21世纪 高职高专新概念教材

杨国兴 主 编
扈健丽 副主编
杨莉 王希辰 尹有仁
许学东 主 审

Visual C++ 6.0 实例教程

21 Shi Ji Gao Zhi Gao Zhuan Xin Gai Nian Jiao Cai

中国水利水电出版社
www.waterpub.com.cn

21世纪高职高专新概念教材

编委会名单

参 编 学 校 名 单

（按第一个字笔划排序）

三门峡职业技术学院
山东大学
山东建工学院
山东省电子工业学校
山东农业大学
山东省农业管理干部学院
山东省教育学院
山西阳泉煤炭专科学校
山西经济管理干部学院
广州市职工大学
广州铁路职业技术学院
中国人民解放军第二炮兵学院
中国矿业大学
中南大学
天津市一轻局职工大学
天津职业技术师范学院
长沙大学
长沙民政职业技术学院
长沙交通学院
长沙航空职业技术学院
长春汽车工业高等专科学校
北京对外经济贸易大学
北京科技大学职业技术学院
北京科技大学成人教育学院
石油化工管理干部学院
石家庄师范专科学校
华中电业联合职工大学
华中科技大学
华东交通大学
华北电力大学工商管理学院
江汉大学
西安外事学院
西安欧亚学院
西安铁路运输职工大学
西安联合大学
孝感职业技术学院
杨陵职业技术学院
昆明冶金高等专科学校
武汉大学动力与机械学院
武汉大学信息工程学院
武汉工业学院
武汉工程职业技术学院
武汉广播电视大学
武汉化工学院
武汉电力学校
武汉交通管理干部学院
武汉科技大学工贸学院
武汉商业服务学院
武汉理工大学
河南济源职业技术学院
陕西师范大学
南昌水利水电高等专科学校
哈尔滨金融专科学校
济南大学
济南交通高等专科学校
荆门职业技术学院
贵州无线电工业学校
贵州电子信息职业技术学院
恩施职业技术学院
黄冈职业技术学院
黄石计算机学院
湖北工学院
湖北丹江口职工大学
湖北交通职业技术学院

湖北汽车工业学院
湖北经济管理大学
湖北药检高等专科学校
湖北商业高等专科学校
湖北教育学院
湖北鄂州大学
湖南大学
湖南工业职业技术学院
湖南计算机高等专科学校
湖南省轻工业高等专科学校
湖南涉外经济学院
湖南郴州师范专科学校
湖南商学院
湖南税务高等专科学校

序

根据1999年8月教育部高教司制定的《高职高专教育基础课程教学基本要求》（以下简称《基本要求》）和《高职高专教育专业人才培养目标及规格》（以下简称《培养规格》）的精神，由中国水利水电出版社北京万水电子信息有限公司精心策划，聘请我国长期从事高职高专教学、有丰富教学经验的教师执笔，在充分汲取了高职高专和成人高等学校在探索培养技术应用性人才方面取得的成功经验和教学成果的基础上，撰写了此套《21世纪高职高专新概念教材》。

为了编写本套教材，出版社进行了广泛的调研，走访了全国百余所具有代表性的高等专科学校、高等职业技术学院、成人教育高等院校以及本科院校举办的二级职业技术学院在广泛了解情况、探讨课程设置、研究课程体系的基础上，经过学校申报、征求意见、专家评选等方式，确定了本套书的主编，并成立了编委会。每本书的编委会聘请了多所学校主要学术带头人或主要从事该课程教学的骨干，教学大纲的确定以及教材风格的定位均经过编委会多次认真讨论。

本套《21世纪高职高专新概念教材》有如下特点：

（1）面向21世纪人才培养的需求，结合高职高专学生的培养特点，具有鲜明的高职高专特色。本套教材的作者都是长期在第一线从事高职高专教育的骨干教师，对学生的基本情况、特点和认识规律等有深入的了解，在教学实践中积累了丰富的经验。因此可以说，每一本书都是教师们长期教学经验的总结。

