中国计算机学会教育专业委员会 推荐
全国高等学校计算机教育研究会 出版
高等学校规划教材

程序设计语言与编译

候文永 编

计算机学科教学计划1993

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
URL: http: // www. phei. co. cn

高等学校规划教材

程序设计语言与编译

龚天富　侯文永　编

電子工業出版社
Publishing House of Electronics Industry

内容简介

本书为计算机专业的广口径教材，它适合各类计算机专业本科教学，经适当剪裁亦可作计算机专业大专参考教材。全书共九章，第一、二、三章讲述高级程序设计语言的共性，特别是数据类型和控制结构。第四、五、六、七章讲述编译实现，第八章介绍函数式、逻辑式、对象式程序设计语言，第九章介绍形式语义学。

本书尚可作为计算机科学研究和工程技术人员的参考书。

丛 书 名：高等学校规划教材
书　　名：程序设计语言与编译
编　　者：龚天富　侯文永
责任编辑：张凤鹏
特约编辑：袁　源
印 刷 者：北京科技印刷厂
出版发行：电子工业出版社出版、发行
北京市海淀区万寿路 173 信箱　邮编 100036　发行部电话 68214070
URL:http://www.phei.co.cn
经　　销：各地新华书店经销
开　　本：787×1092　1/16　印张:16　字数:406.4 千字
版　　次：1997 年 6 月第一版　1997 年 6 月第一次印刷
书　　号：ISBN 7-5053-3849-8/G·289
定　　价：18.00 元

出 版 说 明

中国计算机学会教育专业委员会和全国高等学校计算机教育研究会(以下简称"两会"),为了适应培养我国21世纪计算机各类人材的需要,根据学科技术发展的总趋势,结合我国高等学校教育工作的现状,立足培养的学生能跟上国际计算机科学技术发展水平,于1993年5月参照ACM和IEEE/CS联合教程专题组1990年12月发表的《Computing Curricula 1991》,制定了《计算机学科教学计划1993》,并组织编写与其配套的18种教材。现推荐给国内有关院校,作为组织教学的参考。

《计算机学科教学计划1993》是从计算机学科的发展和社会需要出发提出的最基本的公共要求,不是针对某一具体专业(如计算机软件或计算机及应用专业),因此它适用于不同类型的学校(理科、工科及其它学科)、不同专业(计算机各专业)的本科教学。各校可以根据自己的培养目标和教学条件有选择地组织制定不同的教学计划,设置不同的课程。本教学计划的思想是将计算机学科领域的知识,分解为九个主科目(算法与数据结构、计算机体系结构、人工智能与机器人学、数据学与信息检索、人-机通信、数值与符号计算、操作系统、程序设计语言、软件方法学与工程)作为学科的公共要求;对计算机学科的教学归结为理论(数学)、抽象(实验)和设计(工程)三个过程;并强调专业教学一定要与社会需要相结合。另外,还提出了贯穿于计算机学科重复出现的十二个基本概念,在深层次上统一了计算机学科,对这些概念的理解和应用能力,是本科毕业生成为成熟的计算机学科工作者的重要标志。

为了保证这套教材的编审和出版质量,两会成立了教材编委会,制定了编写要求和编审程序。编委会对编者提出的编写大纲进行了讨论,其中一些关键性和难度较大的教材还进行了多次讨论。并且组织了部分编委对教材的质量和进度分片落实,有的教材在编审过程中召开了部分讲课教师座谈会,广泛听取意见。参加这套教材的编审者都是在该领域第一线从事教学和科研工作多年,学术水平较高,教学经验丰富,治学态度严谨的教师。这套教材的出版得到了电子工业出版社的积极支持。他们把这套教材列为出版社的重点图书出版,并制定了专门的编审出版暂行规定和出版流程,组织了专门的编辑和协调机构。

