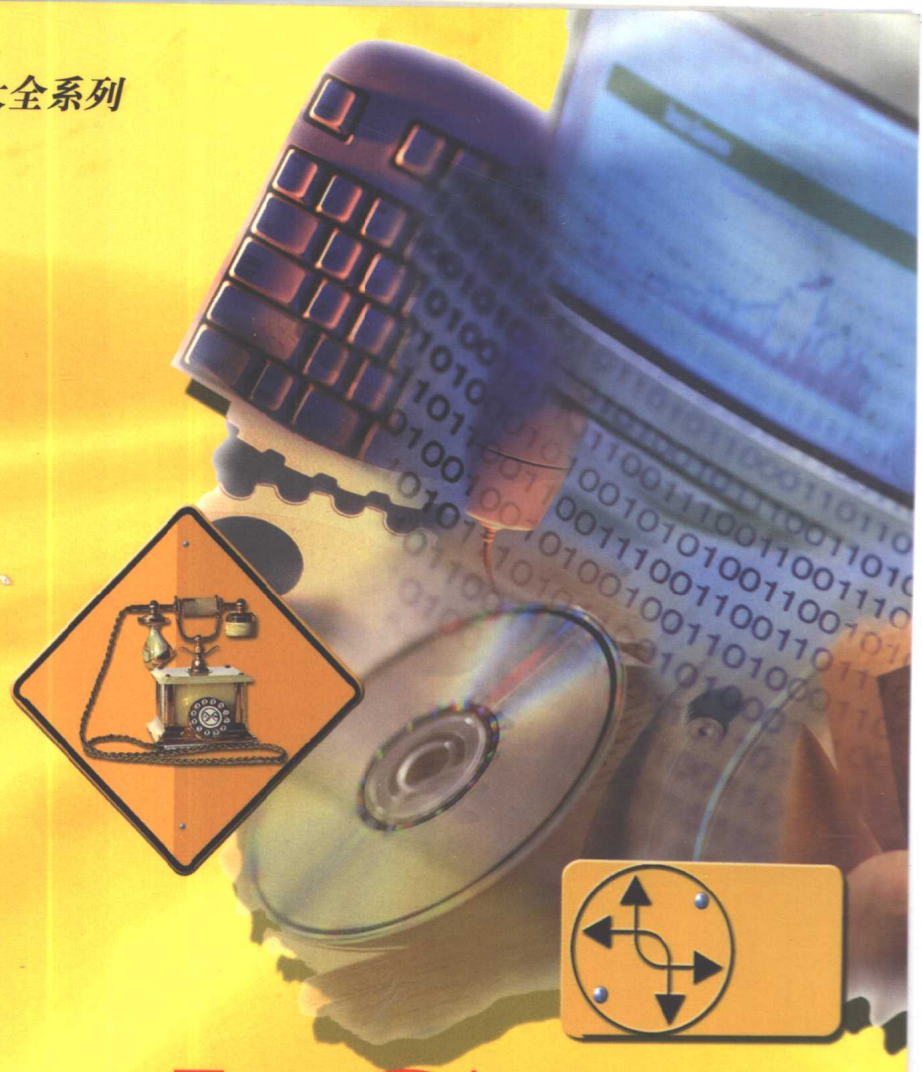
万水计算机技术实用大全系列

- 深入理解C++语义，分析C++对象的内存布局
- 深入了解Win32操作系统的细节信息
- 深入分析基于SDK的Windows程序文件中的每一句代码
- 深入理解MFC的建构思想和基本技术，透彻理解MFC程序中的每一句代码



