高等学校教材

程序设计语言概论

郭浩志等 编著

国防科技大学出版社

程序设计语言概论

郭浩志（主编）
宁　洪　张李秋
侍　晖　王怀民
编　著

国防科技大学出版社

内 容 简 介

本书旨在介绍现代程序语言的基本概念、思想和设计方法。主要内容包括程序语言的设计原则、强制式语言的框架、数据类型、数据抽象、并行机制和异常处理，各种作用式语言（包括函数型、数据流、逻辑型）和面向对象程序设计语言的程序范型、风格和特征。重点在于讨论语言的基本设计思想，而不是具体程序设计细节。本书既概要总结传统语言的概念和思想，又反映70年代中后期以来程序语言的最新研究成果。各章之后均附有适量习题和主要参考文献。

本书可作为高等理工科院校计算机科学、工程或应用诸专业高年级本科生或低年级研究生的教材，也可作为教师或科研技术人员的参考书。

程序设计语言概论

*

郭浩志 （主编）
宁 洪 张李秋
侍 晖 王怀民
编 著

*

责任编辑 戴东宁

*

国防科技大学出版社 出版发行
国防科技大学印刷厂印装

*

开本：787×1092 1/32 印张：12 字数：285千字
1989年6月第1版第1次印刷 印数：1—3000册

ISBN 7-81024-068-4

TP·13 定价：2.45元

出 版 说 明

根据国务院关于高等学校教材工作分工的规定，我部承担了全国高等学校、中等专业学校工科电子类专业教材的编审、出版的组织工作。由于各有关院校及参与编审工作的广大教师共同努力，有关出版社的紧密配合，从1978年至1985年，已编审、出版了两轮教材，正在陆续供给高等学校和中等专业学校教学使用。

为了使工科电子类专业教材能更好地适应“三个面向”的需要，贯彻“努力提高教材质量，逐步实现教材多样化，增加不同品种、不同层次、不同学术观点、不同风格、不同改革试验的教材”的精神，我部所属的七个高等学校教材编审委员会和两个中等专业学校教材编审委员会，在总结前两轮教材工作的基础上，结合教育形势的发展和教学改革的需要，制订了1986～1990年的“七五”（第三轮）教材编审出版规划。列入规划的教材、实验教材、教学参考书等近400种选题。这批教材的评选推荐和编写工作由各编委会直接组织进行。

这批教材的书稿，是从通过教学实践、师生反映较好的讲义中经院校推荐，由编审委员会（小组）评选择优产生出来的。广大编审者、各编审委员会和有关出版社为保证教材的出版和提高教材的质量，作出了不懈的努力。

限于水平和经验，这批教材的编审、出版工作还会有缺点和不足之处，希望使用教材的单位、广大教师和同学积极提出批评建议，共同为不断提高工科电子类专业教材的质量而努力。

电子工业部教材办公室

前　言

本教材系按电子工业部的工科电子类专业教材1986—1990年编审出版规划，由计算机与自动控制教材编审委员会计算机编审小组征稿，推荐出版，责任编委为陈火旺教授。

本教材由国防科技大学郭浩志副教授担任主编，华中理工大学（原华中工学院）刘键教授任主审。

本课程的参考时数为40学时。本书旨在介绍现代程序语言的基本概念、思想和设计方法。全书共分十章。第一章概述，介绍程序语言的简史、设计原则、发展特征以及强制式语言和作用式语言的主要区别；第二章顺序控制结构，给出强制式语言的框架；第三至第六章依次讨论了几种主要语言机制（数据类型、数据抽象、并发与异常）的基本概念和特征；第七至第九章初步研究三种作用式语言（函数型、数据流和逻辑型）的设计背景、语言特性和风格，其中函数型以LISP，FP和ML为代表，数据流以VAL为主，逻辑型以PROLOG为例；第十章面向对象的程序设计，以Smalltalk为背景，叙述其五个基本概念、特征和环境。

