精通

Internet

C++

for Windows 95/NT

胡 俭 丘宗明 翟继增 游建设 编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
URL: http://www.phei.co.cn

精通 Visual C++ for Windows95/NT

胡俭 丘宗明 翟继增 游建设 编

電子工業出版社
Publishing House of Electronics Industry

内 容 简 介

Visual C++是目前在国内应用最广泛的高级程序设计语言。本书以在 Windows95 和 NT 平台上进行编程的 Visual C++4.0 为例，介绍微软基础类(MFC)编程的概念和方法。全书共分三篇，特别适于使用过 C 语言，但没有 Windows 编程经验的程序员阅读，旨在帮助读者顺利地从 C 语言 16 位编程过渡到 MFC 在 Windows 中进行 32 位编程，使读者能够充分利用面向对象技术带来的巨大优越性。

本书适于从事计算机软件工作的程序员和工程师阅读，也可作为相关专业的师生的教学参考书使用。

书　　名：精通 Visual C++ for Windows 95/NT
编　　者：胡俭　丘宗明　崔继增　游建设
责任编辑：郭立
特约编辑：刘艳芳
印 刷 者：北京科技大学印刷厂
装 订 者：三河市双峰装订厂
出版发行：电子工业出版社
北京海淀区万寿路173信箱　邮编 100036　发行部电话 68214070
URL:http://ww.phei.co.cn
经　　销：各地新华书店经销
开　　本：787×1092/16　印张：35　字数：852千字
版　　次：1997年5月第一版　1997年5月第一次印刷
印　　数：6000 册
书　　号：ISBN 7-5053-3934-6
TP·1706
定　　价：40.00 元

前　言

微软基础类库(MFC)是微软为 C++程序设计者提供的一个面向对象的 Windows 编程接口，而 Visual C++ 4.0 则是支持 MFC 类库、用于开发 Win32 应用程序的可视化开发工具。本书的目的是试图通过对 Windows 编程基本概念和方法的介绍、对 MFC 类库较详细地讲解以及对 Visual C++ 4.0 开发工具的一般性说明，使用过 C 语言编程的读者能够尽快地利用 Visual C++ 4.x 和 MFC 编制 Windows 95 和 Windows NT 的应用程序。

本书共有三篇：第一篇“基础篇”，主要提供 Windows 应用程序开发的基础知识，介绍一些有关 Windows 系统的发展背景、术语、概念、应用编程接口 API 和系统的三个基本内核——即核心(KERNEL)、图形设备接口(GDI)和用户接口(USER)，以及 C++语言编程基础和面向对象的编程原理。

第二篇“ MFC 篇”，着重介绍 MFC 类库中的各个主要类的作用和使用方法，内容包括 MFC 的组织与结构、窗口的创建、对话框的创建和管理、菜单和工具条的建立、鼠标和键盘输入、 GDI 图形输出和内存管理等内容。

第三篇“应用篇”，介绍如何利用 MFC 的文档/视窗架构建立应用程序，以及 Visual C++ 4.0 开发工具的使用方法，本篇最后提供了大量的示例程序。

本书由具有实际编程经验的软件工作人员编写，内容全面、示例丰富、图文并茂、深入浅出。在叙述上尽量采用对比的方法，如对 C 与 C++的异同、 Win16 与 Win32 的区别、 Windows 95 与 Windows NT 的差异等都进行了较深入地剖析。我们希望通过这样的讲解，可以帮助具有不同编程经验的读者，特别是没有 Windows 编程经验的 C 语言读者，能够顺利而轻松地由各自的相应水平快速提高到利用 MFC 编制 Windows 95 或 Windows NT 的 Win32 应用程序的水平。

由于本书立足点是提供运用 MFC 编制 Windows C++应用程序的基础，重点在概念和方法的介绍上，且 MFC 本身又具有很强的移植性，因此本书除了适用于软件编程人员学习 Windows 编程之外，也适于作为大专院校讲授 C++的参考教程。

为了配合理解和掌握书中内容，本书提供了大量的示例程序，我们建议读者配置一个与 MFC 库配套的编译器，如 Windows 95 或 Windows NT 下的 Visual C++ 4.x 、 Unix/Xwindow 下的 Wind/u ，以便能够运用书中示例更好地帮助理解和掌握本书的内容。