Visual C++ 编程深入引导

伍红兵 编著

中国水利水电出版社
www.waterpub.com.cn

万水计算机技术实用大全系列

Visual C++编程深入引导

伍红兵　编著

中国水利水电出版社

内 容 提 要

本书分四部分：深入理解 C++、深入理解 Win32 操作系统、深入理解 SDK 程序设计、深入理解 MFC。本书全面地介绍了精通 Visual C++程序设计所应当了解的知识——以 C++最新国际标准为蓝本，通过对 C++语法机制汇编级的分析，深入介绍了 C++的对象模型和语法机制的实现细节，内容有相当的深度，角度独特；以 Windows 98/Windows 2000 为蓝本，介绍了操作系统的内部机制；结合 Visual C++自动生成的 SDK 程序，全面分析了基本 Windows 程序的每一句代码以及这些代码与操作系统之间的互动关系； MFC 实现技术的介绍，引导读者深入了解 MFC。总体上，该书对使用 Visual C++编程的读者能够起到全面而深入的引导，适合中、高级软件开发人员及广大编程爱好者。

图书在版编目（CIP）数据

Visual C++编程深入引导/伍红兵编著. —北京：中国水利水电出版社，2002
（万水计算机技术实用大全系列）
ISBN 7-5084-1007-6

Ⅰ. V… Ⅱ. 伍… Ⅲ. C 语言—程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2002）第 013333 号

书　　名	Visual C++编程深入引导
作　　者	伍红兵　编著
出版、发行	中国水利水电出版社（北京市三里河路 6 号　100044） 网址：www.waterpub.com.cn E-mail：mchannel@public3.bta.net.cn（万水） sale@waterpub.com.cn 电话：（010）68359286（万水）、63202266（总机）、68331835（发行部）
经　　售	全国各地新华书店
排　　版	北京万水电子信息有限公司
印　　刷	北京市天竺颖华印刷厂
规　　格	787×1092 毫米　16 开本　46 印张　1016 千字
版　　次	2002 年 3 月第一版　2002 年 3 月北京第一次印刷
印　　数	0001—5000 册
定　　价	68.00 元

凡购买我社图书，如有缺页、倒页、脱页的，本社发行部负责调换

前　言

本书为精通 Visual C++编程提供了一个引导。精通 Visual C++编程应该遵循这样一个学习过程：深入理解 C++——深入理解 Win 32 操作系统——深入理解 SDK——深入理解 MFC。对于 C++语言，本书通过汇编级的分析，深入揭示了 C++的方方面面，许多内容实际上是理解 C++所必不可少的。对于 Win 32 操作系统，本书通过讨论 Win 32 家族中最具代表性的两个平台——Windows 95 和 Windows 2000 的体系结构，及其内部重要的机制，深入介绍了 Win 32 操作系统中对于 Windows 程序设计非常重要的信息。对于 SDK，通过深入分析基于 SDK 的 Windows 应用程序，从而深入揭示了 Windows 应用程序的组成部分。及各部分的运作机制。对于 MFC，通过对关键宏以及类库源代码的分析，揭示了 MFC 许多细节信息。

本书并不是一本内容简单的入门性的读物，许多内容是对相关知识的深入讨论。

全书分为 4 部分：

第一部分深入介绍面向对象编程思想和 C++语言的语法机制的细节。通过分析 C++源程序生成的汇编代码，深入地揭示了 C++的语法机制细节。除了介绍 C++的语法机制本身，还介绍了这些语法机制产生的原因和目的，使读者能更为全面而深刻地领悟 C++。这一部分以 ANSI C++ 3.0 为蓝本，全面介绍了 C++的最新内容。

第二部分以 Windows 95 和 Windows 2000 为蓝本，深入介绍了操作系统的深层信息。从编程的角度看，只要是编写 Windows 应用程序，不管用什么语言编程，这些信息都应该深入了解。

第三部分全面而深入地介绍了一个用 SDK 编写的 Windows 应用程序，几乎对程序每一个文件中的每一条语句都作了详尽的分析（大多数初学者对 Visual C++的 AppWizard 生成的许多语句可能并不是很清楚）。Windows 程序的运行是程序本身与操作系统之间的互动过程，因此，本书不仅介绍了每一条语句的含义和作用，还深入介绍了语句背后操作系统所作的工作。

