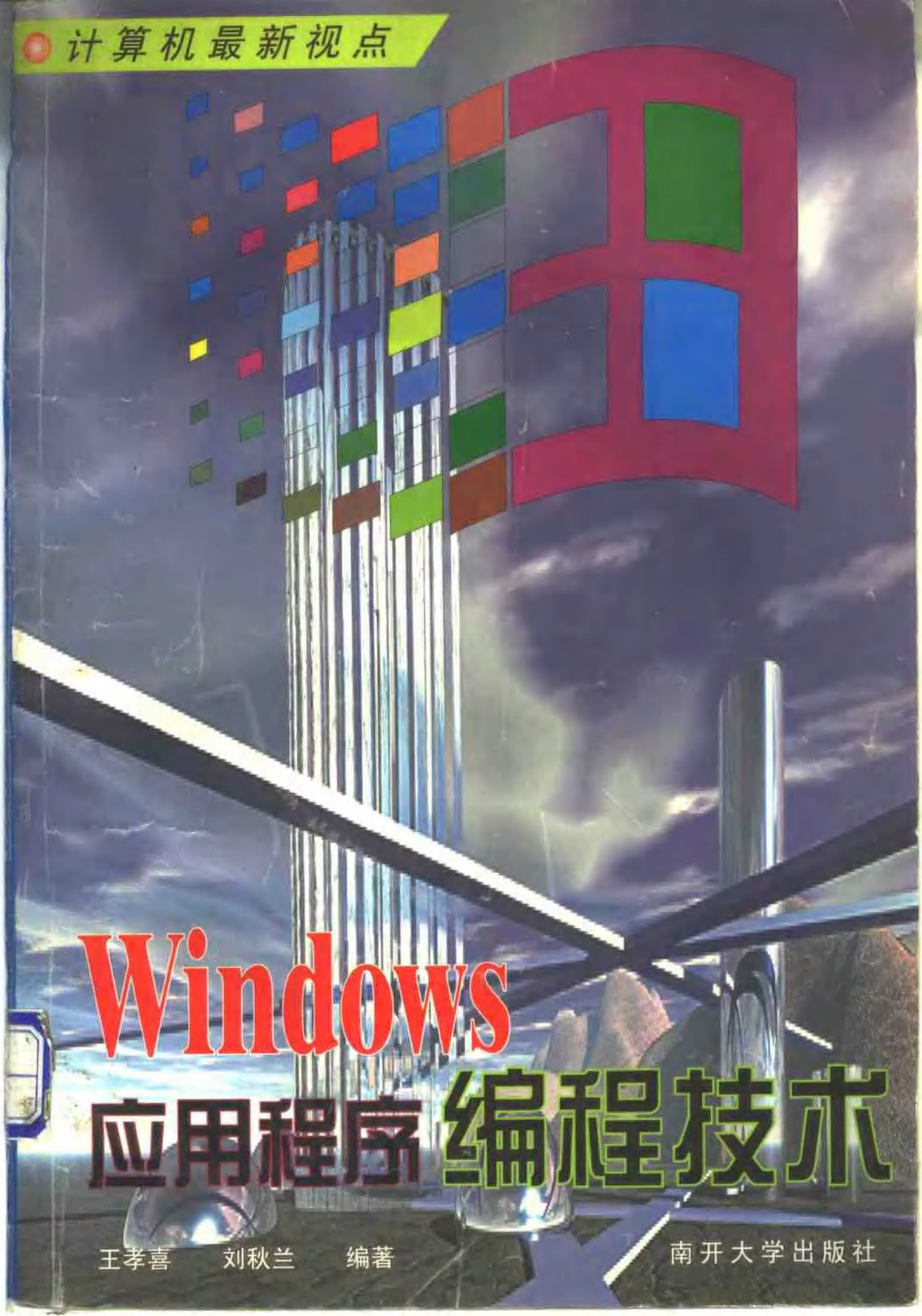
计算机最新视点



Windows 应用程序编程技术

王孝喜　刘秋兰　编著

南开大学出版社

Windows
应用程序编程技术

王孝喜　刘秋兰　编著

南开大学出版社

内 容 提 要

本书在引导读者全面了解 Windows 环境下编程的基本风格、组织结构和系统工作原理的基础上，详细介绍了文本显示、菜单、计时器、控制框、对话框、图形设备接口、动态数据交换和多文档接口等内容。本书力争把握关键问题，既描述彼此之间的紧密联系，又解释容易混淆的地方，既深入讨论每个实例，又将所涉及的函数和结构解释清楚。

