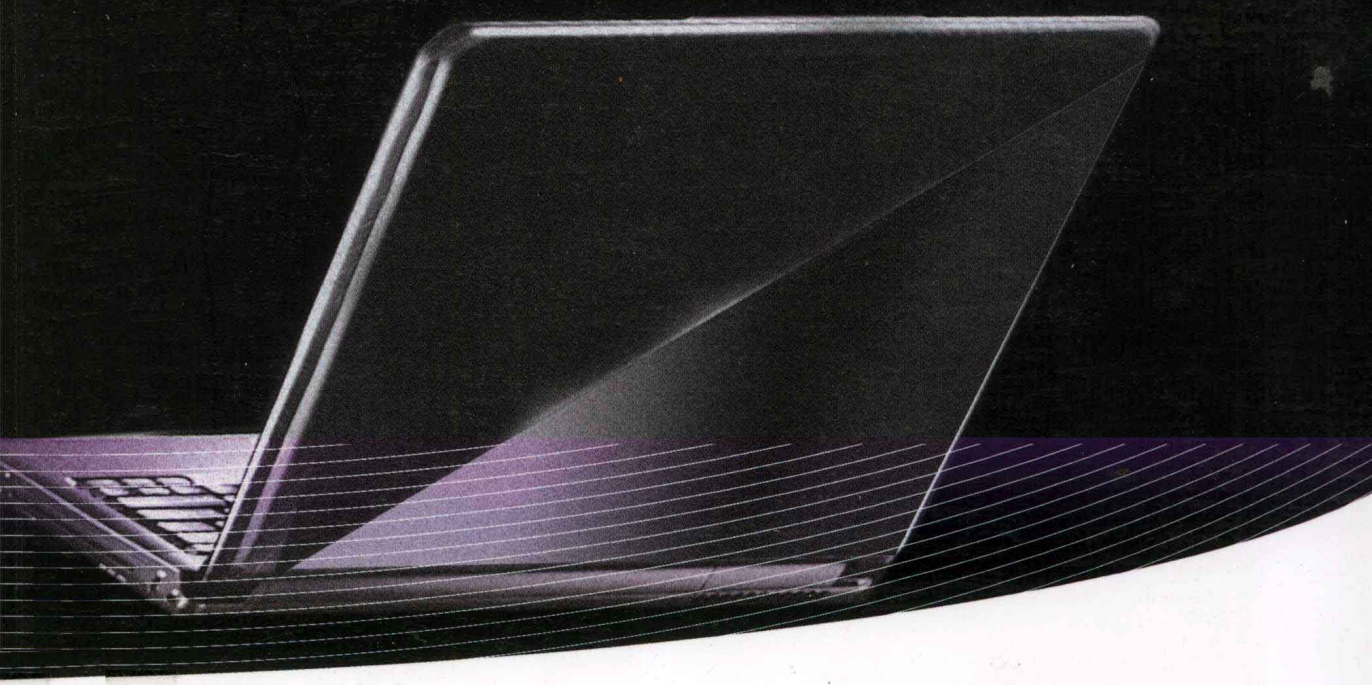
网博
黄金屋

鲁 威 夏曹俊 主编

C++实训教程
应用篇



東南大學出版社
SOUTHEAST UNIVERSITY PRESS

内容提要

本书作为C++实训的应用篇，主要立足于应用最广的Windows编程领域，编写的目的是为学习Windows编程的读者提供一个良好的学习方法。本书循序渐进地介绍了Win32程序运行原理和最基本的Win32 API编程，并详细讲述了Win32程序的设计方法和内部工作机制，而且指出了这些机制是如何对用户程序造成影响的。

本书内容按照教学要求组织，并遵循“从简单到复杂、循序渐进”的原则进行讲解，很多基本概念、工作原理均通过示例代码向读者逐一阐述。

本书适用于Windows程序设计的初学者，只要熟悉C++语言，有一定使用过Windows操作系统经历的读者都能掌握书中的内容。本书语言通俗易懂，条理清晰，内容丰富，非常适合作为高等院校、培训学校的教材，也可供读者自学。

图书在版编目(CIP)数据

C++实训教程：应用篇/鲁威，夏曹俊主编. —南京：东南大学出版社，2011.12

ISBN 978-7-5641-3195-1

Ⅰ.①C… Ⅱ.①鲁…②夏… Ⅲ.①C语言—程序设计 Ⅳ.①TP312C

中国版本图书馆CIP数据核字(2012)第030886号

C++实训教程：应用篇

出版发行：东南大学出版社
社　　址：南京四牌楼2号　邮编：210096
出 版 人：江建中
责任编辑：丁志星
网　　址：http://www.seupress.com
经　　销：全国各地新华书店
印　　刷：江苏凤凰扬州鑫华印刷有限公司
开　　本：787mm×1092mm　1/16
总 印 张：51.5
总 字 数：1300千字
版　　次：2011年12月第1版
印　　次：2011年12月第1次印刷
书　　号：ISBN 978-7-5641-3195-1
定　　价：150.00元(共2册)