（2）以《基本要求》和《培养规格》为编写依据，内容全面，结构合理，文字简练，实用性强。在编写过程中，作者严格依据教育部提出的高职高专教育“以应用为目的，以必需、够用为度”的原则，力求从实际应用的需要（实例）出发，尽量减少枯燥、实用性不强的理论概念，加强了应用性和实际操作性强的内容。

（3）采用“问题（任务）驱动”的编写方式，引入案例教学和启发式教学方法，便于激发学习兴趣。本套书的编写思路与传统教材的编写思路不同：先提出问题，然后介绍解决问题的方法，最后归纳总结出一般规律或概念。我们把这个新的编写原则比喻成“一棵大树、问题驱动”的原则。即：一方面遵守先见（构建）“树”（每本书就是一棵大树），再见（构建）“枝”（书的每一章就是大树的一个分枝），最后见（构建）“叶”（每章中的若干小节及知识点）的编写原则；另一方面采用问题驱动方式，每一章都尽量用实际中的典型实例开头（提出问题、明确目标），然后逐渐展开（分析解决问题），在讲述实例的过程中将本章的知识点融入。这种精选实例，并将知识点融于实例中的编写方式，可读性、可操作性强，非常适合高职高专的学生阅读和使用。本书读者通过学习构建本书中的“树”，由“树”找“枝”，

顺“枝”摸“叶”，最后达到构建自己所需要的“树”的目的。

（4）配有实验指导和实训教程，便于学生练习提高。

（5）配有动感电子教案。为顺应教育部提出的教材多元化、多媒体化发展的要求，每本教材都配有电子教案，以满足广大教师进行多媒体教学的需要。电子教案用 PowerPoint 制作，教师可根据授课情况任意修改。

（6）提供相关教材中所有程序的源代码，方便教师直接切换到系统环境中教学，提高教学效果。

总之，本套教材凝聚了数百名高职高专一线教师多年的教学经验和智慧，内容新颖，结构完整，概念清晰，深入浅出，通俗易懂，可读性、可操作性和实用性强。

本套教材适用于高等职业学校、高等专科学校、成人及本科院校举办的二级职业技术学院和民办高校。

新的世纪吹响了我国高职高专教育蓬勃发展的号角，新世纪对高职教育提出了新的要求，高职教育占据了全面素质教育中所不可缺少的地位，在我国高等教育事业中占有极其重要的位置，在我国社会主义现代化建设事业中发挥着日趋显著的作用，是培养新世纪人才所不可缺少的力量。相信本套《21 世纪高职高专新概念教材》的出版能为高职高专的教材建设和教学改革略尽绵薄之力，因为我们提供的不仅是一套教材，更是自始自终的教育支持，无论是学校、机构培训还是个人自学，都会从中得到极大的收获。

当然，本套教材肯定会有不足之处，恳请专家和读者批评指正。

21 世纪高职高专新概念教材编委会

2001 年 3 月

前　言

本书以编程实例和解析的方式，介绍使用 Visual C++开发应用程序所需的主要知识。主要内容包括：Visual C++基础知识与开发环境，文档和视编程，对话框与控件，菜单、快捷键与控制条，图形操作、文件操作，MFC 通用类，异常处理和诊断，多线程编程。

本书以程序设计为主线，通过对实际例题的分析引出基本知识，书中的实例是根据作者多年的程序开发经验和教学经验精心编排的，既适合于课堂教学，又可以进行一些修改用于读者自己的程序中。

本书是高等职业学校、高等专科学校计算机专业程序设计教材，供高职高专计算机专业学生使用；同时本书并未停留在初学者的水平上，书中的大部分实例都取材于作者以往的程序开发实践，因此，对于使用 Visual C++进行程序开发的技术人员也有一定的参考价值。

根据高等职业教育和 Visual C++本身的特点，本书加强了实际编程能力的训练，从实际程序开发的需要出发，注重培养学生运用基本知识解决实际问题的能力，而不过分追求知识的系统性和完整性。