本书内容丰富、全面、系统、具体，结构清晰，每章都包含了大量的演示程序实例和完整的程序清单。它既可以作为大学计算机专业的本科生和研究生的教科书，又可以作为软件工程技术人员的参考书。

Windows 应用程序编程技术

王孝真　刘秋兰　编著

南开大学出版社出版

（天津八里台南开大学校内）

邮编 300071 电话 23508542

新华书店天津发行所发行

天津宝坻第二印刷厂印刷

1998 年 9 月第 1 版　　1998 年 9 月第 1 次印刷

开本：787×1092　1/16　　印张：25

字数：638 千　　印数：1-2000

ISBN 7-310-01101-5

TP·83　定价：40.00 元

前　言

由 Microsoft 公司研制开发的 Windows 系统是一个基于图形的多任务、多窗口环境。自从 1983 年 11 月颁布初版 Windows 系统以来，经过几次更新，目前已推广使用了多种版本，主要有 Windows 3.1，Windows 3.2 等 Windows 3.x 系统和 Windows 95 系统以及 Windows NT 系统。

当前 Windows 系统风靡世界，它能为应用系统提供统一的用户界面，允许用户同时运行多个程序，并且能够以多种方式进行应用程序间的数据交换和共享，使得广大微机用户耳目一新，现在它已成为当今世界最受欢迎的软件开发环境之一。Windows 系统的网络功能以及对应用多媒体技术的支持更为各行业微机应用软件系统的发展展示出美好的前景。因此，许多著名的软件公司为了适应 Windows 的发展，都在 Windows 开发环境下开发软件，竟相推出在 Windows 环境下运行的软件版本，可见 Windows 系统流行之广。在我国，随着 Windows 系统的发展，以标准配置的 486 和 586 微机为工作平台的 Windows 3.x 系统和 Windows NT 3.x 系统及其开发工具应用在科学研究的各个领域，其汉化系统比较完备，中文技术资料比较齐全，迫切要求程序设计人员必须改变在 DOS 环境下开发软件的模式，只有熟悉和掌握 Windows 环境下新的程序设计思想和方法，才能适应软件发展的潮流。

为了开发 Windows 的应用程序，软件开发者必须全面了解 Windows 系统的工作原理及其庞大的应用程序接口(API)和图形设备接口(GDI)。Windows 系统 API 开启了工程革新之门，为软件人员免除了所有直接写硬件的操作，这是以前的 DOS 所做不到的，任何能通畅运行 Windows 机制的计算机都能支持全部通过 Windows 应用程序接口(API)编制的软件。在 Windows 系统下，用户关注的是硬件的性能，而不是硬件的一致性，因而，Windows 系统 API 能够真正地做到与硬件设备的无关性。

然而，当我们试图转向 Windows 程序开发时，许多人面对上千个 API 函数、三百多个 Windows 消息以及众多的数据结构和消息结构组成的应用程序接口，感到 Windows 程序开发似乎束手无策。利用 Windows 系统所提供的环境进行软件的开发，确实要比其它环境开发软件复杂得多，编程风格也区别于常规的高级语言程序设计手段。

实际上，对于大多数软件工作者而言，只要打破传统的程序设计方法并全心全意地接受 Windows 程序设计思想，认真地掌握基本的代码结构和程序设计方法，学习 Windows 编程本身并不比学习其它结构化程序设计困难。如果以一种开放的思维方式去学习 Windows 编程，就会很快地适应在 Windows 系统下新的程序设计的概念和方法。