目 录

第 1 章 基础部分 …… (1)
1.1 学习前准备 …… (1)
1.1.1 Windows 环境准备 …… (1)
1.1.2 工具准备 …… (1)
1.1.3 文档准备 …… (1)
1.2 第一个 Windows 程序 …… (2)
1.2.1 控制台项目 …… (2)
1.2.2 Win32 项目 …… (2)
1.3 Unicode …… (6)
1.3.1 字符集 …… (6)
1.3.2 美国信息交换标准代码(ASCII) …… (6)
1.3.3 双字节字符集(DBCS) …… (6)
1.3.4 为什么使用 Unicode …… (7)
1.3.5 如何在 C++中使用 Unicode …… (7)
1.4 常用数据类型 …… (8)
1.5 小结 …… (10)

第 2 章 动态链接库 …… (11)
2.1 动态链接库介绍 …… (11)
2.2 静态链接库 …… (12)
2.3 库的调试与查看 …… (13)
2.4 extern "C" …… (14)
2.5 动态链接库示例 …… (15)
2.5.1 准备工作(目录结构创建) …… (15)
2.5.2 基本的 DLL …… (15)
2.5.3 声明导出函数 …… (19)
2.5.4 DLL 的调用方式 …… (20)
2.5.5 DllMain 函数 …… (21)
2.5.6 _stdcall 约定 …… (23)
2.5.7 DLL 导出变量 …… (24)
2.5.8 DLL 导出类 …… (25)
2.6 小结 …… (29)

第3章 多线程编程 …… (30)
3.1 进程 …… (30)
3.1.1 进程的概念 …… (30)
3.1.2 进程与线程 …… (30)
3.1.3 使用 CreateProcess 创建进程 …… (31)
3.1.4 进程的终止 …… (31)
3.1.5 从主线程的入口点函数返回 …… (31)
3.1.6 使用 ExitProcess 函数 …… (32)
3.1.7 使用 TerminateProcess 函数 …… (33)
3.1.8 进程终止后操作系统的工作 …… (34)
3.1.9 子进程 …… (35)
3.2 线程的基础知识 …… (36)
3.2.1 线程的创建 …… (37)
3.2.2 线程的复杂性 …… (38)
3.2.3 线程入口函数 …… (38)
3.2.4 CreateThread 函数 …… (39)
3.2.5 终止线程的运行 …… (42)
3.3 线程的调度 …… (44)
3.3.1 暂停和恢复线程的运行 …… (45)
3.3.2 暂停和恢复进程的运行 …… (45)
3.3.3 睡眠方式 …… (47)
3.3.4 转换到另一个线程 …… (47)
3.4 线程的同步 …… (48)
3.4.1 原子访问：互锁的函数家族 …… (48)
3.4.2 高速缓存区 …… (52)
3.4.3 高级线程同步 …… (53)
3.4.4 关键代码段 …… (55)
3.5 线程与内核对象的同步 …… (64)
3.5.1 等待函数 …… (66)
3.5.2 事件内核对象 …… (68)
3.5.3 信号量内核对象 …… (77)
3.5.4 互斥对象内核对象 …… (79)
3.6 小结 …… (82)

第4章 内存管理 …… (83)
4.1 Windows 内存结构简介 …… (83)
4.2 虚拟内存 …… (84)
4.2.1 操作系统的基本信息 …… (84)
4.2.2 监视虚拟内存的状态 …… (85)

4.3 程序中使用虚拟内存 …… (86)
4.3.1 在地址空间中申请内存区域 …… (86)
4.3.2 在保留区域中的提交存储器 …… (87)
4.3.3 同时进行区域的保留和内存的提交 …… (88)
4.3.4 何时提交物理存储器 …… (89)
4.3.5 回收虚拟内存和释放地址空间区域 …… (90)
4.3.6 何时回收物理存储器 …… (91)
4.4 内存映射文件 …… (91)
4.4.1 步骤1：创建或打开文件内核对象 …… (92)
4.4.2 步骤2：创建一个文件映射内核对象 …… (93)
4.4.3 步骤3：将文件数据映射到进程的地址空间 …… (96)
4.4.4 步骤4：从进程的地址空间中撤销文件数据的映像 …… (97)
4.4.5 步骤5和步骤6：关闭文件映射对象和文件对象 …… (99)
4.5 小结 …… (99)

