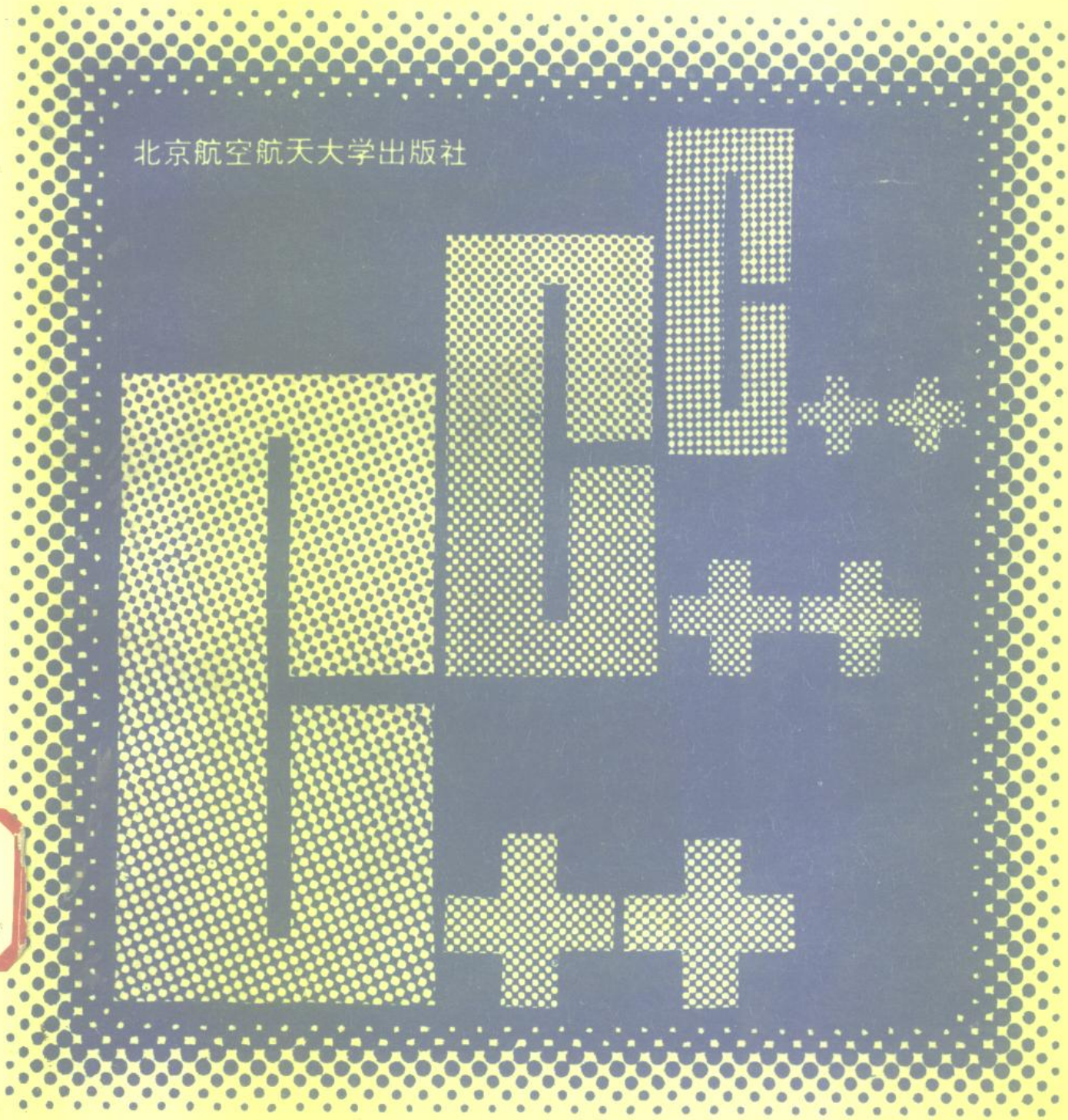
C++
程序设计语言教程
（语言基础）

麦中凡

北京航空航天大学出版社



C++程序设计语言教程

（语言基础）

麦中凡　刘书舟　严建新　著

北京航空航天大学出版社

(京)新登字 166 号

JS19.104

内 容 简 介

本书介绍C++的语法和OOP的基本概念，并讨论了利用C++语言提供的机制编写面向对象程序的方法。在书中，不仅比较了C与C++之间的细微差别，更着重介绍C++提供的面向对象机制：类、派生类、虚函数、运算符重载等等，尤其是C++高版本中新增的模板和异常。书中用大量例子来说明这些概念的使用，并在每章之后附有习题以备读者练习。在学习本书之前，读者应至少已掌握了一门程序设计语言。

本书既可以作为学习C++语言的教材，也可作为面向对象技术的入门资料。

图书在版编目(CIP)数据

C++程序设计语言教程：语言基础/麦中凡等著. —北京
：北京航空航天大学出版社，1995.7
ISBN 7-81012-578-8

Ⅰ.C… Ⅱ.麦… Ⅲ.C语言-程序[illegible] TP312C

中国版本图书馆 CIP 数据核字（95）第[illegible]670号

C++程序设计语言教程(语言基础)

C++CHENGXU SHEJI YUYAN JIAOCHENG (YUYAN JICHU)

麦中凡 刘书舟 严建新 著

责任编辑 许传安

* * *

北京航空航天大学出版社出版

(北京市学院路 37 号，邮编：100083，发行电话：2015720)

朝阳科普印刷厂印刷

新书书店总店科技发行所发行 全国各地书店销售

* * *

787×1092 1/16 印张：22.75 字数：582 千字

1995 年 7 月第 1 版 1995 年 7 月第 1 次印刷 印数：5000 册

ISBN 7-81012-578-8/TP·166 定价：25 元

