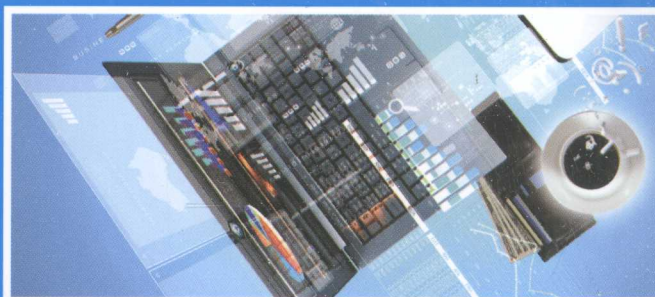




普通高等院校“十二五”规划教材

COMPUTER



VC++2010应用开发技术

VC++2010 YINGYONG KAIFA JISHU

主编 张晓民



机械工业出版社
CHINA MACHINE PRESS

赠电子课件

TP312/4896

:2010

2013

普通高等院校“十二五”规划教材

VC++ 2010 应用开发技术

主 编 张晓民

参 编 刘春茂 齐立磊 袁东峰

北方工业大学图书馆



C00348198

机械工业出版社

本书是一本关于 Visual C++ (以下简称 VC++) 2010 编程入门与提高的书籍,以当前流行的 Visual Studio (以下简称 VS) 2010 为开发平台,主要讲述了如何使用 VC++ 开发 Windows 应用程序。本书的特点是内容由浅入深,采用项目与案例驱动,每章均有工程案例,引导读者逐步掌握复杂的 Windows MFC 编程技术。全书共分 12 章,第 1 章介绍了 Windows 编程基础及 VS 2010 开发环境;第 2~7 章介绍了对话框应用程序、Windows 通用控件、文档与视图应用程序、标准用户界面设计、绘图工具的使用、MFC 类库与 ActiveX 控件;第 8~12 章介绍了线程与程序执行效率、动态链接库和 ActiveX 开发、文件流操作、数据库访问技术和 Windows 网络编程。

本书可作为高等学校计算机及相关专业 VC++ 程序设计的教材,也可作为使用 VC++ 为开发工具的工程技术人员的参考资料。

为了方便教学,本书配备电子课件等教学资源。凡选用本书作为教材的教师均可登录机械工业出版社教材服务网 www.cmpedu.com 下载,或发送电子邮件至 cmpgaozhi@sina.com 索取。咨询电话:010-88379375。

图书在版编目(CIP)数据

VC++ 2010 应用开发技术/张晓民主编. —北京:机械工业出版社,2013.10
普通高等院校“十二五”规划教材

ISBN 978-7-111-43563-1

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

中国版本图书馆 CIP 数据核字(2012)第 204661 号

机械工业出版社(北京市百万庄大街 22 号 邮政编码 100037)

策划编辑:王玉鑫 责任编辑:王玉鑫 范成欣

责任校对:张 力 封面设计:张 静

责任印制:杨 曦

北京富生印刷厂印刷

2013 年 10 月第 1 版第 1 次印刷

184mm×260mm·17 印张·421 千字

0 001—3 000 册

标准书号:ISBN 978-7-111-43563-1

定价:32.00 元

凡购本书,如有缺页、倒页、脱页,由本社发行部调换

电话服务

社服务中心:(010)88361066

销售一部:(010)68326294

销售二部:(010)88379649

读者购书热线:(010)88379203

网络服务

教材网:<http://www.cmpedu.com>

机工官网:<http://www.cmpbook.com>

机工官博:<http://weibo.com/cmp1952>

封面无防伪标均为盗版

前言

Windows 是当今最流行的操作系统之一，其上运行的软件 and 应用程序可以由多种开发工具完成。VC++ 以其强大的功能、高效的代码和应用框架的编程思想，一直受到专业软件开发人员的青睐，但由于其涉及面广，加之要求开发人员有比较扎实的 C++ 和面向对象的基础，所以 VC++ 的学习一直是件令人头痛的事情。

本书是作者多年在 VC++ 方面的教学和编程经验的集结，力求使读者通过本书的学习快速进入 VC++ 的开发殿堂。本书突出实用，以项目案例为驱动，内容由浅入深，以当前流行的 VS 2010 为开发平台，介绍 VC++ 基础应用开发方法，项目案例力求简洁明了，以能说明问题为度，舍弃复杂和综合性的项目案例。

学习 VC++，必须对 Windows 环境有一定的了解和认识，第 1 章向读者介绍了 Windows 基础知识及 VS 2010 开发环境。通过这一部分的学习，使读者对 Windows 程序设计及 VS 集成开发环境有一个基本认识。