第5章 界面编程 …… (100)
5.1 Windows界面编程基础 …… (100)
5.1.1 创建第一个应用程序的窗体 …… (100)
5.1.2 Windows编程的特点 …… (103)
5.2 Windows简单编程 …… (104)
5.3 MFC界面编程 …… (108)
5.3.1 MFC简介 …… (108)
5.3.2 设计一个MFC程序 …… (108)
5.4 控件介绍 …… (111)
5.4.1 Windows标准控件 …… (111)
5.4.2 控件的创建方法 …… (111)
5.4.3 控件的消息以及消息映射 …… (112)
5.4.4 控件的数据交换和数据校验 …… (113)
5.5 静态按钮和控件 …… (114)
5.6 编辑框和旋转按钮控件 …… (114)
5.6.1 编辑框的基本操作 …… (115)
5.6.2 旋转按钮控件 …… (115)
5.7 列表框 …… (115)
5.8 组合框 …… (116)
5.9 进度条、滚动条 …… (116)
5.9.1 进度条 …… (116)
5.9.2 滚动条 …… (116)
5.10 小结 …… (116)

第 6 章　网络编程 …… (117)
6.1　网络协议基础 …… (117)
6.1.1　TCP/IP 协议概述 …… (117)
6.1.2　IP 协议 …… (118)
6.1.3　TCP 协议 …… (119)
6.1.4　UDP 协议 …… (120)
6.2　Winsock 基础 …… (121)
6.2.1　Winsock 的启动和终止 …… (121)
6.2.2　创建套接字 …… (123)
6.2.3　指定本机地址——bind() …… (123)
6.2.4　建立套接字连接——connect()和 WSAConnect() …… (124)
6.2.5　监听连接——listen() …… (125)
6.2.6　接受连接请求——accept()和 WSAAccept() …… (125)
6.2.7　数据发送——send()和 sendto() …… (126)
6.2.8　数据接收——recv()和 recvfrom() …… (126)
6.2.9　I/O 多路复用——Select() …… (127)
6.2.10　释放连接——closesocket()和 shutdown() …… (127)
6.2.11　getpeername() …… (128)
6.2.12　getsockname() …… (128)
6.2.13　gethostbyaddr() …… (128)
6.2.14　Gethostbyname() …… (129)
6.2.15　文件下载函数 …… (129)
6.2.16　在程序中显示 SOCKET 错误信息 …… (130)
6.2.17　初始化 Socket IP 地址的一个例子 …… (130)
6.2.18　通讯程序中应该注意的长度问题 …… (131)
6.3　小结 …… (131)

第 7 章　Services 程序 …… (132)
7.1　服务介绍 …… (132)
7.2　CreateService 函数 …… (132)
7.3　安装服务 …… (133)
7.4　卸载服务 …… (133)
7.5　启动服务 …… (133)
7.6　小结 …… (134)

第 8 章　COM 技术 …… (135)
8.1　COM 基本概念 …… (135)
8.2　基本元素的定义 …… (136)
8.3　使用和处理 COM 对象 …… (136)

8.4 COM 的生命周期与引用计数 …… (139)
8.5 综合示例 …… (141)
8.6 COM 技术的优劣 …… (143)
8.7 小结 …… (144)

第 9 章 项目实例 Web 服务端 …… (145)
9.1 HTTP 协议详解 …… (145)
9.1.1 HTTP 协议 URL …… (145)
9.1.2 HTTP 协议请求 …… (146)
9.1.3 HTTP 协议响应 …… (147)
9.1.4 HTTP 协议消息报头 …… (147)
9.1.5 查看 http 协议的通讯过程 …… (150)
9.2 源代码 …… (152)
9.2 小结 …… (154)

第 10 章 项目实例 FTP 客户端 …… (155)
10.1 FTP 协议详解 …… (155)
10.1.1 数据表示与保存 …… (155)
10.1.2 FTP 命令 …… (156)
10.1.3 典型 FTP 过程 …… (162)
10.2 源代码 …… (163)
10.2.1 FtpClient.h …… (163)
10.2.2 FtpClient.cpp …… (164)
10.3 小结 …… (171)

参考文献 …… (172)

第1章 基础部分

本书主要介绍 Windows 下程序编写的方法。程序使用 C++语言和 Windows 的 API。我们介绍的是 Windows 下编程的基本方法，适用于 Windows XP、Windows Vista、Windows 7等一系列 x86 平台的 Windows 系统(不涉及移动平台)，我们通过调用 Windows 的 API 来实现这些程序，由于 Windows 的 API 有很好的兼容性，所以对于我们来说，Windows 下各个平台的开发方法基本一样。