前　言

按目前C++的规模和复杂程序，与最庞大复杂的Ada语言相差无几。这是近代软件工程技术的需要。完全掌握并熟练地使用各种语法机制应该是高级程序员水平。但这对急于想入门面向对象技术和学会C++作OOP的读者来说，也并不可怕，先入门再深钻，正如小学生、饭店服务员、大学生、秘书、外商助理均可以学英语一样。

本书设定为已掌握一门程序设计语言并独立编写调试过百句以上程序的人编写的(最好是C语言)，如果完全掌握本书内容，结合某个C++系统的上机练习，一般难度和小规模(<1000句)的抽象数据类型程序和面向对象程序不应有问题。至于深层排错、多继承的复杂应用、微妙的语义差别还要精读手册和其它文献，多调多练才能达到，这不是本书的目的。

本书与《C++程序设计语言教程(编程技术)》是配套教材。两书内容既有联系，又相对独立。两书合一是一个完整教材。但考虑读者选择阅读方便，按两册出版。

《C++程序设计语言教程(语言基础)》介绍C++的语法和OOP的基本概念。以用C++语法编制简单面向对象程序为目的。全书11章，除绪论外，第2～5章是C++相对于C增加的语法。为了顾及从其它语言转来的程序员，对C语言作粗略介绍，仅对C语言的难点相对仔细一些，围绕能上机运行最简单的C++程序。第6～10章是C++支持面向对象程序设计的各种机制，一直到C++3.0版的模板和异常，围绕能用C++编制简单的面向对象程序。第11章是C++的输入/输出。各章均有小结并附习题，为了便于读者将习题编程上机，书末附有四个附录。

《C++程序设计语言教程(编程技术)》全书9章，是对C++语言语法结合实际应用更深入的解释和示例：如何编制ADT程序，如何灵活地使用虚函数，如何编制OOP程序、类库等，精选的几个2000句左右的应用例题贯穿其中。希望读者学会组织OOP程序并编程实现，但本书不是面向对象开发(OOD)和OOP软件工程教程。仅限于面向对象模型已清楚的OOP程序设计。

