第一章 引 言

本章对本书进行了概括介绍，描述了本书的内容和使用方法。本章从解释什么是 Visual C++及其工作环境开始，然后介绍了本书的编排方式，并对每章内容进行了简要概述。

1.1 什么是 Visual C++

Visual C++是为开发 Windows 应用程序而创建的一种新的循序渐进的编程工具。它的高性能及高度集成的开发环境，可使用户比以前更快更方便地创建、建立和调试 Windows 应用程序。Visual C++基于以下两个重要概念：首先，Visual C++在 Windows 环境下运行，所以用户可以使用一组丰富的基于 Windows 的全部工具，这可以帮助用户创建和管理 Windows 下的项目和应用程序；其次，它使用了目前著名的包括新工具的“可视”用户界面（在某些方面类似于 Visual Basic）。

1.2 Visual C++ 2.0 环境预览

Microsoft Visual C++基于以下三个元素：C/C++编译器和连接程序，Visual Workbench 以及 Microsoft Foundation Classes（MFC，Microsoft 基础类）库。其中，作为综合开发环境，每一组件均对 Visual C++的成功作出了重大贡献。

1.2.1 Visual Workbench

在 Visual C++ 2.0 以前的版本中，Visual Workbench 是 Visual C++开发环境的核心（在 2.0 版本中，整个环境只称为“开发环境”）。Visual Workbench 包括了一个优秀的代码编辑器，且该代码编辑器具有许多卓越的功能，如自动标记功能——一旦用户在放置代码的地方做了标记，则下一行将自动地在同一点处开始放置，该功能对缩进排列代码非常有用。Visual Workbench 提供了完整的项目管理，可使用户将各种代码模块连接到一个项目中，然后该项目可作为一个单元来处理，以便用户建立自己的应用程序。资源编辑器允许用户创建和编辑各种类型的 Windows 资源，而且集成的程序调试器允许用户在监控方式下运行自己的程序、检查和改变数据以及在程序中设置断点。最新的 2.0 版甚至允许用户实时调试，以便当错误刚出现时，用户可迅速转入程序调试器。

Visual Workbench 也采用了 Microsoft 的专用辅助工具 AppWizard 和 ClassWizard，这两种工具可使用户很容易地创建和定制基于 MFC 的应用程序。用户可使用它们从 MFC 中选择模块，以建立初始的框架程序，然后把新的函数和类加到该框架中。正如本书练习中介绍的那样，这种工具很有益处，它可以帮助用户得到所需要的各种完整的支持，而不会被 MFC 的复杂性所吓倒。

1.2.2 Microsoft Foundation Classes 库(MFC)

就本书而言,由 Visual C++ 2.0 提供的编程环境就是学习和使用 Visual C++很有力的理由。尽管如此,不论用户是否相信,还有更强有力的理由,即该程序可以访问 MFC 库。

与其它任何开发方法相比,MFC 使得用 C++编写 Windows 应用程序更快且更方便。此类库由一系列代表应用程序框架的 C++类构成。这些类被设计成一起使用,以建立应用程序结构框架,它可提供大部分用户界面和功能,而这些正是用户在 Windows 应用程序中期望得到的。因为应用程序框架使用 C++类,所以对用户而言,仅需要通过采用提供新函数的新类去替换适当的类,这样可方便地扩展框架应用程序功能。该新函数正是用户希望加入到内置函数中的函数。

Visual C++环境通过内置在 Visual Workbench 中的两个 Wizard(AppWizard 和 ClassWizard)连接到 MFC 中。这些工具可使用户既方便又安全地选择和定制自己的框架应用程序,并增加新的处理功能,该功能可直接连接到当前的代码库中。另外,这两个工具均可生成代码,调用 MFC 后可完成基本任务。在本书的后面,读者将会发现,当用户使用这些工具时,Wizard 是采用大部分 MFC 功能的最重要组件。

1.2.3 Visual C++版本

目前,已有多种版本的 Visual C++:版本 1.0、版本 1.5 及当前版本,同时还包括为 16 位开发环境研制的 Visual C++ 1.51 和为 32 位开发环境研制的 Visual C++ 2.0。

本书尽可能覆盖了各种版本的 Visual C++。此书的编写目的是帮助用户学习 Visual C++的基本内容。从第一版开始,其环境就被优化设计了,因此用户在本书中学到的内容对所有的版本都适用。尽管如此,本书还是主要面向拥有 1.5 版和 1.51 版的用户。所以,每一帧画面和例子都是基于这种版本的。如果用户使用的是 1.0 版或 2.0 版,则在这些方面需要做些小幅度调整,而且用户屏幕上所显示的内容与本书所看到的会略有不同。本书给出的所有代码均可以由上述版本的 Visual C++生成,且不应该会出现什么大的困难。因此,读者所需的调整量很少。

Visual C++的第一版(1.0 版)有特定的限制。特别地,该版不支持 OLE 2.0,OLE 2.0 仅包含在 1.5 版及其后续版本的 MFC 中。如果使用 1.0 版,用户将不能做第十三章中的练习,因为这些练习需要 OLE 支持。要做这些,就得需要支持 OLE 2.0 的 MFC 版本。

运行 Visual C++ 2.0 的用户需要阅读第十四章。开始练习前,必须把应用程序代码由 16 位转换成 32 位。尽管在 Visual C++ 2.0 中的主要变化是支持 32 位编程的,但在其它方面仍然有所不同,特别是在使用 AppWizard 和使用资源编辑器方面。在开始其它章节的练习前,读者必须了解它们才行。再者,变化是很小的,主要是在方便性和明确性方面的改进,用户应该能够很容易地把文字上的描述轻松地转换到应用环境中去。