本书假定用户有一定的 C++基础，不再做 C++基础的说明。Windows 编程有一定的复杂性，不建议初学者刚开始就学习 Windows 编程，应该先直接通过控制台编程学习 C++，有了一定的 C++基础后再来阅读本书。

1.1 学习前准备

工欲善其事必先利其器，所以准备好工具是非常必要的，我们主要要做好下面的准备工作：Windows 环境准备、工具准备、文档准备。

1.1.1 Windows 环境准备

Windows 不需要多做介绍，绝大部分的个人电脑采用的都是 Windows 平台。当然现在的平板电脑很多采用了 Linux 平台(Android 也是 Linux)。

我们可以安装 Windows XP 专业版(32 位)或者 Windows 7(32 位)，本书的示例都是在 Windows XP 专业版中编写测试的。

为了不要牵涉太多复杂的问题，我们统一采用 32 位的环境，暂不使用 64 位。

1.1.2 工具准备

Windows 程序可以使用多种语言编写，但是使用 C/C++却能提供最佳的性能、最强大的功能和在发掘 Windows 特性方面最大的灵活性。可执行文件相对较小且运行时不要求外部链接库。

由于我们使用的是纯粹的 C++编写 Windows API 程序，对于编程工具来说我们不需要采用太高的版本。高版本开发出来的程序涉及一些环境的安装，低版本在任何机器上都可以直接运行，降低了部署成本。

本书的示例都采用 VC++ 2003 编译，从 Visual Studio NET 2003(简写为 VS)开始编程界面相对之前有些改动，为了学习方便，你可以使用和本书相同的编程工具。

1.1.3 文档准备

本书只是你开始学习 Windows 编程的指导用书，在实际的项目开发中，你一定要准备

好 Windows API 的说明文档。由于 Windows API 很多，并且接口参数比较复杂，我们在短时间内不可能精确地记得这些接口，我们在开发过程中需要经常查阅接口说明。

你需要安装 MSDN，版本越新越好，新的版本有更多的文档说明，如果不安装也可以通过网络直接访问 http://www.microsoft.com/msdn/。

1.2 第一个 Windows 程序

几乎所有的程序开发入门都是以“Hello World”开始，对于初学者来说，学习的信心非常重要，所以我们后面的程序示例都会尽量简单。我们缩写的程序首先应该能运行，然后再从性能和错误处理上完善程序，使得大家能循序渐进地来学习。Win32 程序有两种，一种是基于控制台，一种是 Windows 应用程序。

1.2.1 控制台项目

一个标准的控制台“Hello World”程序如下所示：

```
#include <iostream>
using namespace std;
int main(int argc,char * argv[],char * env[])
{
  cout<<"Hello World"<<endl;
  return 0;
}
```

main 函数是程序的入口函数，包含三个参数和返回值，三个参数可以省略不写。

1.2.2 Win32 项目

下面我们来创建一个等价于上面的控制台项目的程序。首先，我们打开 VS，点击菜单上的【文件】—>【新建】—>【项目】，打开如图 1-1 所示窗口，然后选中 win32 项目，输入项目名称。