现今世界上的程序设计语言数以千计，但其中提出新概念，开创新思想，在基本原理、方法和风格上别具一格的不过一、二十种。本书重点讨论这些典型语言中的基本概念、思想、机制、风格和方法。为帮助读者消化本教材所述基础知识，并进一步跟踪专业前沿知识和发展动态，每章之后均附有形式多样的各类习题和主要参考资料。

本教材由郭浩志编写第一、四、五、六章，宁洪编写第二、九章，张李秋编写第三、七章，王怀民和侍晖两位博士生

分别编写第六和第十章。郭浩志统编全稿。编写过程得到国防科大及其研究生院各级领导的支持和帮助，俞成宜教授对第九章，齐治昌副教授对编写大纲均提过很好的建议，这里一并表示诚挚的感谢。由于编者水平有限，书中难免还存在一些缺点和错误，殷切希望广大读者批评指正。

编　者

1988年10月

目 录

第一章

程序设计语言概述

1.1 历史的回顾

1.1.1 高级语言的崛起

五十年代中，美国纽约IBM公司一个以John Backus 为首的研制小组开始探讨所谓代数语言翻译器。当时，人们想得较多的并非是这样一种语言是否能够被设计和实现，而是估计这种语言很可能是极端低效的。这种担心大大刺激了该语言的设计。结果表明，很多FORTRAN 语言编译程序都可生成十分高效的代码。第一个设计的FORTRAN 0（FORmula TRANslation）诞生于1954年，其编译程序约在两年半后研制成功。当时没有FORMAT语句和用户自定义函数，条件语句和变量也与今天的不同。FORTRAN 0 编译程序尚未最后完成，又开始设计 FORTRAN Ⅰ和Ⅱ．也许是由于IBM 的发起和赞助，FORTRAN 一经发表，便很快流行起来。当今流行的FORTRAN 的许多特色在早期 FORTRAN 中已清晰可见，例如，程序书写采用固定三个区的格式，隐式类型说明，三分支的算术IF语句，简单的DO语句，借助FORMAT 语句给出 I/O 控制等等。在机器语言盛行时期，FORTRAN语言摆脱开早期程序员顽固的习惯势力，脱颖而出，在计算机语言领域中起着开辟新纪元的作用，成为国际上公认的第一个通用高级语言．

并牢牢树立了高级语言在计算机科学中的地位。

出于交流算法和精确描述计算机能接受的计算过程等目的，1958年美国的ACM和西德的GAMM在瑞士苏黎世会议上，交流了近三年各自的研究工作，并成功地联合建立一个所谓国际代数语言IAL或ALGOL 58 (ALGOrithmic Language)，后经一再修改，于 1960 年 5 月发表正式报告，即 ALGOL60. ALGOL60 首次引入很多现代程序语言的新概念。例如，书写的自由格式，系统预定义保留字，必须显式说明的类型，形式更一般的循环语句，分程序结构，作用域规则，递归过程，值参数，换名参数和动态数组等等。

随后很多陆续开发的语言均以ALGOL 60的风格为模式，我们统称为类ALGOL语言。其主要特征为：

- 描述计算过程的算法；
- 强制式；
- 基本计算单位是分程序和过程；
- 具有类型概念和类型检查；
- 采用词法作用域规则；
- 实现采用编译方法

实际上，ALGOL 60 本身并未在世界范围内得到广泛使用。原因是多方面的，其中没能得到象IBM这样强有力的公司的支持，技术上缺少标准 I/O等都是重要原因。从今天的观点看，还可进一步指出，它不支持定义新类型，类型检查不严密，程序中的个别修改会导致整个程序的重新编译，作用域规则有局限性，无数据抽象设施等等。

尽管如此，我们也不应抹煞ALOOL 60的贡献。它的诞生标志高级语言开始成为一门科学。它是第一个用严格语法规则定义的高级语言。它的发展激发了对程序语言和编译程序理