学习 Windows 应用程序设计，必须能够熟练地使用 Windows 的基本元素及其控件，因此第 2~7 章从最简单的 Windows 应用程序——对话框应用程序开始，逐步介绍了 Windows 通用控件、文档与视图应用程序、标准用户界面设计、绘图工具的使用、MFC 类库与 ActiveX 控件。通过这一部分的学习，使读者可以编写出基本的 Windows 应用程序。

Windows 操作系统本身就是一个非常复杂的系统，包含大量的技术和方法，第 8~12 章主要讲述 Windows 程序设计的一些高级的话题，包括线程与程序执行效率、动态链接库和 ActiveX 开发、文件流操作、数据库访问技术以及 Windows 网络编程。通过这一部分的学习，使读者可以使用这些高级开发技术和方法编写出一些复杂和综合性的实际应用程序。

本书由南阳理工学院张晓民主编，河南工业职业技术学院刘春茂，南阳理工学院齐立磊、袁东锋参编。具体分工如下：第 1、2、10、11 章由张晓民编写；第 3、4、12 章由齐立磊编写；第 5、6 章由刘春茂编写；第 7、8、9 章由袁东锋编写。

在本书的编写过程中，郭爽、姬巧春、王森、闻彪对书中的部分代码进行了测试，在此表示由衷的感谢，同时也要感谢南阳理工学院软件学院各位同仁的大力支持和帮助。

由于时间仓促，加之编者水平有限，书中错误及疏漏之处在所难免，敬请读者批评指正。

编者

目 录

前言

第1章 Windows 编程基础及 VS 2010

开发环境.....1

1.1 Windows 程序结构.....1

1.1.1 Windows 操作系统简介.....1

1.1.2 应用程序结构.....2

1.1.3 应用程序开发模式.....7

1.2 VS 2010 开发环境.....7

1.2.1 VS 2010 主要功能介绍.....7

1.2.2 VS 2010 功能窗口.....8

1.2.3 VS 2010 中的菜单及工具栏.....8

1.2.4 VS 2010 中应用程序向导的使用.....16

1.2.5 VS 2010 中的资源编辑器.....18

第2章 对话框应用程序.....21

2.1 第一个 Windows 应用程序.....21

2.2 VS 2010 开发 MFC (EXE) 应用 程序的类型.....23

2.3 设计一个简单的加法器.....23

2.3.1 生成应用程序框架.....24

2.3.2 界面设计.....25

2.3.3 数据的存储.....27

2.3.4 功能实现.....28

2.4 对话框的种类.....30

2.4.1 消息框.....30

2.4.2 模态对话框.....32

2.4.3 非模态对话框.....32

2.4.4 Windows 通用对话框.....33

2.5 对话框应用.....35

2.5.1 利用向导建立记事本项目.....36

2.5.2 为记事本增加加法器.....37

第3章 Windows 通用控件.....42

3.1 基本控件.....42

3.1.1 Static Text、Group Box 和

Picture 控件..... 42

3.1.2 Edit Box、List Box 和 Combo Box

控件..... 43

3.1.3 Button、Radio Button 和 Check

Box 控件..... 45

3.1.4 Spin 与 Slider 控件..... 47

3.1.5 Scroll Bar 控件..... 49

3.2 基本控件应用..... 50

3.2.1 实例说明..... 50

3.2.2 实例具体的实现步骤..... 50

3.3 高级控件..... 57

3.3.1 Progress 控件..... 57

3.3.2 Animate 控件..... 57

3.3.3 Tab 控件..... 58

3.4 高级控件的应用..... 59

3.4.1 实例说明..... 59

3.4.2 实例具体的实现步骤..... 59

3.5 高级对话框的应用..... 61

3.5.1 属性页对话框..... 61

3.5.2 向导对话框..... 62

3.6 制作一个安装向导..... 63

3.6.1 实例说明..... 63

3.6.2 实例具体的实现步骤..... 63

第4章 文档与视图应用程序..... 70

4.1 消息映射机制..... 70

4.1.1 消息的种类..... 70

4.1.2 消息传递与消息循环..... 72

4.1.3 消息映射机制介绍..... 78

4.2 文档与视图结构..... 81

4.2.1 文档与视图结构概述..... 81

4.2.2 Document/View 结构..... 81

4.3 文档与视图类..... 83

4.3.1 文档类	83	8.1.2 多线程的分类	143
4.3.2 视图类	86	8.2 线程的启动和终止	144
4.4 特殊视图类	87	8.2.1 创建用户界面线程	144
4.4.1 滚动视图类	87	8.2.2 启动用户界面线程	145
4.4.2 控件视图类	88	8.2.3 使用用户界面窗口	145
4.4.3 编辑视图类	88	8.2.4 启动工作者线程	146
4.5 文档与视图应用	88	8.2.5 线程的终止	146