第四部分深入介绍了 MFC 的编程观念和内部机制。虽然 MFC 只是利用面向对象技术和 C++语言编写 Windows 应用程序，但是精通 C++语言、精通 Windows 操作系统，同时还精通 SDK 编程的人同样会对 MFC 编程感到困惑。原因有两点，一是 MFC 不仅仅简单地封装了 Win32 API，它还提供了一个规范的应用程序框架，而 MFC 提供的应用程序框架有着比较复杂的思想背景。其二，MFC 的众多机制是通过复杂的宏来实现的，宏的大量使用不仅隐藏了实现细节，也使程序代码显得很不规范。上述两点构成了迈进 MFC 的两道门槛。因此，本书最后一部分深入讨论了 MFC 应用程序框架和 MFC 关键宏，使得用户能对 MFC 有一个较为透彻的了解，帮助读者越过这两道门槛。

本书的读者定位是：中、高级软件开发人员。

编者

2002 年 1 月

责任编辑：李伟

本书为精通Visual C++编程提供深入引导，内容翔实

本书以深入理解C++——Win32操作系统——SDK——MFC为主线，全面地介绍了精通Visual C++程序设计所应当了解的知识。本书以C++最新国际标准为蓝本，通过对C++语法机制汇编级的分析，深入介绍了其对象模型和语法机制的实现细节，内容深广，角度独特；以Windows 98/Windows 2000为蓝本，介绍了操作系统的内部机制，讲述深入细致；结合Visual C++自动生成的SDK程序，分析了基本Windows程序的每一句代码以及这些代码与操作系统的互动关系；MFC实现技术的介绍，引导读者深入了解MFC。

本书分为四大部分

- 深入理解C++　以ISO/IEC 14882标准为蓝本，介绍C++面向对象的编程思想、语法机制和对象模型
- 深入理解Win32操作系统　以Windows 98/Windows 2000为蓝本，介绍Win32操作系统平台的内部机制
- 深入理解SDK　结合Visual C++自动生成的SDK程序，不仅分析了每一条语句的含义和作用，还深入介绍了语句背后操作系统所做的工作
- 深入理解MFC　通过对关键宏以及类库源代码的分析，揭示MFC的细节信息

北京万水电子信息有限公司
Beijing Multi-Channel Electronic Information Co., Ltd.
地址：北京市西直门外榆树馆一巷永康商务写字楼
邮编：100044
电话：(010)6835.9286, 6835.9167
传真：(010)6835.9284
E-mail: mchannel@public3.bta.net.cn

ISBN 7-5084-1007-6/ TP•389
定价：68.00元

ISBN 7-5084-1007-6
9 787508 410074 >

目　　录

第二部分 深入理解 Windows 操作系统

第三部分 深入理解 SDK 程序设计

第四部分 深入理解 MFC

第一部分　深入理解 C++

C++是 MFC（Microsoft Foundation Class Library）的基础。MFC 是利用 C++语言为开发 Windows 应用程序而构造的一个基本 C++类库，因此深入理解 C++是深入掌握 MFC 的前提。

千万不要因为“C++是 C 的扩展”这一简单的论断，而误以为 C++与其他高级语言一样简单。事实上，C++是非常难学的。C++难学不仅在于其广博的语法，语法背后的语意，以及语意背后的深层思维和深层思维背后的对象模型；C++难学，还在于其主要特性都是为支持大程序设计而设置的，初学者一般缺乏大程序设计的经验，因此在学习过程中很难领会其精妙之处。

对于 C++，既要求有效率，又要求有弹性；既要能复用以前的成果，又要面向未来，还要兼顾过去——世界上没有免费的午餐，要实现这些要求，C++有着非常复杂的机制。但仍有众多追随者知难而进，如果真能深入其中，相信你的收获一定大于你的付出。

