ET PUBLISHING
今日電子

美国IDG“宝典”丛书

Standard C++ Bible

丛书累计印数32万册

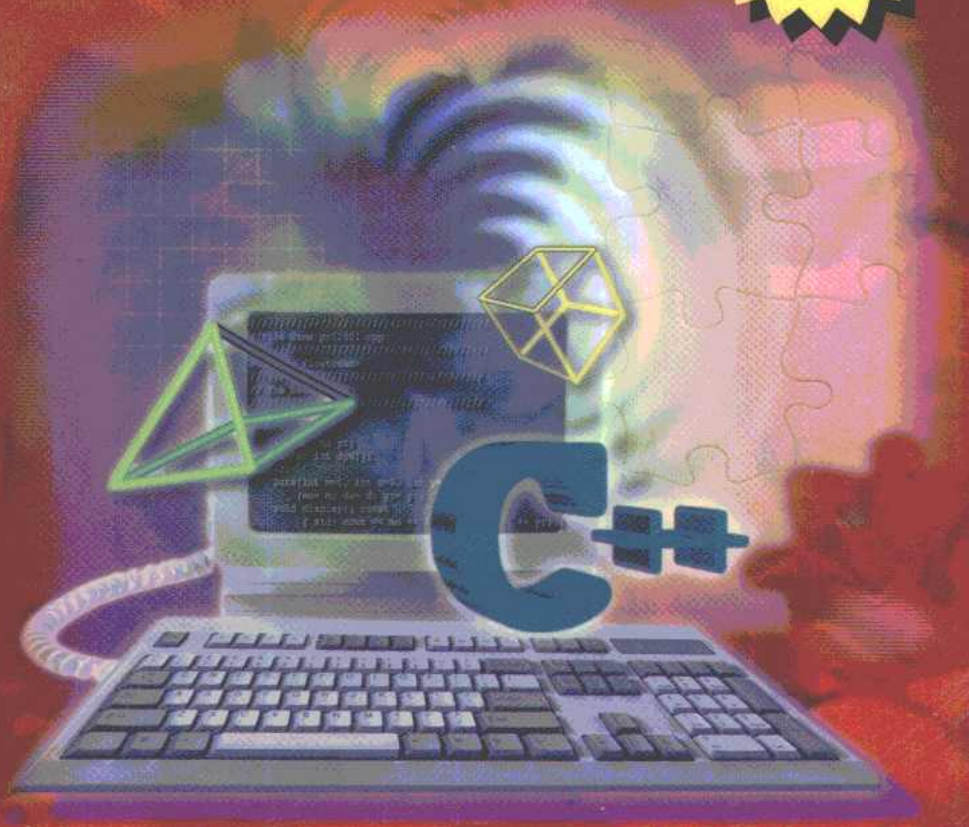
100%

内容丰富、权威

精通C++编程

使用类并编写第一流的面向对象的程序

制作库和新的ANSI/ISO标准



[美] Al Stevens, Clayton Walnum 著

林丽闽 别红霞 等译
林丽闽 审校

标准C++宝典

本书附带的CD-ROM光盘包含所有范例的程序源代码和一个名为Quincy的集成开发环境

IDG BOOKS WORLDWIDE
PHEI
電子工業出版社
Publishing House Of Electronics Industry

美国 IDG“宝典”丛书

标准 C++ 宝典

Standard C++ Bible

[美]Al Stevens　Clayton Walnum 著

林丽闽　别红霞　等译

林丽闽　审校

電子工業出版社

Publishing House of Electronics Industry

北京·BEIJING

内容简介

标准 C++ 是 1998 年正式推出的关于 C++ 的国际性标准版本，本书以此标准为基础，对 C++ 进行了全面地介绍。

本书从最基本的内容讲起，对 C++ 强大的功能作了详细介绍，并运用 C++ 的最新功能，详细论述了面向对象编程的思想。本书针对那些比较高深难懂的主题也做了深入浅出的介绍，比如模板（包括标准模板库，即 STL）、名字空间以及运行时类型信息（RTTI）等，这些功能对开发大型复杂系统的程序设计人员非常重要。因此，无论是新手还是有经验的程序员，都可以从本书中找到丰富的信息。使用本书，读者不仅可以学到技术，还可以得到最好的练习，使自己的程序设计水准达到专业水平。