如果用户对 Visual C++ 2.0 还不太熟悉,则可能会奇怪为什么不能直接从最新的版本开始,并让它顺其自然呢?问题是 Visual C++ 2.0,它是以 32 位应用程序为目标研制的开发环境,本身就是一个 32 位的应用程序。这就意味着,它只能运行在 Windows NT 3.5 和 Windows 95 上。而且,在 Visual C++上开发的应用程序,作为 32 位应用程序,只能在这些

系统上运行。因此，大多数打算学习 Visual C++的用户，很可能正在使用 Windows 3.0 或 3.1，并准备开发在这些环境下的应用程序。本书主要面向这部分用户，正如以前指出的，这似乎并不是一种严格的限制，因为 Visual C++ 2.0 的用户可以很容易地利用这些进行练习。

1.3 本书的目的及编排方式

本书的目的就是向用户介绍(更确切地说，是帮助用户自学)基本的 Windows 应用程序编程。为完成此任务，本书采用了 Visual C++开发环境和 MFC 库，从而使用户创建和增强一个简单的样本应用程序。

本书的目的不在于向用户详细介绍如何进行 Windows 编程。当用户致力于开发 Windows 应用程序时，还会存在很多问题。在本书中，用户使用 MFC 就可基本避开大部分 Windows 应用编程中固有的复杂问题，这就是本书的初衷。尽管如此，如果准备开发具有专业水准的 Windows 应用程序，用户就应意识和了解到很多这方面的内容。尽管未介绍很多 MFC 细节，用户仍会发现，这种理解在开发时会有意外的收获。

问题在于，本书只能覆盖那么多内容。本书主要集中研究使用 Visual C++环境和 MFC。Windows 编程问题只有与它们有关时才予讨论，而不使用通用术语。因此，用户只能在练习中学习 Windows 编程基础。为了得到详细的理论，可在其它课程中深入钻研。

本书包含许多实例代码，详细解释了这些代码的产生、编辑和作用。在学习这些实例时，用户就能明白一个典型的 Windows 应用程序是如何组织的，明白如何使用 C++的类和继承，以及 MFC 如何构造以使用户增加自己的代码来完成案头的任务。

1.3.1 目标

帮助用户自学 Windows 编程这一目标太广了，以致于在组织和选择本书的材料时涉及面十分广泛。在帮助设计和确定本书内容时，有四个特定的目标一直占主导地位。

第一个目标是使用户方便地使用 Visual Workbench 和其它 Visual C++工具。本书结束时，用户可以很容易地在 Visual Workbench 上处理任何大小或类型的项目。这就意味着，查阅任何东西都不必使用参考资料。也就是说，在设置某一编译器或连接器选项，或调用在简单应用程序中没有涉及到的高级 API 时，都可不用参考资料。

第二个目标是让用户明白如何使用 MFC。因为在 Visual Workbench 内，MFC 构成了大多数工作的基础。用户必须熟悉如何访问和使用 MFC，利用这些工具(例如 AppWizard 和 ClassWizard)，可将它集成到自己的应用程序中。

第三个目标与第二个目标有关但又有些不同。当学完本书时，读者应该熟悉构成 MFC 核心的基础类。完整的 MFC 包含了很大范围的类和函数，要精通这些有效的大量组合需要一定的时间，而且必须处理多种 Windows 应用程序。在本书范围内，相信读者能够熟悉构成 Windows 应用程序所必需的组成部分的那些类，这些类很可能出现在用户创建的任何应用程序中。

最后，本书打算帮助用户学习典型的 Windows 应用程序的组织和结构。单击或多次单击鼠标，用户就可利用 MFC 建立一个应用程序。但是，用户必须明白一个应用程序是如何

构成的以及为何必须用特定的方式来组织。否则，就像一名仅仅明白如何操作自动驾驶仪的驾驶员一样，冒险进入自己并不太熟悉的环境。在这样的开发环境中，自己的应用程序未能正常地发挥作用，但并不知道它为何不能工作，而且不知道如何解决这些问题。

这些目标是密切相关和互相交叉的，每一个目标均是互相支持和互相利用的。像其它好的书籍一样，本书没有区分每一个目标，而是设法把它们联系在一起。因此，用户可用同样的方式学习 Visual C++、MFC 和 Windows 编程。

1.3.2 要求

本书对读者有一些要求，要求分为两大类：学习工具和编程知识。如果喜欢的话，也可称为“硬件”和“软件”。首先从实际要求开始，读者需要有 Visual C++的拷贝版。如果还没有在自己的系统里安装它，那样正好，可以在此阅读一些安装指南。如果已经安装了它，那也不错，只需简单地跳过前面的内容，去学习开发环境和第一个练习就可以了。除此之外，所需的其它内容则都在本书或本书随附的磁盘中给出了。或者，本身就是 Windows 环境中的正常内容。

在编程知识方面，读者至少需要对 C 语言有所了解。读者需要基本的编程知识，且能够阅读和理解 C 语言代码。

除了 C 语言之外，本书还包含创建和修改标准 Windows 应用程序所需的其它基本信息。本书没有探讨创建 DLL 或可视控制的有关更高层领域，而是坚持探讨创建一个典型 Windows 应用程序所需的基本工作及其用户界面。

本书不是传授 C++，而是解释 C++的最基本的概念。如果读者不懂 C++，仍可从实例中学习和了解代码。C++是一种丰富而又多变的语言。笔者强烈建议读者在开始用 Visual C++和 MFC 大量编程前，应认真学习和掌握该语言。许多 C++概念是 MFC 如何工作和处理项目的核心，对 C++的正确理解将有助于读者使用 MFC。

