目　录

第一周

第二周

第三周

附 录

第一周一瞥

现在，请准备好开始进行 Visual C++的学习。在今后的三个星期中，本书将介绍如何使用 Visual C++进行面向对象的程序设计(object—oriented programming，即 OOP)。如果读者对 OOP 是什么一无所知，这二十一天的学习将揭开它神秘的面纱，这本书由课文、练习、说明以及供复习用的问答题和练习构成，全部习题的答案在附录 D 中给出。所有这些部分完美地组合在一起，每一部分对于理解 OOP 的概念，以及掌握使用 Visual C++编写面向对象的程序都有极大帮助。

为了巩固当天所学的知识，在每一天课程的末尾都配有复习和小测验。在学完一天的课程之后，读者应该能够回答测验中的题目，并且编写出符合练习要求的程序(在练习的过程中，务必记住：同一程序往往有若干种实现方法，因此，如果读者编写的程序和本书提供的答案不一样，该程序也可能是正确的)。

欢迎进行第一周的学习

在第一个七天中接触到的 Visual C++并不涉及到 OOP 概念。读者买本书就是为了学习 OOP，而本书的主旨也是为了介绍 OOP，但是，掌握面向对象的程序设计方法的过程可以明确地分为三个阶段，这和本书三个星期式的结构正好相符，在这第一周的学习中，读者将通过本书对 Visual C++中 OOP 概念以外部分的介绍，理解 OOP 方法的基础(由 Visual C++改进了的这部分语言表面上看起来和 OOP 无关，事实上却正是用来支持 OOP 的)。

读者一旦接触到 OOP，就会发现 Visual C++提供了一种比 C 语言更好的程序设计方法，即使只掌握了第一个星期所教的知识，而没有学习 OOP，读者仍将发现 C++比 C 语言更加先进(确实如此)。

> **注意**：本书假定读者已具备了一定的 C 语言基础，如果读者想要复习一下 C 语言的基础知识，可以参看本书的附录 E。

C++的设计者并不仅仅是为了改进 C 语言，他们设计这种语言是为了获得一种支持面向对象的程序设计方法的语言，结果 C++比 C 语言更加先进，但是所有这些改进不仅支持了 OOP 设计方法，还使非 OOP 程序设计有了长足的发展。

在第一个星期中，读者将对如何使用 Visual C++的编辑器来编辑或者编译 C++的程序有所了解，并且马上就会意识到 C++对程序的数据输入输出进行了怎样的改进，也将知道为什么 C++的程序员从不使用 printf()和 scanf()(恰恰在读者以为自己熟练掌握了它们的时候)。

第一个星期的课程还提出了改进设计和编写函数的技巧，C++增加了许多省时省力方便快捷的方法，在头两天的学习中读者就会用到其中一些方法。通过对注释语句的一个简单的改变，读者可以重写一些内部函数使它们完成所需要的功能。C++中含有一些不涉及 OOP 思想的命令，同时还对 C 语言进行了改进。在这个星期里，本书提供了这两方面内容的有关知识。

另外，用 Visual C++编程的一个优点是读者所掌握的有关 C 语言的知识仍旧有用，Visual C++支持几乎所有 C 的特性，C 的命令、语法并且读者已经知道并掌握的所有的头文件和函数在 Visual C++的程序中都能正常运行。通常情况下，当学习一种新的语言时，一切都必须从头开始学，如数据的类型、赋值语句等，而学习 Visual C++，却可以一开始就编写大段的程序。

完全掌握了第一星期的课程后，就可以准备运用这些知识用 Visual C++编写面向对象的程序了。

第一天　C++概览

欢迎阅读本书！第一章旨在激发读者学习C++的兴趣。在本天，读者将学习以下三个内容：

- [ ] C++的历史。
- [ ] 面向对象的程序设计方法(OOP)的功能。
- [ ] OOP程序员使用C++的优越性。

1.1　欢迎来到面向对象的C++世界

很多人认为C++是C语言的扩展，但实际上并非如此——C++是一种独立的语言。C++大大提高了C的性能，不过它的基础仍然是C。一个C++程序员必须认识到这一点，否则将会产生许多问题。学习C++的读者中有许多在此之前已经是C程序员，这一点是显而易见的，因为毕竟C在1980年公布，而C++在1990年才公布。

因为C++面向对象的特性，许多程序员正在由C或者别的编程语言转向C++。面向对象的程序设计，即OOP，如同众所周知的那样，把1980年以来的以过程为核心概念的传统程序设计方法所支撑的软件产业向前推进了一步。

面向对象的程序设计方法提供了许多强有力的工具，帮助程序员达到预期目标(编制和调试应用程序)，这种方法比以往的面向过程的设计方法更富有效率。面向对象的程序设计把数据封装在一个安全的外壳中，并使它们具有活动性，变量能够对自己进行主动的操作而非被动地等待程序中的代码对它们进行操作。当需要打印一个变量的内容时，你无须去打印它，而只需要告诉变量去打印自己！这种魔术般的编程方法正是一种最自然的编程方法。

