

万水电脑软件全攻略宝典系列

Visual C++ 6.0

全攻略宝典

倪丽娜 赵茂先 张秀娟 编著
康创团体 策划



电脑软件全攻略宝典系列

Visual C++ 6.0 全攻略宝典

倪丽娜 赵茂先 张秀娟 编著

康创团体 策划

内 容 提 要

本书从实际应用出发,以“全攻略”的方式向您提示 Visual C++ 6.0 的许多功能。本书精选许多实例,提供给您“边读、边学、边做”的高效率学习方法,使您在最短时间内掌握 Visual C++ 6.0 编程的基本方法和技巧。

本书分为四章。①第一章,主要为初学者设计,使初学者在最短时间应用 Visual C++ 6.0 建立基本程序以及处理程序中的错误的方法;②第二章,介绍 Visual C++ 6.0 界面处理方法;③第三章,全面介绍 Visual C++ 6.0 的基本控制;④第四章,介绍 Visual C++ 6.0 的高级编程技巧,包括图形图像处理 and 数据库编程技术。

本书浅显易懂,实例丰富,特别是“全攻略”的风格使得本书的实用性很强,适用面很广,适于初学者全面掌握 Visual C++ 6.0 的使用。

图书在版编目(CIP)数据

Visual C++ 6.0 全攻略宝典 / 倪丽娜, 赵茂先, 张秀娟编著. —北京: 中国水利水电出版社, 2000.5

(电脑软件全攻略宝典系列)

ISBN 7-5084-0374-6

I.V… II. ①倪… ②赵… ③张… III.C 语言-程序设计 IV.TP312

中国版本图书馆 CIP 数据核字 (2000) 第 60622 号

书 名	Visual C++ 6.0 全攻略宝典
作 者	倪丽娜 赵茂先 张秀娟 编著
策 划	康创团体
出 版、发 行	中国水利水电出版社 (北京市三里河路 6 号 100044) 网址: www.waterpub.com.cn E-mail: sale@waterpub.com.cn 电话: (010) 63202266 (总机)、68331835 (发行部)
经 售	全国各地新华书店
排 版	北京万水电子信息有限公司
印 刷	北京牛山世兴印刷厂
规 格	787×1092 毫米 16 开本 32.75 印张 747 千字
版 次	2000 年 6 月第一版 2000 年 6 月北京第一次印刷
印 数	0001—5000 册
定 价	50.00 元

凡购买我社图书,如有缺页、倒页、脱页的,本社发行部负责调换

版权所有·侵权必究

前 言

当今计算机技术迅猛发展、日新月异，这对于软件工作者是一个极大的挑战。目前在个人计算机上，Windows 是现代操作系统的一个工业标准，所以选择一个合适的 Windows 应用程序开发工具，无疑是很重要的。

Microsoft Visual C++6.0 是微软公司推出的支持 32 位操作系统的开发工具，主要用于开发 Windows98 和 Windows NT 应用程序。它提供了对 Microsoft 的最新技术的集成，提供了一个可视化的 C++ 编程模式，是 Windows 系列平台上理想的开发工具。Visual C++ 竭力推荐 Visual Studio 98 作为基于 Windows 的程序开发工具，因为与其它开发工具相比，Visual Studio 98 有以下显著的优点：

1. Windows 和 Visual C++6.0 是一个公司的不同产品。Visual C++ 是与 Windows 系统紧密联系在一起，可以说对于所有系统支持的功能都能在 Visual C++6.0 下找到其原型。

2. 提供了更为强大的 MFC 编程接口，它具备如下的功能：

- 对线程和进程的直接 MFC 支持
- 对多任务的 MFC 支持
- 对数据库应用的 MFC 支持
- 对 Windows 注册表的编程
- 对 Internet 的支持等等

3. 提供了一致的可视化界面

- 提供 ResourceView，可以方便地完成工程资源的编辑。
- 提供 ClassView，能够十分方便地查看工程的各个类的成员函数和成员变量等。
- 提供 ClassWizard，能够十分方便地添加成员函数、将成员函数与 Windows 消息相对应。