4.5.1 实例说明	88	8.2.6 等待函数	147
4.5.2 实例具体的实现步骤	88	8.3 线程同步与通信	147
第 5 章 标准用户界面设计	91	8.3.1 消息队列	148
5.1 菜单与加速键	91	8.3.2 同步对象	148
5.2 菜单应用	92	8.4 用户界面线程的应用	151
5.3 工具栏与状态栏	97	8.5 工作者线程的应用	155
5.4 工具栏与状态栏的应用	99	第 9 章 动态链接库和 ActiveX 开发	166
第 6 章 绘图工具的使用	104	9.1 编写自己的 DLL	166
6.1 Windows 绘图机制	104	9.1.1 DLL 概述	166
6.2 绘图工具和绘图工具类	105	9.1.2 MFC 中 DLL 的分类	167
6.2.1 画笔	106	9.1.3 DLL 中的导入/导出函数	168
6.2.2 画刷	108	9.1.4 DLL 中的数据与资源	170
6.2.3 画笔和画刷的应用	110	9.1.5 DLL 与应用程序的链接	171
6.2.4 字体	114	9.1.6 DLL 开发举例	172
6.2.5 字体的应用	118	9.2 编写自己的 ActiveX 控件	183
6.2.6 其他绘图工具	124	9.2.1 ActiveX 控件的事件、属性 和方法	183
第 7 章 MFC 类库与 ActiveX 控件	130	9.2.2 ActiveX 控件开发实例	184
7.1 VC 中的代码重用	130	第 10 章 文件流操作	194
7.2 利用 MFC 类库	131	10.1 MFC 中的文件操作	194
7.2.1 MFC 类库的优越性	131	10.2 文件类 CFile 和 CStdioFile	194
7.2.2 MFC 常用类	132	10.2.1 CFile 类	194
7.3 ActiveX 控件的使用	138	10.2.2 CFile 的成员函数	196
7.4 ActiveX 控件的应用	138	10.2.3 CStdioFile 类	198
7.4.1 创建项目	139	10.3 文件对话框	198
7.4.2 往项目中添加 MediaPlayer 控件	139	10.4 文件操作举例	200
7.4.3 使用 MediaPlayer 控件	139	10.4.1 创建项目	200
第 8 章 线程与程序执行效率	142	10.4.2 设计应用程序界面	200
8.1 多线程概述	142	10.4.3 添加程序复制功能	201
8.1.1 多线程的概念	142	10.4.4 为程序添加粘贴功能	202

第 11 章 数据库访问技术	204
11.1 数据库访问方式	204
11.1.1 ODBC 访问	204
11.1.2 DAO 访问	207
11.1.3 ADO 访问	208
11.1.4 ADO.NET 访问	208
11.2 ODBC 访问 SQL Server 技术	209
11.2.1 记录集与记录视图	209
11.2.2 数据库操作	210
11.2.3 记录集的操作	211
11.2.4 ODBC 应用举例	214
11.2.5 多表操作应用举例	231
11.3 DAO 访问 Access 数据库	238
11.3.1 记录集与记录视图	238
11.3.2 增强的数据库管理功能	238
11.3.3 DAO 应用举例	239
11.4 ADO 访问 Access 数据库	244
11.4.1 VC++ 使用 ADO 的 3 种方式	244
11.4.2 使用 #import 的编程步骤	245

第 12 章 Windows 网络编程	248
12.1 网络应用开发基础	248
12.1.1 网络应用开发概述	248
12.1.2 网络应用开发基本流程	249
12.2 CSocket 类的功能和函数	250
12.3 服务器端应用程序开发	253
12.3.1 创建项目	254
12.3.2 增加和修改项目资源	254
12.3.3 增加 CListeningSocket 类	254
12.3.4 增加 CClientSocket 类	256
12.3.5 实现网络功能	257
12.4 客户端应用程序设计	259
12.4.1 创建项目	259
12.4.2 增加和修改项目资源	259
12.4.3 增加 CRequestSocket 类	260
12.4.4 实现网络功能	261
12.4.5 网络应用程序运行测试	265
参考文献	266

第 1 章

1.1 Windows 程序结构

1.1.1 Windows 操作系统简介

Microsoft Windows 是一个基于 Intel x86 微处理芯片的个人计算机上的具有图形用户的多任务和多窗口的操作系统，它是对 MS-DOS 操作系统的扩展和延伸。与 MS-DOS 操作系统相比，它有许多优越之处：①提供了比 MS-DOS 字符界面更为直观、友好的图形用户界面；②可以一次运行多个程序（即多任务），方便了用户的操作，提高了机器的利用率；③Windows 环境下的应用程序具有一致的外观和用户接口，用户只需要熟悉其中一两个程序，就可以触类旁通地学会使用其他的 Windows 应用程序。另外，Windows 还具有更好的虚拟内存管理和设备无关等特性。由于 Windows 具有以上突出优点，Windows 平台上的软件开发和程序设计已成为主流。Windows 之所以取得成功，主要在于它具有以下优点：