编　者
于北京航空航天大学计算机科学与工程系
1995年2月

目　录

第1章 绪 论

程序设计语言是表达软件的工具，是人机通信的媒介。程序员利用语言提供的各种机制把自己要作的计算告诉机器。从应用的角度，程序设计语言就是一台抽象机器，程序员就利用这个抽象机器的各种功能（语言机制）编制出绘声绘色的软件。一般说来，编制良好的软件不完全依赖程序设计语言，正如东方人也能以自己的语言表达荷马、莎士比亚的诗句，西方人也能欣赏李白、李清照的诗词，拿程序设计的行话说，数学模型程序设计方法学对程序质量影响远比程序设计语言更胜。然而，从软件是人类思维产品而言，它和任何艺术品一样，有其浑然天成的特征，软件和软件质量又和程序设计语言好坏密不可分。好的语言，编制出的软件清晰明白，便于查错、修改，容易集成、扩充，值得保留，可以重用。质量不高的语言，却要依靠程序员高超技艺，并耗费大量精力，才能得到高质量软件。

从软件产业角度，当然希望得到高表达能力、高生产率，易读易改的好语言。特别是计算机工业的龙头，计算机硬件突飞猛进的发展（从1945年每秒5000次到1990年500～600MPS，计算速度提高了十万倍）软件始终处在跟不上，赶不上，不好用的危机之中，寻求好的语言始终是软件产业的头等大事。

程序设计语言随软件技术的发展而快速发展，反过来它又推进了软件产业。六十年代初期，软件界主导的程序设计语言是FORTRAN，COBOL和Algol 60。由它们编制的软件奠定了最初的软件产业基础。由于它们类型单调，程序控制过多依赖程序员的技巧，滥用goto使程序难读、难改、不易移植，程序很难编得更大，六、七十年代兴起了结构化程序设计，希望以规范格式的软件结构消除复杂软件内部的混乱。于是结构化程序设计语言Pascal，C成了明星，并迫使老语言纷纷改版结构化，FORTRAN-77，COBOL-74就是这个时代的产物，七十年代因受益于结构化，最大软件已达385万句（美国导弹预警系统）。

硬件不停地发展，软件危机不仅没有缓解反倒加剧，主要的原因是只要硬件有可能算得出来，人们就想编制更大的软件，软件一大就无法管理和维护，稍有缓解的结构化程序设计在更大的软件面前也是问题百出。所以，七十年代中期兴起软件工程，除了有成套的管理方法之外，软件技术强调模块性、抽象性，易维护、可修改、可移植。将软件应做什么和应怎么做分开的表达技术导致基于模块开发过程和软件工程原则。八十年代受益于软件工程，最大的软件已达四千万句（美国航空飞机监控系统）。

但又过了将近10年，证实传统的软件工程对软件危机并没有多大的缓解，大型软件投资失败，不可靠、不可维护、不可移植依然大量存在，寻求新的开发和表达技术的努力未有稍懈。九十年代初，在抽象数据类型和交互式环境思想基础上，以Smalltalk-80为代表的面向对象技术出现，给处于危机重重的软件产业带来希望之光。面向对象以其封装的模块性；类和实例的抽象性；继承机制提供的软件重用性；重载多态和动态束定的灵活性所支持的软件的可扩充性，无疑成为当代最成功的软件技术。于是软件产业的各主导语言都纷纷向面向对象技术扩

