计算机程序设计与软件开发系列丛书

Visual C++ 6.0
编程实例·技巧

李国徽　王涛　潘琳　编著

卢炎生　审校

华中理工大学出版社

前　言

Visual C++ 6.0 是 Microsoft 目前综合性最高、最复杂的软件开发产品。它为用户提供了前所未有的编程能力和便利，它提供的工具能适合各种编程风格。

然而学习 Visual C++ 6.0 决不是一件容易的事情。本书旨在给 Visual C++ 6.0 程序员提供一些基本的编程方法和有关 Visual C++ 6.0 高级应用的编程技巧。帮助程序员充分利用 Visual C++ 6.0 提供的各种便利。

本书的特色

本书不是一本简单的 Visual C++ 6.0 使用手册，本书提供了大量的简单易学的程序示例，包括相关的技术文档和程序说明。由于考虑到大多数 Visual C++程序员没有受过正规的程序设计训练，所以我们还介绍了一些有用的编程技巧和系统应用程序的开发经验。总之，我们希望通过此书的学习使您感到掌握 Visual C++ 6.0 是一件容易、有趣的事情。在仔细阅读本书之后，您不仅能学到新的技术知识，熟悉某些方法的细节，同时能提高编程的技术和经验，更加适应现实的要求。

本书的内容

本书几乎涉及了 Visual C++ 6.0 的全部内容，包括 Visual C++ 6.0 的基本概念，应用框架及 Windows 消息机制，文档和视之间的关系，图形设备接口，文件、持久性和序列化等 Visual C++ 6.0 编程中的一些基本问题，同时还讨论了在 Visual C++ 6.0 编程中比较高级的专题如数据库访问（主要是 ODBC），多媒体编程，进程、线程及它们之间的同步与通讯，Internet 编程及 ActiveX 编程等。

本书的读者

本书的初衷是不仅带领读者走进 Visual C++ 6.0 的广阔天地，同时尽可能地为读者展现 Visual C++ 6.0 的无限魅力，提供丰富的学习资料。因此，本书的主要对象包括 Visual C++ 的程序开发人员以及掌握了最基本的 Visual C++知识（安装和启动 Visual C++）的学习者。前者可以从书中获得许多实用的编程技巧、开发技术和经验，而后者则可以通过学习、实践，掌握 Visual C++ 6.0 开发环境和开发工具，提高编程能力和水平。

怎样使用本书

当您开始学习使用 Visual C++ 6.0 的时候，最好从头到尾阅读此书，它会给您提供最好的学习指导。然后您就可以将此书作为一本参考手册，通过目录及索引来经常查阅一些内容。由于许多应用框架元素之间是彼此紧密相关的，许多概念也都是彼此联系的，因此本书不可能像百科全书那样包罗万象。在使用本书时，最好还需要一本《类库参考手册》，同时帮助文件也要经常用到。

好了，要介绍的内容已经差不多了，下面我们将开始深入地介绍 Visual C++ 6.0，希望这本书能成为您学习 Visual C++的好帮手，这将是我们的努力劳动所获得的最大回报。因此，如果有什么其它问题，请和我们联系。最后，祝您愉快地开始 Visual C++ 6.0 的学习之旅。

编　者

1999 年 9 月于武汉华工园

面向对象程序设计的基本概念

1.1 导　论

自从 20 世纪 50 年代以来，计算机硬件技术发生了翻天覆地的变化．为了让计算机能发挥更大的作用，人们必须充分地利用它的硬件性能，在软件设计与开发方法上取得突破性的进展．面向对象技术(包括面向对象分析与面向对象设计)是人们在这方面迈出的重要的一步．面向对象程序设计是一种全新的设计，是一种实现软件系统的方法，它的主要目标是

- 用增加软件可扩充性和可再用性的手段来改善程序员的生产率；
- 减少软件维护的复杂性和费用．

所谓可扩充性指的是“软件产品能适应于规格(说明)变动的难易程度”．所谓可再用性指的是“软件产品能在新的应用中被整个地或部分地予以重新利用”的一种性能．所谓维护指的是“使程序处于良好工作状态的活动，包括检查、测试、调整、更换、修理等”．

当使用面向对象程序设计时，软件开发的设计阶段是与实现阶段紧密相联系的．正如许多的软件设计和实现途径一样，面向对象程序设计这一方法的潜力可能会超过实际上所能达到的潜力．虽然在各个孤立的应用领域里已有面向对象程序设计取得巨大成就的报导，但在软件设计和开发的领域中，评估面向对象方法学的相对成功为时尚早．纵然如此，面向对象技术在编程、DBMS(数据库管理系统)分析/设计工具(面向对象数据库已经成为数据库研究领域的一个重要分支)等领域里的地位日趋提高，并正被认为是 90 年代最重要的软件技术之一．据有关方面对美国 1600 家单位进行调查，在导入面向对象技术方面，已经导入的单位占 21.0%，正在试验性导入的单位占 26.8%，没有打算导入的单位占 52.2%．