这套教材的编审出版凝聚了参加这套教材编审教师和关心这套教材的教师、参与编辑和出版工作者、以及编委会成员的汗水,他们为此作出了努力。

这套教材还得到电子工业部计算机专业教学指导委员会的支持,其中11本被选入1996~2000年全国工科电子类专业规划教材。

限于水平和经验,这套教材肯定还会有缺点和不足,希望使用教材的单位、教师和同学积极提出批评建议,共同为提高教学质量而努力。

中国计算机学会教育专业委员会
全国高等学校计算机教育研究会

教材编审委员会成员名单

前　言

程序设计语言是人与计算机进行通信的工具，也是软件开发的最重要工具。随着计算机科学技术的发展，计算机得到越来越广泛的应用，高级语言的发展对计算机的广泛应用起了重要的作用。计算机科学技术的发展与程序设计语言的发展有着密切的关系，因此，学习和研究高级程序设计语言是非常必要的。

程序设计语言这一领域，不仅仅是会用某种程序设计语言，还有语言设计和实现的理论和实践。中国计算机学会教育专业委员会公布的《计算机学科教学计划 1993》把它作为九个主科目之一，说明了它在计算机科学中的重要地位。

程序设计语言这一科目应包括程序设计语言的设计、编译和使用，以及它们相关的理论和实践，它的对象是程序设计语言的设计者、实现者和使用者。本书作为各类计算机专业的广口径教材，只讲述了程序设计语言的特性和主要实现技术，这些内容是完成计算机学士学位教育必不可少的。

本书的目标是，力求使各类计算机专业本科的学生，掌握设计和实现程序设计语言的最基本的理论和技术，从而提高鉴赏和评价程序设计语言（或语言设计方案）的能力；了解语言概念、功能（应用领域）和限制，以便选择一种恰当的语言为某一目的而使用的能力；具有设计一种语言或扩充现有语言的能力；实现一个语言的能力，以便他们能鉴赏、评价、选择、设计和实现程序设计语言。

全书共分九章，是四个学分的课程。第一、二、三章讨论传统程序设计语言的共性，特别是它们的数据类型和控制结构；第四、五、六章讲述基本的编译技术；第七章讲述运行时存储空间管理；第八章简单介绍函数式、逻辑式和对象式程序设计语言；最后在第九章简单介绍了形式语义学。

计算机软件专业的学生必须学完全书的内容，其他各类计算机专业本科的学生可以不学抽象数据类型、并发单元、第八章和第九章。适当剪裁，还可作为计算机专业大专的参考教材。

本书的第一、二、三、八和九章由电子科技大学龚天富编写，第四、五、六和七章由上海交通大学侯文永编写，龚天富还对全书进行了统稿。

本书由哈尔滨工业大学王义和教授、北京航空航天大学麦中凡教授、高仲义教授主审，电子科技大学江明德审阅了第八章和第九章，他们提出了许多宝贵的意见，在此表示衷心的感谢。特别要感谢北京航空航天大学杨文龙教授，他为本书的出版做了大量工作。

由于我们学识浅薄，编写时间仓促，谬误之处在所难免，恳切希望读者不吝赐教。

作者

一九九六年七月于成都

目　录

第一章 绪 论

第一节 引 言

语言是人们交流思想的工具。人类在长期的历史发展过程中,为了交流思想、表达感情和交换信息,逐步形成了语言。这类语言,如汉语和英语,通常称为自然语言(Natural Language)。另一方面,人们为了某种专门用途,创造出种种不同的语言,例如旗语和哑语,这类语言通常称为人工语言(Artificial Language)。专门用于计算机的各种人工语言称为程序设计语言(Programming Language),本书将讨论这类语言的特性和它的实现(Implementation)。

1946年出现了第一台电子数字计算机(Electronic Digital Computer),它一问世就成为人们强有力的计算工具。只要针对预定的任务(问题),告诉计算机"做什么"和"怎么做",计算机就可以自动地进行计算,对给定问题求解。为此,需要人们将有关的信息告诉计算机,同时也要求计算机将计算的结果告诉人们。这样,人与计算机之间就要进行通信(Communication)。既然要通信,就需要信息的载体。人们设计出词汇量少,语法简单和意义明确的语言作为这类载体,这样的载体通常称为程序设计语言。这类语言有别于人类在长期交往中形成的自然语言,是人直接设计创造的,故称人工语言。