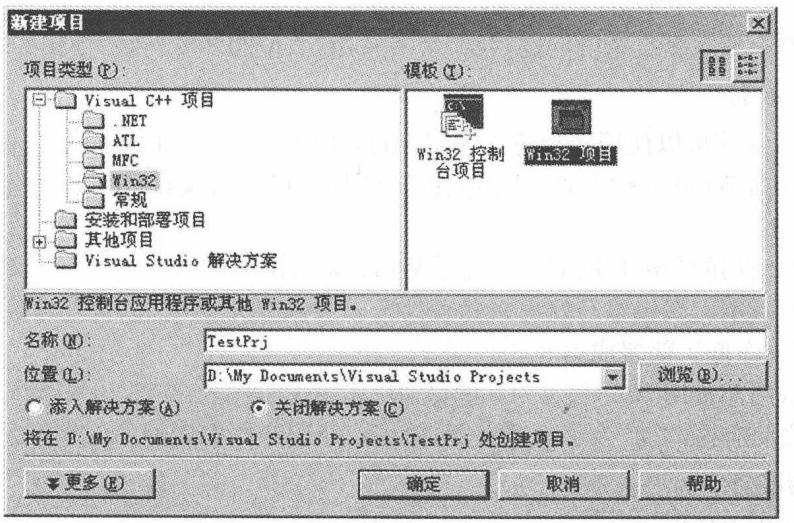


图 1-1 新建项目窗口

选中应用程序设置，附加选项选中空项目。

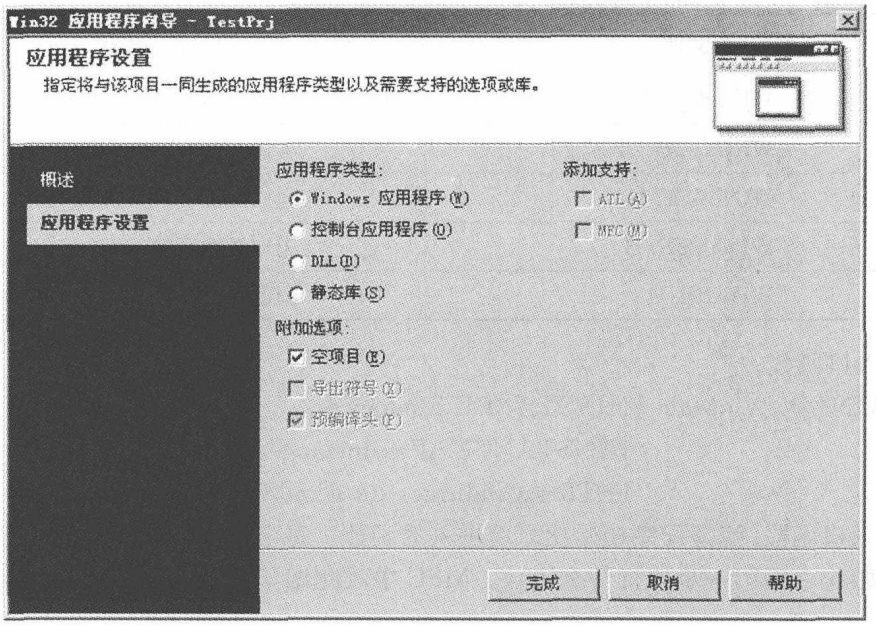


图 1-2 选中“空项目”

项目创建好后在解决方案上的源文件目录上点击右键，选中【添加】—>【添加新建项】，选中 cpp 输入文件名。源代码如下：

```
#include <Windows.h>
int WINAPI WinMain (HINSTANCE hInstance,
                    HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
  MessageBox (NULL,"Hello World!","HelloMsg", 0);
  return 0 ;
}
```

这是最简单的 Windows 程序，包含了一个引用头文件(Windows.h)；一个入口函数 WinMain，WinMain 函数等同于控制台程序的 main 函数；还包含了一个窗口输出函数 MessageBox 和一些符号。

我们开始分析这段代码。

(1) 头文件的引用

头文件的引用方法是#include <Windows.h>，这种引用方式是到环境变量指定的目录中查找文件。你可以手动查看这个文件，首先找到你的 VS 安装目录，然后就可以在"Microsoft Visual Studio .NET 2003\Vc7\PlatformSDK\Include"目录中找到它们。

当然版本不同也会有些不同，你可以搜索这个文件，你也可以直接在编辑器中右键点击 include 代码，选中打开<Windows.h>，就可以看到这个文件。

Windows.h 是 Windows 程序中主要的头文件，在这个头文件中包含了一些其他必须的

头文件。这些头文件及说明如表 1-1 所示。

表 1-1 头文件及相关说明

头文件	说 明
WINDEF. H	基本类型定义
WINNT. H	支持 Unicode 的类型定义
WINBASE. H	内核函数
WINUSER. H	用户界面函数
WINGDI. H	图形函数库

(2) 入口函数

```
int WINAPI WinMain (HINSTANCE hInstance,
                    HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
```

所谓入口函数,就是程序初始化之后进入的函数。每个程序都必须要有一个入口,来决定程序初始化之后第一步执行什么函数。MFC 中对此做了封装,但是它内部还是要有一个入口函数。

此函数类似于 main 函数,函数名称和参数是固定的,WinMain 函数调用返回一个 int 类型的值。下面我们来说明 WINAPI 的含义和一些特殊的类型。

● WINAPI 的含义

WINAPI 关键字在 WINDEF. H 中定义,语句如下:

```
#define WINAPI _stdcall
```

也就是说 WINAPI 是一个宏,直接被替换为_stdcall,后面我们会发现很多 Windows 函数定义前的各种标识,最终都是转换为_stdcall。现在我们就要知道_stdcall 是什么。

■ _stdcall

_stdcall 调用约定:函数的参数自右向左通过栈传递,被调用的函数在返回前清理传送参数的内存栈。_stdcall 调用约定在输出函数名前加上一个下划线前缀,后面加上一个“@”符号和其参数的字节数,格式为_functionname@number。例如 function(int a, int b),其修饰名为_function@8。