为了理解 Windows 编程的特点，首先应该弄清楚标准 DOS 应用程序和 Windows 应用程序之间的主要区别，这其实也是学习 Windows 编程的关键。我们编写在 DOS 环境下的应用程序时，是假定以标准操作环境为前提并假设该环境由基本字符作为输入和输出，应用程序单独占有对系统的存储权及对输入输出设备的使用权。但 Windows 系统的应用程序和其它应用程序必须共享包括计算机 CPU 在内的所有资源，所以 DOS 环境下编程的假定在 Windows 环境下不再成立。另外 Windows 应用程序还必须通过基于图形的显示器、键盘和鼠标器与用户进行交互会话。尽管在 DOS 下也允许人机交互作用，但这是完全不同的两种风格。

DOS 应用程序主要使用顺序的、过程驱动的程序设计方法。顺序的过程驱动的程序有一个明显的开始、明显的过程及一个明显的结束，因此程序能直接控制程序事件或过程的发生和

执行顺序。虽然在这种程序中也允许含有许多异常处理的方法，但这样的异常处理也仍然是顺序的和过程驱动的结构。

区别于DOS程序设计方法的是，Windows程序设计方法使用事件驱动。事件驱动程序不由事件的顺序来控制，而是由事件的发生来控制，它以一种非顺序的方式处理事件，从而回避了顺序的、过程驱动的方法。

事件驱动程序设计是密切围绕着消息的产生与处理而展开的，消息是由事件的发生和处理而产生的。例如无论何时按下一个键盘按键或鼠标按键，都会发出一个具体的消息，而在松开它们时，又会发出另一个消息。作为一个Windows程序员，就是要对正在开发的应用程序所要发出或接收的消息进行排序和管理。由于Windows消息是事件驱动的，所以消息不会以任何预定义的顺序出现。

为使读者不至于茫然不知所措，本书先以一些简单的Windows应用程序为实例，介绍Windows应用程序设计的一些基本概念和Windows应用程序的主要组成部分，使读者对Windows应用程序设计有一个初步的了解，然后再逐步地在模仿的基础上加以理解，由浅入深，以实用为前提，深入到Windows程序设计的各个部分，让读者顺利地从“只知其然”过渡到“知其所以然”。

但是如果仅用一些简单的Windows应用程序为例，介绍Windows应用程序的一些基本概念，说明Windows应用程序的组成部分，则只能使读者对Windows应用程序有初步的了解。如果只是罗列一大堆详尽资料，又会使人陷入难以自拔的窘境。程序设计最好的教学方法是仔细地选择和描述一些标准的程序实例，把主要内容合理分类，既独立地讨论各部分的特性，又描述彼此之间的紧密联系，既能将每个实例深入讨论，又将所涉及的函数和结构解释清楚。当然面面俱到是不可能的。本书力争把握关键问题，既解释容易混淆的地方，又便于使读者举一反三，触类旁通。

C语言是基于Windows应用程序的首选开发语言，Windows的许多编程特性都是针对C语言而设计的。尽管基于Windows的应用程序也可以用其它语言开发，但是C语言访问Windows函数最直接也最简单。而且C语言也是目前最流行也最便于使用的编程语言。正是由于这个原因，所有语法描述和程序范例都是用C语言写成的。在开发过程中，既可以使用Microsoft C/C++，也可以使用Borland C++等。本书所有的程序代码都是用Borland C++写成的，稍加修改，不难移植到Microsoft C/C++系统中。对于熟练掌握C语言、有多年编程经历的程序设计人员而言，只要熟悉Windows应用程序的开发环境，了解Windows应用程序的组织结构，掌握Windows应用程序的编程和调试并不十分困难。

Windows系统API使用了标准C语言中没有的类型、宏和结构。这些类型、宏和结构使得基于Windows应用程序的建立工作更为简单，也使应用程序源代码更加清晰、易懂。本书中讨论的所有类型、宏和结构都在Borland C++的C语言的相关头文件中给出定义。