每当设计出一种类型的计算机,随之孪生一种该机能够理解并能直接执行的程序设计语言,这种语言称为机器语言(Machine Language)。用机器语言编写的程序,是二进制代码,计算机可以直接执行。对人来说,用机器语言编写程序既难写,又难读。为了提高程序的可读性(Readability)和可写性(Writabilily),人们将机器语言符号化,于是产生了汇编语言(Assemble Language)。机器语言和汇编语言都是与机器有关的语言(Machine-dependent Language),通常称为低级语言(Low-level Language)。其他的程序设计语言都是与机器无关的语言(Machine-independent Language),称为高级语言(High-level Language)。由于机器只理解机器语言,执行机器语言编写的程序,因而由汇编语言和高级语言编写的程序机器不能直接执行,必须将它们翻译成功能完全等价的代码(机器语言程序),机器才能执行。这个翻译工作是自动进行的,由一个特殊的程序来完成。翻译汇编语言的程序称为汇编程序(Assembler),又称汇编器,翻译高级语言的程序称为编译程序(Compiler),又称编译器。编写一个高级语言的编译程序的工作,通常称为对这个语言的实现。

每种高级语言都有一个不大的字汇表(Vocabulary),一种显式定义的文法(Grammar),以及构造良好的语法(Syntax)规则和语义(Semantics)解释。这些基本属性的严格规定,便于由高级语言到低级语言的机器翻译。

高级语言较为接近数学语言和自然语言,它比较直观、自然和易于理解。用高级语言编写的程序易读、易写、易交流,以及易出版和存档。由于易理解,使程序员容易编出正确的程序,以便于验证程序的正确性,发现了错误也容易修改。因此,用高级语言开发软件的成本要比用低级语言的低得多。今天,绝大多数的软件都是用高级语言开发的,因此,高级语言是软件开发的最主要工具。

由于高级语言独立于机器，用高级语言编写的程序容易从一种机器搬到另一种机器，因而具有较好的可移植性(Portability)。

高级语言至今还没有完全取代机器语言，在一些地方还必须使用机器语言。例如，编译程序的目标程序，各种子程序库，以及实时应用系统中要求快速执行的代码段等。

本书讨论的对象是高级语言，利用抽象的方法讨论高级语言的一些共同性的概念和属性，讲述编译的重要技术。为了叙述的简洁方便，在不引起混淆的情况下，把高级语言简称为语言(Language)。

一种语言涉及设计者、实现者和使用者，有了设计者和实现者，才可能有使用者。本书既不是为语言的使用者编写的，也不是为语言的实现者编写的，本书的目标是，力求使大学各种计算机专业的本科学生，掌握程序设计语言的设计和实现的最基本的理论和技术，从而提高鉴赏和评价语言(或语言设计方案)的能力；能了解语言的重要概念、功能和限制，以便具有为某一目的选择一种恰当的语言的能力；设计一种新语言或扩充现有语言的能力；实现一个语言的能力。以便他们能鉴赏、评价、选择、设计和实现程序设计语言。

在计算机科学中，有些概念在各个科目中重复出现，ACM/IEEE-CS 的《计算教学计划1991》将它们称为重复出现的概念。重复出现的概念是重要的思想、原则和过程。由中国计算机学会教育专业委员会和全国高等学校计算机教育研究会推出的《计算机学科教学计划1993》也肯定了这些重复出现的概念。深入了解并适当使用这些概念的能力，被看成毕业生成为成熟的计算机科学家和工程师的标志之一。这些重复出现的概念共有十二个，它们是

绑定(Binding)
大问题复杂性(Complexity of Large Problems)
概念和形式模型(Conceptual and Format Models)
一致性和完备性(Consistency and Completeness)
效率(Efficiency)
演化(Evolution)
抽象层次(Levels of Abstraction)
按空间排序(Ordering in Space)
按时间排序(Ordering in Time)
重用(Reuse)
安全性(Security)
折衷和结论(Tradeoffs and Consequences)