鸣　　谢

首先我们要感谢电子工业出版社领导、编辑和工作人员，没有他们的关心和支持，便不可能有本书的出版。他们为本书提供了大量的参考资料和 Visual C++ 4.0 的工作环境、设计了本书的封面，并最终促成本书的出版、发行。

我们要特别感谢刘艳芳，她为本书进行了全面而仔细的校对，并提出许多中肯的修改意见。本书占用了她大量的宝贵时间。

感谢胡俭、丘宗明、翟继增、游建设，他们是本书的主要执笔者。参加本书编写的还有徐铭政先生。我们也要感谢王曦、韩燕、王燕、杨美婷、刘小春、姜桥，他们对本书的编写、版式设计、录入和排版提出了很多建议，并付出了大量艰辛的劳动。

目　录

第一篇　基础篇

第二篇 MFC 篇

第三篇 应用篇

第一篇　基 础 篇

Windows 编程是一种新的编程方式，无论程序控制、资源使用，还是程序的调试及运行，与 DOS 程序的开发都有很大的不同。

DOS 程序是单线程、顺序执行的，在运行时它独占系统资源，其输入和输出完全由程序本身控制；而 Windows 程序则是多线程、事件驱动的，在运行时单个程序不能独占系统资源，系统资源被多个同时运行的程序所共享，其输入和输出必须通过 Windows 的输入/输出功能来实现。这些差异导致 Windows 程序的设计与编制依赖于 Windows 所提供的编程机制和系统功能，因此，Windows 应用程序开发人员必须掌握这些编程机制、系统功能和应用开发环境及其工具。

本篇主要提供 Windows 应用程序开发的基础知识，介绍一些有关的术语、概念和 Windows 系统的三个基本内核（即系统核心(KERNEL)、图形设备接口(GDI)和用户接口(USER)），以及 C++基本知识和面向对象编程原理。

第一章 Windows 应用编程接口

以前，程序员若想编制 Windows 应用程序，必须采用 C 语言、使用微软的软件开发工具包(SDK)、并直接调用 Windows 应用编程接口(API——一组由操作系统提供的骨干函数)，然而使用 SDK 开发 Windows 应用程序相当繁琐。

如今，出现了许多开发 Windows 应用程序的不同语言的编译器，如 C++、Pascal、FORTRAN、BASIC 和 COBOL 编译器等等，程序员有了较大选择余地，C 语言已不再是开发 Windows 应用程序唯一的选择了。

用于编制 Windows 应用程序的语言和开发工具都有一个共同特点，即它们最终都将依赖于 Windows API，因此，深入理解 Windows API，有助于更好地开发 Windows 应用程序和充分发挥 Windows 系统的能力。比如，在利用 Visual Basic 开发应用程序时，可以通过直接调用 Windows API 函数完成某些特定的功能，甚至可以从应用系统如表格处理器（Speadsheets）和文字处理器中直接调用原始 API 函数实现用户所期望的功能。但是，原始 API 函数的数量巨大（数以千计），调用的组织性很差，直接调用原始 API 并非易事，它需要长时间的学习和经验积累，为了简化和方便 Windows 应用程序编程，软件开发人员开发出了封装 Windows 数据结构和 API 函数的 C++类库，它为 Windows 编程开辟了一个新天地，目前，市场上这种 C++类库很多，如 Borland 公司的对象窗口库(OWL)和微软基础类库(MFC)，本书将主要介绍 MFC 库。在 MFC 库中，关键的 Windows 数据结构和大多数 API 函数都被封装起来，许多 MFC 类的成员函数都和 Windows API 函数具有相同的名字，实际上，大多数情况下，MFC 类的成员函数只是对 Windows API 函数的调用，MFC 类库的基础仍是原始的 Windows API。

各种版本的 Windows 操作系统只支持两种不同的编程接口，即 16 位的接口 Win16 API 和 32 位的接口 Win32 API。这两种接口非常相似，MFC 与它们都兼容，因此，只要有一个与 Win16 或 Win32 兼容的编译连接器，就能很容易把 MFC 程序联编成 Win16 或 Win32 的运行代码。尽管 MFC 试图缩小 Win16 和 Win32 之间的差异，但同时了解这两种 API 能更好地发挥它们各自的长处。

操作系统与应用编程接口

准确地理解和正确地区分操作系统与应用编程接口是十分重要的，对于大多数程序员而言，它们是同义词，因为多数操作系统仅有一种编程接口，譬如MS-DOS和UNIX操作系统，而Windows系统，从它的第一版本开始就支持多种编程接口，如Windows 1.01既可运行Win16程序又可运行MS-DOS程序。