读者成者程序员，在能够编写Windows应用程序之前，必须首先了解Windows和DOS这两个不同的操作系统环境中应用程序源代码的差异，其中包括结构、类型、消息和运行方式。在第一章，以一个完整的应用程序介绍Windows环境下编程的基本风格以及组织结构。本章中还介绍了编程环境和编程工具。为了读者学习方便，本章中还给出了一个样板程序，它将用于本书以后的演示程序中，并以Template.I为名的include文件形式出现。

第二章介绍文本信息的显示技巧，包括字体的选择和使用。

第三章主要介绍键盘输入和鼠标输入，其中包括键盘消息处理以及虚拟键码，在此将会看

到一个简单的文本编辑程序，用来展示虚拟键码是如何产生和被解释的。对于 Windows 这样的系统，鼠标是与键盘同样重要的。没有鼠标，使用 Windows 系统几乎是不可能的。本章还介绍各种鼠标消息，同时还附有一些演示程序以阐明这些鼠标消息是如何起作用的。

第四章讨论系统计时器的设置和使用，计时器的利用是周期性地完成某些工作的必要条件。

第五章开始介绍资源，主要讨论菜单资源，因为它几乎用于所有的 Windows 应用程序中。本章详细介绍了菜单的类型、创建和使用、有关菜单的消息处理和与菜单有关的函数等。本章还附带简介了加速键和字符串两种资源类型。本章的大量实例程序充分演示了各种资源的使用方法。

第六章着重介绍各种控制器或子窗口的使用，包括静态控制框、编辑框、按钮、列表框、组合框和滚动杆，它们如同机械零件一样随时用于各种场合，同时也是第七章讨论的对话框的组成元素，因而是学习下一章内容的准备。

对话框的创建和使用在第七章讨论。对话框是应用程序创建的用于获取用户输入的一个临时窗口，使用它来提示用户输入命令的附加信息。除菜单之外，对话框是另一种重要的接收用户输入和控制的资源类型，不论是模式对话框还是非模式对话框，都是 Windows 应用程序的重要组成部分。而组成对话框的成分，就是前一章介绍的各种控制器或子窗口。

前面这七章所介绍的内容是 Windows 应用程序的编程基础，也是较为简易的部分。但它确实是今后各章节作更高层次的探讨的基础。熟练掌握这一部分的基本操作，对于了解和运用随后各章节所介绍的更为复杂和更强功能的操作，是必不可少的。

第八章介绍图形设备接口(GDI)和绘图功能。这一章讨论的内容包括设备场景和映射模式、绘图工具绘笔和画刷、常用的图形函数，还介绍设备无关的位图(DIB)和位图文件结构以及如何捕捉位图。作为演示，展示一些简单的商业用图综合说明本章的有关内容。

随后的三章讨论应用程序之间的数据交换的种种方法。第九章介绍元文件操作，它是另一种绘图操作方式，它允许一个完整的绘图操作被记录下来，并被当前应用程序或其它应用程序所重显；或者使用剪贴板的特性传递给其它应用程序，由它们加以重显。

剪贴板的数据传递是第十章讨论的主题，简单介绍剪贴板的数据格式以及如何使用剪贴板传递数据。动态数据交换(DDE)被认为是 Windows 编程中最困难也最复杂的问题之一，尽管它是实现不同应用程序之间数据交换的有力方法，但多数程序员认为它太难，因而往往避免使用它。动态数据交换管理库(DDEML)提供了一种新方法进行动态数据交换，使得用户可以更简便和更安全地将 DDE 功能放入 Windows 应用程序中。DDE 和 DDEML 的使用和实例在第十一章给以说明。

第十二章讨论多文档接口(MDI)。多文档接口是定义 Windows 应用程序的标准用户接口规范。MDI 应用程序允许用户工作在多个文档上，最简单的实例是让用户同时编辑多个文本，由于此原因，MDI 被广泛用于 Windows 应用程序中。

最后一章讨论打印机驱动程序的使用。由于在实际应用中常常需要打印输出文本或图形，因而要求程序员应熟悉这方面的知识。

本书内容丰富、全面、系统，它既可以作为大学本科生或研究生的教科书，又可以作为软件技术人员的参考书。本书中每章都包含一些演示程序实例，它们完整的程序清单附在各章末尾，包括资源描述文件、头文件、模块定义文件以及相关的源代码文件等。当然每个演示程序实例仅用于说明某一编程任务是如何实现的，并展示 Windows 系统的某个特征是如何被使用