本书是学习和使用 C/C++ 的优秀工具书，其内容综合、完整，叙述清晰、易懂，实例典型、丰富，适合于学习 C 和 C++ 语言的读者使用。

图书在版编目（CIP）数据

标准 C++ 宝典/（美）史蒂芬（Stevens, A.）著；林丽闽译 .-北京：电子工业出版社，2001.2
（美国 IDG“宝典”丛书）
书名原文：Standard C++ Bible
ISBN 7-5053-6503-7
Ⅰ.标… Ⅱ.①史…②林… Ⅲ.C 语言－程序设计 Ⅳ.TP312
中国版本图书馆 CIP 数据核字（2001） 第 05365 号

丛 书 名：美国 IDG“宝典”丛书
书　　名：**标准 C++ 宝典**
著　　者：[美]Al Stevens, Clayton Walnum
译　　者：林丽闽　别红霞　等
审 校 者：林丽闽
责任编辑：梁卫红
印 刷 者：北京朝阳隆华印刷厂
出版发行：电子工业出版社　URL: http: //www.phei.com.cn
北京市海淀区万寿路 173 信箱　邮编 100036
经　　销：各地新华书店
开　　本：787×1092　1/16　印张：49.25　字数：1182 千字
版　　次：2001 年 2 月第 1 版　2001 年 2 月第 1 次印刷
书　　号：ISBN 7-5053-6503-7 / TP·3572　定　价：88.00 元（含光盘一张）
著作权合同登记号：图字：01-2000-1149
凡购买电子工业出版社的图书，如有缺页、倒页、脱页者，请向购买书店调换。若书店售缺，请与本社发行部联系调换。联系电话：68159356　68279077

前　言

* *

C++已成为了全世界专业编程人员的首选语言。C++已经有了一个公认的标准，并且许多编译器支持该标准中的绝大多数特性，所以应该抓紧时间深入学习这门内容丰富功能强大的语言，并且用它进行编程。本书对 C++语言的定义和介绍遵循 ANSI/ISO 国际标准。

如果不是 C 程序员

本书自成体系，是 C++语言的完整教程，不要求读者预先掌握 C 语言。在过去的几年中，所出版的大多数关于 C++的书，都假设读者已经懂 C 语言。到现在为止，大多数 C 程序员已经学习过 C++。因此本书的主要读者是那些对 C 和 C++都不懂，但是希望学习 C++的程序员。

按照传统的观念，学习 C++之前，应该学 C。这种观念现在已经过时了。学习了 C 风格的程序设计，那么未免会带有一些不必要的不合理的偏见。如果还没有学习 C，那么就应该直接学习 C++，这是目前流行的比较合理的观念。因此我决定把本书作为 C++的一个完整的学习教程，主要对象是那些懂得程序设计的人，但是不要求一定掌握某个具体的编程语言。当然 C 程序员也可以使用本书学习 C++。

已经是 C 程序员

对 C 程序员来说，从第 2 章到第 8 章的大部分内容，都是关于 C 语言的内容。C 程序员也许很想跳过这些章节，但是我建议尽量不要这样做。比如在第 2 章中，对 C++控制台输入/输出 iostream 对象进行了介绍，这些对象是 C++程序设计中使用的，而在标准 C 函数中，则使用 getchar()、putchar()、gets()、puts()、printf()和 scanf()等一组函数。在本书这几个章节的许多地方，都可以看到 C++语言与 C 语言的细微差别。例如，在第 4 章中讲述了 goto 语句，它与 C 中的 goto 语句就不相同；再比如，C++中的 Void 指针如果不经过强制类型转换，就不能够赋值给一个指针类型变量；再比如，C++程序中 main()主函数不能够被递归调用；对 const 对象必须进行初始化；变量不能被隐含声明为 int 类型；enumerator 是一种数据类型，不是整型；等等。在这里对这些差别不再一一赘述。本书中 C++是作为一个完整的新的主题被介绍的，因此不要完全依赖于已经掌握的 C 的有关知识，而忽略从第 1 章到第 8 章所讲述的内容。

例子程序

从第 1 章开始，本书在讲述 C++的过程中提供了一系列例子程序。每一个程序都包含了 C++源代码，可以进行编译和执行。如果希望从这个教程中获得最大的收益，那么应该一边学习一边把这些例子程序编译运行。

