程序消息队列接受的消息,调用相应的处理过程;
- 当消息循环检索到 WM_QUIT 消息时终止程序运行。

WinMain 函数有三个基本的组成部分:函数说明、初始化和消息循环。

(1) 函数说明

WinMain 函数的说明如下:

```
int WINAPI WinMain
   ( HINSTANCE hThisInst,       //应用程序当前实例句柄
     HINSTANCe hPrevInst,       //应用程序其他实例句柄
```

```
    LPSTR lpszCmdLine,          //指向程序命令行参数的指针
    Int nCmdShow                //应用程序开始执行时窗口显示方式的整数值标识
)
```

由于 Windows 操作系统是多任务的操作系统，能进行多任务管理，因此，Windows 应用程序可能被并行地多次执行，因而可能出现同一个应用程序的多个窗口同时存在的情况，Windows 系统将应用程序每一次的执行称为该应用程序的一个实例(instance)，并使用一个实例句柄来唯一地标识它。

(2) 初始化

初始化包括窗口类的定义、注册、创建窗口实例和显示窗口四部分。

1) 窗口类定义

在 Windows 应用程序中，窗口类定义了窗口的形式与功能。窗口类定义通过给窗口类数据结构 WNDCLASS 赋值完成，该数据结构中包含窗口类的各种属性，在窗口类定义过程中常用到以下函数：

● LoadCursor 函数，其作用是在应用程序中加载一个窗口光标。

LoadCursor 函数的原型为：

```
HCURSOR LoadCursor
    (HINSTANCE hInstance,     //光标资源所在的模块句柄，为 NULL 则使用系统预定义光标
     LPCTSTR lpCursorName     //光标资源名或系统预定义光标标识名
    )
```

● LoadIcon 函数，其作用是在应用程序中加载一个窗口图标。

LoadIcon 函数的原型为：

```
HICON LoadIcon
    (HINSTANCE hInstance,     //图标资源所在的模块句柄，为 NULL 则使用系统预定义图标
     LPCTSTR lpIconName       //图标资源名或系统预定义图标标识名
    )
```

● GetStockObject 函数，其作用是获取已经定义的画笔、画刷、字体等对象的句柄。

GetStockObject 函数的原型为：

```
HGDIOBJ GetStockObject(int fnObject); //fnObject 为对象的标识名
```

2)注册窗口类

Windows 系统本身提供部分预定义的窗口类，程序员也可以自定义窗口类，窗口类必须先注册后使用。窗口类的注册由注册函数 RegisterClass()实现。其形式为：

```
RegisterClass( & wndclass);     //wndclass 为窗口类结构
```

RegisterClass 函数的返回为布尔值，注册成功则返回真

3) 创建窗口示例

创建一个窗口类的实例由函数 CreateWindow()实现，该函数的原型为：

```
HWND Create Window
            (LPCTSTR lpszClassName,          //窗口类名
```

```
    LPCTSTR lpszTitle,          //窗口标题名
    DWORD dwStyle,              //创建窗口的样式
    int X,                      //窗口左上角坐标
    int y,                      //窗口左上角坐标
    int nWidth,                 //窗口宽度
    int nHeight,                //窗口高度
    HWND hwndParent,            //该窗口的父窗口句柄
    HWENU hMenu,                //窗口主菜单句柄
    HINSTANCE hInstance,        //创建窗口的应用程序当前句柄
    LPVOID lpParam              //指向一个传递给窗口的参数值的指针
)
```

4) 显示窗口

窗口类的显示由 ShowWindow 和 UpdateWindow 函数实现。应用程序调用 ShowWindow 函数在屏幕上显示窗口,其形式为:

```
ShowWindow(hwnd,nCmdshow);
```

显示窗口后,应用程序常常调用 UpdateWindow 函数更新并绘制用户区,并发出 WM_PAINT 消息,如果更新区域是空的,就不发送任何消息。UpdateWindow 函数的形式为:

```
UpdateWindow(hwnd);
```