书中所给出的实例全部在 Visual C++ 6.0 环境下调试通过，为了方便读者区分 VC++自动生成的代码和程序员添加的代码，在本书中，凡是第一次出现的需要程序员添加的代码，均以暗底色给出。

为了方便教师教学，用 PowerPoint 制作了与本教材配套的电子教案，教师在使用时可以根据需要进行必要的修改。

本书由杨国兴主编，杨莉、王希辰、尹有仁、扈健丽任副主编，参加本书编写工作的还有连瑞永、艾澍雨、王希辰、李明等。其中，第 1 章由艾澍雨编写，第 2 章由尹有仁编写，第 3 章由蒋伟荣编写，第 4 章由杨莉、扈健丽合编，第 5 章由连瑞永编写，第 6 章由李明编写，第 7 章、第 8 章、第 9 章由杨国兴编写。

北京科技大学许学东教授审阅了全部书稿，并提出许多改进意见，北京科技大学谢平老师打印了部分书稿，在此表示衷心的感谢。

由于编者水平有限，书中不妥或错误之处在所难免，恳请专家和读者批评指正。

编　者

2001 年 3 月

目　　录

第 1 章　VC++基础知识与 Visual C++ 6.0 开发环境

本章学习目标

本章首先简单介绍 VC++的一些基础知识，然后利用应用向导创建一个单文档应用程序和一个基于对话框的程序。通过本章学习，读者应该掌握以下内容：

- 了解 VC++的一些基本概念
- 了解 Visual C++ 6.0 开发环境
- 利用应用向导建立一个应用程序框架

1.1　VC++基础知识

1.1.1　Visual C++和 MFC 的历史

在 Windows 平台下 32 位应用程序开发中有很多计算机开发语言，但几年来 Visual C++一直以它独特的魅力在众多编程工具中独树一帜，那到底是为什么呢？与其他编程工具相比，Visual C++ 不但提供可视化的编程环境，更在编写直接对系统进行底层操作的程序方面独领风骚，并且其生成代码的质量，也在众多开发工具中名列前茅。还有 Visual C++提供的 Microsoft 基础类库（Microsoft Foundation Class Library，简写为 MFC），对 Windows 的 Win32 应用程序接口（Win32 Application Programming Interface）进行了十分彻底的封装，使得可以使用完全面向对象的方法来进行 Windows 应用程序的开发，不但避免了对某些技术细节需要深入了解的麻烦，而且大大节省了应用程序的开发周期，降低了开发成本，也使得 Windows 程序员从大量的复杂劳动中解脱出来。

Visual C++的核心是 Microsoft 基础类库，即通常所说的 MFC。MFC 相当彻底地封装了 Win32 软件开发工具包（Software Development Kit，即通常所说的 SDK）中的结构、功能，它为编程者提供了一个应用程序框架，这个应用程序框架为编程者完成了很多 Windows 编程中的例行性工作，如管理窗口、菜单和对话框，执行基本的输入和输出、使用集合类来保存数据对象等等，并且，MFC 使得在程序中使用很多过去很专业、很复杂的编程课题，如 ActiveX、OLE、本地数据库和开放式数据库互联（Open Database Connectivity，简写为 ODBC）、Windows 套接字和 Internet 应用程序设计等，以及其他的应用程序界面特性，如属性页（也叫标签对话框）、打印和打印预览、浮动的和可定制的工具条变得更加的容易。

早在 1989 年，Microsoft 的程序员们开始试图将 C++和面向对象的编程概念应用于 Windows 编程中，以编写出一个可以使 Windows 编程更加简便的应用程序框架。他们把这个应用程序框架叫做 AFX （AFX 这个词来源于 Application Framework，但奇怪的是这个词组中并没有包含“X”这个字母）。直到今天，AFX 小组早已不存在了，AFX 这个名称也于 1994 年初不再使用，但在 Visual C++和 MFC 中，AFX 的影子却随处可见，很多全局函数、结构和宏的标识符都被加上了 AFX 的前缀。