本书的例子程序按照内容顺序编排，先从比较简单的概念开始，循序渐进，一直到比较复杂的内容。后面的程序是建立在前面程序所介绍知识的基础之上的，因此应该按照例子程序出现的先后顺序阅读。

例子程序比较小，它们不是那种写得非常完整，可以直接在实际中应用的程序，这不是设计这些程序的目的。每一个程序对 C++的某个特定的功能作了例证。这些例子是可以进行独立编译和链接的完整的程序。

编译器

书后所附的 CD-ROM 中包含了 Quincy 99，它是一个基于 Windows(Windows 95 或者更高版本)的 C++编译系统，可以用来编译和运行书中的例子程序。Quincy 99 可引导读者使用本书中所有例子程序，程序的编号与书中一致，可以根据编号从列表中进行选择。

教授与学习的方法

程序设计是一门复杂的课程，需要通过循序渐进的方法进行学习。要学好每一课的内容，都依赖于对前面一课内容的掌握，有时还依赖于新的没有讲到的内容。比如 C++中的 iostream 类，这个类定义了一些对象，实现从控制台的输入和输出，要充分理解这个概念，需要掌握 C++中有关类和重载运算符的概念，这些都是 C++的高级内容。然而要学习这些高级内容，必须学习有关键盘和显示设备的知识。iostream 类集合实现了这些设备的功能。所以在使用系统定义的 cin 和 cout 对象时，并不要求知道其实现的细节，但是应该相信其操作的正确性，最终在后面的学习中把它弄懂。

由于本书所采用的这种特殊的循环方法，对已经学习过 C++的程序员来说，也许对本书的组织结构有疑问。对于有经验的 C++程序员来说，书中的一些例子程序在代码结构上也许看来不够传统，不够适当或者根本没有必要。这样做是有意的，目的就是构成本书的学习顺序。本书最终都会介绍预先使用而未讲述的内容。C++还具有许多高级特性，并且 C++允许出现奇怪的外来编码，本书并不包含这些内容，因为作为面向初学者的教程，这些内容不适合。在已经熟练掌握C++语言之后，可以从其他的书中学习有关这一部分的知识，实现对 C++的全面掌握。

要对本书抱有信心并且保持耐心，所有的知识都会讲述清楚。在本书中会出现这样的情况，就是有许多提前用到但是尚未讲述的知识。如果对这些知识不够明白，可以做一个记号，在后面学习到有关内容之后，再回到所标记之处，重新温习。

本书的组织结构

第 1 章介绍了 C++ 编程概念。指明了读者需要预先掌握的知识，以及使用 CD-ROM 中的程序需要了解的内容。在本章中，对 C++ 的历史做了简要介绍，对 C++ 编程语言进行了概述，还介绍了主函数 main()(它是 C++ 程序的入口点)。

第 2 章引导读者开始初步编写程序。其中对 main()函数做了更详尽的介绍，并且介绍了如何为代码加注释，如何把标准库的头文件包含在程序中，如何实现简单的控制台输出以观察程序运行结果。本章还讲述了 C++ 的表达式和赋值语句，以及怎么从键盘读入数据并在屏幕上显示。

第 3 章集中讨论了函数。包括如何在 C++ 程序中声明、定义和调用函数，如何向函数传递参数，如何从函数返回值，以及如何书写 C++ 函数的语句块结构。此外，本章还介绍了如何把 C++ 的程序模块与其他语言(比如 C 语言)编写的程序模块链接起来。

第 4 章介绍了程序流程控制。其中包括 if...else, do, while, for, goto, switch, break, continue 和 return 语句。

第 5 章讲述了 C++ 的数据类型，包括字符型、整型和浮点类型。讲述了如何使用 constants，介绍了变量有效范围的概念，以及如何用基本数据类型构造数组、结构和联合等聚合数据类型。