的，而不是作为一个完整无缺的用户应用程序。因此，编者对演示之外的实际使用并不提供任何意见，如果读者乐意的话，欢迎将这些实例中的成分以任何适当的形式用在自己的应用程序之中，并请对使用中发现的问题提出宝贵的改进意见。

由于编者的水平所限，书中难免有错误和不妥之处，恳请广大读者特别是同行专家批评指正。

编者　1997．10

目 录

第一章 Windows编程初步

对于熟悉DOS环境下的编程人员来说，编写Windows 3.x（Windows NT）应用程序的风格与其所熟悉和接受的编程风格有明显的、突出的不同，以致许多编程人员怀疑自己是否在使用不同的程序设计语言和操作系统。因为，他们在Windows应用程序的C语言源代码中发现含有许多类似PASCAL语言的成分，并且发现那些他们过去在编程中常用到的捷径和技巧不再适用，甚至在某些情况下，这些捷径和技巧对应用程序和操作系统来说是致命的。

本章将通过实例说明这些不同，突出Windows的特点并说明学习Windows编程本身并不比学习其它结构化程序设计语言困难，学习Windows编程只需要学习基本的代码结构和程序设计方法。对于大多数程序员而言，学习Windows程序设计的最大障碍在于必须打破传统的程序设计方法并全心全意地接受Windows程序设计思想，只有以一种开放的思想方式学习Windows编程，才能尽快地适应这个新的程序设计的概念和方法。

Windows程序设计方法与MS-DOS程序设计方法的不同就在于Windows是事件驱动的，事件驱动程序不由事件的顺序来控制，而是由事件的发生来控制。

事件驱动程序设计是密切围绕着消息的产生与处理而展开的，每当发生一个事件时，都会发出一个具体的消息。例如，无论何时按下一个键或鼠标按钮，都会发出一个具体的消息，在放开它时，又会发出另一个消息。当窗口改变大小或窗口内容需要重画时，也要发出相应的消息。消息传递是实现通讯和控制的主要手段，整个系统以消息驱动的方式工作，系统中发生的用户输入操作，显示信息的变化，系统环境参数的改变等所有的事件都打包成消息出现在相应的应用程序及其窗口。另外，在应用程序之间和各窗口之间的通讯和控制也是利用消息完成的，发送者可以通过发消息要求接收者完成相应处理，这些消息可以是系统定义的，也可以是用户自己定义的。程序设计的任务就是为这些消息的处理设计代码或根据消息对相关窗口的显示状态进行调整。作为一个Windows程序员，就是对正在开发的应用程序所要发出的或所要接收的消息进行排序和管理。由于Windows消息是事件驱动的，所以消息不会以任何预定义顺序出现。

本章先简要地介绍Windows应用程序的特点，涉及的对象以及开发的过程，使初学者先接触一些在Windows应用程序开发过程中涉及到的新概念。

1.1 WinHello：一个入门性的Windows程序

传统上，在DOS环境下在屏幕上显示消息"Welcome You to the wonderful world！"仅需编制一个有数行代码的C语言程序即可。而对于Windows系统，要显示这样一段消息却比较复杂，需要大约130多行代码，相当于传统的C代码的许多倍。这主要是因为对于任何Windows应用程序，不管它如何简单，要在一个比DOS复杂的环境下运行，就需要在程序中增加一些最基本的环境准备语句，以适应这一新环境。然而，这并不意味着所有的Windows应用程序都比相应的DOS应用程序大而复杂，恰恰相反，对于大一些的应用程序，它在Windows下的代码要比在DOS下的代码小得多。道理很简单，许多在DOS下要由应用程序自己提供的功能和服务，在Windows下只需作为系统函数直接调用。

