计算机软件工具应用与开发系列

Visual C++ 6.0 编程指南

鸿志创作组　编著

科学出版社

计算机软件工具应用与开发系列

Visual C++ 6.0 编程指南

鸿志创作组 编著

科学出版社
1999

内 容 简 介

本书介绍 Visual C++ 6.0 的基本知识、编程技术以及常用的一些技巧。全书分为三篇。第一篇介绍有关 Visual C++ 6.0 和 Windows 程序的基本知识，包括 MFC 的一些基本知识等；第二篇首先讲解 AppWizard，ClassWizard 以及文本编辑器的使用，然后以实际例程讲述文档界面、数据库管理等应用程序的开发以及对话框、菜单、状态栏和工具栏等的基本编程应用；第三篇着重介绍 ActiveX 控件、调试器的使用和自定义 Visual C++ 6.0 等内容。并在书后附有 Visual C++ 的 Internet 网上资源等内容。

本书内容丰富，适合于从初学者到中、高级程序员的所有 Visual C++ 开发人员。

图书在版编目（CIP）数据

Visual C++ 6.0 编程指南/鸿志创作组 编著．－北京：科学出版社，1999.5

（计算机软件工具应用与开发系列）

ISBN 7-03-007453-X

Ⅰ.V… Ⅱ.鸿… Ⅲ.C 语言-程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（1999）第 08792 号

科学出版社 出版

北京东黄城根北街 16 号

邮政编码：100717

新蕾印刷厂印刷

新华书店北京发行所发行 各地新华书店经售

*

1999 年 5 月第 一 版 开本：787×1092 1/16

1999 年 5 月第一次印刷 印张：21 1/4

印数：1—5 000 字数：489 000

定价：28.00 元

前　言

C语言是在1987年由微软开发成功的，它好比是当时在前进中徘徊的软件业的一针强心剂，对整个计算机行业起了极大的推动作用。

伴随着计算机硬件、操作系统和编程思想等的迅速发展和革新，在激烈的竞争趋势推动下，C语言已经发展到了基于Windows95/98/NT等操作系统下程序开发的C++语言，并且有好几块大的分支：如Microsoft的Visual C++和原Borland的Borland C++等等，其中Visual C++的势头更猛，这与微软强大的技术支持和投入有关。

Visual C++已从1994年底的2.0版，经过4.3和5.0等版本，发展到了今天的6.0版，而且与Visual Basic，Visual FoxPro，Visual J++等一起封装在Visual Studio 6.0中。

Visual C++是目前效率最高的C++开发系统，它将优秀的开发环境，集编辑、编译、连接、调试、向导等多项功能集成为一体，所提供的著名的MFC（Microsoft Application Foundation Classes）类库也已成为了业界标准。

对于一名软件开发人员来说，选择功能强大和高效率的优秀开发工具往往会起到事半功倍的作用。面向对象的可视化编程已是目前的大势所趋，其中最流行的开发工具有三种：Visual C++6.0，Visual Basic 6.0和Inprise公司（原Borland公司）的Delphi。作为本书的作者，我们极力向您推荐Visual C++，不仅仅是因为它继承了原C语言的重要特性，还因为它的用户界面、编译调试技术、语言特性等各方面的功能之强大，以及微软公司的技术发展前景。相信在阅读完此书后，您会找到更多的理由和根据。

本书共分为三篇。第一篇的内容比较基础，讲述Visual C++的特点和MFC编程基础，面向有一定C语言编程基础的初级程序员；第二篇的内容是用Visual C++6.0开发文档界面、对话框、数据库管理以及动态连接库等应用程序的一般方法步骤和各种控件应用的重要技巧，面向VC的中级程序员；第三篇则主要讲解关于ActiveX控件的使用、开发以及Visual C++ 6.0的设置等高级应用，面向VC的高级程序员。本书内容丰富，图解详细，循序渐进，易懂易学，不失为一本从入门到精通的完整学习资料。

本书的第一章至第三章由张军同志编写，第四、五章由刘伟平和金文戈编写，第六章至第八章由罗勇开、肖逖共同执笔，第九章至第十五章的内容由黄健和张东完成，最后两章由尚学伟和曹竹平完成。全书由张东同志统稿，由黄健同志审稿。周祖胜、郝水侠、张运涛、胡宏、刘吉强等人也参加了部分编写工作。