第 6 章讲述了如何在程序中使用指针，如何使用 typedef 操作符定义新的数据类型标识符。还介绍了有关递归的概念，以及 C++ 引用型变量。

第 7 章讨论了标准 C 库函数中的一部分内容。介绍了如何使用头文件实现在程序中调用标准函数，还介绍了字符串函数、内存分配函数、数学函数等。

第 8 章介绍了 C++ 中的预处理功能，包括定义宏、编写条件编译表达式来控制程序编译过程。

第 9 章对结构体进行扩展并引入类机制。介绍了内部数据类型和用户定义数据类型的数据抽象特性，还介绍了数据成员、成员函数和访问控制符。

第 10 章介绍了函数模板。使用函数模板可以创建通用函数，用于不同类型的数据。本章介绍了有关模板参数的内容，还介绍了关于模板的高级主题，比如模板重载。

第 11 章继续讨论类的相关内容，包括构造函数、析构函数、转换函数、赋值函数、类对象数组、类对象的内存分配。

第 12 章讨论了重载运算符，重载运算符给类对象赋予了行为，当其用于含有算术、关系等运算符的表达式中时，重载运算符使类对象可以模仿内部数据类型的行为。

第 13 章介绍了类的继承。使用类的继承，可以建立面向对象的类层次结构，层次结构由基类和派生类构成。本章还介绍了多态的概念，它是一种面向对象特性。

第 14 章讲述了多种继承，它是一种语言特性，允许派生类从多个基础类继承属性。

第 15 章讨论了类模板，它是 C++ 语言的特性，可以根据参数生成类。

第 16 章描述了面向对象编程的技术，包括数据抽象、封装、继承和多态性等。

第 17 章讨论了标准 C++ 的库。对 iostream 控制台输入/输出类进行了扩充介绍，还介绍了标准字符串类和复数类，以及 Standard Template Library(标准模板库)的容器类集合。

第 18 章介绍了 iostreams 的管理和格式化。

第 19 章详细介绍了 C++流的概念和磁盘文件操作。在读写文件时可以使用不同的流类。本章还介绍了如何处理文本文件和二进制文件。

第 20 章介绍了 Standard Template Library (STL)中定义的各种类型的类。包括序列、结合容器、算法和指示器。

第 21 章讲述了如何使用 STL 的序列类进行编程。序列类可以实现链表、队列、堆栈、向量等。

第 22 章介绍了如何使用结合容器类。结合容器类可以实现映射、集合和位集合。

第 23 章讨论了函数库中的通用算法。这些算法可以使程序处理定义在 STL 中的对象数据。处理过程包括计数、排序和分组。

第 24 章介绍了 STL 指示器,它是一种特殊类型的指针,可以使程序在容器中沿正序或者倒序移动。本章还介绍了如何用指示器访问存储在容器中元素的信息。

第 25 章详细介绍了异常处理,包括如何使程序以有序的方式处理异常。

第 26 章介绍了名字空间,包括如何定义名字空间、名字空间的范围、没有名称的名字空间、名字空间的别名。

第 27 章介绍了 C++的运行时类型信息(RTTI)和新风格类型转换,包括动态类型转换、静态类型转换和常类型转换。

第 28 章讲述了 C++中的 locale 类。包括如何使用 locale 类实现应用程序的国际化,如何对 locale 进行设置,以及如何根据 locale 要求的形式显示信息。

附录介绍了附于本书的 CD-ROM 中的内容。

C++ 语言

第 1 部分

◆ ◆ ◆ ◆ ◆ ◆

本部分包括：

◆ ◆ ◆ ◆ ◆ ◆

第 1 章

C++ 程序设计概述

本章包括：

C 语言和 C++ 语言的历史

C++ 语言概述

编写最简单的 C++ 程序

C 和 C++ 程序设计语言要比它们的许多使用者年龄更长。C 语言发展于 70 年代初，C++ 则最早出现在 1980 年。在这段时期内，C 和 C++ 语言从一个小工具(仅仅用于程序员们的研究实验)发展成为世界性的面向对象的语言，被几代程序员选择使用。现在 C++ 已经有了公认的标准。C 语言的标准在多年前就确定了，C++ 语言的标准也在最近提出并且得以确认。实际上，几乎所有种类的计算机和操作系统都有相应的 C++ 编译系统，并且当代大多数应用软件，不论是运行于小型机还是一般微机上的，都是用 C++ 语言编写的。