1.3.3 编排方式

本书是按一系列讲座和练习形式编排的，目的在于使读者能够用所学习的每个新概念和功能迅速工作。为了帮助读者了解 Windows 开发过程的本质，应用程序作为实例贯穿全书，这可帮助读者明白实际的 Windows 应用程序是如何建立的，一个完整而又复杂的 Windows 应用程序是如何从开始逐步演变而成的。

这里有一种方法，即通过对 Visual C++工具和 MFC 的使用而不断加深理解和掌握，这样，应用程序的开发便会水到渠成。很多开发人员都会提供帮助，而且是完全正确的，即在开始编程前，读者必须完全明白自己的应用程序如何工作。事实上，大多数人，即使像编者这样的专业开发人员，开始时对所做的工作也只有一般的想法。甚至对于一个详细的设计，读者仍然会发现需要钻研某一领域或方向，而这是开始时所预料不到的。这里要看到的过程，是编者尽力模拟的创建一个简单的 Windows 应用程序的实际过程。

通过使用 MFC 和 Visual C++，读者可比以前有更大的灵活性去修改和调整自己的应用程序，这的确是一个好消息。这里的练习将展示给读者如何使用这些工具，既安全又方便地改变和维护自己的应用程序。

■ 本书约定的表示法

本书使用标准的印刷惯例来帮助读者明确特定的项目。

int i=x+y; 采用单一间隔字体的文本是代码或代码段，例如函数名、变量名或已定义的常数。

File 文本中某个字母下面划线时，代表命名项目的访问键。这种方法主要是帮助用户使文本与显示器相匹配。在屏幕上的文本尚未检验时，用户不能使用这些键去访问。

本书还采用“本书习惯用法”来标出具有特殊含义的文本。所有“本书习惯用法”如下：

注意：提醒读者这些信息特别有趣或特别重要。

警告：提醒读者工作时可能遇到的特定问题或危险。

概念与术语：表示C++术语和概念的定义和讨论。如果读者熟悉C++，可以跳过该项下的正文段，该处很可能是读者已知道的知识。因为有些读者不像其它读者一样熟悉C++，所以编者尽量补充那些对理解或使用实例中的程序有必要的C++的概念。

捷径：捷径可使用户迅速简便地完成某些任务，在正文中却需用很长的形式代表。

■ 各章概述

以下内容是本书内容的简要概述。这些章节从安装Visual C++开始，逐渐向读者介绍如何使用Visual C++和MFC建立一个完整的可执行的OLE应用程序。

1. 第一章：引言

这是读者正在阅读的一章。它提出了本书的概述，描述了本书的内容，介绍了使用本书的方法。

2. 第二章：安装Visual C++

本章讨论了安装问题，例如安装资料的总量、安装选项以及安装要求等。同时，本章还给出了一个简短实例，向读者介绍如何检验安装的Visual C++是否正确。

3. 第三章：使用Visual Workbench

Visual Workbench是Visual C++的核心。开始工作前，读者必须熟悉Visual Workbench的工具、设置和选项。本章向读者介绍如何建立一个项目以及如何访问Visual Workbench下的工具。

4. 第四章：使用AppWizard

本章讨论了AppWizard，这是在Visual C++中创建的基于MFC的应用程序的基础。本章同时还提出了标准的Windows程序开发周期，并说明了AppWizard如何适应和缩短这种周期。在本章中，我们将引导读者学习一系列标准步骤，这是使用AppWizard生成一个框架应用程序所必需的知识。

5. 第五章：在文档中绘图

在本章中，介绍了在Windows应用程序下绘图的基础。同时还介绍了ClassWizard，这是MFC类的类生成器和管理器。本章中，还向读者介绍了用于鼠标消息和Windows消息

的演示应用程序的基本消息处理过程。

6. 第六章:理解面向对象的编程和 MFC

至此,几乎已介绍了本书的一半内容,读者对 Visual C++环境已有了较深的认识。在开始增强演示应用程序以及进一步使用 Visual C++ Wizards 之前,本章向读者简单介绍面向对象的编程和 MFC(Microsoft 基础类)。本章介绍了在应用层次上面向对象编程的基本原理,讨论了诸如类等级、关键词及继承等问题。读者同时还可了解 MFC 框架应用程序,并将了解 MFC 的整体结构以及学会如何使用它来增强演示应用程序的功能。

7. 第七章:视窗与文档

大多数 Visual C++编程集中在视窗和文档上进行,本章描述了视窗和文档如何交互作用,以及读者如何修改各种层次以得到希望的功能。例如,读者可向演示应用程序中增加滚动功能。

8. 第八章:异常和调试

调试是任何编程者面对的事实。Visual C++提供了出色的工具,用于避免出错以及发现疏忽的错误。本章详细讨论了在 Visual C++应用程序中如何使用演示应用程序去调试和跟踪问题。

9. 第九章:使用资源

Visual C++备有集成资源编辑器和资源处理界面。在 Visual C++的早期版本中(到 1.51 版),这是一个独立的且是集成的应用程序,称为 AppStudio。在 Visual C++ 2.0 中,它被完全集成到 Visual Workbench 中。在任何情况下,读者都会从本章中了解到在资源编辑器创建资源后如何向项目文件中增加资源。本章同时还讨论了资源的主要基本类型。

10. 第十章:控制和消息

本章向读者介绍了在应用程序中如何使用资源控制,如何把这些控制连接到使自己的应用程序正确响应的消息上。例如,读者可在演示程序中增加对话框处理功能。

11. 第十一章:增强应用程序

本章包含了大量的增强功能,读者在大多数应用程序中将会用到它们。本章向读者介绍了如何为自己的应用程序存储和检索永久性信息,如何创建一个分割窗口以便在窗口中观察一个文档的几段,以及如何使用辅助的 MFC 类(例如 CTime),以在自己的应用程序中更新状态条显示。