在此对这些为本书编写付出了辛勤劳动的工作人员和支持编辑出版本书的同行表示衷心的感谢，殷切希望广大读者提出宝贵意见和建议。

编　者

1999.3

目　录

第二篇 知识积累

第三篇 高级应用

第一篇　初 级 入 门

本篇的内容主要向您介绍一些有关 Visual C + + 的发展进程和最基本的 Visual C + + 6.0 编程基础知识。如果您对 Visual C + + 已经有了一定的基础和了解，并进行过一些 Visual C + + 语言的编程，您可以跳过本篇的内容，直接进入第二篇的学习。

1 关于 Visual C++

1.1 Visual C++简史

我们知道,Visual C++的根源不是开始于 Microsoft,而是 Borland。您可能会记起,给 DOS 带来集成开发环境或称"IDE"的想法的 Turbo Pascal。这里的"IDE"是指某个领域内的另一个缩写。它意味着,编辑器和编译器共同工作,二者可从同一个地方访问。在编辑器中写下源代码,单击 Compile(编译)按钮来启动编译器。当编译器发现错误时,它将编辑光标设到出错语句处,以便于您改正问题。"IDE"的想法是,为程序开发提供一个整体环境,程序员无须从中离开到其他程序中。

C 语言正是在这个时候诞生的(公元 1987 年),Turbo Pascal 发行了 Turbo C。Microsoft 相应地推出类似产品,称为 QuickC。QuickC 作为一个产品单独发售,但它被封装于称为 Big C 的 Microsoft's C 编译器中,Big C 版本号为 5。当时在 C 产品上 Microsoft 的竞争对手有很多,其中包括 Computer Innovations, Datalight, Lattice, Manx 等,这些公司现在已经被别的公司吞并或已经根本不存在了。其他有些公司幸存下来了,如著名的 Borland 和 Watcom(现在是 PowerSoft)等。它们开发的 C 产品继续同 Microsoft 公司竞争。

把 QuickC 同 Big C 放在一起的目的是,使程序员能在 QuickC 方便的"IDE"下编写代码。QuickC 编译起来很快,主要是因为它对代码仅作些微优化。QuickC 优化时只是登记一些变量,插入几条 LEAVE 指令。这样可以使编译时间缩短。程序调试过后,再在 QuickC 中运行时,程序员就能用 Big C 创建一个正式版,它在代码优化方面做得更为正式些。当用 Big C 编译时,程序的大小减少 15%或更多。

QuickC 和 Turbo C 给 C 编程引进了许多东西,但从没有对开发者产生永久性的影响。首要原因就是二者的编辑器都不怎么好(QuickC 编辑器后来合并到 Microsoft QuickBasic 中,现在它作为 DOS 编辑器 Edit.com 仍存在于 Microsoft Windows 95 中)。DOS 下"IDE"的另一个问题就是,它们占据大量内存,只给开发环境下的程序运行留下很少的空间。您不得不经常离开"IDE"来运行和调试程序。许多在开发工作中使用 QuickC 的程序员仅仅使用它的命令行版本。

随后,Windows 3.0 出现了。Windows 3.0 尤其是 3.1,为个人计算机引入了正式的 IDE 时代。内存的约束消失了,而且,如果您准备做 Windows 编程,Windows 环境看来是天生的好地方。人们很容易明白,在 Windows 环境下进行 Windows 编程,会产生更好的结果。Windows 是个优秀的软件,长期在 Windows 下工作,会使您对一个程序应该做什么,不应该做什么有更好的认识。

让许多人惊奇的是,Microsoft 把精力更多地集中在 C 编译器的内部支持上,而不是把它的界面升级到一个新的时代。当 QuickC 7 版发行时,它仍然是一个基于 DOS 的产品,它既

可以在 Windows 下的 DOS 模式下运行,也可以在扩展内存管理器下运行(它的包装盒中带有 Qualitas's 386Max)。做为让步,QuickC 7 版提供了一个名为 Programmer's Workbench 的字符模式的 IDE,它已成为现有标准的累赘。然而,Workbench 展示了自 QuickC 时代以来的自然而然的演变结果。许多菜单中的命令看起来还是很现代化的,例如 New,Open,Save As,Build 和 Open Project 等。

QuickC 7 版对编程工作的重要贡献不是它的"IDE",而是它对 C++ 的支持。Microsoft 首次把编译器指明为"C/C++",用来强调其新的双重性。它不再只是简单地扩展编译器来辨认 C++ 超集的新命令了,而是牵涉到更多的东西。C/C++ 7.0 版也引进了 Microsoft Foundation Class 库 1.0 版,完善了源代码。如果没有 Microsoft 发送给开发者的这套优秀的预写类,C++ 就不会成为现在这么流行的 Windows 编程工具。