- 功能强大的调试工具，使你轻松地查找程序中的错误。
- 提供众多功能强大的控件，大大减轻程序员编程的工作量。
- 提供功能强大、信息丰富的 MSDN，使程序员容易找到自己所需的资料等等。

本书将针对 Visual C++6.0 的上述特点，详细讲述它的强大功能。本书从整体上可分为两大部分：第一部分结合面向对象思想，讲述了基于 Windows 和面向对象技术的编程要点和编程模式以及一些基本的 Windows 编程如菜单、工具条、状态条和对话框，而后讲述了 Windows 的基本控制，包括静态文本控件、按钮、编辑控件、分组控制等等。这一部分不仅适合于新用户学习 Visual C++6.0，对于熟悉 Visual C++ 6.0 的读者也有很大的参考价值。第二部分主要针对 Visual C++6.0 的功能支持，详细介绍了如何用 Visual C++6.0 来开发 Windows 高级应用程序。在这一部分我们讲述了 Windows 的图形图像处理，包括现在流行的 OpenGL 程序设计。另外我们还讲述了有关数据库的程序设计方法和快速创建

Windows 标准对话框的途径。这些内容对于长期从事 Windows 应用程序开发的读者来说具有很高的阅读价值。

本书的最后给出了 Microsoft 基本类 (MFC) 库中常用的消息映射函数、C++ 语言快速进阶和本书用到的全功能索引等内容。这是十分有用的参考资料, 如果作为第一手查阅资料可以大大方便编程, 如果读者有机会接触 Internet 网, 到作者花费大量时间查阅的有关 Visual C++ 的网址去看看, 这些网址中提供了大量的源代码和说明, 对于程序员来讲无疑是十分重要的。

本书作为一本使用和开发指南, 通过攻略式的方式介绍 Visual C++6.0 的各项功能, 并对程序员经常碰到的问题以注意事项的形式给予明确的说明, 目的是使用户尽快掌握 Visual C++6.0 的使用方法、使用技巧和编程技术。对于每一种任务, 我们都用两种以上的攻略加以实现, 力求使读者从多方面、多角度地灵活掌握 Visual C++6.0 的内容。

本书内容新颖, 技术全面, 实用性强, 适用于从事开发 Windows 应用程序的软件开发人员, 也可以作为大中专院校师生的参考书。参加本书编写的有山东科技大学的倪丽娜、赵茂先、张秀娟、张金泉以及张大军、丁玮、刘鹏等人。

由于时间紧迫和个人的能力有限, 本书中不可避免地存在一些错误和缺点, 恳请大家批评指正, 共同学习。

编者

2000 年 2 月

全书导读

本书内容简介

第一章 Visual C++6.0 基础

本章是为初学者设计的，我们将向您展示 Visual C++ 6.0 的工作界面。您只要用鼠标单击几个按钮，再输入工程名就可以得到一个可以运行的程序。不过，此时程序还不能做实质性的工作。紧接着，介绍了向程序中添加代码以完成相应工作的方法。最后，详细介绍如何发现程序中的错误以及处理这些错误的方法。

第二章 Visual C++界面处理

在这一章，我们将介绍如何利用 Visual C++ 6.0 中由 MFC 提供的文档和视图结构来处理用户的大量信息。最后，介绍通过在程序中添加菜单、工具栏和状态栏来提高程序的可操作性和界面的可读性。读完本章，您可以定制出功能强大的菜单系统，绘制出各种需要的工具。

第三章 Windows 基本控制

在这一章，首先介绍如何在程序中添加功能强大的对话框，包括有模式对话框、无模式对话框和属性表（为完成某一功能的特定的一组对话框）。通过这些对话框大大加强了 Visual C++ 6.0 与用户交换信息的能力。最后介绍了如何在对话框中添加单选按钮、复选框、列表框和组合框。读完本章，您可以定制出功能强大的向导，它们由一系列的属性表构成。

第四章 Windows 高级技术

在这一章，首先介绍利用 Visual C++ 6.0 提供的类快速建立一些对话框的方法。然后介绍了在 Visual C++ 6.0 中添加图形图像的方法。最后，介绍使用 ODBC 和 DAO 进行数据库编程。