■ _cdecl

这是默认的函数类型,可以省略。

_cdecl 是 C 和 C++程序的默认调用方式,所有参数采用从右到左的压栈方式。每一个调用它的函数都包含清空堆栈的代码,所以产生的可执行文件会比调用_stdcall 函数的大。注意:对于可变参数的成员函数,始终使用_cdecl 的调用方式。_cdecl 调用约定仅在输出函数名前加上一个下划线前缀,格式为_functionname。

● 参数含义

■ hInstance

当前应用程序的实例句柄。在 Windows 程序设计中,句柄就是一个数字,这些数字可以用来识别为某个应用程序。该句柄可以唯一地标识该程序,实例句柄还可以作为其他

Windows 函数调用的参数。

■ hPrevInstance

在 Windows 的早期版本中，当多次运行同一程序时，你便创建了该程序的多个实例。

程序通过检查 hPrevInstance 参数就能够确定自身的其他实例是否正在运行，然后它可以略过一些繁杂的工作并从前面的实例中将某些数据移到自己的数据区域。

在 32 位和 64 位的 Windows 版本中，此参数已被抛弃，传给 WinMain 的第二个参数总是 NULL。这个参数只在 16 位的系统开发中会碰到，我们现在基本不会开发 16 位的系统，所以不需要考虑这个参数。

■ szCmdLine

用于执行程序的命令行。我们可以通过命令执行某个程序来给他传递参数，类似于 main 函数的 argc 和 argv。

■ iCmdShow

程序最初显示的方式，可以是正常的，或者是最大化显示，或者是最小化显示在任务列表栏中。

(3) MessageBox 函数

MessageBox 函数用于提示消息。MessageBox 对话框本身不具有什么功能，一般用来给客户提示，或者是应用在调试中。

MessageBox 的第一个参数通常是窗口句柄。

第二个参数是显示在小窗口内的字符串。

第三个参数是出现在小窗口标题栏中的字符串。

第四个参数可以是在 WINUSER. H 中定义的一个以 MB_开始的常数或常数的组合运算来指定按钮的数目及形式，使用的图标样式等。如果将第四个参数设置为 0，则仅显示“OK”按钮。

可以用第四个参数来指定按钮的样式：

```
#define MB_OK                    0x00000000L
#define MB_OKCANCEL              0x00000001L
#define MB_ABORTRETRYIGNORE      0x00000002L
#define MB_YESNOCANCEL           0x00000003L
#define MB_YESNO                 0x00000004L
#define MB_RETRYCANCEL           0x00000005L
```

指定默认按钮：

```
#define MB_DEFBUTTON1            0x00000000L
#define MB_DEFBUTTON2            0x00000100L
#define MB_DEFBUTTON3            0x00000200L
#define MB_DEFBUTTON4            0x00000300L
```

还可以使用一个常数指定对话框图标样式：

```
#define MB_ICONHAND              0x00000010L
#define MB_ICONQUESTION          0x00000020L
#define MB_ICONEXCLAMATION       0x00000030L
```

```
#define MB_ICONASTERISK          0x00000040L
```

这些图标中的某些有替代名称：

```
#define MB_ICONWARNING           MB_ICONEXCLAMATION
#define MB_ICONERROR             MB_ICONHAND
#define MB_ICONINFORMATION       MB_ICONASTERISK
#define MB_ICONSTOP              MB_ICONHAND
```

更多的内容可以直接查看 MSDN 中此函数的说明。

在本程序中，MessageBox 返回数值 1，有一个对应的宏 IDOK。IDOK 在 WINUSER. H 中定义，等于 1。根据对话框中显示的其他按钮，MessageBox 函数还可返回 IDYES、IDNO、IDCANCEL、IDABORT、IDRETRY 或 IDIGNORE。

1.3 Unicode

Unicode 是 Windows 程序的核心部分，Windows 内部的字符串处理都是采用 Unicode 来实现，对于非英文国家的人来说，Unicode 更加重要，因为涉及宽字符的存储方式。

1.3.1 字符集

软件本地化要解决的真正问题，实际上就是如何来处理不同的字符集。多年来，许多人一直将字符串作为一系列单字节字符来进行编码，并在结尾处放上一个零。对于我们来说，这已经成了习惯。当调用 strlen 函数时，它在以 0 结尾的单字节字符数组中返回字符的数目。

问题是，汉字的字符集中的符号太多了，因此单字节（它提供的符号最多不能超过 256 个）是根本不够使用的，为此出现了双字节字符集（DBCS），以支持这些文字和书写规则。