元，以求在新的一轮竞争中立于不败之地，C++就是在这个背景下从C语言演变而成的。

1.1 C++程序设计语言概述

在正式学习C++程序设计语言各语法特征和语义机制之前，先介绍C++的概貌以及与学习相关的特点也许是有益的。

(1) C++是C语言的超集

C++是AT&T公司1985年正式推出的(1.0版)，它是C语言的超集，是C语言向面向对象的扩充，即它除了C语言能编制过程式程序的所有语法机制而外增加了类和实例、继承、重载运算符、虚函数、友员、内联等支持面向对象程序设计的机制。这些机制的某些实现是C语言做出的，所以编译并不拒绝纯C语言编写的过程式程序，和某些语言改造为面向对象后成为全新语言不一样。C++完全容纳C并成为多范型语言，即支持多种程序设计风格，C是C++的真子集就决定了学习C++不仅要学习面向对象程序设计技术，还要学习C语言已有的、行之有效的编程技巧和特征。这里就有一个是先学C然后增加C++的特征，还是先学C++反过来补充C的生疏点的问题。我们主张后者。因为面向对象程序设计风格和过程式程序设计风格相差很大，而人总有语言惯性。一旦习惯用C编过程式软件改成面向对象反倒麻烦，故本书认为C和C++是一个语言，C++的具体的语法借重于C，在程序设计风格上有意忽略C的过程风格，它只是C++的基层语句，在类方法(函数)的实现中，离不了C语言的数组、指针、联合、for循环等基本技巧。

(2) C++集中反映了软件新技术

八十年代软件和程序设计语言技术的新特征是：

●数据抽象和抽象数据类型。

用户可以定义复合的数据类型，且这种抽象的数据类型的使用和它的实现可显式分离，便于开发和修改。

●数据隐藏。

程序模块有可控的可见性，因而块中局部数据对块外是隐藏的。

●支持重用的类属机制，类型、变量、函数均可参数化。

●事前控制程序失败的异常机制。

●支持并发程序设计。

●支持重载与类型的多态，以利于程序扩充。

●支持重用的继承机制。

●支持模块通信的编程风格。

C++或多或少或完整或初步地反映了上述特征，本书将结合语法详细介绍。正是由于它反映了较多的软件新技术，它不是最终用户首选的语言，也不是程序设计入门的语言，而是应用开发者和系统程序员的最佳语言。因而，本教程篇幅不算太小，还假定读者已学习过一门结构化程序设计语言，并有数十小时上机经验。

(3) C++可能成为软件产业的主导语言

C 语言以其简洁的表达、实现的高效(与 UNIX 紧密相连有关)从 1972 年问世以来,以无可辩驳的优势席卷系统软件领域,比与它同期诞生的 Pascal 增长快许多倍,深受程序员的喜爱。"除了汇编,C 是最方便的,而 C 又脱离具体机器便于移植,生产率也高。"特别是七十年代后,微机、工作站大量普及,C 语言又以小巧占据了优势。但 C 并不好读,也不易学,可一经掌握程序员都不愿放弃它。

八十年代软件工程环境大发展,以 UNIX 工具箱开创的环境工具开始了第一代 CASE(计算机辅助软件工程环境)的功能,用 C 开发的各种工具(包括在 DOS 上和 Windows 上的)充斥市场,C 已牢牢地占据了系统软件市场,奠定了在软件产业中的地位。

C++采取与 C 完全兼容的策略,使得它不仅继承 C 所取得的成就,在用八十年代软件工程思想改造后(成为 C++)使它在未来一、二十年间将成为无可匹敌的软件开发工具。值得一提的是,美国国防部 1980 年为统一三军通用程序设计语言而支持的 Ada——Pascal 后裔,具备八十年代软件工程要求的全部特征,且为强类型,是模块式语言的典范,但正因为过于强调安全和静态评审检查,使得该语言缺乏灵活性(特别是动态束定实现较复杂),表达能力较 C++为差,十年来由于面向对象技术冲击的结果,除了军方强行推行而外,并未像想象中的那样迅速普及。然而,一个语言的成功除了本身设计、实现好坏而外,还要看有没有人支持和投资。C 语言 C++除了设计实现被程序员看好而外,民间软件投资很大是个事实。军方投资在 Ada 上,而且至今美国国防部仍坚持 Ada,并积极地把不适应面向对象的 Ada-83 改版为 Ada-9x,相比之下,其它语言就没有这个幸运了,所以在通用的系统软件领域将是 Ada-9x 和 C++的天下,超大型软件 C++仍不及 Ada-9x,一般软件 C++有较大优势,而在人工智能领域将是 Common Lisp 的后继 CLOS 的天地。