(3) 消息循环

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 WinMain 函数的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);
    }
```

其中函数 GetMessage 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,            //用于消息过滤的最小消息号值
  nMsgFilterMax             //用于消息过滤的最大消息号值
```

值得注意的是，GetMessage 函数中的参数 nMsgFilterMin 和 nMsgFilterMax 可实现对消息的过滤，即程序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似键转换为字符信息，其形式为：

```
TranslateMessage(lpMSG)
```

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

```
DispatchMessgae(lpMSG)
```

当 GetMessgae 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。

2. 编写窗口消息处理函数

窗口消息处理函数(以下简称窗口函数)定义了应用程序对接收到的不同消息的响应，它包含了应用程序对各种可能接收到的消息的处理过程。通常，窗口函数由一个或多个 switch...case 语句组成，每一条 case 语句对应一种消息，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响应程序模块。

窗口函数的一般形式如下：

```
LRESULT CALLBACK WndProc(HWND hwnd,UINT messgae,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，但有时用户操作时会出现误动作，也就是说发送错误的消息，这些错误的消息一般不可能在消息处理函数中出现，因此，应用程序通过在消息处理函数加入如下语句，为未定义处理过程的消息提供缺省处理：

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

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

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。如前所述，

该消息是关闭窗口时发出的。一般情况下，应用程序调用函数 PostQuitMessage 响应这条消息。PostQuitMessage 函数的原型如下：