最初的 AFX 版本在经过一年的艰苦之后诞生，却未能被大多数 Windows 程序员所接受。AFX 的确是经过了精心的规划和编码，并且，它也提供了对 Windows API 的高度抽象，建立了全新的面向对象的 AFX API，但最致命的缺点是 AFX API 库根本不兼容于现有的 Windows API。由此导致的最严重后果是大量的 SDK 代码无法移植，而程序员将学习两种完全不同的编程方法。

AFX 不得不重新做所有的一切，它所创建的新的应用程序框架是一套扩展的 C++类，它封装和映射了 Windows API，这就是 MFC 的前身。过去的 AFX 小组也变成了 MFC 小组。最终，MFC 的第一个公开版本于 1992 年 3 月随 Microsoft C/C++ 7.0 （而不是 Visual C++ 1.0）一起推出。那时距 Windows 3.1 发布尚有好几个月。在 MFC 1.0 中还没有文档/视结构，但有类 CObject 和 CArchive。在 12 个月之后，MFC 2.0 随 Microsoft 新的编程工具 Visual C++ 1.0 一道出炉。与 MFC 1.0 一样，MFC 2.0 仍是 16 位的，因为 32 位的 Windows NT 3.1 直到 1993 年 7 月才问世。在 MFC 2.0 中，增加了对文档/视结构、OLE 1.0、Windows 3.1 公用对话框的支持和消息映射等。在 Windows NT 3.1 面世一个月以后，Microsoft 推出了 32 位版本的 Visual C++和 MFC 2.1，它实际上是 MFC 2.0 的 Win32 接口。

最后一个 16 位的 Visual C++编译器是 1993 年 12 月推出的 Visual C++ 1.5，直到今天，一些为 Windows 3.1 编写 16 位应用程序的程序员还在使用这个版本。1994 年 9 月，32 位的 MFC 3.0 伴随着 Visual C++ 2.0 一道面市，在今天的计算机图书市场上，还有着大量关于 Visual C++ 2.0 和 MFC 3.0 的图书出售，因此，你可以想象得出 Visual C++ 2.0 所取得的成功和它所产生的影响。发展到今天，Visual C++ 已经推出了 6.0 版本， MFC 也已发展成一个稳定和涵盖极广的 C++类库，为成千上万的 Win32 程序员所使用。MFC 库是可扩展的，它和 Windows 技术的最新发展到目前为止始终是同步的。并且，MFC 类库使用了标准的 Windows 命名约定和编码格式，所以有经验的 Windows SDK 程序员很容易过渡到 MFC。MFC 结合了 Windows SDK 编程概念和面向对象的程序设计技术，从而具有极大灵活性和易用性。

1.1.2 帮助的使用

随着应用程序越来越复杂，程序员所需掌握的知识也越来越多，然而，在很多情况下，我们几乎不可能把所有的知识都记到大脑里。

因此，在学习使用 Visual C++进行应用程序设计之前，先学习一下如何从 Visual C++的集成开发环境 Developer Studio 中获得帮助是很有必要的。

最常用的一种方法是通过上下文相关的帮助快速获得所需的信息。打开上下文相关帮助的快捷键是 F1。上下文相关的帮助可以用于多种场合。

常见的情况是从代码编辑器窗口获取与关键字、函数和类的相关的帮助，其步骤如下：

（1）将光标定位于所需获取帮助的关键字、函数或类名及类成员名，按下快捷键 F1。

（2）当与指定的关键字、函数或类及成员相关的帮助条目仅有一条时，Developer Studio 直接在帮助窗口中打开该主题，否则，将在 Results List 窗口中列举相符合的所有主题，双击其中 Title 栏下的某一项以打开相应主题。