在各种面向对象程序设计语言中，特别是在 Smalltalk 和 C++这两种当前较广为流传的面向对象程序设计语言中，C++语言它具有一些独到之处：它的表达力较强，可移植性好；与程序员的传统基础接近(一般的程序员对 C 语言都比较熟悉，虽然在 OOP(面向对象程序设计的简称)的纯度方面稍逊于 Smalltalk)；语言及其环境规模不大，较易掌握使用；适合开发大型程序．

面向对象程序设计包含了几个主要的概念：抽象数据类型和类、类型层次结构(子类)、继承及多态性．本章将介绍这些基本概念．

抽象数据类型是面向对象程序设计的核心。抽象数据类型是一种模型，它包含一类型与一组相关的操作。这些操作是为“作为基础的该类型的行为”而定义的，并且刻划该行为。

在大多数面向对象程序设计语言中，用定义“对所有这些操作(它们能施于作为基础的该类型上)的接口”的办法，类定义借此描述作为基础的抽象数据类型的行为。类定义也确定该类型的各实现细节或数据结构。通常，这些实现细节只在该类的作用域内才是可存取的。我们称这样的类型为私有类型。数据类型的全部或一部分在该类的作用域之外是可存取的，我们就称该类型的这些部分为公开的。

在该类型之上定义的那些操作一般也分成公开的或私有的。在该类的作用域之外可存取的那些操作是公开(public)的操作。私有(私用)(private)的操作只在该类的作用域内才是可存取的。用面向对象的说法则是为一个类而定义的那些操作称作方法。这些方法类似于非面向对象语言中的过程和函数。如果一个类的各公开操作在很多应用场合是被普遍使用的，那么，该类就构成一可再用软件成分的基础。

对象是一变量，该变量被说明是一特定类的变量。这样的一个对象用包含“定义在类定义中的(私有的和公开的)数据所有各域的复件(拷贝)”的办法，把“状态”封装起来。调用定义在该类定义中的一个或多个方法，可施行动作在此对象上。调用方法的过程称为“发送消息(message)给此对象”。这样的消息一般地要修改存储在该特定对象中的数据。

每一类变量或对象表示该类的一个实例。如果几个对象均被定义成是同一类的，那么，一般地它们所含的值组是互不相同的。

面向对象语言是可扩充的，这是因为：程序员能够创建一些新的类型，它们可赋以各特殊性质并且它们的行为被刻划在类定义中。像对待程序设计语言本身所提供的各预定义类型那样，用几乎同样的方式就能处理这些新类的各对象。

通过各子类，面向对象的“范型”(paradigm)提供各类所形成的层次结构。子类定义刻划一个对象集合的行为，这些对象继承父类的某些特征，而又获得不为该父类所共享的一些特殊化的特征。容许创建各子类可以降低软件开发的费用和复杂度。

各子类能导致渐增式的问题求解。修改，或更坏的情况，重写现有的各软件成分(假定这些成分的源代码是可利用的)，一些新的子类就能从一个基准类集合中创建出来。这些新子类的各对象形成软件总体结构的基层结构。

父类中定义的数据类型和操作，子类可拿来袭用。此外，子类还可以增加自己的数据类型和操作。这就是面向对象软件系统中的(类)继承(机制)。

1.2　面向对象程序设计范型

本节讨论面向对象程序设计范型这一概念，并据此对现有的程序设计语言进行分类。为了了解 OOP 范型，首先论述面向对象的问题求解方法，然后再明确地定义 OOP 范型。

1.2.1　程序设计范型

程序设计范型指的是程序设计的体裁，正如文学上有小说、散文、诗歌等体裁，程序

设计体裁是用程序设计语言表达各种概念和各种结构的一套设施．它首先是对程序设计这一抽象级上来说的，其次才是对程序运行这一动态的具体级上来说的。由此，当今的程序设计范型就分成：过程(式)程序设计范型、函数(式)程序设计范型、面向约束(逻辑)程序设计范型、面向对象程序设计范型，还有进程(式)程序设计范型、类型系统程序设计范型和事件程序设计范型。每一程序设计范型可有多种程序设计语言，例如，Pascal 和 C 均体现过程式程序设计范型，用来进行过程式程序设计。

(1) 过程式程序设计范型

程序设计归纳为选定数据结构、设计算法过程或函数。程序执行被看作各过程调用的一(偏序)序列，以此处理数据结构。此范型最为普通，它被诸如 Algol，Pascal 和 C 语言所支持。

(2) 函数程序设计范型

程序被看作“描述输入与输出之间的关系”的一个数学函数。在函数程序设计范型中，完全消除状态或变量这一概念。也就是说，函数程序设计范型是无变量的程序设计。它被诸如 FP 和 Lisp 语言所支持。

(3) 面向约束(逻辑)程序设计范型

程序被看作“描述输入与输出之间的各关系”的一组方程．具体地说，程序设计归纳为列举事实、定义逻辑关系(规则)、以提问方式求得(或证实)解。像函数程序设计范型一样，在面向约束程序设计范型中也消除了状态这一概念．Prolog 语言是支持面向约束程序设计范型的最主要的一种语言。