与 DOS 环境不同，在 Windows 环境下的源代码除 *.C 或 *.CPP 源程序文件外通常还有一个资源描述文件 *.RC 和一个模块定义文件 *.DEF。

1.1.1 应用程序窗口

本章末尾给出 WinHello 的 C 语言源程序的完整代码。我们暂不看 WinHello 程序的具体内容，先看一下程序运行后显示出的图形式样。图 1.1 表示它运行后所显示的 Windows 窗口式样和缩小后的图标。我们首先介绍一下窗口所涉及到的，能直接看到的对象的含义。

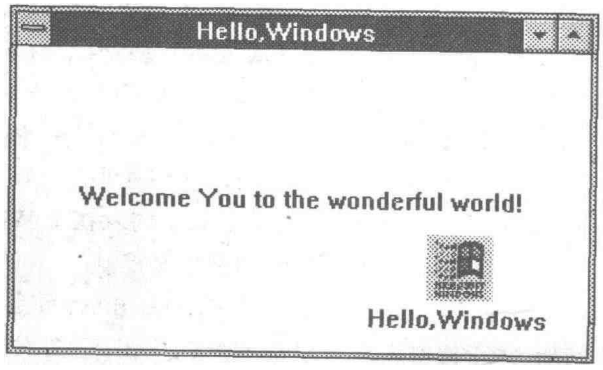


图 1.1 WinHello 程序运行后的图形及其图标

1. 边界

边界是组成窗口边框的两条水平线段和两条竖直线段。它们不仅是屏幕区域的界限，而且还指示出若干重叠窗口中哪一个是活动窗口。在边界上操作鼠标可移动窗口和改变大小。

2. 标题栏

图 1.1 中显示的信息“Hello,Windows”称为标题栏。标题栏用于显示应用程序定义的一行正文，通常是应用程序的名字或说明该窗口的用途，由应用程序在创建窗口时指定。用户可以用鼠标或其它指定设备通过操作标题栏移动窗口。

3. 系统菜单框

标题栏左边的带阴影的内含水平横条的矩形框称为系统菜单框，它是一个小位图。用鼠标单击它或同时按下 Alt 和空格键会弹出如图 1.2 的系统菜单，系统菜单是一个由 Windows 系统创建和管理的菜单，其中含有标准的菜单项设置，用户可通过它改变窗口的大小、对窗口重新定位或关闭应用程序等。

4. 最小化框

标题栏右边带阴影的内含一个向下箭头的矩形框称为最小化框，它是一个小位图，只能用鼠标激活它。单击它使整个窗口缩到一个最小的图像，这个最小的图像称为图标，它是一个 32×32 的位图。图 1.1 中右下角即是图标，原标题栏内的信息“Hello, Windows”显示在它的下面。单击最小化框与在系统菜单中选择最小化(Minimize)菜单项的功能一样。

5. 最大化框

标题栏右边带阴影的内含一个向上箭头的矩形框称为最大化框，它是一个小位图，只能用鼠标激活它。单击它使整个窗口放大并充满整个屏幕，同时 Windows 系统把最大化框变成恢复框。单击最大化框与在系统菜单中选择最大化(Maximize)菜单项的功能一样。恢复框同样

图 1.2 系统菜单

也是一个位图,单击它则把窗口恢复到原先的位置和大小。单击恢复框与在系统菜单中选择还原(**Restore**)菜单项的功能一样。

6.**客户区**

客户区占据整个窗口的最大部分,用于显示正文和图形。应用程序必须提供一个称之为窗口过程的函数,来处理窗口的输入并在客户区显示输出。图 1.1 的客户区内在中心对称位置显示一行正文"Welcome You to the wonderful world !"。

除图 1.1 窗口所包含的组成外,绝大部分的应用程序窗口还包含菜单栏和滚动杆。

7.**菜单栏**

菜单栏紧贴标题栏的下面,列出了应用程序所支持的命令,菜单栏中的项是命令的主要分类。从菜单栏选中某一项通常都会显示一个弹出式菜单,其中的项是对应于指定分类中的某个任务,用户可选择一个菜单项让应用程序完成相应的任务。选择菜单项的方法可以借助于鼠标、定义的热键或者同时按下 Alt 和某个字符等。菜单通常由资源描述文件提供。

