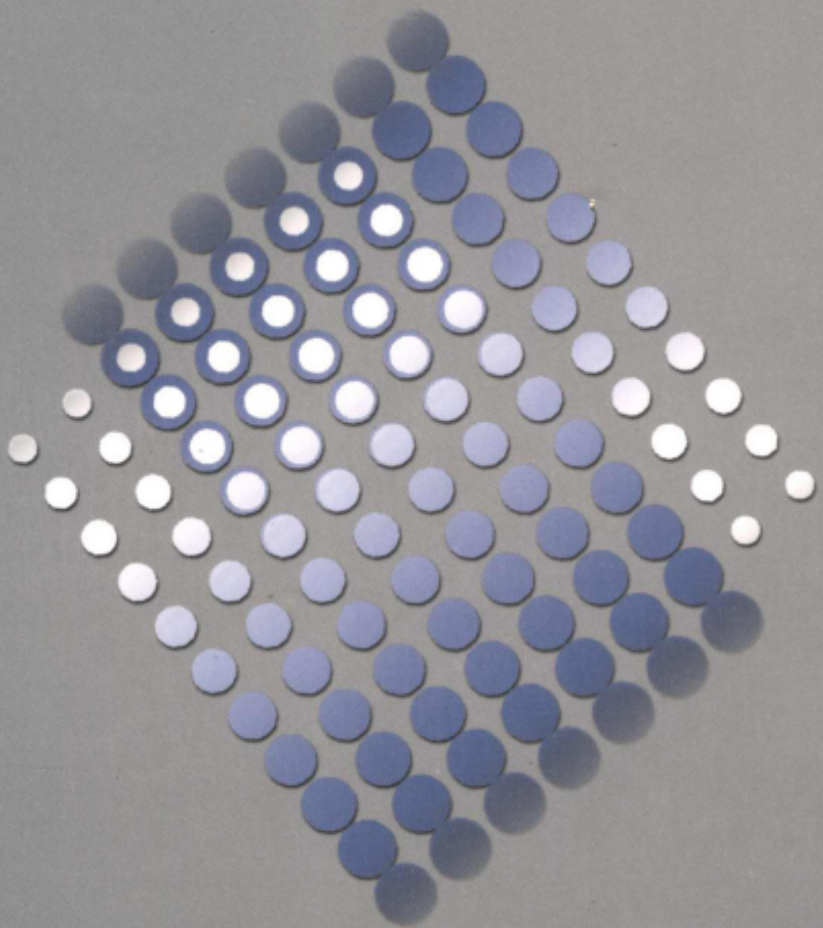
21世纪信息通信系列教材

编译程序设计原理与技术

BIANYI CHENGXU SHEJI YUANLI YU JISHU

李文生 编著



北京邮电大学出版社
www.buptpress.com

编译程序设计原理与技术

李文生 编著

北京邮电大学出版社
·北京·

内 容 简 介

在较多地参考了国内外权威人士著作的基础上，本书系统地介绍了编译程序设计的基本原理和技术。其主要内容包括词法分析、语法分析、类型检查、中间代码生成、代码生成和代码优化等。本书旨在培养学生发现问题、提出问题，进而分析和解决问题的能力。本书内容充实，图文并茂，各章节内容循序渐进，并注意理论与实践的结合，可作为高等院校计算机科学与技术专业的教材或参考书，也可供其他专业的学生或从事计算机工作的工程技术人员阅读参考。

图书在版编目(CIP)数据

编译程序设计原理与技术/李文生编著. —北京：北京邮电大学出版社，2002

ISBN 7-5635-0623-3

Ⅰ.编… Ⅱ.李… Ⅲ.编译程序—程序设计—高等学校—教材 Ⅳ.TP314

中国版本图书馆 CIP 数据核字(2002)第 058225 号

书　　名：编译程序设计原理与技术

作　　者：李文生

责任编辑：张学静

出 版 者：北京邮电大学出版社(北京市海淀区西土城路 10 号)邮编：100876

发行部电话：(010)62282185　62283578(传真)