本书在许多地方体现了这些重复出现的概念，读者应当注意这些概念，并学会理解和使用这些概念。

第二节　语言的定义

我们读一个程序的时候，总想了解程序的意义是什么，编译程序是如何编译这个程序的。这就要求对每种语言加以明确的定义，以便语言的使用者和实现者以及人和机器都能确定程序是否有效；对有效的程序，了解它的含义和作用是什么。

程序设计语言是用来推述计算机所执行的算法的形式表示，它由两个主要部分组成：语法和语义。语法是一组规则的集合，用以构造程序及其成分。任何实际有用的语言都具有无限

多个句子，要把这些句子一个个地枚举出来是不可能的，所以往往采用有限描述来对语言加以定义。常用的方法有两种：一种是生成的观点，即通过一组有限的规则（产生式）把有效的句子生成出来；另一种是识别的观点，即通过一组转换图（识别图）来识别合法的句子。

语义也是一组规则的集合，用以规定语法正确的程序或其成分的含义。常用描述语义的方法是通过映射每一语言结构到已知其含义的论域（Domain）来表示的。最早用自然语言来描述语言结构的含义，这种描述是非形式的、冗长的、易于引起二义的，但它能给出一个语言的直觉梗概。语义的形式描述是计算机科学的一个重要研究领域，目前已有指称语义学（Denotational Semantics）、操作语义学（Operational Semantics）、代数语义学（Algebraic Semantics）和公理语义学（Axiomatic Semantics）等多种描述方法（参见第九章）。

一、语法

每种语言都是建立在字汇表的基础上的。字汇表的元素通常称为字（Word），在形式领域中把它们称为（基本）符号（Symbol）。本书将采用符号这一术语。符号是由字母表（Alphabel）上的字符串（String）构成的，字母表是该语言允许使用的所有字符（Character）的集合，字符是字母表中的元素（Element）。符号是由字符组成的有限串，由词法规则（Lexical Rules）规定什么样的字符串可以构成语言的有效符号。例如，我们通常所说的“以字母打头的字母数字串”即为符号“标识符”的词法规则。

语言的特征是，有的符号序列被认为是正确的、符合形式规定的句子（Sentence），而另一些符号序列则认为是不正确的，不合格式的。用什么来确定一个符号序列是正确的还是不正确的句子呢？是用语法，或语法的结构来确定。实际上语法定义为一组规则（Rule）或公式（Formola），它确定（形式上正确的）句子的集合。更重要的是，这样一组规则不仅使我们能够决定一个符号序列是否为一个句子，而且还提供了句子的结构，这对于确定句子的含义是有帮助的。因此，语法和语义显然是紧密相关的。虽然结构的定义总是考虑从属于更高的目的——语义，但并不妨碍我们只从结构方面来进行讨论，而不管其意义和解释。

以英语为例，取一个句子“I study”，其中“I”是主语，“study”是谓语，可用下列规则来定义：

〈句子〉——→〈主语〉〈谓语〉　　（规则 1）

〈主语〉——→I | students　　（规则 2）

〈谓语〉——→study | run　　（规则 3）

上述三行说明下列意义：

（1）一个句子由主语后面跟一个谓语构成。

（2）主语由单词“I”或“students”构成。

（3）谓语由单词“study”或单词“run”构成。

上述三行中的每一行称为一条语法规则（Syntax Rule），其中“→”读为“是”或“被定义成”，而“|”读为“或”。描述这些规则的记号系统称为巴科斯-诺尔范式，即通常所说的 BNF（Backus-Naur Form），它由巴科斯发明，诺尔最先用于 ALGOL 60 的定义中。句子的构成部分：〈句子〉、〈主语〉和〈谓语〉称为非终结符（Nonterminal Symbol），或称语法范畴（Syntactic Category），非终结符总是用尖括号括起来的；单词 I students、study 和 run 称为终结符（Terminal Symbols）；规则称为产生式（Production）。用来描述别的语言的语言称为元语言（Metatanguage），BNF 是一个描述别的语言的元语言，它所使用的符号“→”、“<”、“>”和“|”称为元语言符号（Metalanlinguistic Symbol），这些符号不是被定义的语言的符号。非终结符中有一个特殊符号，称为开始符（Start