1.3.2 美国信息交换标准代码（ASCII）

标准 ASCII（American Standard Code for Information Interchange）码包括 26 个小写字母、26 个大写字母、10 个数字、32 个符号、33 个控制字符和一个空格，总共 128 个字符。

ASCII 码有许多优点。例如，26 个字母是连续的（在 EBCDIC 代码中就不是这样的）；大写字母和小写字母可通过改变一位元资料而相互转化；10 个数字的代码可从数值本身方便地得到（在 BCDIC 代码中，字符“0”的编码在字符“9”的后面！）

最棒的是，ASCII 码是一个非常可靠的标准。在键盘、视频显卡、系统硬件、打印机、字体文件、操作系统和 Internet 上，其他标准都不如 ASCII 码流行而且根深蒂固。

1.3.3 双字节字符集（DBCS）

在双字节字符集中，字符串中的每个字符可以包含一个字节或包含两个字节。例如，汉字，如果第一个字符在 0x81 与 0x9F 之间，或者在 0xE0 与 0xFC 之间，那么就必须观察下一个字节，才能确定字符串中的这个完整的字符。使用双字节字符集，对于程序员来说是个很大的难题，因为有些字符只有一个字节宽，而有些字符则是两个字节宽。

如果只是调用 strlen 函数，那么你无法真正了解字符串中究竟有多少字符，它只能告诉你到达结尾的 0 之前有多少个字节。ANSI 的 C 运行期库中没有配备相应的函数，使你能够对双字节字符集进行操作。但是，Microsoft Visual C++的运行期库却包含许多函数，如

_mbslen，它可以用来操作多字节(既包括单字节也包括双字节)字符串。

为了帮助你对 DBCS 字符串进行操作，Windows 提供了一组帮助函数(见图表 1-2)。前两个函数 CharNext 和 Char Prev 允许前向或逆向遍历 DBCS 字符串，方法是每次一个字符。第三个函数 IsDBCSLeadByte，在字符为双字符集中的第一个字节时返回 TRUE。

表 1-2 ASCII 码表

高位 / 低位	0—	1—	2—	3—	4—	5—	6—	7—
—0	NUL	DLE	SP	0	@	p		p
—1	SOH	DC1	!	1	A	Q	a	q
—2	STX	DC2	"	2	B	R	b	r
—3	ETX	DC3	#	3	C	S	c	s
—4	EOT	DC4	$	4	D	T	d	t
—5	ENQ	NAK	%	5	E	U	e	u
—6	ACK	SYN	&	6	F	V	f	v
—7	BEL	ETB	'	7	G	W	g	w
—8	BS	CAN	(	8	H	X	h	x
—9	HT	EM	)	9	I	Y	I	y
—A	LF	SUB	*	:	J	Z	j	z
—B	VT	ESC	+	;	K	[	k	{
—C	FF	FS	,	<	L	\	l	|
—D	CR	GS	—	=	M	]	m	}
—E	SO	RS	.	>	N	^	n	~
—F	SI	US	/	?	0	_	o	DEL

1.3.4 为什么使用 Unicode

自从 Windows 2000 开始，Windows 系统内核开始完全支持并完全使用 Unicode 编写，所有 ANSI 字符在进入底层前，都会被相应的 API 转换成 Unicode。所以，如果你一开始就使用 Unicode，则可以减少转换的用时和 RAM 开销。

对于 JAVA/. NET 等这些“新”的语言来说，内置的字符串所使用的字符集已经完全是 Unicode。

最重要的是，目前世界上大多数的程序使用的字符集都是 Unicode，因为 Unicode 有利于程序国际化和标准化。

1.3.5 如何在 C++中使用 Unicode

在 C++中使用 Unicode，就意味着你可能要抛弃已经非常熟悉的 char 和 ANSI 版字符串处理函数，取而代之的是使用 wchar_t，并且在使用字符串的时候，在""前加 L。比如：

```
wchar_t* sz = L"Test For Unicode";
```

而且，我们要使用 Unicode 版的字符串来处理函数，比如 wcscpy/wcscat. etc。

对于 std 的 I/O，我们则需要利用 wcout 代替 cout。

在 VC 编辑器中，利用 TCHAR 来替代原有的字符类型，并定义了_TEXT 和_T 宏用以支持 Unicode 字符，两者定义如下：

```
1  #ifndef _UNICODE
2  #define TCHAR    wchar_t
3  #define _T(x)   L##x
4  #define _TEXT(x)   L##x
5  #else
6  #define TCHAR    char
7  #define _T(x)   x
8  #define _TEXT(x)   x
9  #endif
```