在这一部分中，首先介绍了面向对象编程思想，然后深入介绍了 C++的语法机制。对于 C++中的关键内容，则通过分析 C++源程序生成的汇编代码，深入揭示了 C++的语法机制的实现细节。在介绍 C++语法机制的同时，还介绍了 C++设计这些语法机制的原因和目的，使读者对 C++能有更深入的把握。

这一部分虽然没有对 C++作完整的介绍，但还是涵盖了 C++的绝大部分内容。

此外，本部分以最新的 ANSI C++ 3.0 为蓝本，涉及了 C++的许多最新内容。

第 1 章　面向对象程序设计

一个好的程序设计语言应具有编程的高效性、运行效率的高效性、安全性、可维护性。对于大型的软件，这些要求尤其显得重要。面向对象技术的产生和发展来源于软件设计实践的强烈需求，而不是源于某些多事的学者为显示其高深莫测的奇思妙想。了解面向对象程序设计思想及其起源，了解软件设计实践中的要求，了解程序设计语言的发展，对于深入理解 C++的诸多语法现象是十分有益的。许多 C++的初学者都认为 C++语言过于复杂和不必要，而且，作什么事都要先仔细分析需求，小心地构造出多种多样的类，反而把问题搞复杂了。之所以有这种心态，往往是由于对面向对象程序设计思想和其在大程序设计中表现出的优良特性缺乏了解。

本章从软件设计需求和程序设计语言的发展过程着手，介绍了结构化程序设计思想和面向对象程序设计思想，以及 C++的版本情况。

1.1 结构化程序设计

对于一个大型软件，往往不是一个人所能胜任的，而是需要有一个研制组分工负责完成。如果编程人员不掌握一套比较规范化、工程化的程序设计方法和技术，以及养成良好的程序设计风格和习惯，很可能花费了大量的人力、物力和时间，得到的却是一个潜伏着各种危机因素的不可靠的程序。而要从一个大规模的程序中发现潜在的错误，是十分困难的，这样的程序将会给调试、维护以及正常使用带来很大的不便。

1969 年，Dijkstra 首先提出结构化程序设计的概念，它强调了从程序结构和风格上来研究程序设计。进入七十年代以后，结构化程序设计的方法日益完善。与此同时，在数据类型抽象、程序的推导和综合，程序变换和程序自动化等方面，也都取得了重大的进展，从而形成了一套完整的程序设计方法。

结构化程序设计基于下面两个前提：

1．必须以一种能理解和易维护的方式设计和编写程序。

2．通过把一个问题细化成为易于处理的元素，以解决复杂的编程问题。

结构性首先表现在表达式的计算中。高级语言内允许使用普通代数形式来书写表达式，使得一组动作汇集在一个表达式中加以体现。

另一个需要结构化的是数据。在 C 语言中，数据有非构造类型数据和构造类型数据之分。所谓构造类型数据，就是结构化的数据。一个构造类型数据的每一个成分都是一个变量，这个变量或属于非构造类型，或仍属于构造类型，从而可以形成诸如数组的数组、结构的数组、数组的结构等较为复杂的数据结构，以适应各种不同应用领域的要求。

再次，由于一个程序是由一系列规定动作的语句编码组成，故需要改进语句编码本身，使其具有结构化的特点。结构化的语句编码体现了能用有限数量的基本结构来表示程序控制逻辑的技术。在 C 语言中，每一个程序都能用如下四种基本结构来表达：顺序结构、条件结构、循环结构、函数调用。顺序结构最为简单，它表示无条件地、逐个地执行语句。条件结构通过对数的判断来选择程序流向。循环结构用于重复执行语句序列。函数调用则可以使我们仅用一条语句来代替一组语句。程序内使用函数调用，不仅可以使程序编码相对集中，更重要的是它提供了构造一个清晰的、具有层次结构的程序的有力手段。