8.**滚动杆**

水平和垂直滚动杆是普遍使用的控制器,通常与屏幕显示相联系,用于水平和垂直方向的定位调整,应用程序可以借助于滚动杆移动客户区的内容。例如,显示一个较长文件的字处理应用程序,就得提供一个垂直滚动杆,以便用户向上或向下翻页。对滚动杆的操作必须使用鼠标,在一般情况下,没有鼠标就无法使用滚动杆。

标题栏、菜单栏、系统菜单、最小化框和最大化框、改变大小的边框以及滚动杆统称为窗口的非客户区,它们差不多都是由 Windows 系统直接管理的;应用程序则是管理窗口的其它事情,特别是管理客户区的外观及各种操作等。

1.1.2 WinMain 过程(函数)

正如 DOS 环境下的 C 程序都含有一个名为 main 的主函数作为其入口函数,任何一个 Windows 环境下的应用程序也有一个类似的入口点,其名为 WinMain(要注意区分大小写,对于所有的 API 函数也要严格遵守这种规定)。同样地,在 DOS 的 C 程序的主入口函数说明中可以包含检索命令行参数的手段,在 Windows 的入口函数 WinMain 中利用参数 lpszCmdLine 提供了类似的手段,不过在 Windows 下,几乎不使用命令行参数。

与 DOS 不同,WinMain 的参数不管是否会用到,必须按如下的次序和格式严格说明。这是因为 WinMain 主函数只能被用户间接调用,实际的调用格式是由 Windows 系统提供的,不

再存在DOS下调用格式的灵活性。

在Borland C/C++中,可以用斜杠"//"来表示源程序代码中的一个单行注释,当然,C语言中的"/* ... */"注释符仍然有效。在资源描述文件中亦可用分号(";")引导一个单行注释。下面给出WinMain过程说明,并详细介绍其各部分的功能:

```
#include  <windows.h>
char  szAppName[] = "WinHello";
int   PASCAL  WinMain(HANDLE  hInstance,//当前实例
                HANDLE  hPrevInstance,      //前一实例
                LPSTR   lpszCmdParam,       //命令行参数
                int     nCmdShow)           //如何显示窗口
  {
   HWND                hwnd;
   MSG                 msg;
   WNDCLASS            wc;
   if (! hPrevInstance)
     {
      wc.style = CS_HREDRAW|CS_VREDRAW;                    //窗口类风格
      wc.lpfnWndProc =(WNDPROC)WndProc;                    //窗口过程函数指针
      wc.cbClsExtra = 0;                                   //窗口类附加数据
      wc.cbWndExtra = 0;                                   //窗口附加数据
      wc.hInstance     = hInstance;                        //实例句柄
      wc.hIcon     = LoadIcon(NULL,IDI_APPLICATION);       //最小窗口图标
      wc.hCursor = LoadCursor(NULL,IDC_ARROW);             //窗口内光标
      wc.hbrBackground= GetStockObject(WHITE_BRUSH);       //着色窗口背景的画刷
      wc.lpszMenuName = NULL;                              //菜单指针
      wc.lpszClassName = szAppName;                        //窗口类名
      RegisterClass(&wc);                                  //注册窗口类
      }
   hInst = hInstance;
   hwnd= CreateWindow(                                     //创建窗口
            szAppName,                                     //与注册时相同
            "Hello,Windows",                               //窗口标题
            WS_OVERLAPPEDWINDOW,                           //窗口风格
            CW_USEDEFAULT,                                 //初始X坐标
            CW_USEDEFAULT,                                 //初始Y坐标
            CW_USEDEFAULT,                                 //窗口的宽度
            CW_USEDEFAULT,                                 //窗口的高度
            NULL,                                          //父窗口句柄
            NULL,                                          //窗口菜单句柄
            hInstance,                                     //实例句柄
            NULL);                                         //附加数据
   if(! hwnd) return(FALSE);                               /创建窗口失败FALSE
```