（1）直观、高效的面向对象的图形用户界面，易学易用。

从某种意义上说，Windows 用户界面和开发环境都是面向对象的。用户采用“选择对象→操作对象”这种方式进行工作。例如，要打开一个文档，首先用鼠标或键盘选择该文档，然后从快捷菜单中选择“打开”操作，打开该文档。这种操作方式模拟了现实世界的行为，易于理解、学习和使用。

（2）用户界面统一、友好。

Windows 应用程序大多符合 IBM 公司提出的 CUA（Common User Access）标准，所有的程序拥有相同的或相似的基本外观，包括窗口、菜单、工具栏等。用户只要掌握其中一个，就不难学会其他软件，从而降低了用户培训学习的费用。

（3）设备无关及图形操作。

Windows 的图形设备接口（GDI）提供了丰富的图形操作函数，可以绘制出诸如线、圆、框等的几何图形，并支持各种输出设备。设备无关意味着在针式打印机上和高分辨率的显示器上都能显示出相同效果的图形。

（4）多任务。

Windows 是一个多任务的操作环境，它允许用户同时运行多个应用程序，或在一个程序中同时做几件事情。每个程序在屏幕上占据一块矩形区域，这个区域称为窗口。窗口是可以重叠的，用户可以移动这些窗口，或在不同的应用程序之间进行切换，并可以在程序之间进

行手工和自动的数据交换和通信。

虽然同一时刻计算机可以运行多个应用程序，但仅有一个是处于活动状态的，其标题栏呈现高亮颜色。活动的程序是指当前能够接收用户键盘输入的程序。

1.1.2 应用程序结构

1. 应用程序的组成

一个完整的 Windows 应用程序通常由 5 种类型的文件组成，见表 1-1。

表 1-1 组成 Windows 应用程序的 5 种文件的扩展名及类型

扩展名	文件类型	备注
.C 或 .CPP	C 语言源程序	源程序包含变量及函数定义
.H	头文件	头文件包含源程序文件需要的外部常量、变量、数据结构和函数声明
.DEF	模块定义文件	模块定义文件定义程序模块的属性，在 Windows 95/98 中，该文件可省略
.RC	资源描述文件	资源描述文件定义源程序使用的资源
.VCXProj	项目文件	各种程序文件组织管理的项目文件，经编译成为可执行文件

2. 源程序组织结构

Windows 应用程序具有相对固定的基本结构，由入口函数 WinMain 和窗口函数构成基本框架，并包含各种数据类型、数据结构与函数等。入口函数 WinMain 和窗口函数是 Windows 应用程序的技术基础。

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 函数可以实现以下功能：

- 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收的消息，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 3 个基本的组成部分：函数说明、初始化和消息循环。

WinMain 函数声明：

```
int WINAPI WinMain
```

```
(
    HINSTANCE hThisInst,      // 应用程序当前实例句柄
    HINSTANCE hPrevInst,     // 应用程序以前实例句柄
    LPSTR lpszCmdLine,        // 指向程序命令行参数指针
    int nCmdShow              // 应用程序开始执行时窗口显示方式，整数标识
);
```

值得注意的是，Windows 应用程序可能并行地多次执行，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，这就是 Windows 操作系统所进行的多任务管理。Windows 系统将应用程序的每一次执行称为该应用程序的一个实例（instance），并使用一个实例句柄唯一地标识它。

Windows 应用程序和 DOS 应用程序的重要区别之一就是 Windows 应用程序在启动时,需要进行初始化工作。

初始化包括窗口类的定义、注册、创建窗口实例和显示窗口 4 部分。

(1) 窗口类定义。

在应用程序中,窗口类定义了窗口的形式与功能。窗口类定义是通过给窗口类数据结构赋值来完成的,该数据结构中包含窗口类的各种属性。窗口类定义常用以下函数完成:

1) LoadIcon 函数。

LoadIcon 函数的作用是在应用程序中加载一个窗口图标。其原型如下:

```
HICON LoadIcon(HINSTANCE hInstance,
    // 图标资源所在的模块句柄; 如果为 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
```

2) LoadCursor 函数。

LoadCursor 函数的作用是在应用程序中加载一个窗口光标。其原型如下:

```
HCURSOR LoadCursor
(
    HINSTANCE hInstance,
    // 光标资源所在模块句柄; 如为 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
);
```