使用上述四种基本结构，使程序增加了可读性，改善了可靠性，便于维护和修改程序。

解决了以上关于数据的结构化和语句编码的结构化，接下来的问题是用怎样的方法来设计和编写程序。结构化程序设计导引出两种基本的方法：自顶向下的方法和自底向上的方法。自顶向下的方法首先定义程序的整体结构，然后具体实现其中的每一个细节；自底向上的方法则是首先实现各个细节，然后将他们组合在一起，以实现总的任务。一般采用较多的是自顶向下的方法，它体现了逐步求精的过程。

若将一个欲解决的问题视作一项任务，那么首先应将这项任务映射成一个相应的体系结构。该结构中包含了若干个定义明确的、相对独立的子任务。当每一个子任务均获得

解决，整个任务亦得到解决。对于这些子任务来说，各自涉及的范围已较前缩小，功能亦趋具体，所需语句编码也比实现整个任务要少。当然，其中还可能有较为复杂的情况。我们可将这样的子任务继续分解为范围更小、功能更具体、所需语句编码更少的若干子任务。如此继续，直到每一个细分了的子任务均可用简单明确的语句编码满意地实现时为止。整个过程体现了逐步求精的特点。当然这里面还会涉及到在每一层次选用怎样的数据结构，上、下层的数据相互之间的联系，以及对各个子任务所选择的具体算法等等。因此可以说，自顶向下的逐步求精过程，就是程序方案及语句编码由粗略变为精细的演变过程。由于应用了结构化程序设计的方法，所编写的程序不仅结构良好、清晰易读，而且易于验证程序的正确性。

结构化程序设计语言前的各种语言早已销声匿迹，目前大多数人所熟知的包括 C 语言、Pascal 语言、Fortran-77 等都是结构化的程序设计语言。如果你编写过 C 语言小程序，你一定熟悉 C 语言程序设计的一般法则。对于一个具体问题，首先要作的就是分析解决这个问题需要几个步骤，每个步骤需要几个独立的功能，每个功能用一个函数来实现，然后在 main() 函数中调用这些功能函数，从而完成程序设计。虽然在实现每一个功能时要仔细地选择数据结构以及算法，但在整个程序设计的过程中主要关注的还是功能（即函数），你要小心地划分功能，以尽可能使不同的功能不相互重叠以节省代码；你还必须仔细考虑功能的粒度，即功能应该分得多大，使得划分的每一个功能既相对独立，又不要过于包罗万象而使代码过于庞大，难以实现。对于一些复杂的功能，你还得仔细考虑如何对这一功能进一步地划分成更小的功能以便于实现……总之，功能的划分及实现是你在使用结构化程序设计语言进行程序设计时所要作的最主要工作。

1.2　面向对象程序设计

虽然结构化程序设计思想很有效，甚至曾一度解决了软件危机，但人们对软件规模的要求似乎永无止境，人们需要编制更大的软件以解决生产实践中的问题。在大程序（一般代码在 50,000 行以上的程序）设计中，用结构化程序设计方法所设计的代码变得越来越不可靠和难以维护，代码的规模也变得难以控制，而且代码中细小的错误也非常难以查找，有时这些困难会变得难以逾越，导致投资失败。这里以结构化程序设计语言之一的 C 语言为例，列举几个结构化程序设计在大程序设计中的缺点。

1. 代码缺乏复用性

C 语言本身几乎没有支持代码复用的语言结构，因此，一个程序员精心设计的代码很难为其他程序复用。结构化程序设计从系统的功能入手，按照工程的标准和严格的规范将系统分解为若干功能模块，系统是实现模块功能的函数的集合，当业务需求或软硬件技术的发展发生变化时，按照功能划分设计的系统模块必然发生变化，有时甚至会造成程序员费尽心血而编制的代码不得不完全重写，因为代码功能（函数）与需求是密切相关的。随之而来的弊端是模块可重用性不高，程序的维护变得非常困难。