论的研究热情，它的风格直接波及整个60年代高级语言的发展。

1.1.2 高级语言的急剧开发

从60年代初开始，随着语言和编译程序理论的进一步发展，很多翻译问题相继得到完善的解决。与此同时，高级语言的数量按指数规律猛增。据Jean Sammet的估计，仅在60年代就开发了大约200多种高级语言。但多数语言或大同小异，或是方言，其中能提出新概念，开创新思想，在基本原理或方法上有独到之处的高级语言只不过十多种。本书的注意力主要也集中在这些有研究价值的高级语言上。

一、FORTRAN和COBOL语言在计算机领域中地位的确立

1959年，美国国防部为使其所有军事部门统一使用一种语言，从政府和工业界召集了很多专家进行研究，并于当年末推出合作成果COBOL（COmmon Business Oriented Language）。一年后，RCA和Remington-Rand-Univac首次研制成功可供使用的COBOL编译程序。COBOL将若干新概念引入语言领域（例如独立于机器的数据部分），采用接近英语的表达方式，具有较强的数据处理能力，适于广大非专业人员使用。COBOL的出现，打破了计算机应用领域仅限于科学工程数值计算的局面，开始进入各种事务处理领域。美国和西欧的多次调查表明，近二、三十年用高级语言编写的程序大多数是用COBOL语言完成的。

FORTRAN主要用于描述数值计算，而COBOL适用于事务数据处理。这两种语言分别被人们公认为上述两个领域的标准语言。当然，它们都还适用于其它很多领域，具有较强的通用性。

二、ALGOL60风格的继承与发扬

虽然ALGOL60的实际使用范围不如FORTRAN和COBOL广泛，但它却激起人们对语言进一步研究的兴趣。60年代，继承和发扬ALGOL60风格的典型语言有ALGOL-W，Euler，SIMULA，PL/1，ALGOL68和Pascal等。

由Niklaus Wirth研制的两个很有意义的类ALGOL语言ALGOL-W和Euler引入了很多新的概念，例如，值——结果参数，case语句，记录和指针。

1967年，由挪威的Dahl和Nygard研制的SIMULA语言是第一个应用于模拟领域的高级语言。它引入的class概念，将数据和操作合为一个模块，被誉为数据抽象的先驱。此外，它的协同子程序（Coroutines）概念是对过程的发展，两个协同子程序可彼此调用，多次进入，实际上已是并行思想的萌芽。

60年代中IBM研制一个集60年代以来出现的新概念和新思想于一身的汇集型语言——PL/1，试图替换FORTRAN和COBOL，并配合当时IBM的新型计算机360系列的设计，除了从FORTRAN，COBOL和ALGOL 60吸收分程序结构、递归过程和记录等等概念外，PL/1还引入两个前所未见的重要设施，即异常处理和多任务。在理论上，PL/1的维也纳文本是用操作语义形式描述语言的成功范例。

在ALGOL60的工作告一段落后，IFIP工作组重新定义该语言的规格说明，并作一些次要修改以及消除大量二义性和模糊之处，最终产生当今使用的ALGOL 60新版本。此后不久，该工作组以1968年报告形式，发表ALGOL 60后继者，即ALGOL 68，尽管名称相似，但在实质上与ALGOL 60是不同的。当时很多人对它提出异议，认为即使对熟练的程序员，ALGOL 68报告也太难读和难掌握。事后证明这个看法是正确

的。后经再三修改，才于1975年公布正式版本。ALGOL 68是一个可扩充语言，它提供了一个语言核心和一组扩充工具。它不仅引进很多新的、有意义的概念，还成功地将通用性和正交性原则应用于语言设计中。在理论上，它的 Wijngaarden 文法是首次严格描述完整语言的零型文法。

由N.Wirth 在 70 年代初设计的 Pascal 语言，成功地处理了小巧与强功能、高效与可移植性以及简洁与冗余之间的矛盾。Pascal 的数据结构设施（尤指用户自定义数据类型）是很有特色的，它促进了对数据抽象的研究。Pascal 是第一个体现 Dijkstra 和Hoare结构程序设计思想的语言。Hoare 和Wirth 给出了它的公理定义。在解决日益增长的软件危机和程序验证等方面，Pascal 也起了巨大的作用。与语言设计大师Wirth 开发的其它语言一样，Pascal 语言以其简洁、设计上的经济性和工程上的高质量而著称。