Symbol)，上述规则中〈句子〉就是开始符。在程序设计语言中，通常〈程序〉是开始符。

语法规则描述什么样的符号序列是合法的。在规则中，出现在"→"左边(通常称为规则左边)的符号总是非终结符，它被定义成"→"右边(规则右边)的非终结符和/或终结符的序列。规则右边出现的非终结符必须至少出现在某条规则的左边，并由这些规则定义该非终结符。语言的一个完整的语法描述称为文法，它具有一个终结符集(即字汇表，记为 T)，一个非终结符集(记为 N)，一个产生式集(即语法规则集，记为 P)和一个开始符(记为 S，$S \in N$)。文法可记为一个四元式(N，T，P，S)。

在形式语言中，上述的英语例子可写成文法 $G_1 = (N_1, T_1, P_1, S_1)$，其中

N_1 = {〈句子〉，〈主语〉，〈谓语〉}

T_1 = {I，students，study，run}

P_1 = {〈句子〉→〈主语〉〈谓语〉，〈主语〉→I | students，〈谓语〉——study | run}

S_1 〈句子〉

我们从开始符〈句子〉出发，重复使用规则右边来替换规则左边，可以推导(Derivation)出句子"I study"，推导步骤如下：

〈句子〉⇒〈主语〉〈谓语〉　　(使用规则 1)

⇒I 〈谓语〉　　(使用规则 2)

⇒I study　　(使用规则 3)

其中，符号"⇒"作为推导步骤的分隔符，它既不是元语言符号，也不是被定义语言的符号。

从文法的开始符出发，使用不同的规则，可以生成不同的句子，所有句子的集合，就是该文法所生成的语言。这就是以生成的观点来定义语言的。

上述文法是一个非常简单的英语文法，仅仅是示意的。若要使这个文法更强更实际，以便能描述全部英语句子，那就要增加许多规则。仅仅使用 BNF 来定义英语或其他自然语言是不可能的。实际上我们给出的英语文法只可能推出四个句子：

I run

students run

I study

students study

现在我们来讨论生成标识符的文法 G_2，它的规则如下：

〈标识符〉→〈字母〉

〈标识符〉→〈标识符〉〈字母〉

〈标识符〉→〈标识符〉〈数字〉

〈字母〉→A | B | … | X | Y | Z | a | b | … | x | y | z

〈数字〉→0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

在上述规则中，非终结符〈标识符〉出现在某些规则的左边和右边，这就构成了自己定义自己的问题，这种形式的定义称为递归定义(Recursive Definition)。正是这些递归定义，使我们能够以有限的规则来定义无限个程序结构。事实上，文法 G_2 允许我们推出无限多个任意长度的标识符。同样，表达式的文法可以定义为文法 G_3，它的规则如下：

〈表达式〉→〈标识符〉　　(规则 1)

〈表达式〉→(〈表达式〉)　　(规则 2)

〈表达式〉→〈表达式〉〈运算符〉〈表达式〉　　(规则 3)

〈运算符〉→ + | − | * | /

文法 G_3 允许我们推导出无限多个任意长度的表达式。其中规则 2 允许推出任意层次嵌套的表达式，而规则 3 允许组合任意多个表达式为新表达式。规则 1 没有递规的形式，从而可以达到推导的终止。

现在让我们从识别的观点来考虑语言的定义，即用所谓识别图(Recognition Graph)，或称语法图(Syntax Graph)来定义给定的语言。事实上，任何以产生式给定的文法，均可以构造一个等效的语法图，用来识别该文法所生成的合法的句子。语法图构造规则如下：

A. 每一非终结符 N 连同相应的诸产生式

$$N\rightarrow\alpha_1|\alpha_2|\cdots|\alpha_n|$$

映射到一个语法图 N，其结构由产生式右边按下面规则 B 至规则 F 决定。上述产生式中的 $\alpha_i(i=1,2,\cdots,n)$是终结符和/或非终结符的序列。

