计算机C/C++语言系列丛书

PROGRAMMING WITH MICROSOFT VISUAL C++ 1.5

SCOTT PALMER

希望

学苑出版社

Programming With Microsoft Visual C++ 1.5

Visual C++ 1.5 程序设计

SCOTT PALMER 著
黄 光 程思乾 译
李 津 万 博 审校

学苑出版社

(京)新登字 151 号

内 容 简 介

本书是针对利用 Visual C++进行 C 和 C++程序设计的人编写的，全面系统地介绍了 Visual C++。首先介绍 Visual C++集成环境(Visual Workbench)，说明集成编辑器和调试器的用法；然后介绍 C 和 C++语言的基础知识，包括操作符和表达式、说明、控制语句、函数、指针、其他高级数据结构和文件输入/输出；接着介绍了类型兼容、宏、条件编译指令、预编译处理，讨论了面向对像的基本编程方法。

欲购本书的用户，可与北京海淀 8721 信箱书刊部联系，电话 2562329，邮编 100080。

计算机 C/C++语言系列丛书

Visual C++ 1.5 **程序设计**

著　　者：SCOTT PALMER
译　　者：黄　光　　程思乾
审　　校：李　津　　万　博
责任编辑：甄国宪
排　　版：万博图书创作社
出版发行：学苑出版社　**邮政编码**：100036
社　　址：北京市海淀区万寿路西街 11 号
印　　刷：**施园印刷厂**
开　　本：787×1092　1/16
印　　张：21.875　　**字数**：495 千字
印　　数：1～5000 册
版　　次：1994 年 9 月北京第 1 版第 1 次
ISBN7-5077-0875-6/TP・24
本册定价：44 元

学苑版图书印、装错误可随时退换

前　言

您买这本书，也许是由于本书论述用最新的 Visual C++ 1.5 进行程序设计。您是对的。但只猜对一半：本书还是一个冒险家的乐园，充满了惊险和乐趣。我们大多数程序员不是为了金钱而编程，而是为了乐趣。本书介绍如何用 C++进行程序设计，如何编写 Windows 程序，如何使用 Visual C++的工具，如何发现乐趣。

C++和 Microsoft Windows 就像是一个程序的游乐场：充满新的思想、功能强大的工具和技巧。有了这本书，将学习到面向对象程序设计的最新概念和技术，如如何创建类数据类型，懂得如何管理类和数据，如何从预定义数据类型中继承函数和数据，如何使用 C++流 I/O 性能，以及如何管理更复杂的工程。最后，还介绍如何使用 Windows Programming Interface (API)，既可通过直接使用，也可通过功能强大的 Microsoft Foundation Class 库 (MFC)来提高开发效率。

本书的内容

本书主要讨论三个主要的中心：首先，介绍 C++和面向对象程序设计的基本技术；其次介绍 Microsoft Windows 程序设计的基础思想和技术；最后，讨论如何 Microsoft Visual C++的主要功能。由于 C++是 C 程序设计语言的超集，有必要让读者了解 C 从而更有效地理解 C++。因此，C 并没有深入讨论，只在第一章中介绍了 C 和 C++之间的差别。另外，本书中说明的 Microsoft Windows 程序设计技术适用于 C、C++和其他能编写 Windows 程序的程序设计语言。

本书的目标

本书适合任何一个有一些编程经验的人想学习 C++和面向对象程序设计，特别是学习用标准和专业版的 Microsoft Visual C++编写 Microsoft Windows 3.X 程序的人。也可以被用其他能编写 Windows 程序的程序设计语言人作为一般 C++和 Windows 程序设计的指导性资料。

本书的组织

本书分成三个部分，第一部分由第一章和第二章组成，介绍 C++的概况。第二部分由

第三至十五章组成，循序渐进说明了习面向对象程序设计和C++程序设计语言主要特性。这部分的内容适用于其他目标环境下任何标准C++编译器的程序设计，目标环境既可以是基于文本的，也可以是基于图形界面的的PC、Macintosh和主机。

第三部分由第十六至第十八章组成，阐述用Visual C++循序渐进进行Windows程序设计技术。内容包括Windows程序设计的基本特点、Windows API和Visual C++中的Microsoft Foundation Class库。

必备条件

由于本书涉及一般的C++程序设计以及Windows程序设计，可以作为任何符合C++标准的语言的参考书，包括Visual C++ 1.0和1.5，Microsoft C/C++ 7，Borland C++，Watcom C++和Zortech C++。由于C程序设计没有深入讨论，因此，需要一个C++编译器，但一些概念和技术和用Quick C for Windows和其他C工具进行Windows程序设计相同。

为了更好地使用本书，需要至少有一个80386以上的PC，安装有Visual C++、Microsoft Windows 3.1和MS—DOS 5.0或更高版本。其中的许多程序设计技术和示例还可适用于Microsoft Windows NT、Windows for Workgroups和老的Windows 3.0。

除此之外，还需要一个脑袋，一点空余时间和冒险精神。

致　谢

虽然我的名字出现在封面上，但这本书是集体努力的结果。对 Random House 的编辑 Kim Fryer 表示诚挚的感谢，没有他的信赖、鼓励和少有的耐心，这本书不可能与读者见面。Random House 的主编 Mia McCroskey 也给予了很大的帮助。