当然，Pascal 也有其缺陷。例如，文件处理设施的功能较弱，没有动态数组，在编译期间无法作完整的类型检查，指针使用虽比PL/1的好，但仍会产生无用单元和悬挂访问等问题，循环体无应付意外的出口等等，随着计算机事业的发展，当将它用于描述大型软件系统时问题显得更为突出。例如，数据抽象程度不高，无分别编译机制，不能描述并行性，无低级程序设计设施，语法不系统等等。80年代提出的Pascal 最新版本和很多计算机上的实现或多或少作了一些改进。

三、非ALGOL风格语言的出现

60年代研制的，与 ALGGL 风格不同的典型语言有APL，LISP和SNOBOL等。

IBM的Kenneth Iverson在60年代初研制的APL语言，其设计目标是引入一种简洁的计算表示，其中包括很多数学概念和

符号。由于它采用非标准字符集，语义又比较隐晦，以致在几年之内仍停留在纸面上。直至60年代中，Iverson, Falkoff和其他人利用 APL 形式描述新的 IBM/360 计算机系列获得成功，才有了实现该语言的希望。但具体实现时问题很多。例如，该语言如何用字符串表示，采用什么字符集，如何确定名字作用域等等。尽管APL违背现代程序理论中若干宗旨，但它本身已被事实证明是一个十分健全和多用途的语言。APL东山再起后，不仅获得蓬勃的发展，而且最终还赢得一批科技人员热情的支持。

John McCarthy 和他在 MIT 的同事，从 50 年代末就开始着手研制LISP（LISt Processor)。McCarthy虽参加过ALGOL 60 的设计，但 LISP的风格却迥然不同，其概念简单，主要用来处理符号表达式。LISP 中引入了很多新概念。例如，作为统一数据结构的表，数据和程序统一表示方法——S 表达式，递归条件表达式，前缀运算符，将递归作为基本控制结构，采用无用单元收集。当然，LISP 成功和始终令人刮目相看的原因，远不止上述这些。更重要的是它开创了一种所谓作用式语言（Applicative Language）的程序设计风格。它取消了赋值语句和副作用，语义简洁清晰，成为研究程序设计的一个理想工具。在函数构造方面，与数学上递归函数的构造方法十分相似。在理论上，证明了存在一组简单而又完备的基本函数，可以构造字符集上的任何可计算函数。它具有自编译能力，是第一个自己描述自己语义的语言。在形式语义方面，它开辟了操作语义的方向。LISP 广泛应用于数学中的符号微积分计算、定理证明、谓词演算以及其它符号处理和人工智能领域。实际上，考虑到效率等因素，当今使用的大多数 LISP 版本都在纯LISP语言中嵌入很多强制式语言（Imperative Language，又

译作命令式语言）的成分。

60年代中期，Farber，Griswold和Polonsky在贝尔电话实验室研制成专用于字符串数据处理的SNOBOL语言。其语句由在符号名字串上运算的规则组成，基本运算有字符处理、模式匹配和替换。此语言的贡献是提出模式数据类型，便于程序员定义数据类型，在串操作方面达到比较完善的水平。SNOBOL后相继改进为SNOBOL2和SNOBOL3。虽然SNOBOL是内容丰富和复杂的语言，但其实现却比PL/1和ALGOL68的简单。它当今的版本SNOBOL 4给出一种与机器相对无关的宏功能，增加了其程序的可移植性，使它得到进一步的推广。SNOBOL主要用于文章编辑、编译级代数表达式的符号处理等等领域。