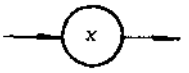

图 1-1

B. α_i 中每出现一个终结符 x，以圆框中的 x 来标记，如图 1-1 所示。

C. α_i 中每出现一个非终结符 N，以方框中的 N 来标记，如图 1-2 所示。

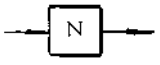

图 1-2

D. 形如

$$N\rightarrow\alpha_1|\alpha_2|\cdots|\alpha_n$$

的产生式，映射到图 1-3。其中，每个带方框的 α_i 是把构造规则 B 至 F 应用到 α_i 得到的。

E. 形如

$$\alpha\rightarrow\beta_1\beta_2\cdots\beta_m$$

的产生式映射到图 1-4。其中，每个带方框的 β_i 是把构造规则 B 至 F 应用到 β_i 得到的，每个 β_i 或者是终结符，或者是非终结符。

F. 形如

$$\alpha\rightarrow\alpha\beta|\gamma$$

的产生式映射到图 1-5。其中，α 是非终结符，β 和 γ 是终结符和/或非终结符序列。

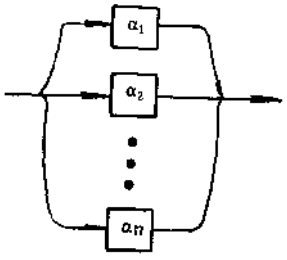

图 1-3

利用构造规则，上述标识符的文法 G_2 的语法图如图 1-6 所示，运算符的语法图如图 1-7 所示，表达式文法 G_3 的语法图如图 1-8 所示。将上述三图归并得到表达式的语法图如图 1-9 所示。

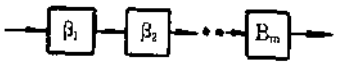

图 1-4

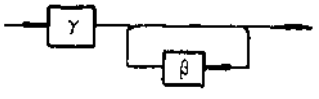

图 1-5

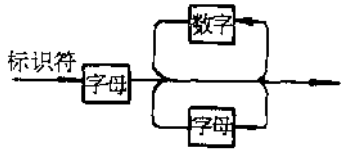

图 1-6

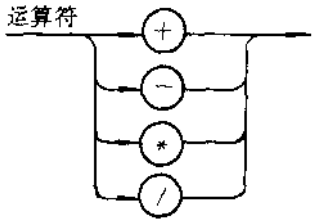

图 1-7

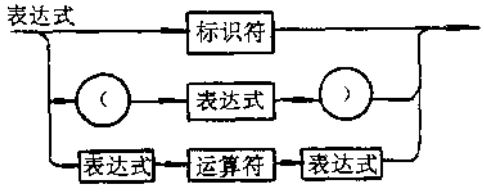

图 1-8

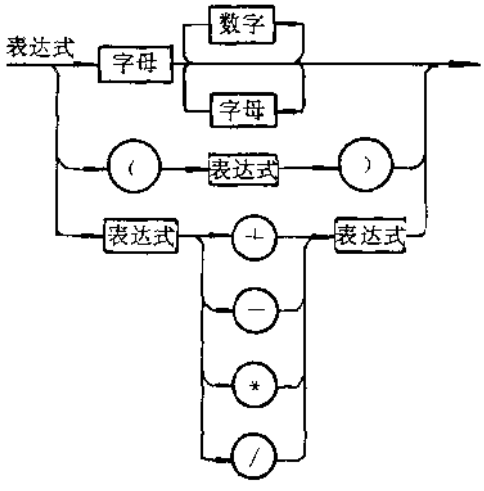

图 1-9

在语法图中,终结符用圆框标记,非终结符用方框标记。任何一个非终结符均可由仅有的一个入口边和一个出口边的语法图来定义。若一个终结符序列是合法的,那么必须从语法图的入口边通过语法图而到达出口边,且在通过的过程中,恰恰能识别该终结符序列。在通过标识终结符的圆框时,标记的终结符与被识别的终结符正好符合,则该终结符被识别;若通过标记为非终结符的方框,那么由通过该非终结符的语法图来识别;若遇到分支,可以经由任一边