(4) C++仍在发展

一个语言的成长,按语言学家统计的规律至少要有 15 年(即概念完善、原型开发、产品应用的三个五(年)规律),目前尚未见到正在开发适合面向对象软件工程的全新语言,故只有相对最好的语言 C++和 Ada-9x,但面向对象技术也在纵深发展,C++也将不断改版。

C++ 1.0 版提供了支持面向对象的主要机制:类和实例、单继承、友员、虚函数;公、私有接口;内联等(1985)。C++2.0 版支持多继承,增加保护接口(1989),C++3.0 版又增加了模板、异常,估计会相对稳定一段时间,因为它已经较为全面地实现了当今软件工程对语言提出的各种需求。

然而,面向对象本身开辟的新领域又提出了新需求。例如,在面向对象数据库支持下的软件工程,把软件的重用提到一个新的高度,所有软件均在带库的环境支持下开发,大量软件(大于 80%)重用。这就要求数据库对数据对象的操纵和程序语言对数据对象的操纵一样方便,中间不需转换。也就是说,C++应提供对持久性对象(放到数据库中的)的描述。C++既是程序设计语言也是数据库数据模式定义和操纵的语言。

除此而外,并发部分有可能成为语言机制(而不是目前用 C++写的类库)。人工智能向各种软件技术渗透势必导致软件产业主导语言扩充模拟 AI 专家系统的机制:增加触发、关系路径追踪(表达语义网)机制都是有可能的。

纵上所述，C++是极有前途的语言，这和它的开发者——AT&T 公司的 Bjarne Stroustrup，是系统程序员出身密切相关。与 C 语言一样，开发者以长期实践经验结合面向对象思想提出并实现的各种机制，强烈体现"解决问题""务实"的思想，不求理论上的完善和一致性（如友员的滥用会毁掉对象的封装，它只是为高效共享操作的一种折衷），在许多问题上把困难和问题留给程序员而使语言不致太复杂（如多继承的可靠性）。本书力图在讲述中指出这些问题。

1.2 C++简短的历史

C++不象 PL/1，Algol-68，Ada 那样，由许多语言专家组成的委员会开发，而是由几个（主要是一人）专家开发的，直到后期（形成正式版本时）才由 AT&T 公司组成委员会。因此，语言的特征与开发者经历密切相关。

对 C++影响最大的两个语言一为 Smalltalk，一为 Simula-67。

众所周知，Smalltalk-80 是第一个正式提出面向对象概念的语言，它的类——实例概念取自 Simula，它又发展了继承体，在无类型、以匹配实现动态束定、运行中选择子归约等方面又学自 LISP。于是形成了前述的面向对象五大特征。向人们展示符合软件工程原则的五大特征可由一个语言体现，这是最大的成就。

正是由于匹配和归约，它的运行效率直到目前还不十分理想（比 C++约慢 3～5 倍）。所以，它一问世就有人想以传统语言模拟它，保留它在面向对象上的所有特征，而在效率上希望有重大的突破，值得一提的是 Objective C，它几乎是 Smalltalk 的 C 语言版本，编程风格与传统 C 相差很大，C 的软件资源不能直接使用。C++在向面向对象扩充时，考虑到 C 已取得的成就和大量软件资源存在的现实，采取与 C 完全兼容的道路。放弃了 Smalltalk 学自 LISP 的那部分，带有更强的 Simula 的烙印。

C++的发展大致也经历了概念形成（1979～1985）、原型开发（1985～1888）、工业成长（1988～）等三个阶段。原著者 Bjarne Stroustrup 在《C++的历史》一文中更仔细地介绍了以下阶段：