本书特点

本书与一些流行软件丛书以介绍菜单命令为主线的写作方法有所不同，本书从方便用户的角度出发，以“全攻略”的方式向您介绍 Visual C++ 6.0 的使用。在每一个任务中包含了大量的知识点。对每一个知识点的讲解使用了多种方法，即攻略。

《电脑软件全攻略宝典系列》

编委会

主 编 曾满平

副主编 陈海军 张翊钦

编 委

曹 宁	蔡长明	常洪生	陈 强	车皓阳	邓 刚
高永英	桂大为	郝延东	何 为	蒋方帅	金 凌
廖邦政	廖 力	李兴栋	李德吉	李 然	刘 鹏
刘旭昌	马勃民	闵应宗	潘晓辉	钱 坤	宋志军
孙 媛	唐青松	唐先武	谢正雄	修 众	姚顾波
叶 彬	张 韬	张 岩	张 玲	张 永	张大军
周家祥	周 舰	周 颖	朱晓东		

目 录

前言

全书导读

第一章 Visual C++ 6.0 基础.....	1
任务一 建立第一个 Windows 应用程序.....	1
任务说明.....	1
攻略一 直接使用 Win32 API 函数.....	3
攻略二 利用 MFC 类库直接开发应用程序.....	13
攻略三 利用 MFC AppWizard 创建程序.....	17
总 结.....	27
任务二 向 Visual C++ 6.0 程序中添加程序代码.....	27
任务说明.....	27
攻略一 利用 ClassWizard 完善程序代码.....	28
攻略二 手工完成代码的添加和修改.....	42
攻略三 从 ClassWizard 看文档和视图的结构.....	48
总 结.....	55
任务三 Visual C++调试与异常处理.....	56
任务说明.....	56
攻略一 传统的错误捕获和处理.....	56
攻略二 TRY...CATCH 处理异常.....	57
攻略三 利用宏 ASSRET 进行程序诊断.....	66
攻略四 利用集成环境进行调试.....	69
总 结.....	79
第二章 Visual C++界面处理.....	81
任务一 处理 MFC 中的文档.....	81
任务说明.....	81
攻略一 使用单文档.....	82
攻略二 使用多文档工程.....	102
总 结.....	108
任务二 对界面进行优化.....	113
任务说明.....	113
攻略一 用菜单对界面优化.....	114

攻略二 利用工具栏优化界面.....	132
攻略三 用状态栏进行界面优化.....	144
总 结.....	155
第三章 Windows 基本控制	156
任务一 改进 Windows 的人机交互功能.....	156
任务说明.....	156
攻略一 有模式对话框.....	158
攻略二 无模式对话框.....	185
攻略三 使用属性表.....	194
总 结.....	214
任务二 文本操作与简单按钮.....	215
任务说明.....	215
攻略一 静态文本处理.....	215
攻略二 使用编辑控件.....	219
攻略三 使用 Windows 基本按钮.....	234
总 结.....	243
任务三 成组控件.....	254
任务说明.....	254
攻略一 使用单选按钮.....	257
攻略二 使用复选框.....	261
攻略三 使用列表框.....	267
攻略四 使用组合框.....	274
总 结.....	281
第四章 Windows 高级技术.....	288
任务一 在 Windows 中快速创建对话框.....	288
任务说明.....	288
攻略一 利用 CFileDialog 类实现快速创建文件处理对话框.....	289
攻略二 利用 CPrintDialog 类实现快速创建文件打印对话框	296
攻略三 利用 CFontDialog 类实现快速创建字体对话框.....	301
攻略四 利用 CColorDialog 类实现快速创建颜色选择对话框	303
总 结.....	305
任务二 处理 Windows 中的图形和图像.....	305
任务说明.....	305
攻略一 使用 Windows 的绘图函数.....	306
攻略二 使用位图.....	337
攻略三 用 OpenGL 显示三维图像.....	347
总 结.....	370