3) GetStockObject 函数。

应用程序经常通过调用函数 GetStockObject 获取系统提供的背景刷子, 其原型如下:

```
HBRUSH GetStockObject(int nbrush); // nBrush: 为系统提供的背景画刷标识名
```

(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, 窗口类的注册由函数 RegisterClass() 实现。其形式如下:

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

RegisterClass 函数的返回值为布尔类型, 注册成功则返回 TRUE, 否则返回 FALSE。

(3) 创建窗口。

创建一个窗口类实例由函数 CreateWindow() 实现, 该函数原型如下:

```
HWND CreateWindow
(
    LPCTSTR lpzClassName,           // 窗口类名
    LPCTSTR lpzTitle,               // 窗口标题名
    DWORD dwStyle,                  // 创建窗口样式, 常用窗口样式见表 1-2
    int x,                          // 窗口左上角横坐标
    int y,                          // 窗口左上角纵坐标
    Int nWidth,                    // 窗口宽度
    Int nHeight,                   // 窗口高度
    HWND hwndParent,               // 该窗口的父窗口句柄
    HMENU hMenu,                   // 创建窗口的菜单
    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参数的指针
);
```

表 1-2 常用窗口样式

标 识	说 明
WS_BORDER	创建一个带边框的窗口
WS_CAPTION	创建一个带标题栏的窗口
WS_HSCROLL	创建一个带水平滚动条的窗口
WS_MAXIMIZEBOX	创建一个带最大化按钮的窗口
WS_MAXIMIZE	创建一个最大化的窗口
WS_MINIMIZEBOX	创建一个带最小化按钮的窗口
WS_MINIMIZE	创建一个最小化的窗口
WS_OVERLAPPED	创建一个带边框和标题栏的窗口
WS_OVERLAPPEDWINDOW	创建一个带边框、标题栏、系统菜单及最大、最小化按钮的窗口
WS_POPUP	创建一个弹出式的窗口
WS_POPUPWINDOW	创建一个带边框和系统菜单的弹出式窗口
WS_SYSMENU	创建一个带系统菜单的窗口
WS_VSCROLL	创建一个带垂直滚动条的窗口

在实际应用中，可定义组合式的窗口样式。例如，下面的语句表示带有垂直和水平滚动条的弹出式窗口：

```
WM_HSCROLL|WM_VSCROLL|WM_POPUP
```

(4) 显示窗口：

```
ShowWindow(hwnd, nCmdShow);
```

其中，hwnd 为窗口句柄，nCmdShow 为窗口显示方式标识。表 1-3 列出了常用显示方式标识及其说明。

表 1-3 常用显示方式标识及其说明

标 识	说 明
SW_HIDE	隐藏窗口
SW_SHOWNORMAL	显示并激活窗口
SW_SHOWMINIMIZE	显示并最小化窗口
SW_SHOWMAXIMIZE	显示并最大化窗口
SW_SHOWNOACTIVE	显示但不激活窗口
SW_RESTORE	恢复窗口的原来位置及尺寸

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，其形式如下：

```
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，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, 其形式如下:

```

GetMessage
(
    lpMSG,                // 指向 MSG 结构指针
    hwnd,
    nMsgFilterMin,        // 用于消息过滤的最小消息号
    nMsgFiltermax         // 用于消息过滤的最大消息号
);

```

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, 其形式如下:

```
TranslateMessage(lpMSG);
```

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

```
DispatchMessage(lpMSG);
```

当 `GetMessage` 函数返回 0, 即检索到 `WM_QUIT` 消息时, 程序将结束循环并退出。

4. 窗口函数

窗口函数定义了应用程序对接收到不同消息的响应, 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` 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。

窗口函数是应用程序处理接收到消息的函数, 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。

窗口函数的一般形式如下:

```

LRESULT CALLBACK WndProc(
    HWND hwnd,                // 窗口句柄
    UINT message,             // 代表所处理消息的消息值
    WPARAM wParam,            // 消息的附加参数
    LPARAM lParam              // 消息的附加参数
)
{
    ...
    switch(message)           // message 为标识消息的消息值
    {
        case...
            ...
            break;
        case WM_DESTROY:
            PostQuitMessage(0);
            break;
        Default:
            return DefWindowProc(hwnd, message, wParam, lParam);
    }
    return(0);
}

```

窗口函数的主体是消息处理语句, 该语句由一系列 `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。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); // 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。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。

(1) 数据类型。

Windows 应用程序的源程序中包含了种类繁多的数据类型, windows.h 是用户调用系统功能的关键。该文件中定义了 Windows 系统使用的数据类型,其中包括许多简单类型和结构类型。部分常用的 Windows 数据类型及其说明见表 1-4。

表 1-4 部分常用的 Windows 数据类型及其说明

数据类型	说明
WORD	16 位无符号整数
LONG	32 位有符号整数
DWORD	32 位无符号整数
HANDLE	句柄
UINT	32 位无符号整数
BOOL	布尔值
LPTSTR	指向字符串的 32 位指针
LPCTSTR	指向字符串常量的 32 位指针

(2) 数据结构。

Windows C 语言程序数据结构比一般的 C 语言数据结构复杂,常用的数据结构如下:

1) MSG。MSG 结构中包含一个消息的全部信息,既是消息发送的格式,也是 Windows 编程中最基本的数据结构之一。有关 MSG 结构的定义可参考有关资料。

2) WNDCLASS。WNDCLASS 结构包含一个窗口类的全部信息,也是 Windows 编程中使用的基本数据结构之一。应用程序通过定义一个窗口类确定窗口的属性,其定义如下:

```
typedef struct tagWNDCLASS
{
    UINT style; // 窗口类的样式,一般设置为 0
    WNDPROC lpfnWndProc; // 指向窗口函数的指针
    Int cbClsExtra; // 分配在窗口类结构后的字节数
    Int cbWndExtra; // 分配在窗口实例后的字节数
    HINSTANCE hInstance; // 定义窗口类的应用程序的实例句柄
    HICON hIcon; // 窗口类的图标
    HCURSOR hCursor; // 窗口类的光标
    HBRUSH hbrBackground; // 窗口类的背景刷
    LPCTSTR lpszMenuName; // 窗口类菜单资源名
    LPCTSTR lpszClassName; // 窗口类名
} WNDCLASS;
```

3) POINT。POINT 结构定义了屏幕上或窗口中的一个点的 X 坐标和 Y 坐标。POINT 结构也是应用程序中最常用的结构之一。有关 POINT 的定义可参考有关资料。

4) RECT。RECT 结构定义了一个矩形区域，其中包含该矩形区域的左上角和右下角两个点的 X 坐标和 Y 坐标。其定义如下：