另外还可以在 Output 窗口或是在对话框中获得上下文相关的帮助，方法也是类似的。在对话框中还可以使用所谓的“What's This”按钮。

除了上面所说的这些方法之外，还可以在 Help 菜单中选择 Search 命令或者单击工具条上的按钮来打开查询对话框。该对话框分成两个选项卡：选项卡 Index 以索引方式来查询在线文档，文档中的每一个主题都与一个或多个索引关键字相联系；反之，一个索引关键字也可能与多个文档主题相联系。另一个选项卡称作 Query （查询）选项卡。该选项卡允许你指定查询字符串，然后从在线文档中查找匹配的所有文档，并且，还可以指定多种不同的查找方式和范围。一般来说，通过 Query 得到的结果要比使用 Index 的庞大得多，并且，由于 Query 还可以对文本，而不仅仅是标题进行匹配，因此，也许会得到一些事实上和所需要的资料无关的结果。这使得使用 Query 没有使用 Index 那么方便和有效。然而，Query 方式也有其先进之处，在 Query 选项卡的 Type in the word（s） to find 处可以使用含逻辑运算符的查询表达式，可以使用的逻辑运行符包括 AND、OR、NOT 和 NEAR，其中，AND、OR、NOT 分别还可以简写为“&”、“|”、“!”。

1.1.3　面向对象的编程语言

面向对象的编程是当前程序设计中的热门话题，在这里，我们仅仅介绍使用 Visual C++ 进行面向对象编程中所用到的一些关键概念，这些概念是进一步学习 Visual C++所必需的。除了介绍面向对象编程和 C++语言的基本概念外，还将介绍在使用 C++的类时所需注意的一些问题。

传统的“结构化程序设计”方法是由荷兰学者 Dijkstra 在 70 年代提出的，他把面向机器代码的程序抽象为三种基本程序结构：顺序结构、选择结构和重复结构，并提出了一系列的设计原则，如自上而下、逐步求精、模板化编程等。根据这些原则，按照程序所需实现的功能，自上而下层层展开。上层是定义算法的模块，最下层是实现算法的模块。按照这样的规范构成的模块是高度功能性的，有很强的内聚力。但各模块的数据处于实现功能的从属地位，因此，各模块与数据间的相关性就较差，无论把数据分放在各个模块里还是作为全局变量放在总控模块中，模块之间都有很大的耦合力。在 Windows 程序中，多个模块是并发执行的，这样，这种耦合力就会极易导致程序系统出现混乱。

为了最大限度地利用已有的资源和减少程序开发的工作量，需要有一种比传统的过程式

结构化程序设计方法抽象能力更强的新方法，面向对象的程序设计方法正是在这种背景下诞生的。

面向对象的程序最根本的目的就是使程序员更好地理解和管理庞大而复杂的程序，它在结构化程序设计的基础上完成进一步的抽象。这种在设计方法上更高层次的抽象正是为了适应目前软件开发的特点。

使用面向对象的程序设计方法绝非是要摒弃现有的结构化程序设计方法，相反，它是在充分吸收结构化程序设计优点的基础上，引进了一些新的、强有力的概念，从而开创了程序设计工作的新天地。面向对象的程序设计方法把可重复使用性视为软件开发的中心问题，通过装配可重用的部件来生产软件，而不是像以前编程所用的那样，通过调用函数库中的函数来实现。这里要注意，我们是基于应用程序这一个层次来阐述这些问题是，事实上，在对象内部的实现上，我们常常使用过程式的结构化程序设计方法，也常常调用 C/C++函数库中的很多有用的函数，然而从程序的总体结构上说，它是由一系列对象构成的，对象之间能够以某种方式进行通信和协作，从而实现程序的具体功能。

面向对象的程序设计中最基本的概念是对象，一般意义上的对象指的是一个实体的实例，在这个实体中包括了特定的数据和对这些数据进行操作的函数。对于面向对象的程序设计，一个对象具有状态（state）、行为（behavior）和标识（identity）。对象的状态包括他的属性和这些属性的当前值。对象的行为包括可以进行的操作以及所伴随的状态的变化。对象的标识用来区别于其他的对象。

对象的核心概念就是通常所说的“封装性”（encapsulation）、“继承性”（inheritance）和“多态性”（polymorphism），下面我们分别阐述其具体含义。