Electric Ink 的 Karl Barndt 和 Lydia Levins 以极度认真的态度处理了文书编辑和生产。

感谢我的家人。

最后要感谢的人是帮助我出版这本书的您，读者。如果本书对您有所帮助，是对我的最大慰藉。

目　录

第一章 序 论

在这一章中，我们先准备对C++作一个综述，即讨论它的演化过程、对流线技术的采纳以及对新的面向对象编程规范的支持。之后我们比较一下C++和C语言的异同点，看一看怎样才能用Visual C++来达到最好的编程效果。

作者先提醒一点，本章的目的只是给出一个概念性的综述。虽然文中不时地会出现一些短的C和C++程序代码，详细讨论编程的内容不属本章范畴，而是留到后续各章中。如果读者不喜欢“概念性”的讨论(正如绝大多数的程序员一样)，那就可能总想着跳过本章内容而直接进入第二章，在那里将介绍Visual C++引入的一些专用工具。不过，作者还是请各位至少把本章的内容大致浏览一下，因为书中的内容安排是前后连贯的，了解本章就有助于更好地理解后面的内容。有些例子中的代码是完整的程序，读者如果有兴趣还可以试着进行调试运行。这些程序有意设计得比较简单，以便把要说明的观点说清楚。

1.1 C++历史的简单回顾

首先请读者注意：虽然C++是基于C语言的一门编程语言，但是学C++之前不必一定要先懂C语言。C语言确实包含了C++的许多特点，但其中的编程技巧有大量是在C++中不需要的，故学过C的读者又得放弃这些技巧。尽管如此，读者们还是能体会到C++编程技巧比C更为简单明了。正如C++的创立者Bjarne Stroustrup所回忆的：“当初提出C++主要是为了使(我自己和)我的朋友们可以不必采用汇编、C和许多现代的高级语言编程，它的主要的目的是使程序员更容易，而且心情更舒畅地编写出好程序来。”

C++对C作了大量的改进，其中的一些内容将在本章讨论到。但C++最重要的创新之处既是语法性的，又是概念性的：它定义了一种新的称为“类”的数据类型，从而对“面向对象的程序设计”提供了显式的支持。事实上，当C++在1980年刚提出时，它叫做“带有类的C”。“C++”这个名字是后来形成的，“++”是指C语言的递增操作符(++)，以表明C++与C的演变继承关系。

似乎在地球上住着的人都应知道，面向对象程序设计是近几年在程序设计领域最热门的话题。但它不仅仅是一种时尚，不会像信息资源管理和时下许多其他唬人的新名词这样的时尚很快就会为世人所遗忘。它也不是只通过学一下C++的“class”这个关键字的语法就能掌握的东西。它需要的是头脑中的概念革命，即在思考和设计程序的方法上来一次彻底的变革。

1.2 过程式程序设计

传统上用 BASIC、Pascal 和 C 等过程语言进行编程时，“处理过程”和“数据”这两者是有严格界限的。程序从用户那里接受一些数据，然后对这些数据执行一系列指令，如“把 X 加到 Y 中，把和拷到另一个地址，再乘以 0.05，最后打印结果”。最早的程序其复杂性莫过于此。

1.2.1 结构化程序设计

随着程序变得越来越庞大和复杂，这种简单的逐条排列程序指令的顺序方法就显得太庞杂和笨拙。对于一个 50 行的程序，要理解其中的思路并不太难，但要是碰到一个 50000 行的程序，我们就需要找出一种方法来把程序分解为几个功能块，以使之更容易理解。

结构化程序设计具有这一优点，它把程序分解为一些子例程，这些例程在 C 和 C++中称为函数，在 Pascal 中称为函数或过程。这样分解之后，数据就可以传给各个子例程，子例程当然要比整个程序小得多也简单得多。处理数据的每一步都可以观察清楚，而子例程的思路也容易理解了，因为它的长度和难易程度能为我们所驾驭。

1.2.2 被动式的数据

结构化程序设计尽管十分重要且有用，仍然只是一个进步，而不是一次概念上的革命。和非结构化程序设计一样，在结构化程序设计中，“处理”就是做某件事，“数据”就是处理的对象。程序中的数据是一个被动的旁观者，就像躺在手术车上的病人一样。病人被推着从一个科室换到另一个科室，又是拍 X 光片，又是听诊，又是打针——我们可以推想他在整个过程中的被动作用多少会使他心里感到有些难受——数据也是如此，它只是等在那儿看着程序对它进行各种处理。即使它想发挥更积极的作用，它也不知道该怎么做：它毕竟只是一堆数而已。

被动式的数据还有另外一层意思：它没有抵抗错误处理的内在保护功能。如果某个子例程经修改后对数据进行了不正确的处理，那么数据就会遭到破坏。正如一个病人吃错药之后疾病会加剧——医生称之为“庸医病”——一样，被动接受各种处理的数据其安全性也低于应有的标准。

在结构化程序设计中，解决这一问题有一个方法，就是使数据只属于用到它的那些子例程，如图 1.1 所示。这样，任何其他的子例程就无法访问这些数据了。但是，这种方法经常会做过头：有些数据项需要在程序中多处被引用。例如，一个职员资源管理系统可能需要包括多个子例程，用于分别计算员工薪水、统计病假天数、记录值得表彰的事绩等等。如果员工的纪录只能被某个子例程所访问，那就排除了需要这些数据的所有其他子例程访问这些数据的可能性；但如果这些数据能为所有子例程所用（即成为全局数据），则可能将从意想不到的地方招致对数据的破坏。