经　　销：各地新华书店

印　　刷：北京源海印刷厂印刷

开　　本：787 mm × 1 092 mm　1/16

印　　张：21.25

字　　数：518 千字

印　　数：1—5 000 册

版　　次：2002 年 9 月第 1 版　2002 年 9 月第 1 次印刷

ISBN 7-5635-0623-3/TP·69　　定价：34.00 元

前　言

近年来，计算机科学得到了迅猛的发展，软件工程方法学也得到广泛的应用和发展，作为人机交互的程序设计语言也不断地推陈出新，软件工具相继涌现，相应地，程序设计语言的编译原理、方法和技术也在不断地发展和完善。本书主要以 Pascal 和 C 语言为背景，就编译程序的设计原理与技术有关的主要课题进行了较为深入的讨论。

全书共分 11 章。第 1 章介绍了有关形式语言与自动机的基本概念，这是学习本书必要的基础理论知识。第 2 章对编译程序的组成、功能及有关的前后处理器等进行了介绍，读者可以从中了解编译程序的概况，这是其余各章节的基础。第 3 章引入了词汇、模式等概念，介绍了利用状态转换图用手工方式实现词法分析器的方法和步骤，并对词法分析器自动生成工具 LEX 作了简单介绍。第 4 章详细讨论了常用的语法分析技术，如适于手工实现的递归下降分析方法、用于分析器生成器的完备的 LR 分析技术等，还介绍了语法分析器自动生成工具 YACC 等。第 5 章讨论了语法制导的翻译技术，引入了属性、语法制导定义和翻译方案等概念，并介绍了它们的实现方法，后继章节将应用这种技术描述语义和所要完成的翻译。第 6 章介绍了实现静态语义分析的思想，详细讨论了类型检查的有关问题。第 7 章讨论了支撑程序运行的环境中存储组织的有关问题。第 8 章介绍了中间语言，讨论了如何利用语法制导翻译技术把一般的程序设计语言结构翻译成中间代码。第 9 章介绍了目标代码生成的思想和一个简单的实现算法。第 10 章简单讨论了常用的代码优化技术。最后一章介绍了编译程序设计及实现的一般方法，并结合本书教学，提供了一个课程设计的题目，通过实际操作，可加深对编译程序设计原理的认识和理解。

本书作者从事本科“编译原理与技术”课程教学多年，在教学过程中参考了许多国内外同类教材，收集了同学们在学习过程中反馈的大量信息，在对多年的教案进行整理的基础上，结合本人丰富的教学经验完成了此书。在本书的编写过程中，徐晖同志根据他从事计算机软件开发工作多年的丰富经验，提

出了很多宝贵的建议，为本书得以和读者见面做出了很大贡献，同时还得到了李怀诚教授、马华东教授、刘辰副院长、林秀琴书记、罗婷老师等的帮助和大力支持，在此一并表示深切的感谢，同时也感谢在教学过程中提出宝贵意见和建议的同学们。

由于作者水平所限，书中难免存在缺点和不妥之处，真诚地希望得到广大读者和同行专家的批评指正。

李文生
北京邮电大学计算机科学与技术学院
2002 年 5 月

目　录

第1章　形式语言与自动机基础

本章简单介绍形式语言与自动机的基本知识，这些知识是学习其他章节内容的基础。

1.1　语言和文法

语言是人类社会生活中必不可少的交流工具，计算机的出现促进了语言学的研究，特别是形式语言学的研究。自从语言学家 Noam Chomsky 于 1956 年建立了形式语言的描述以来，形式语言学理论发展得很快。这种理论对计算机科学有着深刻的影响，对程序设计语言的设计和编译程序的构造有着重大的作用。程序设计语言是形式化的语言，是人与计算机之间相互传递信息的工具。这一节我们就语言和文法进行简单的讨论。

1.1.1　字母表和符号串

术语字母表指的是符号的非空有限集合。典型的符号是字母、数字、各种标点和运算符等。如集合{0,1}是二进制数的字母表，计算机使用的字母表常见的有 ASCII 字符集和 EBCDIC 字符集。

1. 符号串

定义在某一字母表上的符号串是由该字母表中的符号组成的有限符号序列。在语言理论中，术语“句子”和“字”常常作为术语“符号串”的同义词。

2. 符号串有关的几个概念

符号串 α 的长度指的是 α 中出现的符号的个数，记作 $|\alpha|$。如空串的长度为 0，常用 ε 表示，这是一个特殊的符号串；符号串“abc”的长度为 3。