有一种观点认为进程式程序设计范型从属于面向对象程序设计范型；类型系统程序设计范型是与面向对象程序设计范型有关的；至于事件程序设计范型，则与 Petri 网模型等有关。

可以有兼备两种或多种范型的程序设计语言，即混合范型语言．例如 C++不是纯粹的面向对象范型的，而是过程与面向对象混合范型的语言，实质上，Lisp 并非是纯粹的函数程序设计范型的，而是过程与函数混合范型的。

1.2.2 面向对象问题的求解

面向对象的软件系统的体系结构是围绕着一个类的集合而构造起来的，这些类刻划系统中所有作为基础的数据的行为．出自每一类的各对象用调用该类的各方法来加以处理；即发送消息给这些对象。这些消息表示在该对象集合上所要采取的各种动作。

面向对象程序设计围绕的核心是所要处理的数据，而不是进行数据处理的各过程。这些数据形成软件分解的基础。实际上，面向对象软件设计的主要挑战是把一软件系统分解成各个作为基础的数据类型或者“各个类与各个子类”，并且定义这些基础类与子类的性质。各对象或各类(各子类)变量对应于真实问题空间中的各物理实体或各逻辑实体。

“定义一面向对象软件系统并形成该系统的高级设计”的总体结构框架仅仅展现一(子)类集合及它们的定义和各个对象。每一类行为用各方法接口来刻划。各方法的实现细节不是该系统的高级设计的部分。

在一典型的面向对象语言中，定义在一类中的各方法的接口能够与这些方法的实现细节相互独立地予以规定，从而容许该系统的设计与它的实现分离开来．一个概念(带有各方法接口的类定义)与它的实现(实现该类数据结构和各算法的代码)之间的这种分离，在面向对象程序设计中以及在实现可再用性和维护费用的控制方面，是十分重要的．

在面向对象程序设计中可再用性得以提高是由于：在一类的各方法接口中提供了封装在该类中的各概念．用户只需了解规定在各方法接口中的该类对象的行为，而无需关心它们的实现．从用户的观点来看，这些方法的实现包含在一个“黑匣子”里，是隐蔽看不见的．

在 OOP 方法中可维护性得以提高是由于：数据结构或算法的实现(即类实现内部的代码)上的变动能被限制在“实现该类或该类一部分”的代码区域内．由于保持了类的接口，从而在该类之外的作用域内不致产生有害的效应．此接口形成依照各动作(从该类的作用域外部所能施行于该类各对象上的那些动作)来“使用”该类的基础．

面向对象软件开发的主要目标是

- 使用“呈各基准类的形态”的那些可再用软件成分，并且应用“利用各子类”的渐增问题求解法，来缩短开发时间并降低开发费用．
- 通过“把变动局限于一个或多个类的实现范围内”，从而降低软件维护的费用．

面向对象系统的可靠性能得以提高是由于在初始设计中所造成的高级整体化．组成该系统的各主要部分从一开始就配置成一定的构形并配合在一起．每一主要部分被它的各抽象性质所定义．在作成或实现许多低级细节之前，可进行高级整体化测试．这就有助于可靠性的改善．

面向对象程序设计提供用于快速原型化的有用平台(工作场地)．在完成“一系统高级分解成各类和(出自这些类的)各对象”之后，为了能察看该系统的这些“部分”配合得如何，可用简单的代码快速地实现“刻划该系统的行为”的许多重要方法．往后再完善实现的细节．

1.2.3 什么是面向对象程序设计范型

面向对象程序设计范型：就程序而论，它是一个类的集合和各类之间以继承关系联系起来的结构，再加上一个主程序，其中定义各对象并规定它们之间传递消息的规律．就程序执行而论，它归结为各对象和它们之间以消息传递的方式进行的通讯．

在 OOP 程序执行过程中，从主程序所规定的各对象的初始状态出发，它们通过消息传递进行通讯，从而对象的状态受到改变而达到新状态．如此不断地进行状态的变换，达到的最终状态，即为计算结果．在对象的状态变换过程中，可能有新的对象产生，参加状态变换．