12. 第十二章:测量与演示

本章主要讨论了由 Visual C++提供的工具,以及用于提供与设备无关的用于绘图的框架应用程序。读者可在演示应用程序上增加类,以便将测量系统改为与设备无关的模式。本章还介绍了如何增加新的打印功能,这依赖于与设备无关的测量。

13. 第十三章:更深入的议题

本章包含了重要的高级议题,它们是 Visual C++的组成部分。特别地,它展示了如何从 MFC 向一个现有应用程序中增加新的代码。例如,读者会更新演示应用程序以提供与上下文相关的帮助,然后学习如何在自己的应用程序中提供 OLE 兼容性。读者可使用同样的增强技术来向演示应用程序中增加 OLE 服务器功能。

14. 第十四章:升级到 Visual C++ 2.0

本章解释了如何把项目和程序从早期的 Visual C++版本(16 位)升级到 Visual C++ 2.0 中。在此,演示应用程序作为一个如何升级项目的实例使用。本章同时还讨论了在 MFC 3.0 中实现的新特点,这是 MFC 库的最新版本。

第二章　安装 Visual C＋＋

本章讨论了安装问题，例如要安装的推荐软件、安装选项以及安装要求等。本章同时亦给出了一个简单实例，以检验读者是否正确地安装了 Visual C＋＋。

本章包含的重要议题如下：

* Visual C＋＋的 CD-ROM 版本
* Visual C＋＋ 1.5 和 Visual C＋＋ 2.0
* 硬件和软件要求
* 安装过程
* 检验 Visual C＋＋的安装

2.1　Visual C＋＋的 CD-ROM 版本

Visual C＋＋ 2.0 制作在 CD-ROM 上，它包含要安装的所有的 Visual C＋＋组件的资料。CD-ROM 含有以下目录：

msv15：本目录包含 Visual C＋＋ 1.51 的开发环境。如果用户正在使用 Windows 3.1，Windows for Workgroups 或 Windows NT 3.1 或更早的版本，则可以安装这个版本。这个环境可使用户开发 16 位应用程序，并在该环境下运行。

msvc20：本目录包含 Visual C＋＋ 2.0 的开发环境，只有当用户正在运行 32 位的操作系统（例如 Windows NT 3.5 或 Windows 95）时，才可以安装。

msvccdk：本目录包含了 Microsoft OLE Custom Control Developer's Kit（CDK）的文件。这可使用户为 MFC 和其它应用程序开发新格式的 OLE 定制控制。这是用 32 位取代过时的 16 位的 Visual Basic 控制，它在开发人员中间曾经十分流行。

win32s：本目录包含了 Microsoft Win32s 的库文件，它允许用户的 32 位应用程序可在 16 位操作系统上运行，例如 Windows 3.1。

wincim：本目录包含了 CompuServe Information Manager 应用程序和支持文件的 Windows 版的免费拷贝，这是为方便用户而提供的。它可使用户与 CompuServe 联网或签约。Microsoft Developer Support 提供了许多有关 CompuServe 的特别讨论，在此可使用户从 Microsoft 专业人员和其它开发人员那里得到一些帮助和建议。

尽管 CDK，Win32s 以及 CompuServe 工具很有价值且很吸引人，但它们并不是本书着重讨论的问题。这里将不再深入讨论它们或讨论如何使用或安装这些库。如果读者打算使用它们，可查阅 Visual C＋＋ CD-ROM 上的联机文献集，以进一步了解有关情况。

2.2 Visual C++ 1.5和Visual C++ 2.0

这是一个技巧问题。当用户的CD-ROM上有两种不同版本的Visual C++时，在开始设置和使用时，用户不会有太多的选择。

Visual C++ 2.0是为创建32位应用程序而设计的开发环境，在Windows NT 3.5或Windows 95操作系统下运行。而且，Visual C++ 2.0本身也是32位的应用程序。如果用户没有在Windows NT 3.5或Windows 95下运行，就不能在自己的系统上安装Visual C++ 2.0。下面将要描述的标准安装步骤明确了当用户没有在32位操作系统上运行时，将会使得用于安装Visual C++ 2.0的按钮无效。

尽管用户不能在16位操作系统(例如Windows 3.1)上安装Visual C++ 2.0，但却可在Windows NT 3.5下安装Visual C++ 1.51。如果这样做，表明用户是在Windows兼容方式下运行它(Windows下的Windows，即WOW)。用户选择这样做有两种理由，首先，如果用户正在开发16位的应用程序，或靠它们自己，或与32位的应用程序相连，则必须在Visual C++ 1.51上才能完成；这是产生16位应用程序的唯一开发环境。其次，用户可选择使用Windows NT或Windows 95作为开发平台，因为它比Windows 3.1或另外的16位操作系统更强大、更安全。这种方法的主要缺点是这些环境中的每一种都需要大量的磁盘空间。

因此，当读者阅读本章时，要牢记Visual C++ 2.0的选项和工具只有在32位操作系统上才行得通。

2.3 硬件和软件要求

从CD-ROM中安装Visual C++的要求，依据以下两种情况而变：用户正在使用的操作系统和安装时用户选择的选项。正如用户所预料到的，运行Visual C++ 2.0的要求比运行Visual C++ 1.51要更精确。

2.3.1 Visual C++ 1.51

下面是在用户系统上安装Visual C++ 1.51的最低要求：

* 一台IBM个人计算机，或100%与80386兼容运行的、或更新的处理器
* 一台VGA或更好的显示器
* 4MB可用内存(推荐8M)。如果用户正在运行Windows NT，则至少需要16MB
* 一个与读者的操作系统兼容的CD-ROM驱动器
* 运行于增强模式下的Microsoft Windows 3.1，Windows for Workgroups 3.1或Windows 95，或Windows NT。Visual C++提供了命令行去访问某些工具和函数，但在安装过程中需要Windows支持。
* 一个硬盘驱动器，应具有足够大的空间安装用户选择的选项。最小的安装必须具有13M的硬盘空间。