符号串 α 的前缀是指从符号串 α 的末尾删除 0 个或多个符号后得到的符号串，如“uni”是“university”的一个前缀。

符号串 α 的后缀是指从符号串 α 的开头删除 0 个或多个符号后得到的符号串，如“sity”是“university”的一个后缀。

符号串 α 的子串是指删除了 α 的前缀和/或后缀后得到的符号串，如“ver”是“university”的一个子串，符号串 α 的前缀、后缀都是它的子串。

对任意的符号串 α，α 的自身、ε 都是串 α 的前缀、后缀，也是串 α 的子串。

符号串 α 的真前缀、真后缀、真子串是指，对任意的非空符号串 β，如果 β 是符号串 α 的前缀、后缀或子串，并且 $\beta \neq \alpha$，则称符号串 β 是符号串 α 的真前缀、真后缀、或真子串。

符号串 α 的子序列是指从符号串 α 中删除 0 个或多个符号（这些符号可以是不连续

的)后得到的符号串。如"uvrit"是符号串"university"的一个子序列。

符号串 α 和符号串 β 的连接 αβ 是把符号串 β 加在符号串 α 之后得到的符号串。如:若 α = ab,β = cd,则 αβ = abcd,βα = cdba。由于 ε 是不包含任何符号的空串,所以,对于任何符号串 α 来说,都有

$$\varepsilon\alpha = \alpha\varepsilon = \alpha$$

符号串 α 的幂,若 α 是符号串,α 的 n 次幂 α^n 定义为:

$$\underbrace{\alpha\alpha\cdots\alpha\alpha}_{n\text{个}\alpha}。$$

当 $n = 0$ 时,α^0 是空串 ε。假如 α = ab,则有:

$$\alpha^0 = \varepsilon$$

$$\alpha^1 = ab$$

$$\alpha^2 = abab$$

$$\cdots$$

$$\alpha^n = \underbrace{abab\cdots ab}_{n\text{个}ab}$$

1.1.2 语言

术语语言指的是在某一确定字母表上的符号串的集合。例如,按照语法构造出来的 Pascal 程序集合是定义在 Pascal 字符集上的语言,正确的英文句子的集合是定义在 26 个英文字母表上的语言。还有一些抽象的语言,如空集 ϕ,和只包含一个空串的集合 {ε} 也都是符合此定义的语言。注意这个定义并没有把任何意义赋予语言中的符号串。

1. 语言的运算

语言也可以进行运算。在此我们考虑语言的"并(union)"、"连接(connection)"和"闭包(closure)"运算。假设 L 和 M 表示两个语言,这些运算的定义如下:

语言 L 和 M 的并记作 L∪M:$L \cup M = \{s \mid s \in L \text{ 或 } s \in M\}$

语言 L 和 M 的连接记作 LM:$LM = \{st \mid s \in L \text{ 并且 } t \in M\}$

语言 L 的 Kleene 闭包记作 L^*:即 L^* 为 L 的 0 次或若干次连接。

$$L^* = \bigcup_{i=0}^{\infty} L^i = L^0 \cup L^1 \cup L^2 \cup L^3 \cup \cdots$$

语言 L 的正闭包记作 L^+:即 L^+ 为 L 的 1 次或若干次连接。

$$L^+ = \bigcup_{i=1}^{\infty} L^i = L^1 \cup L^2 \cup L^3 \cup L^4 \cup \cdots$$

还可以把幂运算推广到语言。定义:$L^0 = \{\varepsilon\}$,$L^n = L^{n-1}L$,于是 L^n 是语言 L 与其自身的 $n-1$ 次连接。

例如:令 $L = \{A, B, \cdots, Z, a, b, \cdots, z\}$,$D = \{0, 1, \cdots, 9\}$,可以从两个角度看 L 和 D:一是可以把 L 和 D 看作是字母表,则 L 是由全部的大写和小写英文字母组成的字母表,D 是由 10 个十进制数字组成的字母表;二是可以把 L 和 D 看作是语言,由于可以把符号看成是长度为 1 的符号串,所以 L 和 D 都是定义在确定的字母表上的符号串的集合,即 L 和 D 都是有限的语言。