```
void PostQuitMessage(int nExitCode)
```

其中，nExitCode 为应用程序退出代码，函数 PostQuitMessage 的作用是向应用程序发出 WM_QUIT 消息，请求退出。

1.3 事件驱动的特点是什么？

解答：Windows 程序设计围绕着事件或消息的产生驱动运行消息处理函数。Windows 程序的执行顺序取决于事件发生的顺序，程序的执行顺序是由顺序产生的消息驱动的，程序员可以针对消息类型编写消息处理程序以处理接收的消息，或者发出其他消息以驱动其他处理程序，但是不必预先确定消息产生的次序。这是面向对象编程中事件驱动的显著的特点。

事件驱动编程方法对于编写交互式程序很有用处，用这一方法编写的程序使程序避免了死板的操作模式，从而使用户能够按照自己的意愿采用灵活多变的操作形式。

1.4 简述 Windows 应用程序中的消息传递机制。

解答：VC 中存在几种系统定义的消息分类、常用的消息有窗口管理消息、初始化消息、输入消息、系统消息、剪贴板消息、系统信息、控制处理消息、控制通知消息、滚动条通知消息、非用户区消息、文档界面消息、DDE(动态数据交换)消息、应用程序自定义消息等。应用程序发送的消息发送至消息队列，系统根据消息到达的顺序对消息进行处理，并调用响应的消息处理模块代码。

第2章　GDI及其应用

2.1　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

刷新有三种可能,它们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。

系统对上述三种刷新提供了相应的方法:

- 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。

2.2　如何定义映射模式?

解答:映射模式是设备描述表的内容之一,其优点是程序员可以不必考虑输出设备的坐标系情况,在一个统一的设备坐标系中完成图形的绘制与操作,Windows 有 8 种映射模式。其中 MM_TEXT 映射模式得到了普遍的应用,是缺省的映射模式,其特点是逻辑坐标和物理坐标都以像素为单位。MM_ISOTROPIC 和 MM_ANISOTROPIC 这两种映射模式通过将图形从程序员定义的逻辑设备窗口映射到物理设备的视口以实现坐标转换。窗口是对应逻辑坐标系上程序员设定的一个区域,视口是对应于实际输出设备上程序员设定的一个区域。定义了窗口和视口后,Windows 系统即可按照窗口和视口的坐标比例自动调整图形。注意:MM_ISOTROPIC 映射模式要求 X 和 Y 方向的映射比例相同,这种要求可能导致系统强制变换视口。

应用程序可以调用函数 GetMapMode(...)得到当前系统的映射模式,调用函数 SetMapMode(...)设置系统定义的 8 种映射模式。然后可以调用函数 SetWindowExtEx(...)设置窗口区域大小,调用函数 SetViewportExtEx(...)设置视口的大小。视口和窗口的缺省原点均为(0,0)。还可以调用函数 SetWindowOrgEx(...)和函数 SetViewportOrgEx(...)来设定窗口和视口的原点。注意最后这两个函数只有在 MM_ISOTROPIC 和 MM_ANISOTROPIC 这两种模式下才有用。定义了映射模式后我们就可以在窗口上绘制相应的图形了。

2.3　请编写程序,要求如下:

(1) 定义一只红色的画笔,绘制一个等边五边形;

(2) 用不同颜色的线条连接互不相邻的两个点；

(3) 用不同颜色的画刷填充用上述方法所形成的图形中的每一个区域。

解答：经分析可以知道本题最后需要绘制的是这样一个图形，如图 2-1 所示。

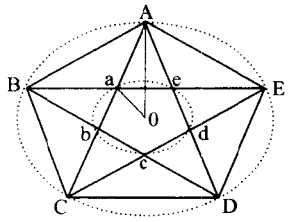

图 2-1　五边形及五角星的填充图案

(1) 首先要生成图中各点的坐标

定义五边形 ABCDE 和 abcde，其外接圆半径分别为 dfRadious0 和 dfRadious1，假设 dfRadious0 = 100，根据几何关系，可以计算出 dfRadious1 的值。由于都是正多边形，利用三角形 oeA 由正弦公式可以计算得到 dfRadiosu1 的值。根据由圆心到各端点的直线夹角为 72°，可以计算得到各顶点的坐标，按照 ABCDE，abcde 的顺序赋值给点结构数组 lpOuterPoints[5]和 lpInnerPoints[5]。存储结果如表 2-1 所示。

表 2-1　各点的坐标所对应的数组元素

点	存储单元	点	存储单元
A	lpOuterPoints[0]	a	lpInnerPoints[0]
B	lpOuterPoints[1]	b	lpInnerPoints[1]
C	lpOuterPoints[2]	c	lpInnerPoints[2]
D	lpOuterPoints[3]	d	lpInnerPoints[3]
E	lpOuterPoints[4]	e	lpInnerPoints[4]

(2) 绘制五边形

在 WM_PAINT 消息处理程序中，首先将映射模式设置为 MM_ANISOTROPIC，将窗口原点移动到(-100，-100)处。这主要是为了绘图方便，不必将点坐标加上平移量平移到窗口中。调用函数 CreatePen(...)创建红色的画笔。调用函数 SelectObject(...)选入红色画笔，利用绘图函数 Polygon(...)绘制五边形 ABCDE。

(3) 绘制五角星

调用函数 MoveToEx(...)将划线的起点移动到 A 点，对五条边进行循环，在循环体中调用函数 CreatePen(RGB(0,i * 51,0))设置各边颜色。分析五角星端点坐标的规律(见表 2-1)，发现调用函数将 LineTo(...)终点依次移动到 i * 2%5 点处，即可分别画出五角星的各条边。

(4) 填充区域

这一部分比较复杂，需要填充周围的 10 个三角形和中心的正五边形 abcde。首先填充三角形，将三角形分为五组：Aae，AaB 为一组，Bba，BbC 为一组等(见表 2-2)。这样编号的目的在于可以保证三角形端点号和正五边形的端点号之间存在着很好的映射关系，便于使用一个循环一次完成填充工作。

表 2-2　三角形分组

分 组 号	三 角 形	顶点存储位置	三 角 形	顶点存储位置
1	Aae	004	AaB	001
2	Bba	110	BbC	112
3	Ccb	221	CcD	223
4	Ddc	332	DdE	334
5	Eed	443	EeA	440

三角形 Aae 的顶点存储位置是指顶点 A 存在 lpOuterPoints[0]、顶点 a 存储在 lpInnerPoints[0]、顶点 e 存储在 lpInnerPoints[4]中，所以简写为 004。从中我们可以发现很强的规律性。以组号 i 为循环变量，则每一组三角形的前两个坐标满足 i%5 的关系，而第 3 个顶点满足(i+1)%5 和(i+4)%5 的关系。由此输入下面的循环体：