1.1 C 语言和 C++ 语言的简要历史

在 19 世纪 70 年代初，AT&T 的 Bell 实验室的程序员 Dennis Ritchie 第一次把 BCPL 语言改编成为被他命名的 C 语言。Ritchie 的目标是提供一种语言，使程序员能够用来对硬件进行操作，带有汇编语言的特点，又能够使用结构性的编程语句，具有高级语言的特性。他把整型和指针数据类型包括在 C 语言中，从而可以直接对 PDP-11 型小型机的硬件寄存器进行操作；他还充分利用了 PDP-11 的堆栈结构特点，提出了局部变量和递归函数的概念。

根据最初的设计，C 语言运行于 AT&T 的多用户、多任务的 Unix 操作系统上。C 语言先后在 PDP-7 和 PDP-11 上运行。后来 Ritchie 用 C 语言本身改写了 Unix C 的编译程序，Unix 操作系统的开发者 Ken Tompson 又成功地用 C 改写 Unix，从此他们开创了编程史上的新篇章。Unix 成为第一个不是用汇编语言编写的主要操作系统。为了把 C 编译系统移植到其他种类的计算机上，Unix 操作系统也被不断扩展，成为第一个能够在不同平台上运行的操作系统——一个真正的可移植的操作系统。实现 Unix 系统的移植变为对 C 编译系统的移植问题。操作系统中的一些驱动程序和接口函数，由于涉及到具体的硬件

特性，原本由汇编语言编写，现在只需用C语言进行改写，并重新编译就可以实现系统的移植。C语言也由此而成为一种非常出众的语言，它可以用来编写系统程序，而以往系统程序的编写语言只有汇编语言；它还可以用来编写可移植的程序，而这对于其他高级语言来说是可望而不可及的。

过去的许多年中，C语言主要运行于Unix操作系统之上。Unix操作系统始终是AT&T的内部操作系统，后来AT&T把Unix操作系统几乎免费提供给各个大学使用，使得整整一代学生都学会使用了Unix和C。这些学生离开学校以后进入到编程领域，他们不仅对Unix和C很有经验，而且对Unix和C还带有特殊的感情。这使得Unix和C的影响很快波及到美国的各个公司。C语言从此成为主流语言，在高级语言领域替代了Cobol语言的位置，在低级语言领域替代了汇编语言的位置。

尽管C发源于大型商业机构和学术界的研究实验室，但是当开发者们为第一台个人计算机提供了C编译系统之后，C语言得以广泛传播，为大多数人所接受。早期的微型计算机使用的是8080和Z80微处理器，运行在CP/M操作系统上。那时BASIC是主流编程语言，其次是汇编语言。后来供应商开始为微型计算机提供C编译系统，从而使得这种所谓的家庭计算机的使用者们转向C语言。1981年IBM PC及其后来的产品系列的推广，使得C语言的流传更加广泛。C逐渐成为大多数程序设计者最喜爱的语言，他们用C编写MS-DOS下运行的程序。PC机一度有成打的编译系统。事实上，大多数PC机上运行的程序和系统软件，都是采用C和C++编写的。对MS-DOS操作系统来说，系统软件和实用程序都是由C编写的。Windows 3.x的图形操作环境是用C编写的。Windows 98和Windows NT大部分也是用C编写的。

C的定义最早出现在1981年的《The C Programming Language》(Prentice Hall公司出版)一书中，这本书是由Brian Kernighan和Dennis Ritchie合著。这本书并不是对C语言的正式描述，而是叙述了Ritchie实现C的方式。这本书非常著名，人们以作者名字的首字母K&R来引用这本书。书中所描述的C多年来被称为K&R C。现在许多程序员把K&R C称为古典C。随着时间的推移，编写编译系统的开发者们不断根据用户的需要扩展C的特性。其中出色的地方已经逐渐被业界接受，并成为标准。

1983年，美国国家标准化协会(ANSI)专门成立了一个委员会，以业界的事实标准为基础，制定C语言的标准。国际标准组织ISO也加入到ANSI的这项工作中。ISO是面向国际的，所制定的语言标准适用于国际化的编程。这两个委员会于1990年共同发布了一个标准文件，其中定义的C称为标准C。K&R所著书的第二版描述了标准C。

同样是在AT&T的Bell实验室，C++程序设计语言由Bjarne Stroustrup设计并开发出来。他的工作大约开始于1980年，最初的目的是创建一种模拟语言，能够具有面向对象的程序设计特色。在当时，面向对象编程还是一个比较新的理念。Stroustrup博士并不是从头开始设计新语言，而是在C语言的基础上进行创建。