以上给出的内存容量是 Microsoft 推荐使用的最小容量，据笔者的经验，这是绝对不可再少的最小容量。要得到足够好的性能，至少需要 16MB 内存。编译和链接时的速度差别也很可观。而且，如果磁盘空间有限，用户应该研究设置一个永久的 Windows 交换文件。缺少 Windows 的运行空间同样会严重降低系统的性能。笔者已在网络论坛上见到过数次对 Visual C++性能的严重抱怨，所有这些抱怨归根结底归于缺乏系统资源。我们不妨把最小系统需求精确比作"要使 Visual C++ 1.51 能够运行而不是有效运行的最少系统配置。

安装 Visual C++ 1.51 的磁盘要求如下：

* 典型安装：70288 KB
* 最小安装：12816 KB
* 定制安装（每项均可选）
  * MS Visual Workbench：11120 KB
  * MS C/C++ 编译器：12304 KB
  * 运行时的库：12720 KB（推荐设置）
  * MS Foundation Classes：28032 KB（推荐设置）
  * 工具：10400 KB（推荐设置）
  * 联机帮助文件：14464 KB（推荐设置）
  * 样本源代码：36448 KB（推荐设置）
  * MFC OLE：5904 KB（推荐设置）
  * MFC 数据库（ODBC）：8112 KB（推荐设置）

以上列出了带有推荐设置的项目选择，用户可选择其它的安装项目，这可能会降低或提高磁盘空间需求。特别需要注意的是，如果用户正在开发 OLE 定制控制，则可能希望为工具和 MFC OLE 而使用不同的选项。

记住，这是 Visual C++软件包要求的总量，在以后的版本中，这些要求可能会改变。尽管如此，这里的列表提供了用户在自己的系统上使用 Visual C++所需要的、有关磁盘空间容量的某些概念。

注意，大多数资料（例如帮助文件和样本源代码），可能与用户日常工作并不直接相关。如果用户缺少可用的磁盘空间（况且有谁没遇到过类似的问题呢？），就必须选择把其中一部分或全部资料留在 CD-ROM 上，并且只从 CD-ROM 上查阅它。当用户打算查阅这些资料时，要花费更长时间，并要求用户有 CD-ROM 驱动器，但这样做会节省磁盘空间。典型的安装过程的确也是如此，大部分的帮助文件要留到从 CD-ROM 上查阅。最小的系统安装在这一点达到极限，使得大部分工具、编译器、库以及样本和帮助文件都留在从 CD-ROM 上访问。在这种情况下，使用 Visual C++时必须时时拥有联机 CD-ROM。

警告：不论什么时候用户把样本文件放在 CD-ROM 上，当用户准备使用样本时，应准备好把其中的一部分或全部文件移到硬盘的工作目录下。记住，CD-ROM 是只读设备。因此，当用户准备从 CD-ROM 中调入一个样本项目时，会接受到不能存储任何改变的警告。此时，用户最好的办法是在硬盘上创建一个目录结构，例如使用文件管理器模拟 CD-ROM 上的结构。当用户使用样本代码时，把需要使用的文件（且只有这些文件）简单地拷贝到硬盘上。

2.3.2 Visual C++2.0

下面是在用户系统上安装 Visual C++ 2.0 的最低要求：

* 一台 IBM 个人计算机，或 100%与 80386 兼容运行的、或更新的处理器(推荐 80486 或更新的)
* 一台 VGA 或更好的显示器(推荐 SVGA)
* 16MB 可用内存(推荐 20MB)
* 与操作系统兼容的一台 CD-ROM 驱动器
* Microsoft Windows NT 3.5 或 Microsoft Windows 95
* 若硬盘有足够大的空间安装用户选择的选项，则安装时最少所需的硬盘空间为 29MB

以上给出的内存容量是 Microsoft 推荐的最少容量。据笔者的经验，组合 Visual C++和 Windows NT 时，在任何实际的工作中这些都显得不够充分。要得到足够的性能，至少需要高速(66 或 90MHZ)运行的 486 处理器，并应认真考虑增加额外内存。一个好的开发系统至少需要 32MB 内存和一个 Pentium 处理器。编译和链接时的速度差别亦很可观。我们可把以上的最小配置视作 Microsoft 的压缩版而不是正常的配置。如果读者打算用 Visual C++做任何有效的工作，那么至少需要 Microsoft 推荐的系统，而且系统越大越便于读者工作。

安装 Visual C++ 2.0 的磁盘要求如下：

* 典型安装：142880 KB
* 最小配置：74432 KB
* CD-ROM 安装：28576 KB
* 定制安装(选择的每项)：
  * MS Visual C++开发环境：14304 KB
  * MS C/C++编译器和库：18382 KB
  * MS Foundation Classes：43200 KB(推荐设置)
  * 数据库(ODBC)文件：4736 KB(推荐设置)
  * 工具：7904 KB(推荐设置)
  * 帮助文件：12112 KB(推荐设置)
  * 源代码：59840 KB(推荐设置)

对于以上列出的带有推荐设置的选项，用户可选择其它的安装选项，这可能会降低或提高磁盘空间的需求。

请记住 Visual C++软件包所需磁盘空间的总量，这些要求在以后的版本中可能会改变。尽管如此，这里的列表仍将提供给用户在自己的系统上使用 Visual C++所需的有关磁盘空间的某些概念。

像以前提出的一样，用户可选择把其中一部分或全部资料留在 CD-ROM 上，并且只能从那里访问这些资料。当用户访问这些资料时要花费更多的时间，并要求用户有 CD-ROM 驱动器，但这样做会节省磁盘空间。最小系统的安装是个折衷方案，因为开发环境、MFC 文