操作系统是个软件产品，可以装在软盘、CD-ROM或计算机系统硬盘上，一个操作系统可以支持多个编程接口，如Windows NT支持五个编程接口：MS-DOS、Win16、Win32、POSIX(可移植操作系统接口)和OS/2控制台API（仅适用于Intel x86机），而Windows 95支持三个编程接口：MS-DOS、Win16和Win32。

对于各种版本的Windows，执行文件一般都只调用一种编程接口，例如，程序CALC.EXE不能同时调用Win16和Win32的API函数，即使Windows NT拥有POSIX和OS/2接口的支持，它的一个执行文件也只能遵循一种API，当程序载入内存运行时，操作系统的加载器会将执行文件与对应的API联系起来。

Win16是用于16位处理器的16位的API，它仅能处理16位数值，而Win32是为新一代32位CPU而建的32位API，它能处理32位数值(有时64位数值)。一般地，微软的Windows对这两种接口都会提供支持。现在，程序员正逐步地移向Win32开发Windows应用程序。

1.1 16位Windows编程接口

Win16是在1985年随Windows第一版发行的，它是Windows 1.x、2.x、3.x主要的编程接口，主要用于80年代中期市场流行的16位CPU 8088和80286。由于Intel公司32位处理器向下兼容，所以Intel 80386、80486和“奔腾”系统都能运行Win16程序。Win16 API是Windows 3.0的主要编程接口，而Windows 3.0在1990年5月打入市场不久就深受欢迎。但是，Win16很不完美，所以微软要引入Win32；然而，Windows 3.0的成功却要求Win32尽可能与Win16兼容，但又不拘泥于Window 3.x那些陈旧而又不易移植的特征。

Win16程序两个显著的特点是其支持的文件格式和内存结构。Win16的执行文件是以“段”来存储代码和数据的，在16位Intel处理器中，段是内存的寻址单位，它可以小至一个字节，大至64K字节。内存分段当初是为了支持老的8位处理器代码的移植，虽然这是一个可行的内存解决方案，但当用到大于64K字节（段的最大尺寸）的数据时，内存分

段就非常令人讨厌，此时需要将整块数据分割成小于 64K 的若干块分别处理。Win16 API 的另一个特点是注重使用 16 位短整数(short)，其值很容易存入早期 Intel 处理器的 16 位寄存器中；只有在极少数情况下，才会用到 32 位数值。例如，Win16 的图形设备接口(GDI)库以 16 位带符号整数作为绘图坐标，域值范围从-32,768 到+32,767 （Windows95 仍依赖于 16 位 GDI，这就是在 Windows 95 中运行的 Win32 程序只能处理 16 位图形的原因）。

Win16 API 的开发者们知道编写一个可移植的 API 具有很大的价值，但由于多方面原因而没能做到，其中原因之一就是 Win16 API 没有内置对文件输入输出的支持，而是依赖于 MS-DOS 读写文件。程序员们为了增强其 Win16 程序的可移植性，通常采用 C 运行库函数来实现，但是，由于有些文件系统功能只能通过调用 MS-DOS 中断才能实现，因此，即使编得最好的 Win16 程序也会包含依赖于 MS-DOS 的代码。例如，若要查询硬盘卷名，就需要调用 MS-DOS 中断。

Win16 依赖于平台的特性明显地表现在一些专门访问 Intel 内存的函数中，例如，函数 AllocSelector( )、AllocCStoDSAlias( )、SetSelectorBase( )、GlobalDos( )、AllocGlobal-PageLock( ) 等，虽然它们不属于原始 Windows API，但这些函数却帮助解决了 Windows 发展过程中出现的一些特殊问题。因此，不管原因如何，Win16 应用程序中只要有这些函数就表明其依赖于特定的处理器而难以移植到其它处理器上。