封装按照面向对象编程原则定义，所谓的封装性是指隐藏类（class）为支持和实施抽象所作的内部工作的过程。类的接口是公有的，它定义了一个类所能完成的功能，而这些接口的实现是私有的或受保护的，它定义了类完成这些功能所作的具体操作。对于使用这些类的编程者来说，只需要知道类所能完成的功能，而不需要知道这些功能具体是如何实现的。拿我们的手表作为例子，在使用手表时，我们只需要知道手表所能完成的功能和如何使用手表来完成这些功能，这些内容相当于对象的接口。我们不需要知道在手表的内部，这些功能是如何实现的，因此，对于手表来说，无论手表使用的是一般的机械摆，还是石英振荡器，只要它们的使用方法是完全一样的，用户就没有必要知道这个不同。另一个例子的集成电路芯片，我们只需要知道该芯片的每一个引脚的电气参数和功能，而不必知道这些功能在芯片的内部是如何实现的，就可以使用该芯片来组装电路。如果仅就封装性而言，这里的手表和集成电路芯片就相当于面向对象的编程中的对象。

一个类在定义数据的同时也定义了对这些数据的操作，这些操作称作方法（method）。按照面向对象的定义，方法就是对对象中的数据的访问。在 C++中，对对象中的数据的访问是通过公有成员函数来进行的，这些公有成员函数可以在对象的外部进行调用，他们提供了对象的外部接口。而对于这些接口的内部实现在对象的外部是不可见的，这些实现包括了类

内部所使用的数据结构和支持公有方法实现的私有成员函数，通常，这些数据成员和成员函数是私有的，他们只能为类中成员函数所访问，而不能从类的外部进行访问。

访问一个方法的过程称为向这个对象发送一个消息（message），对象的工作是靠消息来激发的，对象之间也是通过消息发生联系的，即请求其他对象做什么或响应其他对象的请求是通过发送或接收消息来实现的。

封装可避免许多维护性问题。如果一个基本数据类型的结构被修改了，例如一个链表修改成了一个数组，除类中访问该数据方法的代码外，软件系统的其余部分是不受影响的，因为基本数据在外部是不可见的，只能通过公有方法的接口与基本数据发生联系，改变一个类的实现，丝毫不影响使用这个类的程序员，从而大大的减少了应用程序出错的可能性。

什么是继承呢？类支持层次机制，因此我们可以借用可重用性部件来很容易的从一个或多个已有的类出发，来生产各种更符合我们要求的新类。假设我们从类 A 出发来派生新的类 B，那么我们称类 A 为类 B 的基类（base class），类 B 为类 A 的派生类（derived class），类 B 继承了类 A 中的各种行为和状态，并可添加自己的成员变量和成员函数。然后，可以将这个结构几乎不作修改或者只需作少量修改地用面向对象的编程来表达，从而大大地缩短了软件系统的开发周期。

继承机制所带来的最大优势在于使软件系统非常易于扩充，程序员不仅可以直接使用各种已有的类，还可以从这些类方便地派生出新的类，新的类继承了基类所包括的所有接口和功能，因此只需要定义和实现与基类所提供的功能中不同的那一部分，这大大降低了软件开发的复杂性和费用，因此面向对象的编程方式非常之适合于进行大型软件系统的开发。

继承机制还使得我们可以将与现实生活空间相一致的思维方式应用于程序空间，即我们可以在程序设计时使用直观的思维方式设计程序中所使用的对象层次结构，然后，直接将此结构映射到面向对象的程序空间，而不需要做任何修改或仅需要作少量修改就可以使用面向对象的程序设计方法来实现该结构。这时，编写程序的过程更类似于“搭积木”。

举一个简单的例子：

```
int print（char*）；
int print（int）；
```

上面的两个不同函数使用同样的函数名，在 C++中，这称为函数的重载。这时，若将一个字符指针传递给 print 函数，如下所示：

```
char *sz="Hello World!";
print（sz）；
```

这时，编译器调用的是 int print（char*），如果将一个整型变量传递给 print 函数，如下所示：

```
int i=0;
print（i）；
```

则编译器调用的是 int print（int）。这种根据所传递的参数的不同而调用不同函数的情形