件、基本工具以及用于 32 位开发的库均安装到用户的硬盘上，而把编译器、样本和帮助文件留在了 CD-ROM 上。至于 CD-ROM 安装，在使用 Visual C++时一直拥有联机 CD-ROM 是这种安装选项所必须的。尽管如此，如果用户选择这些选项中的一个，可参见上面的“警告”作为正式的提示。

有一点仍需强调，如果用户正在开发 32 位的应用程序，并希望在 16 位的机器或操作系统上运行，就必须拥有一台独立的计算机，或者一个双引导设置(这样，读者可以按自己的选项从 Windows NT 或 Windows 3.1 中进行引导)，而且用户必须在 16 位的系统上安装 Win32s。最佳的处理是在一台独立的计算机上安装 Win32s 库。或者，用户必须在计算机上拥有 CD-ROM 驱动器，或者需要用户在网络中可访问兼容的 CD-ROM 驱动器。这些 DLL 需要 Microsoft Windows 3.1 或 Windows for Workgroups 3.1 或更高的版本，以及 DOS 5.0 或更高的版本支持。

2.4 安装过程

开始安装任何版本的 Visual C++时，首先要运行位于 CD-ROM 驱动器根目录下的 SETUP.EXE 程序。在启动对话框里，用户会看到不同的选项，这些选项取决于用户的操作系统。

下面将要描述在以下两种情况下用户能安装什么内容：一是在用户使用 16 位的系统如 Windows 3.1 时；二是用户使用 32 位的系统如 Windows NT 3.5 时。

2.4.1 在 Windows 或 WFW 3.11 下安装

在 Windows 3.1 下运行 CD-ROM 中的 SETUP.EXE，就开始了安装 Visual C++。要做到这一点，可在 Program Manager 的File 菜单下使用Run 命令。键入 d:\setup(或其它的 CD-ROM 驱动器字符)作为命令行，这将使 CD-ROM 中的 Setup 程序开始运行。此时，用户可观察到 CD-ROM 上的指示灯闪烁。

注意：用户亦可用Browse 按钮，在 CD-ROM 驱动器中找到 SETUP.EXE 文件，然后执行应用程序，这样花费时间要多点。

安装程序将显示 Setup 对话框，如图 2.1 所示。正如读者所看到的，最上边的按钮代表 Visual C++ 2.0，它变得模糊而且不能使用。但是，其它的三个按钮是可以使用的。在此，用户可选择其中的任何一个进行安装。

* OLE 控制
* Visual C++ 1.51
* Win32s 库

因为本书的目标在于 Visual C++，故这里不讨论 OLE 控制或 Win32s 的安装。如果用户希望安装它们，可查阅 CD-ROM 上的资料和 Visual C++ 2.0 所携带的文档。

一旦用户选择了 Visual C++ 1.51 按钮，如果这是用户初次安装 Visual C++ 1.51，将会看到一个 Registration 对话框，这是要求用户输入用户名和公司名(如果可能的话)。这种信息不是必须的，但当用户寻求帮助时可能有用。

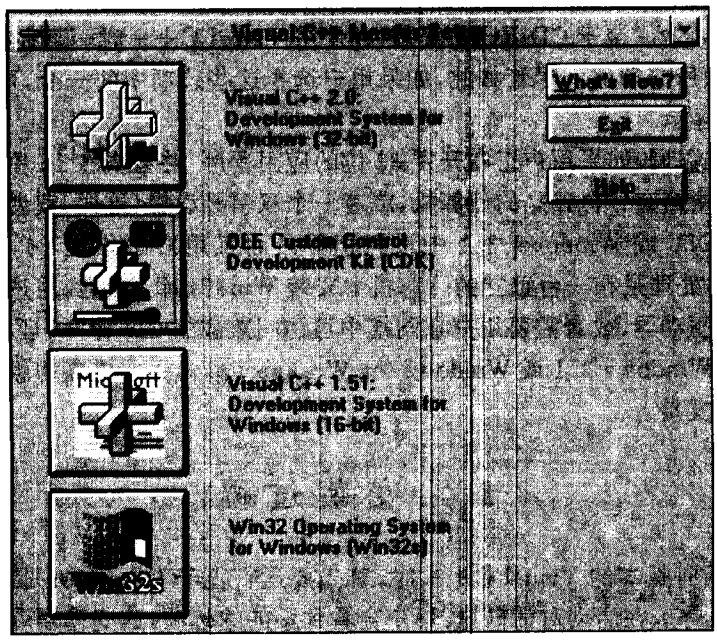

图 2.1 在 16 位环境如 Windows 3.1 下运行时的 Setup 对话框

然后就会看到一个 Welcome 对话框，万一用户按错按钮时，它允许用户在开始安装前退出系统。如果用户打算继续安装过程，按一下Continue 按钮就会看到 Installation Options 对话框，如图 2.2 所示。

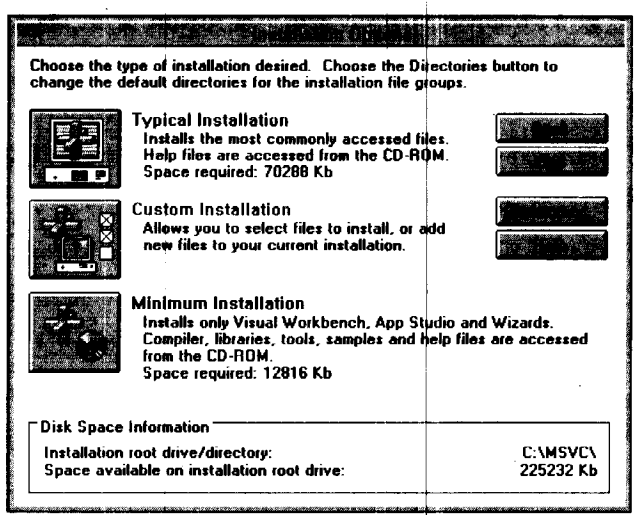

图 2.2 Installation Options 对话框允许用户根据需要选择 Visual C++的安装组件

图 2.3 显示了 Visual C++安装程序使用的缺省目录结构，如果用户打算观察或改变它，可在图 2.2 中的 Installation Options 对话框上按一下 Directories 按钮。