OOP 的最主要的特征是(各对象之间的)消息传递和(各类之间的)继承．

1.3 类、对象和封装

本节中较为精细地考察类与对象以及它们的封装．

类描述涉及定义所有那些性质和性能，它们刻划“作为此类的一实例”的任一对象的行为。

类定义的私有段通常包括：

● 通常定义作为基础的数据类型的(各)数据结构。

● 规定对那些方法的接口，它们仅仅在该类的作用域内是可存取的。这些方法经常用来支持各公开方法的实现。

类的私有段有时可含有“出自别的一些类”的各对象，这些对象只能在该给定类的作用域内予以处理。

类的公开段通常规定对那些方法的接口，它们形成“该类的跨越很多应用领域的可再用性”的基础。用发送消息给该类的各对象的办法，能够从该类的作用域之外调用这些方法。

封装是用来定义各个软件对象的一种技术。封装定义：

① 一清晰的界限，它围绕所有这些对象的内部软件的作用域。

② 一接口，它描述此对象如何与别的对象相互作用。

③ 一受保护的内部实现，它给出该软件对象所提供的功能的细节。在定义该对象的那个类的作用域之外，这些实现细节是不能存取的。

封装这个概念本来是与类描述有关的，但是，它也提供关于一问题解法的各个不同成分是如何分组的。这样，封装的单元是对象，它具有由它的类描述所描述的那些性质。这些性质被这同一个类的别的对象所共享。封装作为对象比“类表示封装”的说法更为具体、明确。用封装的这一定义，类的每一实例就是在一问题解法中的一个分立的封装或成分。

下例引入类、对象和封装，简要地探讨二叉搜索树的抽象。

在最高级上，为了表示二叉搜索树，识别出两个必要的对象。这两个对象的第一个被表示成“称之为 tree”的实例。因为二叉搜索树由各个结点组成，所以，第二种对象用“称之为 treenode 的类”的实例来表示。

非形式地刻划类 treenode 的各实例是“形成一个二叉搜索树”的各对象。每个 treenode 含有一个对象，此对象用来确定结构中数据的顺序关系。一个二叉 treenode 含有“指向左子树和右子树”的两对象。

下列各方法用来定义“能施行于类 treenode 的各对象之上”的那些动作：

● new-node　创建类 treenode 的一新实例。

● key　返回“存储在给定结点中的数据”的域，此域确定树的顺序。

● left　返回给定结点的左子结点。

● right　返回给定结点的右子结点。

图 1.1 描绘了类 treenode。

非形式地刻划类 tree 如下：

● 类 tree 和各实例是一些二叉搜索树。

● 一个二叉搜索树由各 treenode 组成，其中有一特殊结点叫做根结点。对树对象的一切存取都是通过它的根结点的。

● 在一个二叉搜索树中，插入和删除各结点必定产生一对象，后者也是一个二叉搜索树。

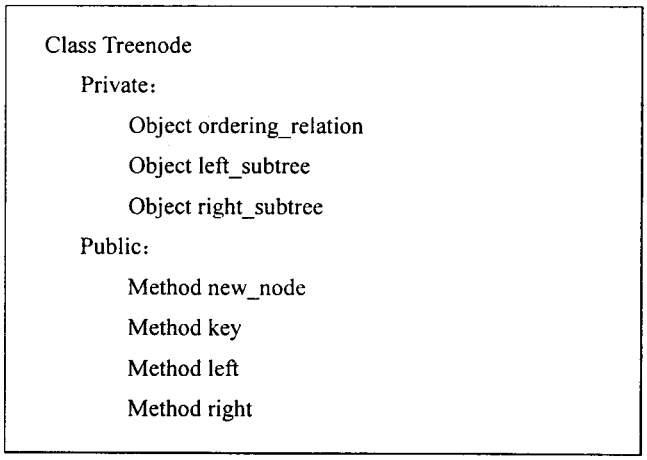

图 1.1　类 treenode 的描述

下列各方法用来定义“能施行于类 tree 的各对象之上”和那些动作：

- define　创建一新树。
- insert　插入一新点到该树中。
- remove　从该树中移去一现有结点。
- is –present　确定一特定结点是否在该树中。

图 1.2 描绘了类 tree。

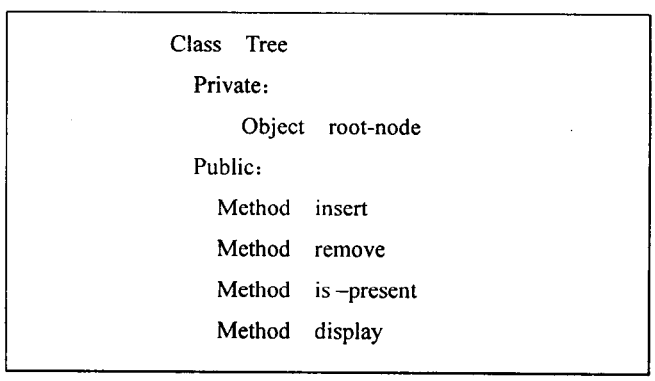

图 1.2　类 tree 的描绘

第2章 创建第一个 Visual C++应用程序

Microsoft Windows 已经被越来越多的人们所接受，人们开始越来越多地体会到图形用户界面(GUI)给人们带来的好处。程序设计与开发环境的发展使得人们不必再像过去那样花大量的精力开发用户界面，而可以把主要的精力集中在实现内在的逻辑功能上。在这方面 Visual C++是一个成功的范例。传统的开发工具要求用户编码实现每一个功能细节(尽管其中的一些功能是大多数开发人员都要实现的功能)，正如我们所发现的那样，Visual C++把大多数开发人员都需要的一些代码提供给程序开发人员。