任务三 数据库编程.....	371
任务说明.....	371
攻略一 使用 ODBC 编程.....	373
攻略二 用 DAO 编程.....	415
总 结.....	458
附录一 Microsoft 基本类 MFC 库中常用的消息映射函数.....	460
附录二 C++语言快速进阶	465
附录三 Internet 网上的 Visual C++资源与源代码	490
附录四 Visual C++ 6.0 安装与组成	494

第一章 Visual C++ 6.0 基础

本章将介绍 Visual C++ 6.0 的一些基本的知识，包括以下三个方面的内容：

(1) 建立一个 Windows 应用程序。这将使你对 Windows 程序有一个比较清晰的了解。

(2) 使用 ClassWizard 完善程序代码。在这儿你将看到如何完善程序的代码以及文档和视图的结构。

(3) 程序的调试。无论是使用系统提供的工具还是自己在程序中添加代码来实现，丰富的调试手段大大提高了程序的开发效率。

任务一 建立第一个 Windows 应用程序

任务说明

☆任务内容

建立一个可执行的 Windows 程序是每个编程人员所期望的结果。在这个任务中我们的目的是生成一个可以运行的 Windows 应用程序。

☆攻略简介

可以采用以下攻略建立 Windows 应用程序：

攻略一：直接使用 Win32 API 函数。使用这种方法，大量的程序代码必须由用户自己编写，在进入特定应用主题之前必须编写大量的重复代码，包括入口函数和初始化应用程序窗口等程序代码。

攻略二：直接使用 MFC 类库。使用这种方法，可以利用 MFC 提供的类和支持代码来处理标准的 Windows 编程任务，如创建窗口、处理消息、添加对话框、工具栏等，而不需要编写入口函数之类的重复代码。

攻略三：使用 MFC AppWizard。使用这种方法，利用 Wizard 生成应用程序的基本框架，然后使用 MFC ClassWizard 生成派生类、增加消息处理函数、进行资源管理等工作，而不需要用户处理程序入口问题，从而将精力集中在特定的主题上。

☆起始材料

Microsoft Visual C++ 6.0 提供了良好的集成开发环境，在这一环境下用户可以输入自己的程序、调试并运行。为了方便用户快速开发程序，Visual C++ 包含了大量的工具供程序员使用。Visual C++ 不仅编译代码，而且产生代码。读者在几分钟之内就可以生成一个

Windows 应用程序，所要做的只是告诉 AppWizard 生成一个具有模板代码的“起始应用程序”。AppWizard 是一个非常有效的辅助开发工具，能生成许多种应用程序，但对于大多数的程序员来讲，他们想要的是可执行程序（.exe）。AppWizard 不仅能够生成可执行程序，而且它还生成了每个程序所必须的模板代码——类、对象和函数。

(1) AppWizard 提供了三种可选择的应用程序类型，并为每一种类型生成不同的代码和类。三种可选择的程序类型如下：

- 单文档界面应用程序

这种应用程序一次只能打开一个文档：当选择 File 菜单的 Open 或 New 时，当前打开的文件在新文件打开或建立前被关闭。在 Windows 中 Notepad 应用程序就是这种文档的典型代表。

- 多文档界面应用程序

多文档应用程序一次可以打开多个文档（通常是多个文件），在菜单栏上有 Windows 菜单并且在 File 菜单上有 close 选项。同时打开的多个文档可以通过 close 命令进行选择地关闭。在 Windows 中如果希望一个文档有多个视图，则必须建立多文档视图。Windows 中的 Word 或 Excel 都是多文档的例子。

- 基于对话框的应用程序

在这一种应用程序中，它没有文档也没有菜单，整个程序看起来就是一个对话框。Windows 的磁盘扫描应用程序就是一个基于对话框的应用程序例子。

(2) AppWizard 允许用户选择是否需要文档/视图结构支持。如果需要 AppWizard 就在应用程序中自动生成了基于 CDocument 类和 CView 类的文档/视图结构，否则生成没有文档/视图结构的应用程序（在这种情况下，从一个磁盘文件打开一个文档视图时将不包含 MFC 支持）。