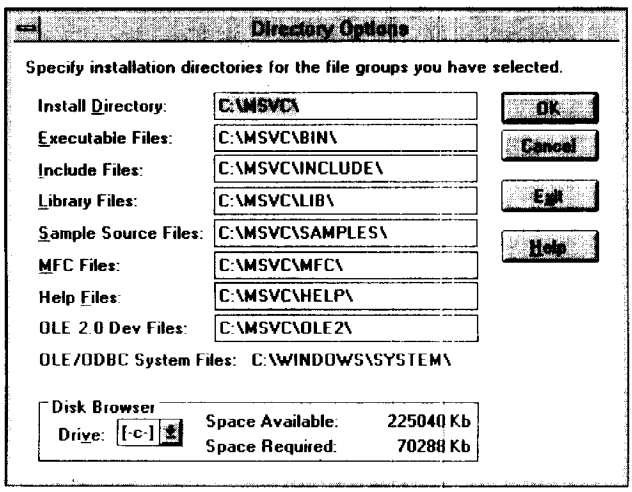

图 2.3　Directories 对话框允许用户把文件放到任何所需的驱动器或目录中

正如用户所看到的，安装时有三个选项：典型安装、定制安装和最小系统安装。

■　典型安装

典型安装完成了一个标准的缺省配置，下面的组件被安装到硬盘上：

* Microsoft Visual Workbench
* Microsoft C/C++编译器
* 运行时的库
* Microsoft Foundation Classes
* 工具
* 样本源代码
* MFC OLE
* MFC 数据库

在这些主要组件中，Setup 程序选择了最普通和最有用的选项，如果用户有足够的磁盘空间，这种典型安装是一个很好的选择。

■　定制安装选项

使用 Custom Installation 对话框，可选择要安装的组件集或其子组件集，Custom Installation 对话框如图 2.4 所示。

正如读者所看到的，很多主要组件有其子组件选项，由与其组件名相邻的按钮指示。按下该按钮，就可选择特定的从属项。

现在，让我们反过来看看每个主要组件，了解根据工作需要可调整什么内容。

首先，两个主要组件：Visual Workbench 和 C/C++编译器没有子组件。必须安装 Vi-

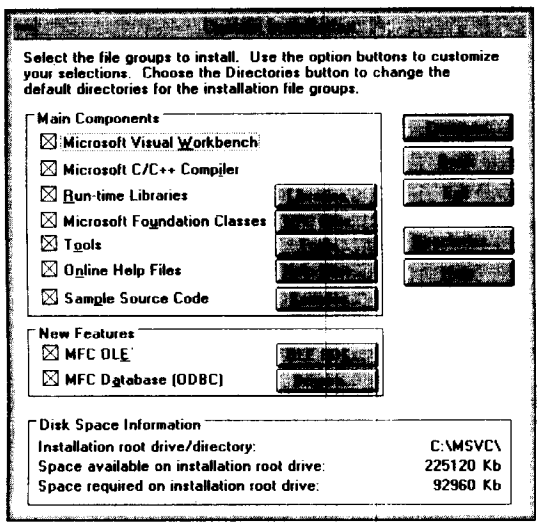

图 2.4 Custom Installation 对话框允许用户精确选择要安装的组件集

sual Workbench，以使用 Visual C++集成开发环境。可以选择把 C/C++编译器留在 CD-ROM 中，但这样做严重减慢了编译过程，如果磁盘空间存在问题，可以在 Installation Options 对话框中选择 Minimum Installation 按钮。

运行时的库是 C 和 C++的支持库，需要其中某些组合来创建和运行用户程序。这些库和选项的缺省设置如图 2.5 所示。

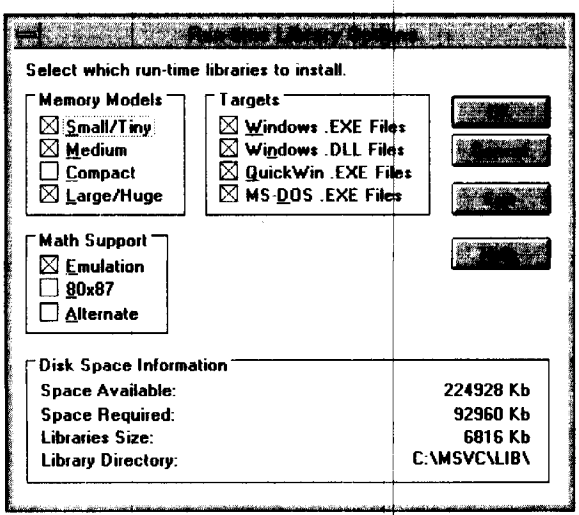

图 2.5 Run-time Library Options 对话框允许用户选择要安装的库

Run-time Library Options 对话框允许用户选择单一内存模块和目标应用程序类型。尽管要求的总的磁盘空间可能较小，但如果空间紧张，用户就该考虑不检查 QuickWin 和 MS-DOS 目标程序，除非用户要专门使用它们。