2.1 编制第一个 Visual C++应用程序

Visual C++不仅能编辑(编译)源代码，而且能产生源代码，利用 AppWizard，可以在几分钟的时间内得到一个 Windows 应用程序。既然你不是第一个需要诸如 Open、Close、Print Setup、Print 等菜单选项的程序员，那么 AppWizard 能把几乎所有应用程序所需要的那些代码加入到你的程序中。

AppWizard 能产生各种各样的应用程序，但大多数初学者需要的是一个可执行程序，有的也许还要产生样板文件(Boilerplate)所需要的一些代码，如其它程序中要用到的类、对象及函数，为了产生这样的程序，首先从开始菜单选择 Projects 选项中的 MFC AppWizard(exe)，如图 2.1 所示。然后输入应用程序的名字并在 Location 栏中确定该应用程序的位置，再按

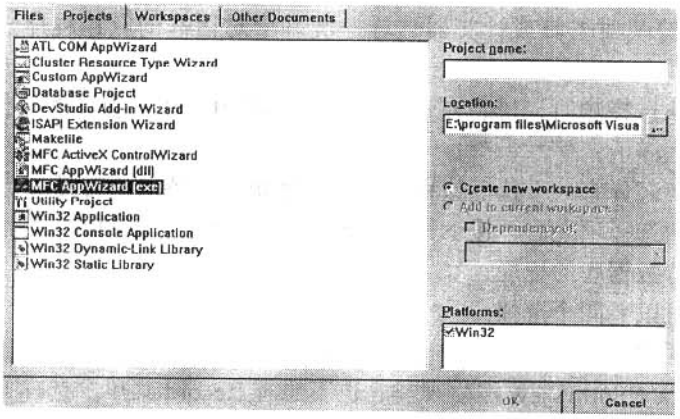

图 2.1 创建第一个应用程序

下 OK 按钮。AppWizard 通过一系列的步骤来完成一个应用程序的编制，在每一步中我们可以作出选择以决定所要的应用程序的类型，然后按 Next 按钮进入下一步。任何时候都可以按 Back 按钮退回到上一步，也可按 Cancel 按钮放弃整个过程。同时也可以按 Finish 按钮跳到最后接受系统的缺省值。下面详细讲述每一步骤。

如图 2.2 所示，首先确定文档类型，即选择是支持单文档界面(Single Document Interface SDI)、多文档界面(Multiple Document Interface MDI)还是基于对话框(Dialog-Based)，也就是决定用户界面的类型：

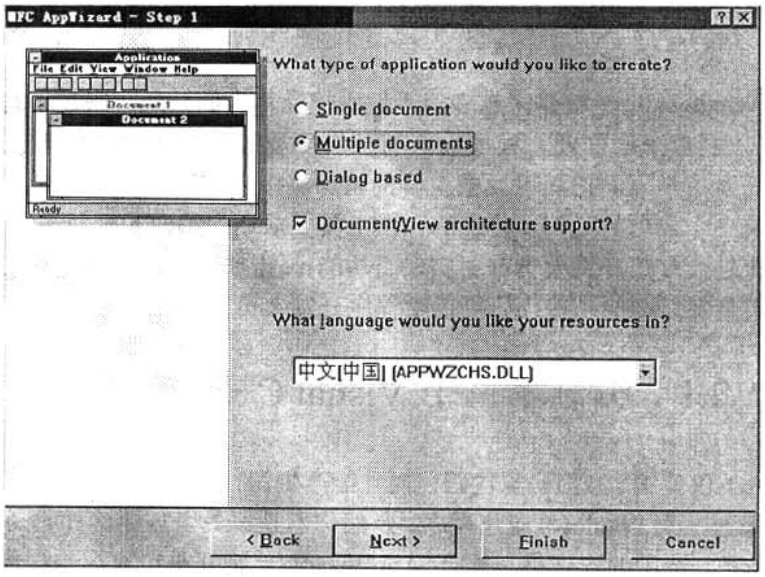

图 2.2　确定文档类型

● 对于 SDI，如写字板(Notepad)，一次只打开一个文档，当你选择 File→Open 菜单而打开一个新的文件时，当前打开的文件就被关闭。

● 对于 MDI，如 Excel 或 Word，一次可以打开多个文档，在 MFC 中，如果要一个文档上有多个视(view)，必须创建一个支持 MDI 的应用程序。