```
for(i=0;i<5;i++)
{
    lpTriangle[0] = lpOuterPoints[i%5];                    //生成三角形区域的坐标
    lpTriangle[1] = lpInnerPoints[i%5];
    lpTriangle[2] = lpOuterPoints[(i+1)%5];
    hBrush = CreateSolidBrush(RGB(i*10,i*20,i*30));        //创新新画刷
    hDefBrush = SelectObject(hDC,hBrush);                  //选入新画刷,并保存系统画刷
    Polygon(hDC,lpTriangle,3);                             //画三角形区域
    lpTriangle[2] = lpInnerPoints[(i+4)%5];                //生成三角形区域的坐标
    SelectObject(hDC, hDefBrush);                          //选入画刷
    DeleteObject(hBrush);                                  //删除画刷
    hBrush = CreateSolidBrush(RGB(i*40,i*30,i*20));        //创新画刷
    hDefBrush = SelectObject(hDC,hBrush);                  //选入画刷
    Polygon(hDC,lpTriangle,3);                             //画三角形区域
    SelectObject(hDC, hDefBrush);                          //选入画刷
    DeleteObject(hBrush);                                  //删除画刷
}
```

对分成的 5 组三角形进行循环，依次将各端点坐标赋值给三角形数组 lpTriangle，数组 lpTriangle 用来存放依逆时针顺序排列的三角形的三个顶点的坐标值。调用函数 CreateSolidBrush(RGB(i*40,i*30,i*20))建立不同颜色的画刷。调用绘图函数 Polygon(...)填充三角形。

具体源程序代码如下(2_3.cpp)：

```
#include <windows.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#define Pi 3.1415926
long WINAPI WndProc(HWND hWnd,UINT iMessage,
                    UINT wParam,LONG lParam);          //消息处理函数声明
BOOL InitWindowsClass(HINSTANCE hInstance);            //初始化窗口类声明
BOOL InitWindows(HINSTANCE hInstance, int nCmdShow);   //初始化窗口声明
```

```
HWND hWndMain;

int WINAPI WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance,
                   LPSTR lpCmdLine, int nCmdShow)        //主函数
{
  MSG Message;
  if(! InitWindowsClass(hInstance))                      //初始化窗口类
      return FALSE;
  if(! InitWindows(hInstance,nCmdShow))                  //初始化窗口
      return FALSE;
  while(GetMessage(&Message,0,0,0))
  {
    TranslateMessage(&Message);                          //消息循环
    DispatchMessage(&Message);
  }
  return Message.wParam;
}

long WINAPI WndProc(HWND hWnd,UINT iMessage,
                    UINT wParam,LONG lParam)             //消息处理函数.
{
HDC hDC;                                                 //定义设备环境句柄
HBRUSH hBrush, hDefBrush;                                //定义画刷句柄
HPEN hPen, hDEfPen; s                                    //定义画笔句柄
PAINTSTRUCT PtStr;                                       //定义包含绘图信息的结构体变量
double dfRadious0 = 100.0,dfRadious1;                    //外部/内部正五边形外接圆半径
POINT lpOuterPoints[5],lpInnerPoints[5];                 //定义外、内正五边形点结构数组
POINT lpTriangle[3];                                     //定义三角形点结构数组
dfRadious1 = dfRadious0 * sin(0.1 * Pi)/sin(126.0/180 * Pi);  //根据外圆半径计算内圆半径
                                                         //计算内、外正五边形的点坐标
for(int i = 0;i < 5;i + + )
{
  lpOuterPoints[i].x = (long)(dfRadious0 * cos(i * 72.0/180 * Pi));
  lpOuterPoints[i].y = (long)(dfRadious0 * sin(i * 72.0/180 * Pi));
  lpInnerPoints[i].x = (long)(dfRadious1 * cos(i * 72.0/180 * Pi + 36.0/180 * Pi));
  lpInnerPoints[i].y = (long)(dfRadious1 * sin(i * 72.0/180 * Pi + 36.0/180 * Pi));
}