C语言已经在大多数系统结构中得以实现，并且支持可移植程序的开发，Stroustrup博士把C++语言系统开发成为一种转译程序，使之把C++源程序处理成C源程序。这使得转译后得到的C源程序能在任何支持C的计算机系统内进行编译。他把这个转译程序称为cfront。cfront程序后来得到不断的移植和扩充。在得到AT&T的许可情况下，语言系统的开发者们可以引用cfront源代码。1985年之后，C++开始在AT&T的外面流行。

经过多年的发展，C++已经有了若干版本。Stroustrup作为创始人，他的贡献在每个C++版本中都有体现。ANSI和ISO的联合委员会于1989年着手为C++制定标准，这是一个很艰

需要注意本书中的例子是C++程序，大多数C++编译器能编译它们。另外，除非有必要，我们使用C++用语而不是相应的C用语。

1.4　C++简述

下面描述了C++的基本概念，但是不要浅尝辄止。在后面各章中有许多例子，可以学到更多的内容。在学到相应内容时，可以回到这部分内容，对其中的含义重新仔细思考。

C++是具有面向对象扩充内容的过程型程序设计语言。这意味着在设计和编写程序时可以使用过程型的模块，同时可以定义对象并且实现对象的实例化(从类中创建实例)。C程序中过程型的模块称为函数。对象的声明称为类。

每个C++程序都由主函数main()开始执行，并在主函数返回时结束。下面一节"主函数main()"详细解释了主函数。main()主函数调用低一级的函数，低一级的函数调用更低一级的函数。函数从函数体中最前面的第一个语句开始执行，结束于函数体内部的最后一条语句，或者遇到return语句返回。每个函数在执行完毕后返回到调用该函数的位置。紧接着执行相邻的下一条语句。

阅读C++程序可以从上至下地进行，但是编写程序却不必采用这个顺序。函数和变量的声明语句，必须出现在调用或者使用它们的语句之前。也就是说，一个函数必须在它被调用之前进行声明，一个变量必须在使用它的语句出现之前进行声明。可以使用函数原型对函数进行声明，在函数原型中描述函数名、返回值类型和参数(关于参数的描述参看注释)，这些描述信息是编译器所需要的信息。在调用函数之前必须对函数进行声明，对函数的定义本身却是可以放在程序中的任何位置。在对函数进行调用时，返回值类型和参数类型必须与函数原型中所声明的相一致。

参数是在函数定义中声明的量。参数所代表的值是调用函数时传递给函数的值。

函数可以带有参数。主调函数把数值通过参数传递给被调函数，这些值在被调函数中用于计算。传递给函数的参数类型必须与函数声明中所给定的参数类型相匹配，或者相互兼容，或者可以进行类型转换。

有些函数有返回值，有些函数没有返回值。如果函数有返回值，那么主调函数可以把函数调用放在赋值语句的右侧，把返回值赋值给某个变量。也可以把函数调用放在另一个函数的参数列表中，让调用函数得到的返回值作为其参数。也可以作为局部数据变量的初始化数值，或者作为表达式中的一个元素。

每个函数包括一个或多个语句块。语句块可以嵌套(一个语句块嵌套在另一个语句块中)。每一个语句块有它自己的局部变量，变量的有效范围就在该语句块内。C程序可以在函数外部定义变量。所定义的变量在全局范围有效，在同一源代码文件中，每个函数中的每个语句都可以引用全局变量。

巨的任务。这个任务在 ANSI 内命名为 X3J16,在 ISO 内部命名为 WG21。1994 年 2 月该委员会出版了第一份非正式的草案,供公众浏览。之后又推出了几份修改草案。到 1998 年这项工作完成,正式推出了 C++的国际性标准。

如同以前的 C 一样,C++开始成为程序员们的首选语言。C++语言已经出现在各种运行环境中,其流行程度不亚于当初的 C 语言。

1.2　C 语言的未来

C++语言和面向对象编程如此流行,也许你会猜想 C 语言的未来会是什么样的。C++是否已经取代了 C 呢? 在大多数情况下,回答是肯定的,但是并不意味着 C 不再是一种可用的语言。好的编程语言很少会永远消失,因为几乎总有使它们具有优势的发展环境,而且有许多已经开发的应用软件需要维护。