(1) 带类的 C（1979～1983）

1979 年 C 语言已得到完善的发展成为系统程序员喜爱的语言，但是对要求模块性和并发性较高的分布式系统表达能力不足，Stroustrup 在 AT&T 公司接受到开发一个新的分布式系统时，自然想到他过去在剑桥大学计算机实验室曾做过的类似的模拟系统。他曾试用过 Simula-67和 BCPL（C 语言的前身）。Simula-67 有较强的类和实例表现能力，模块性好，但效率太低，而 BCPL 相反效率高但表达能力差。于是就想到扩充 C 使之具有 Simula 的类。他写了一个 Cpre 预处理器，增加了类的机制，并完成支持协例程（co-routine）编程风格的任务库。这就是带类的 C。

带类的 C 顾名思义主要是扩充类机制。类是用户定义的复杂数据对象的样板，是封装了数据和操作的"数据类型"，可以按类型定义变量一样生成很多对象。由于类比一般类型复杂，它在生成对象时就要有复杂的初始化工作。为此提供构造对象的构造函数。当不再需要某个对象时，设析构函数以取消它，复杂的对象对程序而言主要是它的行为而不是行为如何一步步实现的动作。因此类样板设有控制可见性的接口，不在接口上的数据和操作（调用这些操作实

现对象行为)对其它对象是不可见的。这就体现了封装性和局部性,为此扩充了公有私有控制,以及缓和封装的友员类和减少冗余的派生类。当然,派生类继承父类的机制完成了面向对象最重要的特征。为了充分体现抽象数据类型把"做什么"和"怎么做"分开,允许程序员在类中只写函数声明,函数体可另写于类外,但为了运行效率,把这些分开写的函数定义和函数声明又连接起来(执行可一贯到底),故设内联函数,此外类型检查和函数参数的类型转换、赋值运算符的重载等都是这一时期扩充的 10 种机制。

带类的C 最初在 DEC PDP/11 机上实现使用,以后又移植到 DEC VAX 系列和 Motorola 68000 系列的机器上。它已应用于网络模拟器的研究之中。

(2) **从带类的** C **到** C++(**1982～1985**)

带类的 C 的原理探索取得非常良好的效果,各种有 C 语言环境的用户竞相扩充带类的C,以致于这些软件没有相互移植性。因此,AT&T 公司就有把带类的 C 作为一个新标准语言的设想:做一个工业化的 C++版本,提供最低限额的标准的支持工具,并定名为 C84 语言(即 1984 年推出正式版本)。

C84 编译器的前端叫做 Cfront,它对语言的语法和语义作完全的检查,建立输入的内部表示,产生代码生成所要的输出。在完成 Cfront 前端的工作中,全部吸收了带类的 C 的 10 种特征,又扩充以下 6 种特征,它们是:

虚函数/函数名和运算符重载/引用机制/常量 const/用户对自由存储的控制/改进了的类型检查以及注释表示的多样化。

由于 C++和 C 各自目标不同,相互 100%兼容显然是不可能的,但商业和软件继承的需要只能采取折衷。由于 C 有几十个版本,ANSI 标准化组织对其标准化产生了 ANSI C。C++和 ANSI C 做到基本兼容,ANSI C 使 C 向 C++更接近。例如,函数调用一律作类型检查,因而设参数原型和缺省参数等机制(本身就扩充了 C)。当然,C++和 ANSI C 依然存在"合理"的不兼容性。

1985 年随着《C++程序设计语言》(本书中译本 1987 年由清华大学出版社出版)一书的出版,C++1.0 版定型。它不象其它语言先有语言规格说明,最后陆续验收合乎该规格说明的编译器,而是在已有编译器原型上写出语言正式定义。

(3) C++2.0 **版**(**1985～1989**)