微软下一个主要的让步是放弃了多数产品对 DOS 的依赖。Microsoft C/C++ 8.0 版成为真正的 Windows IDE,它就是大家所知道的 Visual C++ 1.0 版。这个名字利用了早期 Visual Basic 的成功,但二者从未互相抗衡过。Visual Basic 允许开发者通过大量单击鼠标和少量代码来建立正在运作的 Windows 程序,而 Visual C++ 仅仅通过向导这样的特殊动态链接库来创建启动器源文件。在第 2 章中,我们将看到:向导挽救了众多开发过程中重复性的工作,这种对 Windows 程序来说非常普通的工作是由 MFC 编写的。

Visual C++ 1.5 版之后,微软决定不再将更多的努力花在支持 16 位编程上。Visual C++ 2.0 版仍提供对 16 位的支持,但从那时起,Visual C++ 仅用来创建 32 位程序。没有 Visual C++ 3.0 版,版本号从 2.0 跳到 4.0,使 Visual C++ 和 MFC 同步,这样就结束了一个小小的混乱来源。然而,这种合并只是短期的,因为 Visual C++ 和 MFC 后来又使用了不同的版本号。

Internet 的流行已经明显地影响了产品设计,在发行 4.0 版时,Visual C++ 引进了为 Internet 编程而设计的新类库。5.0 版也增加了一些新类,但注意力更多地集中在改善产品的界面上,以提供一个更好的在线帮助系统、更高级的宏能力和对在开发者组内进行类和其他代码共享的支持。5.0 版也合并了 Active Template Library,并显著地改善了编译器优化代码的能力。在后面的章节中我们将看到:Visual C++ 6.0 版作了更进一步的改进。

1.2 编程的发展

前面介绍了 Visual C++ 的发展历史,让我们看到了微软公司对其的大力投入。它究竟给编程人员带来多大的方便,为什么要使用 Visual C++ 呢?相信本书后面的内容将会给您一个具体的答案。

进行小规模的程序设计通常是一件令人兴奋的事情。小程序一般能很快地被编译并运行。不愿意编写一些小规模程序的程序员是极少的,因为他总是能从目睹程序的初次运行中得到心理上的满足。事实上,计算机尤其是微型计算机,最初仅仅是设计成用来处理小程序的。在当时,只有少数计算机业余爱好者拥有个人计算机,如果能拥有几千个字节以上的 RAM,那就是一件很幸运的事情了。

随着时间的推移、技术的进步,计算机配置的价格开始下调。磁盘容量愈来愈大,可供使用的 RAM 越来越多;同时,CPU 的运行速度也越来越快。微型计算机也从过去人们对它

的业余爱好变成了现在的一个产业,在这个产业中要求硬件和软件具有更多更强的功能。随着硬件系统的飞速发展,程序在规模上与过去十几年前相比也发生了惊人的变化。但人们很快发现编写一个是原来十倍的程序却不一定要比以前的操作复杂十倍。

运行标准的程序设计语言,往往会出现一些相反的效果:一个程序在规模上增长得有多快,那么它难以处理和难以编制的程度增长得较之还快,并且错误可能增多、并有可能渗透到程序的每个部分,一组代码会同另一组发生冲突。调试一个大程序将变得不再是一种练习而是一种职业了。这是因为标准程序设计语言(按今天的标准来衡量)是为小型项目设计的。某些程序段或子程序经常被其他程序段调用,特别是当它的许多变量是全局变量的情况时更是如此。当程序员在对任何一部分代码加工时,都必须时时牢记整个程序存储和使用数据的方法。在开始调试程序时,程序员就会发现:自己同时在对 10000 行相互联系的代码和数据(即整个程序)进行调试,而不是仅仅对一个个小的易管理的部分进行调试。

1.3 Windows 的流行

Windows 的出现以及后来随之发展的 Visual C++ 等编程工具大大简化了编程工作,为软件行业提供了一个广阔的天空。

下面是为什么在相当短的时间内 Windows 变得如此流行的几个原因:

(1) Windows 允许写出独立于设备的程序。

这就意味着在写应用程序时,不必关心打印机、鼠标、显示器、键盘、声卡、Modem、解压卡、光驱、刻录机等设备的类型。不管使用的硬件如何,应用程序都应该工作得很好。是否这就意味着用户可以使用任何声卡?并非如此。把声卡安装到 PC 机上是用户的责任,在安装时,Windows 要求用户安装驱动程序。因此,Windows 要么接受,要么拒绝这个声卡。如果用户从声卡销售商处拿到的硬件与软件(驱动程序)是按照 Windows 的要求来设计的,那么 Windows 就会接受这个声卡。只要 Windows 接受了这个声卡,所写的 Windows 应用程序就会在这个声卡上工作得很好。这对于其他设备也一样,如打印机、显示器、Modem 和 CD-ROM 驱动器等硬件。

(2) 在用户的 PC 机上已安装有许多代码。

只要安装了 Windows,PC 机上就有了许多与 Windows 相关的软件,这些代码存在于开发人员的 PC 机及用户的 PC 机上。这就意味着在开发人员开始写自己的第一行程序之前,用户在自己的 PC 机上已经有了这个程序的一大半,开发人员不仅不需要写这些代码,甚至不需要自己向用户散发这些代码。

(3)标准的用户界面。

所有 Windows 应用程序的用户界面机制是一样的。例如,不必读应用程序的文档,用户就知道如何执行这个应用程序。可以用出现在应用程序窗口角落上的图标来使窗口最小化,知道 OK 和 Cancel 按钮的含义,知道 About 对话框是什么,甚至在开发人员开始写这个应用程序之前就明白了这个程序的许多特征!

无论用于开发应用程序的编程语言是什么,这些使用 Windows 的理由都是适用的。问题是为什么写 Windows 应用程序要使用 Visual C++,而不是带 SDK for Windows 的"常规"C 语言?

1.4 为什么要使用 Visual C++

Visual C++中的 C++就要求使用 C++来写代码,而不是用以前所说的普通意义上的 C。在本书中,您将学到用 C++编程语言写专业的功能强大的 Windows 应用程序时所需要的 C++的知识。

在学习这本书之前,它并没有要求您必须有 C++经验或 Windows 编程经验。但您需要有一定使用"常规"C 编程语言的编程经历。这就是说,只要有一点 C 的 DOS 编程经验,您就能很容易通过此书学好 Visual C++;当然,如果您已有了一定的 Windows 编程经验,那就更好不过了。

现在已知道了 Visual C++中的 C++了,但 Visual 又是什么呢? 它的意思是:可用键盘或鼠标来完成可视化设计及编写应用程序的许多编程工作。用鼠标来选择按钮或滚动条这样的控件,把它们拖入自己的应用程序,再确定大小——在执行这个应用程序之前就可以看到这个应用程序是什么样子。这是它最大的优点:节约了相当可观的时间(不必编译/链接就可以看到应用程序的样子),只需用鼠标就可以改变编辑框、按钮和其他对象的位置与大小。

Visual C++最强大的特征是被称为 ClassWizard 的程序。ClassWizard 能为程序员写代码! 在工业上这种类型的程序被称作 KASE(计算机辅助软件项目)程序。当然,ClassWizard 并不是万能的程序,您还得告诉它想让它写什么程序。例如,假如要把按式按钮(用鼠标)可见地放在应用程序窗口中,只要放好了按钮,就得写代码以便在有鼠标单击时执行这个按钮。这时就该用 ClassWizard。单击 ClassWizard 的不同按钮,告诉 ClassWizard 来准备所有的通用代码。ClassWizard 作出反应,准备好所有的通用代码,然后指示程序员应在何处插入自己的代码。因此,程序员的工作是在 ClassWizard 准备好的地方加入自己的代码。

Visual C++是一个使用起来让人觉得饶有兴趣的软件包,因此它允许在很短的时间内开发很复杂的 Windows 应用程序。

大家均知道 Windows 3.1 和 Windows 95/98 是分别基于 16 位和 32 位的。本书的内容主要是讲解 32 位的 Windows 95/98 操作系统下的 Visual C++应用程序的开发。至于用于 Windows 3.1(16 位)下早期版本的 Visual C++应用程序的开发可参见早期相应的计算机书籍。

下面将分成两章向您介绍几个基本的概念:C++ Windows 应用程序和 OOP(面向对象编程)。

2 C++ Windows 应用程序分析

如果您已经用 C 在 Windows 下编写过程序，通过下面的学习就能发现在这方面 C++ 是非常杰出的。Windows 编程界面很宽广且很复杂，程序员要在窗口类型、程序的运行方式、窗口维数、内存分配等方面做大量的选择。然而，大多数的 Windows 程序不需有这样一系列的选择，这就是为什么说 C++ 对它们来说是非常完美的。并且所有的选项都围绕类进行，从这些类中能派生自己所需的类。当程序员偶然想完成一些其他事情时，可以把前面的函数替换掉。通过这种方法，细节问题被视图隐藏，并且 Windows 程序界面的"Surface area"会很快收缩到一个可管理的尺寸。