(3) AppWizard 生成可执行应用程序允许用户选择是否支持数据库（在后面的任务中将详细介绍对数据库的支持）。有关数据库支持的选项一共有四个：

- None

不是编写数据库应用程序。

- Header files only

不从 CFormView 派生自己的视图或有一个 Record 菜单。

- Database view without file support

从 CFormView 派生自己的视图并有一个 Record 菜单，但不需要对文档进行序列化，从而可以借助 CRecordset 类更新数据库记录。

- Database view with file support

从 CFormView 派生自己的视图并有一个 Record 菜单，同时又需对文档进行序列化。

(4) AppWizard 提供了确定应用程序界面外观的选项。影响界面外观的选项有：

- Docking Toolbar

AppWizard 为用户创建了工具栏，用户在以后可以利用 Resource View 对它进行编辑。如：删除不要的按钮，增加新的按钮，这将在后面工具栏的使用时讨论。

- Initial status bar

AppWizard 生成了初始状态栏以显示菜单提示和其它消息，用户可以在以后对状态栏进行编程以显示自己所需要显示的信息。用户可以在状态栏上增加指示器或其它组件。

- Printing and print preview

决定 AppWizard 是否生成处理打印、打印机设置和打印预览的代码（这些产生的代码是通过调用 MFC 库中 CView 类的成员函数实现的）。需要注意的是：只有在 AppWizard 选择了文档/视图结构的程序才能使用此项功能。

- Context-sensitive Help

决定 AppWizard 是否要为上下文帮助产生一些文件。这一选项的选择需要产生帮助文件的编译器，如果没有此编译器，你可以重新安装 Visual C++ 6.0。

- 3D controls

决定应用程序的界面是否是 Windows 风格。

- MAPI

应用程序将能够利用 Messaging API 来建立、处理、传送和存储邮件消息。

- Windows Sockets

决定程序是否支持 Windows socket。利用 Windows socket 你可以编写一些基于 TCP/IP 协议的通信程序。

攻略一 直接使用 Win32 API 函数

通过例子程序 UseWinAPI 来说明如何使用 Windows 提供的 API 函数，程序的执行结果如图 1-1 所示。

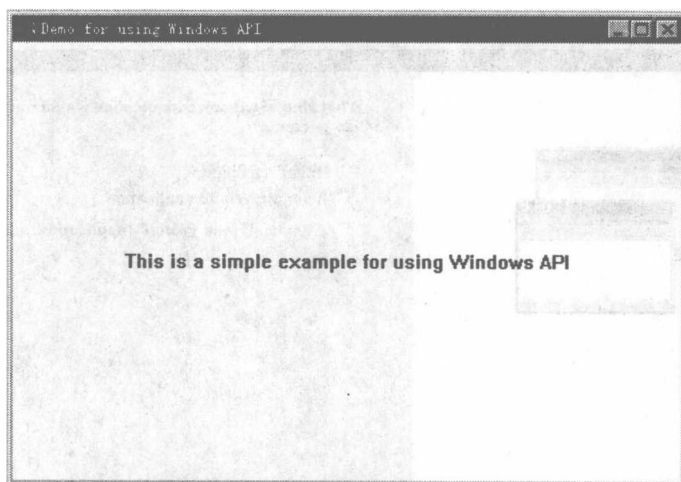


图 1-1 利用 Win32 API 建立应用程序

一、建立一个空工程文件

在 Visual C++ 6.0 的工作环境中，选择 File 菜单中的 New 菜单项，打开 New 对话框。

在 New 对话框中, 选择 Project 选项卡, 然后选择新建工程类型为 Win32 Application, 输入新建工程文件名 UseWinAPI, 然后在 Location 编辑框中输入新建工程所在的目录(也可以单击右边的按钮通过对话框进行选择), 设置 Platforms 为 Win32 平台, 如图 1-2 所示。