另一个将部分 Win16 程序捆绑在 MS-DOS 上的特征是 MS-DOS 的驻留程序(TSR: Terminate-and-Stay-Resident)。MS-DOS 作为一个单任务环境，通常一次只能运行一个程序，然而，通过载入 TSR 并将其驻留于内存之中，便可以为其它应用程序提供服务，这种特点使 TSR 成为设备驱动程序理想的选择，而实际上，这也是许多硬件适配器开发商所采用的手段，大量的 MS-DOS 程序主要是通过 MS-DOS 的 TSR 来作设备驱动用以控制特定硬件设备的。Win16 程序可以利用 MS-DOS 的 TSR，这种特点使 MS-DOS 软件开发人员可以将其应用系统移植到 Windows 上，作为一种短期过渡手段，微软允许 Win16 程序利用 TSR，但希望 TSR 开发者能尽快地编制与 Windows 更为兼容的驱动软件，而如果这些程序要在 Win32 中运行，就必须用 Windows NT 或 Windows 95 系统下的设备驱动程序替换掉 MS-DOS 的驱动程序。

尽管 Windows 3.x 系统下的 Win16 API 具有分段式内存结构、面向 16 位运算和面向 MS-DOS 的特点，但它在市场上仍取得了巨大的成功，这一成功促使许多硬件平台（包括非基于 Intel 的系统）都支持 Win16 程序，表 1-1 列出了 Win16 的支持平台。

在 Win16 程序中即使有面向特定平台的函数调用，仍能移植到其它不同的平台上，从表面上看，这种 API 似乎具有可移植 API 的一切特性，它不同于 Win32 API 需要重新编译源码来适应不同的处理器。Win16 具有在不同的硬件平台之间二进制兼容的优点，如表 1-1 所示，Win16 程序通过 Intel x86 指令集仿真，甚至可以在非 Intel 处理器上运行。

虽然 Win16 程序通过仿真具有很强的可移植性，但却需要付出很多性能上的代价，比如 Win16 程序能够在 UNIX 工作站上运行，但在大多数情况下比运行在基于 Intel 处理器的系统上要慢得多，这种可移植性作为一种短期应付手段还是有些用途的，但实际上没有人愿意花 RISC 工作站的钱来换取 Intel 80386 的性能。

表 1-1 支持 Win16 的平台

处理器	操作系统
Intel x86(80386 和更高)	Windows 3.x，Windows 95 和 Windows NT（内部二进制码支持）
Silicon Graphics MIPS R4000	Windows NT 3.1 和 3.5 上的 Win16 和 Intel x86 仿真
Digital Alpha	同上
Motorola PowerPC 601	同上
HP 工作站和 Sun SPARC	HP-UX 和 Sun OS 上的二进制仿真及 WABI（Windows 应用二进制接口）仿真
其它	UNIX 上的源码兼容及 Bristol 技术公司的 Wind/U
Intel x86 (80386 和更高)	内部机器指令和 OS/2 2.x 上的 Win16 API 仿真
其它	通过 Insignia Soft Windows 在 Macintosh 系统 7 和各种不同的 UNIX：IRIS、NextStep、HP、DEC、SUN 和 SGI 上的仿真
Intel x86	带 WINE（Windows 仿真器）的 Linux 操作系统；仅适用于 Intel 版本的 Linux

微软好象有一个策略，那就是在其他人抛弃其产品之前，自己先将其丢弃。应用程序开发者们可以在许多图形 API 中进行选择，包括 Macintosh、Motif 和 OS/2 Presentation Manager，而微软却特别推出了 Win32 作为其 Win16 API 的后继产品。虽然 Win32 与 Win16 兼容，但 Win32 抛弃了 Win16 必须依赖 Intel x86 处理器和 MS-DOS 操作系统这种特定平台的特点。

1.2 32 位 Windows 编程接口

微软于 1991 年开始研制 Win32 API，其首要的设计目标是 Win32 与 Win16 API 兼容，由于 Windows 3.x 的成功大部分要归功于众多的 Win16 应用系统，因此，只有让软件开发者能容易地将 Win16 代码移植到 Win32 API 上才有实际的意义。第二个设计目标就是充分利用 32 位处理器的能力。随着硬件的发展，内存和 CPU 越来越便宜，开发 Win32 不仅仅是为了适应当前 32 位（和 64 位）处理器，而且还要考虑正在开发的新的处理器。为了摆脱操作系统对 Intel 处理器的依赖性，Win32 的第三个设计目标就是可移植性，使其可以运行在各种处理器平台上。以前，微软的操作系统是建立在 Intel 的处理器基础上，而现在微软决定通过增强 Win32 的可移植性来加强对其它处理器的支持，虽然微软的 Windows 95 操作系统只能运行于 Intel 平台，但是，它的 Windows NT 操作系统已被移植到许多非 Intel 的处理器上。