● 对于像字符映射工具或计算器这样一类基于对话框的应用程序，它根本就没有文档。

在选择不同的选择项时，屏幕左边的图片会提醒用户作出当前的选择时，应用程序界面看起来是个什么样子。

在屏幕的下半部分有一下拉框让程序员选择系统所支持的语言类型。

在作出当前的选择项之后，按 Next 按钮进入下一步，此时屏幕如图 2.3 所示，应用程序员选择程序支持数据库的程度。

从屏幕上看可以有四个选择：

● 如果没有写数据库的应用程序，选择 None；

● 如果不想从 CFormView 派生视类，也不想有 Record 菜单，选择 Header files only，即只用到 ODBC 必需的一些头文件；

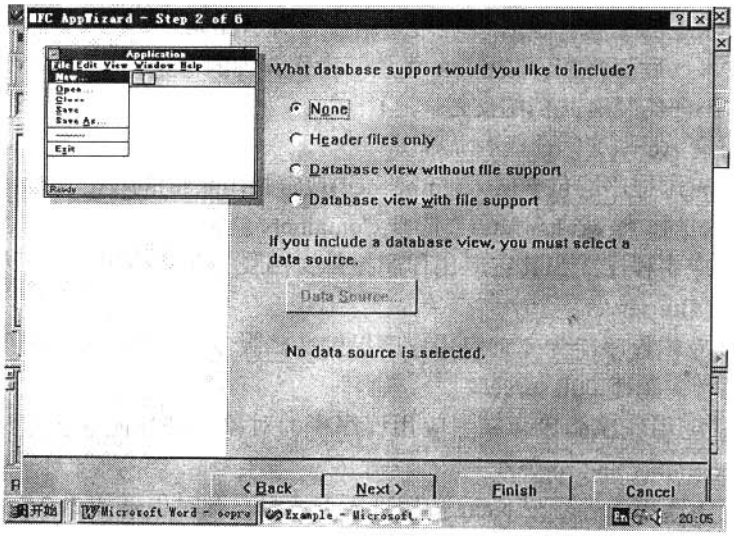

图 2.3　选择数据库的支持程度

● 如果想从 CFormView 派生视类，并且有 Record 菜单，但不必序列化(Serialize)一个文档，选择 Database view without file support，这样程序就可以通过 CRecordset 来修改数据库的元组(记录)，在后面的章节中将详细讨论这个。

● 如果想如前一点所述要求数据库的支持，同时要序列化文档，选择 Database view with file support。

当选择数据库的支持时，必须指定数据源。

在作出了上述选择之后，按 Next 按钮进入下一步，此时屏幕如图 2.4 所示，选择程序包括复合文档的数量。

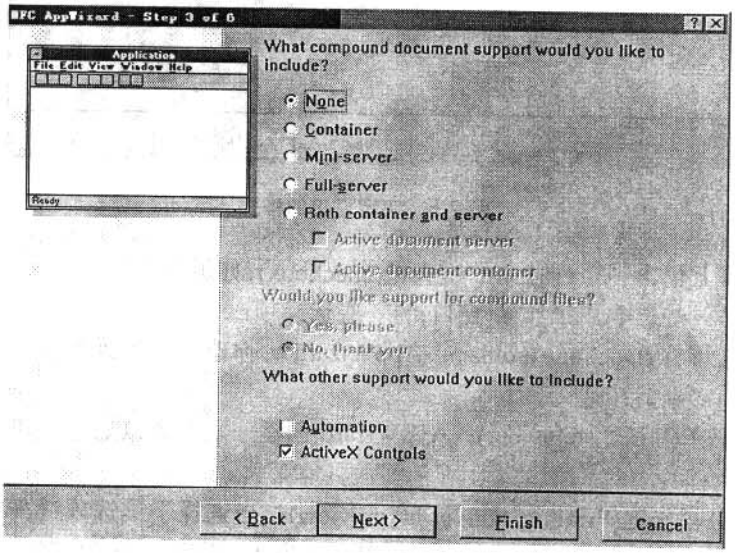

图 2.4　选择包括复合文档的类型

为了反映技术上的发展，OLE 已经被重新命名为 ActiveX，ActiveX 和 OLE 技术被统称为复合文档技术。后面将在“ActiveX 概念”中作详细的讨论。

可以有五种不同的复合文档的支持：

- 如果不想有 ActiveX，选择 None；
- 如果要求应用程序能包含嵌入(embedded)或链接(linked)的 ActiveX 对象，如 Word 文档或 Excel 中的工作表(worksheets)，选择 Container；
- 如果希望应用程序能为其它应用程序提供文档服务，且应用程序不必作为一单独的应用程序，选择 Mini server；
- 如果希望应用程序能为其它应用程序提供文档服务，且同时应用程序能够作为一单独的应用程序运行，选择 Full server；
- 如果希望应用程序能包含其它应用程序中的对象，并且能为其它应用程序提供对象，选择 Both container and server；