应用上述关于语言的运算,L 和 D 可以生成新的语言,如表 1.1 所示。

表 1.1 语言运算举例

语 言	描 述
$L \cup M$	全部字母和数字的集合
LD	由一个字母后跟一个数字组成的所有符号串的集合
L^4	由 4 个字母组成的所有符号串的集合
L^*	由字母组成的所有符号串(包括 ε)的集合
$L(L \cup D)^*$	以字母开头,后跟字母、数字组成的所有符号串的集合
D^+	由一个或若干个数字组成的所有符号串的集合

1.1.3 文法及其形式定义

这里介绍有关文法的形式定义,文法的分类,以及书写约定。

1. 文法的形式定义

所谓文法就是描述语言的语法结构的形式规则。任何一个文法都可以表示为一个四元组 $G=(V_T,V_N,S,\varphi)$,其中:

V_T 是一个非空的有限集合,它的每个元素称为终结符号。

V_N 是一个非空的有限集合,它的每个元素称为非终结符号,并且 $V_T \cap V_N = \phi$,即 V_T 与 V_N 的交集为空。

S 是一个特殊的非终结符号,称为文法的开始符号。

φ 是一个非空的有限集合,它的每个元素称为产生式。

产生式的形式为:$\alpha \rightarrow \beta$,其中 α 是产生式的左部,β 是产生式的右部,"$\rightarrow$"表示"定义为"(或"由……组成"),并且 $\alpha、\beta \in (V_T \cup V_N)^*$,$\alpha \neq \varepsilon$,即 $\alpha、\beta$ 是由终结符号和非终结符号组成的符号串。

开始符号 S 至少必须在某个产生式的左部出现一次。

为了书写方便,对于若干个左部相同的产生式可以进行缩写,如:

$$\alpha \rightarrow \beta_1$$
$$\alpha \rightarrow \beta_2$$
$$\cdots\cdots$$
$$\alpha \rightarrow \beta_n$$

可以缩写为:

$$\alpha \rightarrow \beta_1 \mid \beta_2 \mid \cdots \mid \beta_n$$

其中"|"表示"或",每个 $\beta_i(i=1,2,\cdots,n)$称为 α 的一个候选式。

2. 文法的分类

1956 年,著名的语言学家 Noam Chomsky 首先对形式语言进行了描述,他把文法定义为四元组,并且根据对产生式施加的限制不同,定义了四类文法和相应的四种形式语言类。这四类文法就是 0 型文法、1 型文法、2 型文法和 3 型文法,从 0 型文法到 3 型文法,文法由强到弱,如表 1.2 所示。

在此,我们着重介绍上下文无关文法及相应的语言。

所谓上下文无关文法是这样一种文法,它所定义的语法单位(或称语法实体)是完全独立于这种语法单位可能出现的上下文环境的。对于现有的程序设计语言来说,许多语法单位的构造中具有固定的递归结构,这样的结构可以用上下文无关文法来描述。以后,如无特别说明,"文法"一词均指上下文无关文法。

例 1.1 有描述算术表达式的文法 G:

$G=(\{i,+,-,*,/,(,)\},\{\langle表达式\rangle,\langle项\rangle,\langle因子\rangle\},\langle表达式\rangle,\varphi)$

其中 φ:$\langle表达式\rangle\rightarrow\langle表达式\rangle+\langle项\rangle\mid\langle表达式\rangle-\langle项\rangle\mid\langle项\rangle$

$\langle项\rangle\rightarrow\langle项\rangle*\langle因子\rangle\mid\langle项\rangle/\langle因子\rangle\mid\langle因子\rangle$

$\langle因子\rangle\rightarrow(\langle表达式\rangle)\mid i$

如果用"::="代替"→",这组产生式可以写为:

$\langle表达式\rangle::=\langle表达式\rangle+\langle项\rangle\mid\langle表达式\rangle-\langle项\rangle\mid\langle项\rangle$

$\langle项\rangle::=\langle项\rangle*\langle因子\rangle\mid\langle项\rangle/\langle因子\rangle\mid\langle因子\rangle$

$\langle因子\rangle::=(\langle表达式\rangle)\mid i$

这种表示方法就是大家比较熟悉的 BNF(Backus-Normal Form)表示法,即所谓的巴科斯范式,它是由 Backus 为了在 Algol60 报告中描述 Algol 语言的语法首先提出使用的。这里元语言:

::=	表示"定义为"或"由……组成"
⟨……⟩	表示非终结符号
|	表示"或"

这个文法产生的上下文无关语言是所有包括加、减、乘、除四则运算的算术表达式的集合。

表 1.2 文法的类型及相应的语言类

文法类型	产生式形式的限制	文法产生的语言类
0 型文法	$\alpha\rightarrow\beta$ 其中 $\alpha,\beta\in(V_T\cup V_N)^*$,$\lvert\alpha\rvert\neq 0$	0 型语言
1 型文法,即 上下文有关文法	$\alpha\rightarrow\beta$ 其中 $\alpha,\beta\in(V_T\cup V_N)^*$,$\lvert\alpha\rvert\leqslant\lvert\beta\rvert$	1 型语言,即 上下文有关语言
2 型文法,即 上下文无关文法	$A\rightarrow\beta$ 其中 $A\in V_N,\beta\in(V_T\cup V_N)^*$	2 型语言,即 上下文无关语言
3 型文法,即 正规文法 (线性文法)	$A\rightarrow a$ 或 $A\rightarrow aB$(右线性),或 $A\rightarrow a$ 或 $A\rightarrow Ba$(左线性) 其中 $A,B\in V_N$, $a\in V_T\cup\{\epsilon\}$	3 型语言,即 正规语言

3. 文法书写约定

(1) 下面的符号常用作终结符号:

- 次序靠前的小写字母,如:a、b、c 等;
- 运算符号,如:+、-、*、/ 等;

- 各种标点符号，如：括号、逗号、冒号、等于号等；
- 数字 1、2、…、9；
- 黑体字符串，如：**id**、**begin**、**if**、**then** 等。

(2) 下面的符号常用作非终结符号：

- 次序靠前的大写字母，如：A、B、C 等；
- 大写字母 S 常用作文法的开始符号；
- 小写的斜体符号串，如：*expr*、*term*、*factor*、*stmt* 等。

(3) 次序靠后的大写字母，如：X、Y、Z 等，常用来表示文法符号，即终结符号或非终结符号。

(4) 次序靠后的小写字母，如：u、v、…、z 等，常用来表示终结符号串。

(5) 小写的希腊字母，如：α、β、γ、δ 等，常用来表示文法符号串。

(6) 通常，可以直接用产生式的集合代替四元组来描述文法，第一个产生式的左部符号是文法的开始符号。

1.1.4 推导和短语

这一小节，我们以上下文无关文法为例，介绍有关推导和短语的概念。

例 1.2 考虑简单算术表达式的文法 G：

$$G=(\{+,*,(,),i\},\{E,T,F\},E,\varphi)$$

其中：E 代表“表达式”、T 代表“项”、F 代表“因子”。

$$\begin{aligned}\varphi:\ & E\to E+T\mid T\\ & T\to T*F\mid F\\ & F\to(E)\mid i\end{aligned}\qquad\text{(文法 1.1)}$$

从文法的开始符号出发，反复连续使用产生式对非终结符号进行替换和展开，就可以得到该文法定义的语言。例如对于文法 1.1，从开始符号 E 出发，进行一系列的替换和展开，可以推导出种种不同的算术表达式，该文法能够推导出的所有表达式的集合就是该文法所定义的语言。

1. 推导

假定 $A\to\gamma$ 是一个产生式，α 和 β 是任意的文法符号串，则有

$$\alpha A\beta\Rightarrow\alpha\gamma\beta$$

其中“$\Rightarrow$”表示“一步推导”，即利用产生式对左边符号串中的一个非终结符号进行替换，得到右边的符号串。我们称 αAβ 直接推导出 αγβ，也可以说 αγβ 是 αAβ 的直接推导，或说 αγβ 直接归约到 αAβ。

如果有直接推导序列：

$$\alpha_1\Rightarrow\alpha_2\Rightarrow\cdots\Rightarrow\alpha_n$$

则说 α_1 推导出 α_n，记作：$\alpha_1\overset{*}{\Rightarrow}\alpha_n$，我们称这个序列是一个从 α_1 到 α_n 的长度为 n 的推导。其中“$\overset{*}{\Rightarrow}$”表示 0 步或多步推导。

对于文法 1.1，表 1.3 描述了从文法开始符号 E 推导出符号串 i+i 的详细过程。