在这一章里，将例举一个用 C++ 编写的 Windows 程序，这个程序是在一个窗口中完成"Hello, world"字符串的显示我们将把程序一步步分割开来进行分析。

2.1 关于 Windows 编程

让我们先来看一看在 Windows 下编写应用程序和在 DOS 下编写有何不同。首先，DOS 程序是连续不断编写的，即一条语句紧跟着一条语句。在 DOS 程序中，程序执行控制或多或少地按程序员的设计进行，并且程序的运行是按语句的编写顺序进行的。从一本关于 DOS C++ 的书中能看到如下的入门程序：

```
# include < instream.h >
void main()
{
  cout < < "Hello";
  cout < < "World. \ n";
}
```

程序控制是一行一行地连续进行的。最后，程序的运行结果"Hello World."将输出到屏幕上。如果还有更多的语句，程序将继续下去，并且是按程序员所设计的循环和顺序进行下去的。然而，Windows 程序却不是这样的。

Windows 环境下，应用程序代表着所有可能的选项(在可见对象的格式中)，并在屏幕上显示这些选项以供用户选择。这个特征是编程的最新方式，此方式叫做事件驱动编程。程序不再完全按用户希望的应用程序流程进行控制。用户的选择表示在所有的选项中，并在程序中做出正确的反应。例如，一个窗口中有三个按钮，很明显，您不可能编写这样的程序，即设想用户是按某种特定的顺序按下它们。

事实上，程序员必须为每一个按钮编写分别不同的代码，并且在通常情况下，这些按钮都能激活。本章将程序划分为更小的几块，其中每一块用来完成一个事件。例如，想显示输

出字符串“Hello, world”,实际编写的程序应该是这样的:

```
void CFirstView::OnDRaw(CDC * pDC)
{
    Cstring hello _ string = "Hello, world.";
    Pdc-> TextOut(0,0,hello _ string, hello _ string,GetLength());
}
```

按上述方法设计的代码是用来处理这种事件类型的:当窗口被“redraw”(刷新)时,典型的Windows C++程序是由上面这样的代码块集中起来的,当然其中还包括另外的"On"事件,如OnMouse,OnButtonPress 等等,事件驱动程序就是按上述方法工作的。大多数情况下设计代码时总是围绕着 I/O 界面进行的,程序没有和编辑器工作模式(例如,插入状态)一样的“modes”模式,实际上,所有的可选项都在同一时刻显示在屏幕等待用户使用。稍后,您将学习到上述这些过程是怎样实现的。

Windows 编程除了进行事件驱动外,还很自然是面向对象的。这个过程从屏幕上看是很容易实现的:选择一个对象,如一个图标或涂刷,并且四处移动它。这个过程密切地对应着面向对象程序。通过 C++能够轻松地把程序变成离散的对象,其中每个对象都具有自己的代码和与它相关的数据。每个对象都不依赖其他对象。

面向对象的程序方式对事件驱动软件来说是功能完美的,因为它可使程序变成离散的、模块化的对象。程序员可以把窗口以及窗口的按钮、文本框等等看作对象。关于面向对象的内容将在下一章中详细介绍。

匈牙利表示法

在开始之前,您还应熟知 Windows 变量的命名约定。Windows 有个约定的变量命名方法——匈牙利表示法(是以此表示法的发明者,Microsoft 开发人员 Charles Simonyi 的祖国命名)。因为 Windows 的程序可能会编写得很长,因此程序中长变量就容易引起混淆。为了便于软件人员记忆匈牙利表示法,用字母表示前缀。这些字母见表 2.1。

表 2.1 匈牙利表示法

前 缀	含 义	前 缀	含 义
A	数组	M _	类的数据成员
B	bool(int)	N	short or int
By	unsigned char(dyte)	Np	near 指针
C	char	P	指针
Cb	字节计数	L	long
Cr,	颜色引用值	Lp	long 型指针
Cx,cy	short(count of x,y length)	S	string
Dw	unsigned long(dwored)	Sz	用 0 终止的字符串
Fn	函数	Tm	文本规格
H	句柄	W	unsigned int (word)
I	integer	x,y	short(x or y coordinate)