在作出了上述选择之后，按 Next 按钮进入下一步，此时屏幕如图 2.5 所示，选择窗口的样式及其它一些选项。

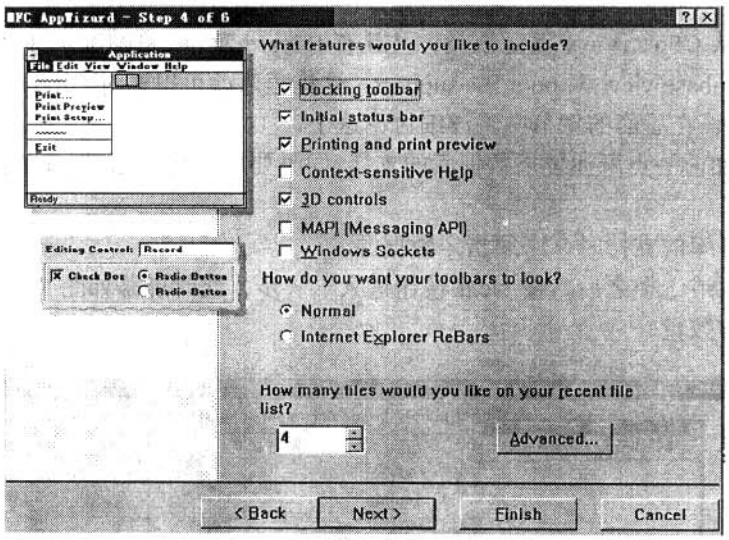

图 2.5　决定窗口的样式

此时的屏幕上包含一系列独立的检查框(checkbox)，如果想要某一特性，则检查(checked)该选项。下面是一些影响用户界面(窗口)的一些选择项：

- 预设的工具条(Docking toolbar)：AppWizard 为你建立一工具条，可以编辑或删除不想要的按钮或添加新的按钮。
- 初始状态条(Initial status bar)：AppWizard 创建一状态条以显示菜单的提示信息或其它信息；
- 打印和打印预览(Printing and print preview)：当选择了该选项时，在文件菜单中就会有打印和打印预览选择项，AppWizard 可以产生实现打印和打印预览的大部分代码。
- 上下文相关帮助：当选择了该选项时，AppWizard 可以产生为了帮助所需要的大部

分代码。

● 三维控件(3D controls)：当选择该选项时，应用程序看起来像一般的 Windows 95 应用程序，否则对话框有白色的背景且在编辑框(edit box)、检查框(check box)及其它的控件周围没有阴影。

● MAPI：当选择该选项时，应用程序可以使用消息 API 来发送 Fax、E-mail 及其它消息。

● Sockets：当选择该选项时，应用程序可以通过如 FTP、HTTP 等协议来直接访问 Internet。在后续的章节中，我们将讲述如何通过使用 Sockets 及 WinInet 类编制 Internet 程序。同时也可以确定在文件的最近文件列表中列出的文件个数。

对于不同的选项，AppWizard 产生不同的文件及类。

由上可知，在 Visual C++环境下开发一个简单的应用程序是多么的简单，这是因为 AppWizard 实现了大部分通用的功能。

2.2　Visual C++集成开发平台

Visual C++集成开发平台(developer studio)提供了一系列访问工程(project)、源文件(source file)、资源文件(resource file)的接口，通过该开发平台，程序员也可以编译、运行及测试应用程序。同时也给程序员提供了帮助。

Visual C++集成开发平台中最具有特色的地方就是其中的工作区(workspace)，如图 2.6 所示，工作区包括 ClassView、ResourceView、FileView。

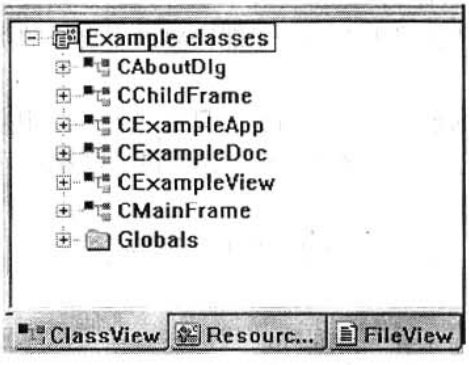

图 2.6　一个典型的工作区

① ClassView 列出组成一个工程的所有的类、类对应的成员函数及成员变量，工程中的全局函数及全局变量。当类嵌在其它类中时，ClassView 窗口提供了类的层次性的描述。双击某一类时能打开定义该类的文件。

② ResourceView 包含工程中所有的资源，如加速键(Accelerator)、对话框(Dialog)、控件(Icon)、菜单(Menu)、字符串表(String Table)、工具条(Toobar)及版本(Version)。

③ FileView 窗口列出工程中所有的文件，双击某一文件打开某一文件编辑。在这里也可以添加文件到一工程或从工程删除一些文件。

应用框架及 Windows 消息机制

本章介绍 Microsoft 基本类库(简称 MFC)应用框架，并着重讲述其优点。然后重点讲述消息循环及消息映射。

3.1 应用框架的概念

应用框架就是“为了生成一般的应用程序所必需的各种软件的集成组合”。这样的定义只能给读者一个比较抽象的概念。3.2 节的例子将是我们学习应用框架的良好开端。

C++之所以倍受欢迎，原因之一就在于它有大量可扩展的类库。有一些类库是随编译器一起提供的，有一些类库是由其它软件公司销售的，还有一些类库则是由用户自己开发的。类库是一个可以在应用中使用的相互联系的 C++类的集合。例如矩阵类库可以用来实现一般的矩阵运算，通信类库可以用来完成串联方式的数据交换。有些情况下人们需要创建一些类的对象，而在有些情况下人们又可能需要派生自己的类，所有这些都完全依赖于特定类库的实现。