注意：第三章提供了一个如何使用 QuickWin 开发系统的简短练习。在删除 QuickWin 目标程序前，可以阅读第三章的那一部分。

至于 Math Support，可选择所要求的支持类型。对大多数应用程序而言，缺省 Emulation 支持是正确的选择。如果有协处理器，仿真器代码就使用一个数学协处理器，否则就使用软件模拟，这就使得应用程序可在最大限度的硬件设置范围内运行。

Custom Installation 对话框中的 Microsoft Foundation Classes 选项在磁盘中安装了 Microsoft Foundation Classes 库的 2.5 版本，这些选择中有很多选项与 Run-time 库的选项相同，如图 2.6 所示。

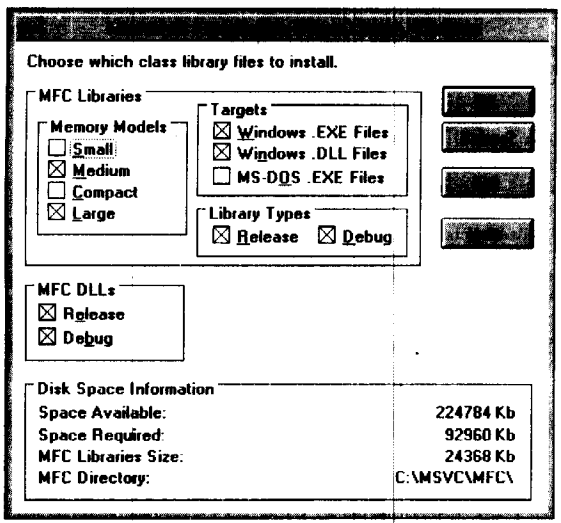

图 2.6　Micrsoft Foundation Class Library Opitions 对话框允许用户选择在 MFC 版本中有效的选项(用户已为应用程序安装 MFC)

这里，笔者唯一的建议就是，用户接受缺省设置而且能确信同时保持这些库的 Release 和 Debug 版本有效。

注意：在本书中，没有提到内存模式。在 DOS 初期，对于一名编程人员，甚至对于初学者来说，了解 DOS 如何处理内存都是基本的要求。为讨论内存模式，并明白为何选择这种模式而不是另一种，则要花费很多精力和篇幅。随着 32 位操作系统的出现，过去与内存有关的约束限制已变得不再相关了。

为了学习 C++和用 MFC 编程，对内存问题不必知道太多，建议读者简单地选用标准设置。当以后有了较多的经验时，如果需要的话，可从 Windows 或 DOS 编程的标准手册上学习有关内存模式的概念及使用。

唯一需要注意的情况是，当用户缺少磁盘空间时，用户可能打算删除其它的内存模式，而只保留一种内存模式。尽管本书的练习中采用了缺省的 Medium 内存模式，但如果用户只准备安装一种模式，仍建议安装 Large 模式。这样，在开发时将会带来很大的灵活性，而且在此模式下本书的练习都能正常运行。

Tools 选项允许用户安装 Visual C++所携带的辅助工具。Tools 的选项如图 2.7 所示。

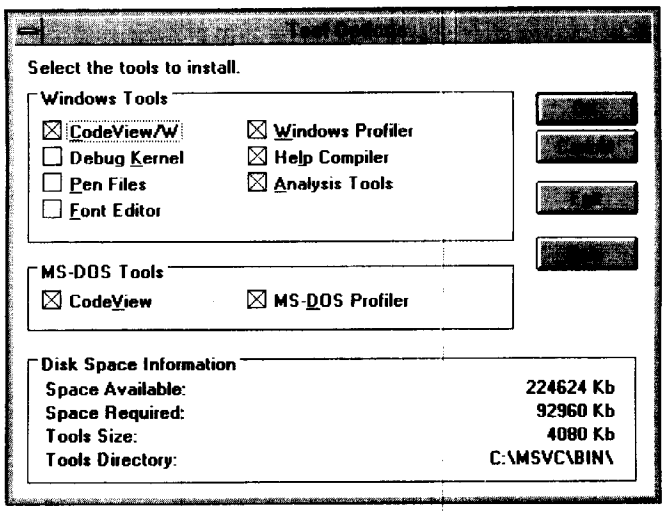

图 2.7 Tool Options 对话框允许用户自由选择要安装的工具

这里的缺省选项包含了几乎所有想用的工具，Pen 和 Font 文件是特殊的，而且大多数工作并不需要它们。

用户希望安装的一个选项就是 Debug Kernel，确认这个框将在 Windows 下安装一个新的代码内核。Windows 下的调试内核提供了强大的查错功能、较好的出错报告功能，并提供了关于调试的符号信息，它可使用户对于应用程序的跟踪达到对 Windows 的实际调用过程跟踪的程度。其实，不必每时都在运行该选项，但当用户的应用程序在 Windows 3.1 下碰到问题时，它将是非常有用的。

注意，要运行在第十三章中的练习，则需要帮助编译器。如果用户不准备开发在 MS-DOS 下运行的任何应用程序，可以不选择 MS-DOS Tools 以节省磁盘空间。

在 Custom Installation 对话框中的 Online Help Files 选项安装了联机帮助。正如以前提到的，如果用户不选择安装这些文件，相关的帮助信息就要到 CD-ROM 上查询。因为帮助文件很大，用户可从硬盘上去掉其中一部分或全部而只保留在 CD-ROM 上。安装这些文件的缺省设置如图 2.8 所示

笔者已经发现，把所有的帮助文件放在 CD-ROM 上，且仅仅在需要时才访问它们，这样做非常容易。如果在 Visual C++组中打开 Books Online 工具，从 CD-ROM 上访问这些项目几乎不用多少时间，但却节省了大量的磁盘空间。

在 Custom Installation 对话框中的 Sample Source Code 选项允许用户自由选择安装哪