C 比 C++更适合于解决某些编程问题。许多 C++的忠实拥护者或许不同意这种看法,但这种观点广泛地被其他人所接受。C++的软件开发环境一般很庞大,它们占用大量磁盘空间并要求大容量、高速度的计算机。这些要求随着各种新版本的语言和编译器的出现而增加。许多 C++编译器的目标就是图形用户界面,比如 Windows 98。除非开发环境包含跨平台的目标,这样的编译器不适合开发运行于小系统上的小程序。面向对象编程使用的是复杂的类层次结构与对象,可以编译大型的模块程序,但是在某些情况下并不比 C 程序更为有效。C 作为传统的面向过程的程序语言,在编写底层的设备驱动程序和内嵌应用程序时,是一个更好的选择。

Windows 98 操作系统几乎全部是用 C 语言编写的,尽管它的开发者 Microsoft 公司也开发了 Visual C++,并在市场上大力推销 Visual C++。并同意销售可视 C++编译程序的产品。对于这种明显的反常现象,我的猜测是微软操作系统的开发者对 C++的理解不够成熟,并且公司尽量维护现有的操作系统,而不去费心思考虑究竟哪个语言更为合适。如果重新开发操作系统,很可能选择 C++而不再是 C。

1.3　C 与 C++

C++是 C 的超集,当然也可以说 C 是 C++的子集,因为 C 先出现。按常理来说,C++编译器能够编译任何 C 程序,但是 C 和 C++之间仍有些小差别表现出不兼容性。

例如,C++增加了 C 不具有的关键字。这些关键字能作为函数和变量的标识符在 C 程序中使用,尽管 C++包含了所有的 C,但显然没有 C++编译器能够编译这样的 C 程序。

C 程序员可以省略函数原型,而 C++不可以,一个不带参数的 C 函数原型必须把 void 写出来。而 C++可以使用空参数列表。第 3 章将学习函数原型。C 的许多标准函数在 C++中都有对应的函数,C++程序员把它们看作 C 语言的改进。下面是一些例子。

- C++中的 new 和 delete 是对内存进行分配的运算符,取代了 C 中的 malloc 和 free
- 标准 C++中的字符串类替代了标准 C 函数库 < cstring > 头文件中的字符数组处理函数
- C++中用来做控制台输入/输出的 iostream 类库替代了标准 C 中 stdio 函数库
- C++中的 try/catch/throw 异常处理机制取代了标准 C 的 setjmp()和 longjmp()函数

语句可以分为如下几类：

◆ 声明：声明变量和函数

◆ 定义：定义变量和函数的实例

◆ 过程语句：在函数内定义的可执行的语句

变量声明语句声明了变量的存储类型、数据类型、名称等。函数声明语句(常称为函数原型)声明了函数的返回值、名称、参数的个数和类型。

变量定义包括变量声明部分，如果数据有初值那么在变量定义中也可以包含变量初始化。在变量定义中定义了变量的实例并且为实例分配了相应的内存。函数的定义包含了函数的可执行代码。

通常，变量的声明和定义放在同一语句内。函数的原型和定义一般在不同位置。如果函数的定义出现在所有调用它的源代码之前，那么函数定义本身已经起到了函数原型的作用。

过程语句可以是赋值语句、表达式语句或者程序流程控制语句。表达式语句是具有返回值的程序语句。它可以独立存在，或者放在赋值语句的右侧。表达式由变量、常量、运算符和函数调用组成。赋值语句本身也是表达式，这一点似乎不容易理解。

C++使用结构化的程序控制结构，它包括顺序结构(一个接一个语句执行)、循环结构(for 和 while 循环)和选择结构(if-then-else 和 switch-case 控制结构)。C++还允许使用 goto 语句实现非结构化流程。

类是对由数据成员和成员函数构成的集合的定义。类把用户定义数据类型的接口和数据封装起来，构成一个抽象的数据类型。

类中的私有数据成员通常是隐含的(对类进行对象的实例化并不能操作私有数据成员)。类的公共接口一般是以方法的形式提供：通过类的成员函数操作类的数据成员。

使用类需要对类进行对象的实例化，然后对该对象引用类的方法，即调用类的成员函数。类是有行为的，这实际上从另一个角度说明，通过方法使用对象就好像对象有行为一样，能够在一定层次上理解方法的含义，不必关心具体的行为细节。