应用框架是一种类库的超集。一般的类库只是一种可以用来嵌入任何程序中的孤立的类的集合，但应用框架却定义了程序的结构。这反映了两者之间的差别。

3.2 应用框架的例子

前面只是泛泛而谈，现在看一些实际的代码，一些可以真正编译运行的代码。该例子是一个具有最短代码同时又可以运行的 MFC 库应用实例。用户可以将它同 Windows SDK 应用作一下比较。下面是我们建立的第一个最简单的应用 MyFirstProgram，其中的头文件 MyFirstProgram.h 如下所示。

```
//application class
class CMyFirstProgramApp : public CWinApp
{
public:
CMyFirstProgramApp();
```

```
// Overrides
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CMyFirstProgramApp)
public:
virtual BOOL InitInstance();
//}}AFX_VIRTUAL

// Implementation

//{{AFX_MSG(CMyFirstProgramApp)
afx_msg void OnAppAbout();
    // NOTE - the ClassWizard will add and remove member functions here.
    //      DO NOT EDIT what you see in these blocks of generated code !
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};
```

对于其它的一些文件在此不详细地介绍，下面介绍程序中的一些元素：

WinMain 函数：Windows 总是要求每个应用都要有 WinMain 函数，我们之所以在此见不到此函数，是因为它被 MFC 隐藏在应用框架内部了，而在 SDK 开发中，编程者要直接处理 WinMain。

CMyFirstProgramApp 类：类 CMyFirstProgramApp 的对象就代表了一个应用．该程序定义了一个单独的全程 CMyFirstProgramApp 对象，其基类 CWinApp 决定了它的大部分行为。

当人们运行该应用时，Windows 会自动调用应用框架内部的 WinMain 函数，WinMain 函数会去查找该应用的全程构造对象，该对象是由 CWinApp 所派生出的类的对象．在 C++ 中全程对象在主程序运行之前就已经构造好了。

当 WinMain 发现了应用对象时，它会自动调用 InitInstance 成员函数，该函数会进一步调用相应的函数来完成主窗口的构造和显示工作。

3.3　理解 Windows 消息机制

Windows 程序与其它类型的程序最根本的区别就是消息机制．例如 DOS 程序，要等待(有时也称为检测)可能的输入，如键盘或鼠标．如果程序不检测鼠标，就不会对鼠标的动作做出反应．在 Windows 程序中以消息作为媒介，在操作系统中用消息来告诉应用程序有情况发生了，例如用户按下某个键、单击或移动了鼠标、打印机可用了等等．一个窗口(每一个元素都是一个窗口)也可以发消息给另外一个窗口．MFC 使得处理消息非常简单，但是你必须掌握消息的原理。

通常使用消息的名称来引用消息，而操作系统则用整数来代表每一消息．系统内部有

一张大的表格用#define 语句将消息的名字与这些整数对应起来，使得 Windows 程序员可以用 WM_PAINT 或 WM_SIZE 或其它消息名来引用各种消息(WM 代表 Windows 消息)。表 3.1 列出了部分消息的宏定义。

表 3.1　消息名称与操作系统内部整数的对应关系

消息名称	系统内部整数
WM_SETFOCUS	0x0007
WM_GETFOCUS	0x0008
WM_ENABLE	0x000A
WM_SETTEXT	0x000B
WM_GETTEXT	0x000C
WM_GETTEXTLENGTH	0x000D
WM_PAINT	0x000F
WM_CLOSE	0x0010
WM_QUERYENDSESSION	0x0011
WM_QUIT	0x0012

消息将知道它被发送到哪一个窗口，消息通常可以带两个参数，而这两个参数可以存放不同的值。

不同的消息是由操作系统的不同部分或由应用程序来控制的。例如，当用户移动鼠标经过一个窗口时，该窗口就收到一条 WM_MOUSEMOVE 消息，这条消息要传给操作系统来处理。操作系统就在新位置画出鼠标指针。当鼠标左键在一个按钮上单击时，这个按钮(也是一个窗口)就收到 WM_LBUTTONDOWN 消息并处理它。

MFC 允许大多数程序员完全忽略底层消息，如 WM_MOUSEMOVE 或是 WM_LBUTTONDOWN。他们只需要处理高级的消息，如“列表框的第二项被选中”或是“某一按钮被单击了”。所有这些高层消息与那些底层消息以同样的方式在程序代码和操作系统的代码中传送。唯一不同的是由谁来处理它们。利用 MFC 在类一级能比较容易地声明每一类可以处理的消息。

3.4　Windows 消息循环

Windows 程序的核心是消息循环机制，通常包含在 WinMain()例程(在 VC++环境下，它被隐含在 AppWizard 自动生成的代码中)中。WinMain()例程像 DOS 或 UNIX 中的 main()函数一样，是运行程序时由操作系统调用的函数。下面是一典型的 WinMain()例程。

```
int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
```