switch(iMessage)
{
case WM_PAINT:                                           //处理绘图消息
   hDC = BeginPaint(hWnd, &PtStr);                       //得到设备环境句柄
   SetMapMode(hDC, MM_ANISOTROPIC);                      //设置映射模式
   SetWindowOrgEx(hDC, - 100, - 100, NULL);              //设置坐标原点
   hPen = CreatePen(PS_SOLID,1,RGB(255,0,0));            //创新红色画笔
   hDefPen = SelectObject(hDC,hPen);                     //将画笔选入,并保存系统原来的画笔
   Polygon(hDC,lpOuterPoints,5);                         //画正五边形
   SelectObject(hDC, hDefPen);                           //恢复系统画笔
   DeleteObject(hPen); !                                 //删除画笔
```

```
    for(i=0;i<5;i++)                                    //填充正五边形的不同区域
    {
      lpTriangle[0] = lpOuterPoints[i%5];               //生成图形中的三角形区域的坐标
      lpTriangle[1] = lpInnerPoints[i%5];
      lpTriangle[2] = lpOuterPoints[(i+1)%5];
      hBrush = CreateSolidBrush(RGB(i*10,i*20,i*30));   //创新新画刷
      hDefBrush  = SelectObject(hDC,hBrush);            //选入新画刷
      Polygon(hDC,lpTriangle,3);                        //画三角形区域
      lpTriangle[2] = lpInnerPoints[(i+4)%5];           //生成图形中的三角形区域的坐标
      SelectObject(hDC, hDefBrush);                     //选入画刷
      DeleteObject(hBrush);                             //删除画刷

      hBrush = CreateSolidBrush(RGB(i*40,i*30,i*20));   //创新画刷
      hDefBrush = SelectObject(hDC,hBrush);             //选入画刷
      Polygon(hDC,lpTriangle,3);                        //画三角形区域
      SelectObject(hDC, hDefBrush);                     //选入画刷
      DeleteObject(hBrush);                             //删除画刷
    }
    hBrush = CreateSolidBrush(RGB(255,255,255));        //创新白画刷
    hDefBrush = SelectObject(hDC,hBrush);               //选入画刷
    Polygon(hDC,lpInnerPoints,5);                       //画中心的五边形
    SelectObject(hDC, hDefBrush);                       //选入画刷
    DeleteObject(hBrush);                               //删除画刷

    //用不同种颜色的画笔来绘制五角星
    MoveToEx(hDC,lpOuterPoints[0].x,lpOuterPoints[0].y,NULL);
    for(i=1;i<=5;i++)
    {
      hPen = CreatePen(PS_SOLID,1,RGB(0,i*51,0));
      hDefPen  = SelectObject(hDC,hPen);
      LineTo(hDC,lpOuterPoints[(i+2)%5].x,lpOuterPoints[(i+2)%5].y);
      SelectObject(hDC,hDefPens);
      DeleteObject(hPen);
    }

    EndPaint(hWnd,&PtStr);                              //结束绘图
    return 0;
case WM_DESTROY:                                        //处理关闭窗口信息
    PostQuitMessage(0);                                 //向应用程序发送 WM_QUIT 消息
    return 0;
default:
    return(DefWindowProc(hWnd,iMessage,wParam,lParam));
    }
}

BOOL InitWindows(HINSTANCE hInstance, int nCmdShow)     //初始化
{
 HWND hWnd;                                             //定义窗口句柄
 hWnd = CreateWindow("Polygon",                         //窗口类名
```