至 1986 年,C++的用户已增至 2000 个,作为一个原型语言它是成功的,于是,如何开发一个更加高效的全新编译器以淘汰 Cfront,以及如何更加完善支持面向对象的各种辅助机制,如流类库、各种部件库等等。这两方面工作同时进行,所以 C++1.x 版陆续出现,1.1,1.2,1.3,…,而扔掉 Cfornt 的版本最后做成 C++2.0,预定 1988 年推出的 C++1.3 版未实现。

1989 年推出的 2.0 版的新特征是:

多继承/抽象类/静态成员函数/const 成员函数/protected 成员函数/运算符的重载/成员指针/赋值和初始化的递归定义。

对一些特征又作了改进:

重载分辨/类型安全连接/用户自定义的内存管理设施。

C++2.0 的效率、安全性都比 1.0 版有显著的改进。C++的用户每 7.5 个月翻一番至

1989 年估计已逾 5 万。但此时仍着眼于功能完善和安全可靠，开发各种机型上的 C++编译器。1986 年 Zortech 在微机上推出 C++编译器，1990 年 5 月 Borland C++，1992 年 Microsoft C++，DEC、IBM 也于 1992 年推出自己的 C++。

但总的说来，类库支持仍然不丰富，而各家类库互不兼容，也还有一些扩充未正式纳入 C++2.0，仅列入语言参考手册(ARM)。

(4) C++3.0 (**1993**)

自 C++2.0 版后，2.1 版扩充了迁移模块，以便向更高版本平滑过渡。

为完善类库提供重用支持，C++3.0 版扩充了模板(template)，即提供抽象类型的数据，可以实例化为需用的类型，实现了真正嵌套的作用域，放弃了 2.1 版扩充的迁移模块。改进了继承，解决了虚基类的多继承的命名冲突/完善了构造函数和析构函数。完善了成员和友员机制/完善了重载运算符放宽内联限制，const 限制/可对函数返回值优化等等。

ARM 参考手册中列出过的异常(exception)虽在不少 C++3.0 编译器中实现，但未纳入 3.0 版。

从以上 3.0 版的改进可以看出，由于 C 语言过多依赖实现且方言不统一，造成上层 C++语法扩充后语义微妙的不同，致使完善和改进拖了很长时间，运用时要求对 C++语言标准化。

(5) **标准化(1989)**

1989 年后 C++开始形成使用热潮，同时也要求尽快标准化，1989 年由 HP 公司联合 AT&T，DEC，IBM 等公司发起建议标准化。为此，美国国家标准局成立了 C++语言标准化小组 X3J16，于 1989 年 12 月召开第一次会议。1991 年 6 月国际标准化组织也为 ISO C++成立了 WG2 委员会，第一次会议在瑞典召开，当然标准化组织是不会发布尚处于快速发展、不太成熟的版本。至今 C++没有正式标准。以上所说 C++1.0，2.0，3.0 版均以 AT&T 公司的版本为准。

类库标准化是继语法编译器标准化之后的重大领域。它涉及到面向对象支持的高度重用性能否实现问题，如果对象定义，类库组织没有统一的标准，重用只是一句空话。所以美国民间于 1991 年成立 OMG(对象管理组)机构，在对象结构标准化之后类库标准化就很方便了。

习　题

1.1 试述软件危机的根本原因，和已经有过的打退危机的软件技术。

1.2 结构化程序设计的主要特征是什么？它比传统过程式程序的优势在什么地方？(效率高？易读易修改？易于设计…)

1.3 就你所知面向对象程序结构的概貌。

1.4 何谓内定义(build-in)、预定义(predefined)和用户定义的类型，你能(用任何语言)定义一个堆栈类型吗(最大能放 100 个整数)？如能编出程序并上机，如不能，说明原因。

1.5 何谓预处理器？何谓编译器？它们能合并吗？

1.6 何谓抽象数据类型？试述它的主要特征。

1.7 面向对象的五大特征是什么？

1.8 论证：程序质量与程序设计语言无关。

1.9 论证：程序设计语言质量与软件生产率密切相关。

1.10 一个成熟的软件要经过那几个发展阶段？