如同当代大多数的编程语言一样，C++程序由多个源代码模块构成，它们被编译为目标代码模块，然后链接成为一个可执行的程序模块。在一个典型的 C++程序中，大多数的目标代码来自于已经编译好的、可以重复使用的类或者函数。

C++源代码模块是文本文件(通常是 ASCII 文件)，对于这种文件，我们可以阅读，也可以用任意的文本编辑器去修改，比如 Windows 98 的写字板小程序。本书附带的 CD-ROM 中的 Quincy 编译器，该编译器集成了一个程序编辑器。Quincy 还集成了一个源代码调试器，与大型 C++开发平台所提供的调试器十分类似。

与 COBOL、BASIC 和 FORTRAN 语言不同的是，C++没有系统内提供的输入/输出语句。输入输出功能是由标准类库中的 C++类来提供。事实上，许多其他语言本身固有的功能，在 C++中都是通过类来实现。数据转换、字符串操作和输出格式化就是其中三个例证，它们说明了 C++实现这些操作不是利用系统本身的功能，而是利用标准库中的类。C++是一个小型语言，它只能声明并定义变量、把表达式的值赋值给变量以及调用函数，只有 main()主函数是语言的一部分。C++的强大功能源自于对函数和类的扩展。标准 C++为一组标准的类和函数定义了形式和功能，这些定义放在一组标准的类和函数库中。用户自定义的类和函数库又可以支持特定的功能，进一步扩充了 C++语言。

1.5　主函数 main()

每一个 C++ 程序都有一个主函数 main(),它是程序的入口和出口。清单 1-1 是一个最小的 C++ 程序。

文件名:pr01001.cpp 文件路径: Quincy99 \ Programs \ Chap01

清单 1-1:最小的 C++ 程序

```
int main()
{
    return 0;
}
```

现在看附录,学习如何使用 Quincy 来运行书中的例子程序。如果愿意可以编译运行清单 1-1,但它除了返回操作系统以外什么都不做。

清单 1-1 声明并定义了主函数 main(),这是程序所做的全部工作。这个程序从主函数的第一条语句开始执行,并当主函数返回时结束。

这个最简单的 C++ 程序也是一个最简单的 C 程序。这两种语言都必须至少有一个主函数。

清单 1-1 基本说明了 C++ 函数的一些主要内容。第一行提供了该函数的返回类型和标识符。主函数 main()返回的是一个整型数据,并且标识符就是 main。关键字 int 具体指明了函数的返回类型。第 2 章将学习整型数据和其他数据类型。

函数名后的圆括弧内包含了参数列表。在本例中 main()没有参数,所以它的参数列表是空的。空的参数列表用圆括弧()表示。

参数列表后面就是函数体。函数体以左侧大括弧({)为开始,以右侧大括弧(})为结束。大括弧中间是函数语句,函数语句是调用函数时执行的代码。清单 1-1 只有一条 return 语句,结束主函数 main(),也结束了整个程序。

注意 return 语句后面的分号(;)。每一条 C++ 语句都以分号结束。

用大括弧包含的语句组称为一个语句块。语句块可以嵌套,在第 4 章可以看到如何使用嵌套控制程序流程。每个函数至少有一个语句块;清单 1-1 的主函数只有一个语句块。

在最后一条语句或 return 语句执行之后函数终止。清单 1-1 中 return 语句返回整型常量 0,然后程序结束。return 语句可以放在函数内部的任何位置。

操作系统调用主函数，而主函数又返回操作系统。这样说不是很确切，但是对于 C++的初学者而言，这样的解释足够说明了主函数的作用。后面学习类的时候，会学习如何执行外部对象的构造函数和析构函数，其中包括如何编写在调用主函数之前或者在从主函数返回之后运行的代码。但是现在，只需将主函数当作程序的入口和出口处。同样必须记住，在程序中不允许从任何其他地方调用主函数。

1.6　小结

本章是学习 C++编程的第一步，对 C++语言有了初步的了解，学习了最简单的 C++程序。第 2 章将学习如何编译和运行简单的 C++程序。