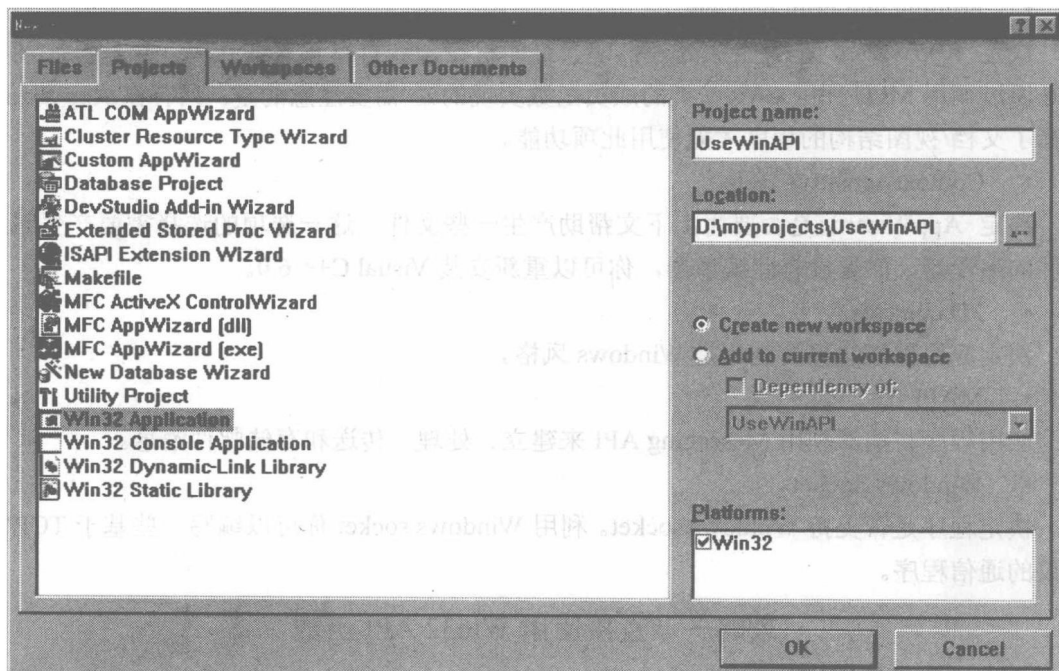


图 1-2 建立工程文件 UseWinAPI

单击 OK 按钮, 出现如图 1-3 所示的对话框。有三个单选项可以使用:

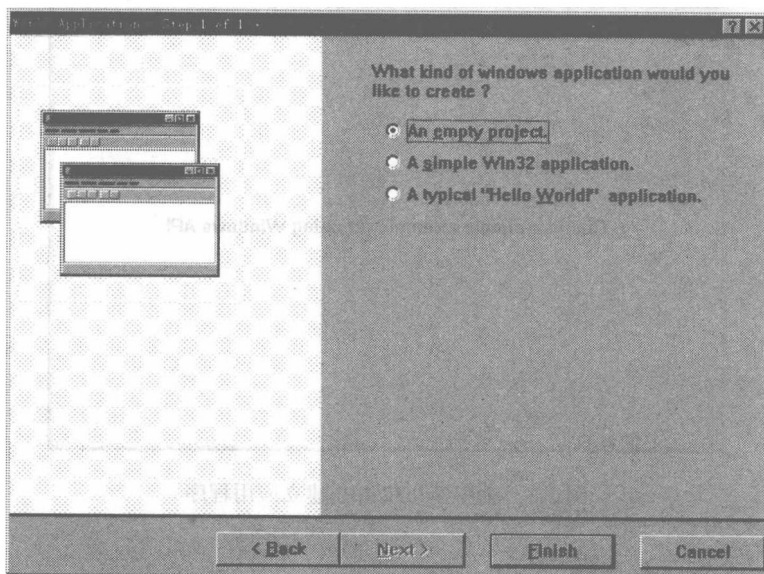


图 1-3 选择 Windows 应用程序的类型

- An empty project

如果选择此项将建立一个空的工程文件（不包含任何文件），在我们的例子中使用这个选项。

- A simple Win32 application

如果选择此选项将建立一个包含头文件 `stdafx.h` 的工程文件和该头文件的实现文件 `stdafx.cpp`，以及一个入口程序（与工程文件同名，此处工程文件为 `UseMFCLiblication`），代码如下：

```
// UseMFCLiblication.cpp : Defines the entry point for the application.
#include "stdafx.h"
int APIENTRY WinMain(HINSTANCE hInstance,
                    HINSTANCE hPrevInstance,
                    LPSTR    lpCmdLine,
                    int       nCmdShow)
{
    // TODO: Place code here.
    return 0;
}
```