很显然，利用 TCHAR 和_T 可以很方便地在 ANSI 和 Unicode 之间切换。

在 VC 中，几乎所有的字符串处理函数都做了类似的第三方宏处理（以_t 开头的），比如_tcscpy 在 Unicode 下编译成 wcscpy，在 ANSI 下编译成 strcpy。

1.4 常用数据类型

- 基本数据类型

BOOL	布尔变量
BOOLEAN	布尔变量
BYTE	字节（8 位）
CCHAR	Windows 字符
CHAR	Windows 字符
COLORREF	红、绿、蓝（RGB）彩色值（32 位）
Const	变量，该变量的值在执行期间保持为常量
DWORD	双字（32 位）
FLOAT	浮点变量
INT	符号整数
LONG	32 位符号整数
LPARAM	32 位消息参数

- 指针数据类型

LPBOOL	指向一个布尔变量的指针
LPBYTE	指向一个字节的指针
LPCOLORREF	指向一个 COLORREF 值的指针
LPCSTR	指向一个以"NULL"结束的 Windows 字符串常量的指针
LPCTSTR	指向以"NULL"结束的 Unicode 或 Windows 字符串常量的指针
LPCWCH	指向一个以"NULL"结束的 Unicode 字符常量的指针

LPCWSTR	指向一个以"NULL"结束的 Unicode 字符串常量的指针
LPDWORD	指向一个无符号双字(32 位)的指针
LPLONG	指向一个符号长整数(32 位)的指针
LPTSTR	指向一个以 NULL 结束的 Unicode 或 Windows 字符串的指针
LPVOID	指向任何类型的指针
LPWSTR	指向一个以"NULL"结束的 Unicode 字符串的指针
PBOOL	指向一个布尔变量的指针
PBYTE	指向一个字节的指针
PCCH	指向一个 Windows 字符常量的指针
PCH	指向一个 Windows 字符的指针
PCHAR	指向一个 Windows 字符串的指针
PCSTR	指向一个以"NULL"结束的 Windows 字符串常量的指针
PCWCH	指向一个 Unicode 字符常量的指针
PCWSTR	指向一个以"NULL"结束的 Unicode 字符串常量的指针
PDWORD	指向一个无符号双字的指针
PFLOAT	指向一个浮点变量的指针
PLONG	指向一个符号长整数的指针
PSHORT	指向一个符号短整数的指针
PSTR	指向一个以"NULL"结束的 Windows 字符串的指针
PSZ	指向一个以"NULL"结束的 Windows 字符串的指针
PTCH	指向一个 Windows 或 Unicode 字符的指针
PTCHAR	指向一个 Windows 或 Unicode 字符的指针
PTSTR	指向一个以"NULL"结束的 Windows 或 Unicode 字符串的指针
PUCHAR	指向一个无符号 Windows 字符的指针
PUINT	指向一个无符号整数的指针
PULONG	指向一个无符号长整数的指针
PUSHORT	指向一个无符号短整数的指针
PVOID	指向任何类型的指针
PWCH	指向一个 Unicode 字符的指针
PWCHAR	指向一个 Unicode 字符的指针
PWORD	指向一个无符号字的指针
PWSTR	指向一个以"NULL"结束的 Unicode 字符串的指针

● 句柄数据类型

HBITMAP	位图句柄
HBRUSH	画刷句柄
HCURSOR	光标句柄
HDC	设备描述表(DC)句柄
HDLG	对话框句柄
HFILE	文件句柄

HFONT	字体句柄
HGDIOBJ	GDI 对象句柄
HICON	图标句柄
HINSTANCE	实例句柄
HMEMU	菜单句柄
HMETAFILE	元文件句柄
HPALETTE	调色板句柄
HPEN	画笔句柄
HRGN	域句柄
HWND	窗口句柄

● 基本数据类型

SHORT	短整数
SPHANDLE	指向一个句柄的指针
TCHAR	Unicode 或 Windows 字符
UCHAR	无符号 Windows 字符
UINT	无符号整数
ULONG	无符号长整数
USHORT	无符号短整数
VOID	任何类型
WCHAR	Unicode 字符
WORD	无符号字(16 位)
WPARAM	32 位消息参数

1.5 小结

本章主要介绍了编写 Windows 程序的准备工作，并且给出了一个例子，写出了第一个 Windows 程序，对 Windows 编程中涉及的新的数据类型做了介绍，知道了 Windows 编程底层字符串都是采用 Unicode 作为编码方式，了解了基于 Unicode 的一些函数。通过本章的学习我们了解了 Windows 程序和普通的控制台程序的区别，使我们认识到要如何学习 Windows 编程。