1.1.3 70年代的高级语言向纵深发展

60年代高级语言蓬勃发展，计算机应用领域和使用人员猛增，用户的要求越来越高，而问题也越来越多，终于爆发了所谓“软件危机”。进入70年代后，高级语言的发展并未止步。但主要不是在数量上增加，而是提高性能，强调风格，引进新的机制，尽量减少程序员犯错的可能。强制式语言逐步走向成熟。此时期的代表语言有ELI，CLU，并发PASCAL，Mesa，Euclid，MODULA，SMALLTALK和Ada。对于70年代的发展，我们不准备逐个语言进行介绍，而是突出这个时期引入高级语言的三个重要机制。

第一个是数据抽象机制。实际上，60年代SIMULA语言的class概念应被认为是第一个引入高级语言的数据抽象机制。只不过70年代的CLU、Euclid，MODULA和Ada诸语言中的数据抽象得到进一步的发展和确认。我们将考察此特征如何

引入这些语言，以及所涉及的语义问题（如作用域和类型等等）。

第二个是异常处理机制。60年代 PL/1 语言的异常概念是现代程序语言异常机制的雏形，它实质上是“中断——转移——返回”设施。学术界后来的争论集中在正常处理应该如何继续执行下去，以及异常引发后对象的作用域如何确定。Ada和CLU的异常处理机制是经过精心设计的，比较完善。

第三个是并行处理机制。实时应用是操作系统、过程控制系统和飞机订票系统等等大型系统中十分有意义的计算部分。旨在支持开发这些应用的语言，首先允许程序员定义可同时执行的程序段（即进程），然后提供一种使进程之间可以同步、互斥和通讯的手段。为此，Edsger Dijkstra 的信号量，Brinch Hansen和C.A.R.Hoare 的管理，Hoare 的消息传递及Ada的任务等等先后被引进高级语言。

以上三种机制，我们将分别在第四、五、六章讨论。

1.2 程序语言的设计原则

在开发Ada过程中陆续推出的一组需求文件(指Strawman, Wooderman, Tinman, Ironman 和Steelman) 也许是至今为止评价、设计程序语言最完整的资料。对它的讨论虽然是有意义的，但是它过于详尽了。我们在这里试图从语言设计者、实现者和使用者的不同角度综合给出比较公认的程序语言的九条设计原则。

1. 定义严密性

指语言的语法和语义内在一致、完整，且无二义性。作为一个好的语言，无论其程序范型 (Programming Paradigms)、风格、规模和应用领域如何不同，都必须满足这一条原则。这

是程序语言设计的一条基本原则。现时普遍采用的BNF或语法图是比较简明的语法定义形式。而几乎所有程序语言的语义都用自然语言描述，这种方法经常隐藏着错误、二义性和不完整性，往往导致语言设计者、实现者和使用者对语义有不同的理解。为精确定义语义，产生了形式语义学。用于程序语言语义定义的形式语义方法主要有三种，它们是指称语义，操作语义和公理语义。

指称语义（Denotational Semantics）学最初是英国牛津大学C.Strachey于1964年提出的。美国的D.Scott为其奠定数学基础。此语义学的观点是：语言成分的含义是该成分本身所固有的，并不依赖所在计算机系统。因此语言成分的语义是执行该成分的最终效果。此效果被认为是该成分所指的外在物体（指称物）。指称语义以语言成分的指称物作为其语义。此方法使语言中每一成分与一个适当的数学对象对应，所以又称为数学语义（Mathemetical Semantics）。指称语义方法的主要优点有两条。一是具有结构性（任一对象的语义总可根据其组成成分的语义构造），二是简洁可读性好，免除了很多细节。利用指称语义已成功地定义了CHILL，Ada等大型语言以及数据库和操作系统等。指称语义方法的主要缺点也有两条。一是用它难以描述并发进程的语义。第二，它是解释性的，难以自动生成语义制导的编译系统。然而，指称语义毕竟是当今最有影响的形式语义方法。

操作语义学将语言成分所对应的计算机系统的操作作为该成分的语义。用一台解释某程序语言的抽象机来定义该语言语义的方法叫做操作语义（Operational Semantics），又称为解释性语义（Interpretive Semantics）。维也纳的IBM实验室在60年代首次利用描述操作语义的元语言VDL（维也纳定义语