- A typical “Hello World” application

如果选择此项将建立一个工程文件并包含一个完整的应用程序。执行该程序将建立一个窗口并在窗口中显示 “Hello world! ”。代码如下（此处工程为 `FirstHello`）：

```
// FirstHello.cpp : Defines the entry point for the application.
#include "stdafx.h"
#include "resource.h"
#define MAX_LOADSTRING 100
// Global Variables:
HINSTANCE hInst;                // current instance
TCHAR szTitle[MAX_LOADSTRING]; // The title bar text
TCHAR szWindowClass[MAX_LOADSTRING]; // The title bar text
// Foward declarations of functions included in this code module:
ATOM            MyRegisterClass(HINSTANCE hInstance);
BOOL            InitInstance(HINSTANCE, int);
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
LRESULT CALLBACK About(HWND, UINT, WPARAM, LPARAM);
```

```

int APIENTRY WinMain(HINSTANCE hInstance,
                    HINSTANCE hPrevInstance,
                    LPSTR     lpCmdLine,
                    int       nCmdShow)
{
    MSG msg;
    HACCEL hAccelTable;
    // Initialize global strings
    LoadString(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);
    LoadString(hInstance, IDC_FIRSTHELLO, szWindowClass, MAX_LOADSTRING);
    MyRegisterClass(hInstance);
    // Perform application initialization:
    if (!InitInstance (hInstance, nCmdShow))
        return FALSE;

    hAccelTable = LoadAccelerators(hInstance, (LPCTSTR)IDC_FIRSTHELLO);
    while (GetMessage(&msg, NULL, 0, 0))
    {
        if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
        {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
        }
    }
    return msg.wParam;
}

```

```

ATOM MyRegisterClass(HINSTANCE hInstance)
{

```

```

    WNDCLASSEX wcex;
    wcex.cbSize = sizeof(WNDCLASSEX);
    wcex.style      = CS_HREDRAW | CS_VREDRAW;
    wcex.lpfnWndProc = (WNDPROC)WndProc;
    wcex.cbClsExtra   = 0;
    wcex.cbWndExtra   = 0;
    wcex.hInstance    = hInstance;
    wcex.hIcon        = LoadIcon(hInstance, (LPCTSTR)IDI_FIRSTHELLO);
    wcex.hCursor      = LoadCursor(NULL, IDC_ARROW);

```

```

    wcex.hbrBackground= (HBRUSH)(COLOR_WINDOW+1);
    wcex.lpszMenuName    = (LPCSTR)IDC_FIRSTHELLO;
    wcex.lpszClassName   = szWindowClass;
    wcex.hIconSm         = LoadIcon(wcex.hInstance, (LPCTSTR)IDI_SMALL);
    return RegisterClassEx(&wcex);
}

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
    HWND hWnd;
    hInst = hInstance; // Store instance handle in our global variable
    hWnd = CreateWindow(szWindowClass, szTitle, WS_OVERLAPPEDWINDOW,
        CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);
    if (!hWnd)
        return FALSE;
    ShowWindow(hWnd, nCmdShow);
    UpdateWindow(hWnd);
    return TRUE;
}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message,
                        WPARAM wParam, LPARAM lParam)
{
    int wmId, wmEvent;
    PAINTSTRUCT ps;
    HDC hdc;
    TCHAR szHello[MAX_LOADSTRING];
    LoadString(hInst, IDS_HELLO, szHello, MAX_LOADSTRING);
    switch (message)
    {
        case WM_COMMAND:
            wmId    = LOWORD(wParam);
            wmEvent = HIWORD(wParam);
            switch (wmId)
            {
                case IDM_ABOUT:
                    DialogBox(hInst, (LPCTSTR)IDD_ABOUTBOX,

```