```
typedef struct tagRect{
    LONG left;           // 矩形框左上角 X 坐标
    LONG top;            // 矩形框左上角 Y 坐标
    LONG right;          // 矩形框右上角 X 坐标
    LONG bottom;         // 矩形框右上角 Y 坐标
}RECT;
```

1.1.3 应用程序开发模式

Windows 程序分为“程序代码”和“UI (User Interface, 用户接口) 资源”两大部分，两部分最后以 Linker 链接器整合为一个完整的 EXE 程序，如图 1-1 所示。所谓 UI 资源是指功能菜单、对话框外观、图标、光标形状等，这些 UI 资源的实际内容（二进制代码）是借助多种工具产生的，并以各种扩展名文件而存在，如 .ico、.bmp、.cur 等，程序员必须在一个资源描述文档 (.rc) 中描述它们。RC 编译器 (RC.EXE) 读取 RC 文件中的描述后将所有 UI 资源文件集中编译成一个 .RES 文件，再与程序代码结合在一起，这才是完整的 Windows 可执行文件。

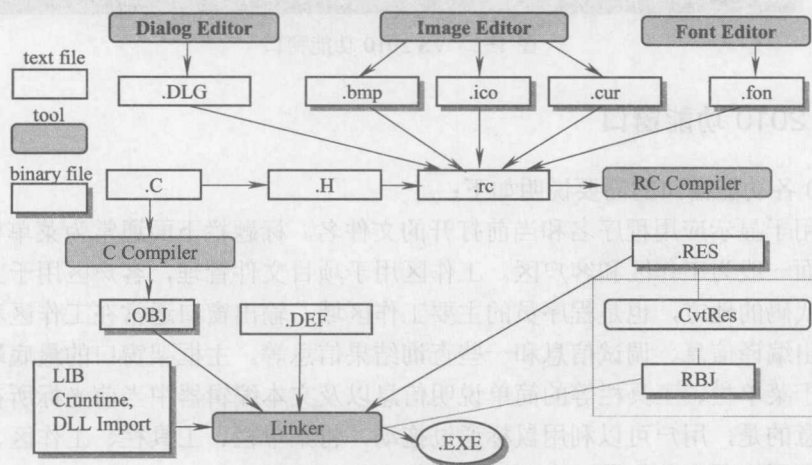


图 1-1 一个 32 位 Windows SDK 程序的开发流程

1.2 VS 2010 开发环境

学习 VS 2010，首要的工作就是熟悉其开发环境。对于初学者来说，可能暂时用不到本节介绍的部分内容，所以在阅读这些地方的时候可以跳过去，等有了一些编程经验以后，再回过头来阅读，可能收获会更大一些。

1.2.1 VS 2010 主要功能介绍

VS 2010 为用户提供了一套良好的可视化开发环境，主要包括文本编辑器、资源编辑器、

项目创建工具、Debugger 调试器等，用户可以在集成开发环境中创建项目，打开项目，建立、打开和编辑文件，编译、链接、运行、调试应用程序，其界面如图 1-2 所示。

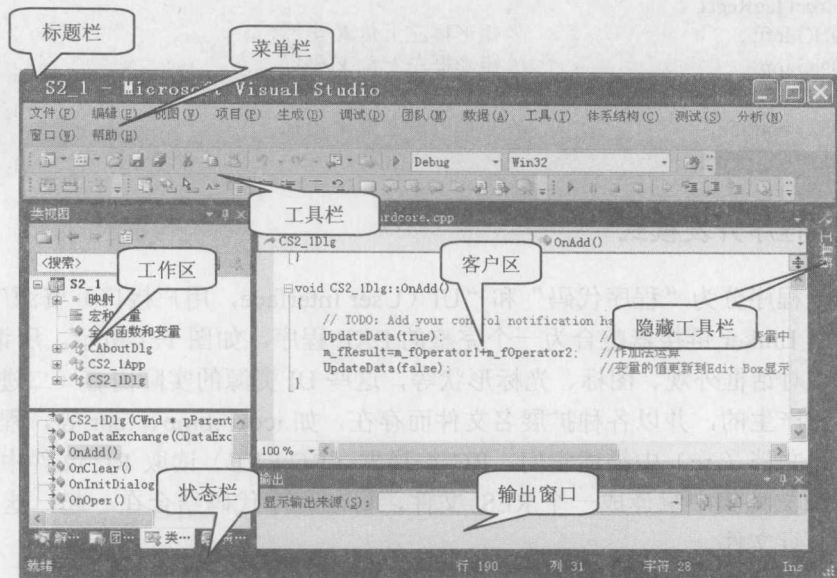


图 1-2 VS 2010 功能窗口

1.2.2 VS 2010 功能窗口

VS 2010 各功能窗口的简要说明如下：

标题栏用于显示应用程序名和当前打开的文件名。标题栏下面通常为菜单栏和工具栏。工具栏的下面一般为工作区和客户区。工作区用于项目文件管理，客户区用于文本文件的编辑，是编写代码的地方，也是程序员的主要工作区域。输出窗口通常在工作区和客户区的下面，用于输出编译信息、调试信息和一些查询结果信息等。主框架窗口的最底端为状态栏，用于显示关于菜单栏、工具栏等的简单说明信息以及文本编辑器中当前光标所在行列号等。