1.1.1　C++的历史

建立起C++应归功于Bjarne Sroustrup。他发展了C++，使程序中的事件成为现实中事件的模拟，从而使C++几年后风靡海内外。Stroustrup揭示出了普通的不具备OOP思想的程序设计方法的缺陷，即它不能像面向对象的程序设计那样在程序中模拟现实中的事件来完成某项任务。

Stroustrup为AT&T公司的Bell实验室工作了许多年，致力于发展和提高C++。美国国家标准学会(ANSI，American National Standards Institute)作为一个制定所有和计算机有关的语言标准的组织也接受了C++的标准。但是他们还在致力于制定一个更完备的标准，这是因为ANSI还没有为C++制定出一个实际可行的标准来和AT&T的标准竞争。

Microsoft公司是Visual C++软件的制作者。AT&T公司需要一个基准测试程序来确定一个编译器是否足够先进可以用作C++的正式编译器。除了不支持样板之外，Microsoft的C++编译器严格遵循了AT&T的标准。样板是一个被激烈辩论的问题，一些程序员坚持认为应该使用样板，但另一些程序员则从不用样板编程。许多Visual C++程序员无论以前是否对样板感兴趣都编制出了许多功能强大的PC机程序，而且也并没有感到需要使用

别的支持样板的C++编译器。之所以出现这种情况，也许是Visual C++提供的丰富的实用工具使程序员不需要使用样板，也许是样板的作用一向被高估了。

注意：多年来，AT&T公司发展了C++的非面向对象方面的性能，使它优越于C。许多增强的功能已被ANSI委员会加进他们的C标准以发展C。例如，函数原型的概念并不是C提出的，但现在它作为C的组成部分已经有好几年了。函数原型概念最早是由C++提出的，因为它非常有利于程序设计，所以ANSI把它加入到C中去了。

1.1.2 非OOP编程方法

当程序员不使用OOP方法时，编程的核心概念是过程。一个面向过程的程序完成一个应用目标的手段是顺序地执行某种指令序列。这种程序在带参量的指令引导下达到某个状态，也就是说是指令决定了程序运行所有可能的方向。C、Pascal、QBasic和COBOL都是这种面向过程的程序设计方法的例子。

今天的计算机发展已不能满足于面向过程的程序设计方法。比如，Windows这样的图形用户接口(GUI)可以使用面向过程的程序设计方法，但它只有在使用面对象的程序设计方法时才更充分地体现出它的优越性能。

图标、对话框、选择框等GUI的屏幕窗口当诸如鼠标或键盘被按下等等的事件发生时被激活。当使用面向过程的程序设计方法编制GUI的应用程序时，用户必须在程序结构中利用数量庞大的C开关量来不停地测试这些事件是否发生。但前面我们已经提到过(而且在本书的其余部分将做更为详尽的解释)，面向对象的程序设计无须这样做，而只需激活数据就可以了。在程序中所有显示在屏幕上的GUI元素都可被激活。

现实世界的程序设计

当用户阅读完本书，将会发现OOP思想比起非OOP思想更像是现实世界的一面镜子。OOP的专家认为OOP从程序中"剔除抽象层"。

剔除抽象层的意思简单来说就是使程序能够更像它所映像的应用目标那样去工作。程序越接近映像的应用目标，编程就越快，调试也就越简单。比如如果用户想用C++编制一段CD-ROM控制程序，以便在计算机的CD-ROM驱动器上放CD来欣赏所喜爱的音乐，用户必须为play(播放)，reward(倒带)，fast forward(快进)和eject(停止)等键定义相应的变量，当play键被按下，play变量将被激活并接管以下的工作来放出一段音乐。

在面向过程的程序设计方法中，每种变量都有它自己的特性。一个Sales变量必须是唯一的，它精确地代表一个数字；一个FirstName变量必须是一个串变量或字符数组，它代表一个人的姓。在OOP语言中，变量不仅有特性，还有行为，行为是指对数据的操作。当用户给出一个变量的行为(通过编写代码)，就使这个变量具有了对自身进行初始化、显示、运算等等的功能。

当用户的变量像拥有特性一样拥有了行为，编程工作就会变得非常简单，用户再也不必在使用这些数据时都编写代码对它们进行操作。通过激活一个变量，C++对用户自定义数据的操作远比C更为清晰。当用户使用关键字struct定义自己的数据类型时，有许多不同的方法向C++中添加这种新的类型。在C中，当用户声明了一个结构后，必须重复使用关键字struct来定义这个结构类型的变量，而在C++中，用关键字struct声明了一个结构后