需要注意的是，用户可以利用鼠标通过拖动，将菜单栏、工具栏、工作区、客户区和输出窗口等随意更换顺序和位置，也可以隐藏其中的某些窗口。

在 VS 2010 中，项目管理采用的是层次结构，最上层是解决方案（.sln），其次是项目文件（.vcxproj），下层是具体代码（.h, .cpp）、资源文件（.rc）等文件。一个解决方案可以包含多个项目，默认情况下是一个项目，每个项目类似一个总管，负责内部各类文件的协调和管理。

1.2.3 VS 2010 中的菜单及工具栏

本节将对一些比较重要的菜单命令进行说明，对于一些常用的工具栏实现的命令在后面的工具栏一节中再进行介绍。

1. 常用菜单

(1) 文件菜单。

1) 文件|新建。

“文件|新建”菜单用于新建项目、文件，从现有代码创建项目。最常用的是项目和文件这两个菜单项。

① 文件。

选择“文件|新建|文件”菜单，将弹出“新建文件”对话框，如图 1-3 所示。用户可以创建多种类型的文件。如果要创建新文件，只需选中所要创建的文件类型，单击“打开”按钮即可。

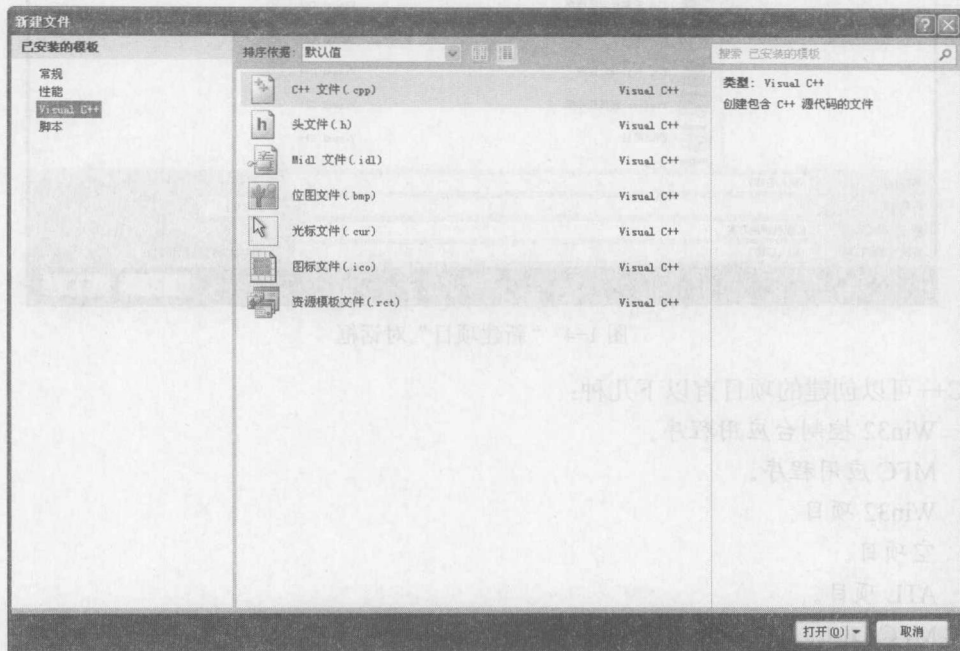


图 1-3 “新建文件”对话框

可以创建的 VC++文件的类型有以下几种。

- C++文件 (.cpp): C++文件。
- 头文件 (.h): C++源文件。
- Midl 文件 (.idl)。
- 位图文件 (.bmp)。
- 光标文件 (.cur)。
- 图标文件 (.ico)。
- 资源模板文件 (.rc)。

② 项目。

项目命令 (见图 1-4) 用于创建多种类型的项目。如果要创建新项目，选中所要创建的项目类型，在“名称”文本框中输入项目名，在“位置”文本框中输入项目所要保存的位置，单击“确定”按钮，即可完成新项目的创建。

新建项目也可通过“起始页”的“新建项目”菜单项来完成。

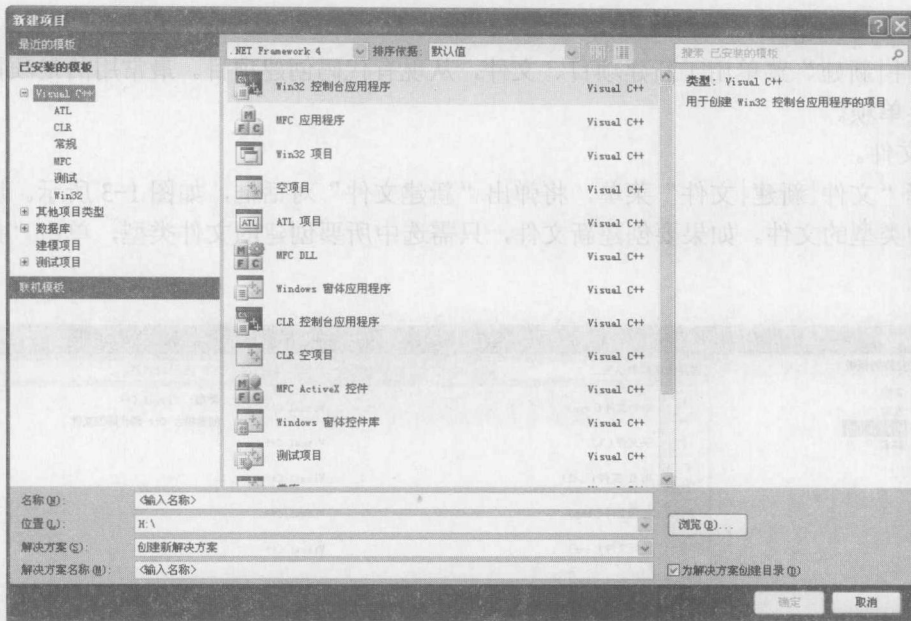


图 1-4 “新建项目”对话框

VC++可以创建的项目有以下几种：

- Win32 控制台应用程序。
- MFC 应用程序。
- Win32 项目。
- 空项目。
- ATL 项目。
- MFC DLL。
- Windows 窗体应用程序。
- CLR 控制台应用程序。
- CLR 空项目。
- MFC ActiveX 控件。
- Windows 窗体控件库。
- 测试项目。
- 类库。
- 生成文件项目。
- 自定义向导。

2) 文件|打开。

与“文件|新建”菜单类似，“文件|打开”菜单项用于打开项目/解决方案、团队项目、文件。最常用的是项目/解决方案和文件这两个菜单项。

① 项目/解决方案。

选择该菜单项可打开已有的工程项目，如图 1-5 所示。该功能也可通过“起始页”的“打开项目”菜单项来完成。