编译程序构造

徐国定　杨宗源　编

华东师范大学出版社

编译程序构造

徐国定　杨宗源　编

华东师范大学出版社

编译程序构造

徐国定　杨宗源　编

华东师范大学出版社出版

（上海中山北路3663号）

新华书店上海发行所发行　　吴县光福印刷厂印刷

开本：850×1168　1/32　印张：12 字数：310 千字

1989 年10月第一版　　1989 年10月第一次印刷

印数：001—5,000 本

ISBN7-5617-0423-2/N·016　　定价：　3.30 元

前　言

编译程序是计算机系统配置的基本系统软件，它在整个计算机的系统软件中占有十分重要的地位。编译程序的作用是把用户用高级语言所书写的源程序翻译成用低级语言所书写的目标程序，从而为计算机执行用户所描述的源程序作必不可少的准备。

本书以作者近几年来在华东师范大学计算机科学系讲授此课程编写的教材为基础，经过修改整理而成。在编写过程中，我们力求理论联系实际，为学生日后研读有关文献资料打下基础。由于编译程序所述及的内容相当广泛，我们只能选择其中最基本的内容作必要的叙述。

全书共分十一章。第一章描述了编译程序的基本结构；第二章介绍形式语言的知识，是学习本书其余各章的基础；第三章涉及有限状态自动机、正则表达式、正则文法和词法分析程序的构造；第四章和第五章详细介绍了句法分析的常用方法；第六章以句法制导翻译为工具，讨论了表达式和程序设计语言常见控制结构的翻译方法；第七章介绍符号表构造；第八章详细讨论了运行时刻存贮的组织和管理；第九章介绍了一个实用的代码生成程序的构造；第十章扼要叙述了代码优化；第十一章讨论了出错恢复的常用方法。在本书的编写过程中，我们将参考的有关文献资料一并列于本书的末尾，在书中就不再一一列出了。讲授本书约需 80～90 学时。

本书的第十一章及第五章 §5.7 由杨宗源编写，其余均由徐国定编写。本书的编写还得到李玉茜等同志的帮助，我们的学生对本书初稿也曾提出过许多改进意见，在此表示衷心的感谢。由

于编者水平有限，书中如有不妥之处，恳请读者批评指正。

编　者

目　　录

第一章　概论 ……(1)
§1.1　程序设计语言的引入 ……(1)
§1.2　编译程序概貌 ……(3)
第二章　形式语言基础 ……(8)
§2.1　文法的形式定义 ……(8)
§2.2　推导树 ……(13)
§2.3　关系及其运算 ……(21)
§2.4　文法的分类 ……(30)
第三章　有限状态自动机 ……(36)
§3.1　有限状态自动机概念 ……(36)
§3.2　有限状态自动机和3型文法 ……(48)
§3.3　有限状态自动机和正则表达式 ……(52)
§3.4　有限状态自动机的最小化 ……(61)
§3.5　有限状态自动机的实现 ……(69)
§3.6　有限状态自动机的应用 ……(74)
第四章　句法分析(一) ……(84)
§4.1　下推自动机 ……(84)
§4.2　LL(K)文法 ……(93)
§4.3　LL(1)文法的句法分析 ……(98)
§4.4　产生式选择集合的计算 ……(111)
第五章　句法分析(二) ……(122)
§5.1　引言 ……(122)
§5.2　优先关系和简单优先文法 ……(125)
§5.3　弱优先文法和简单混合策略优先文法 ……(143)
§5.4　运算符优先文法和优先函数 ……(150)

§5.5 LR(0)文法 ……(165)
§5.6 SLR(1)文法和LALR(1)文法 ……(181)
§5.7 LR 句法分析控制表的安排 ……(198)
第六章 句法制导翻译法 ……(208)
§6.1 中间代码 ……(208)
§6.2 句法制导翻译文法 ……(217)
§6.3 表达式的翻译 ……(229)
§6.4 顺序控制结构的翻译 ……(234)
第七章 符号表 ……(243)
§7.1 符号表概况 ……(243)
§7.2 符号表的数据结构 ……(247)
§7.3 分程序结构语言的符号表 ……(253)
第八章 运行时刻存贮管理和环境的建立 ……(261)
§8.1 引言 ……(261)
§8.2 存贮管理 ……(262)
§8.3 分程序结构语言的非局部量的访问 ……(266)
§8.4 数组的存贮分配 ……(278)
§8.5 形式参数和实在参数的通讯 ……(283)
§8.6 过程的调用和返回 ……(291)
第九章 代码生成 ……(299)
§9.1 引言 ……(299)
§9.2 寄存器的分配 ……(302)
§9.3 临时变量的存贮分配 ……(307)
§9.4 简单算术表达式的代码生成 ……(312)
§9.5 代码生成的进一步讨论 ……(318)
第十章 代码优化 ……(329)
§10.1 引言 ……(329)
§10.2 表达式树的代码优化 ……(333)
§10.3 基本块的代码优化 ……(339)

§10.4 数据流分析 ……(346)
第十一章 出错恢复 ……(355)
§11.1 引言 ……(355)
§11.2 词法错误的出错恢复 ……(358)
§11.3 运算符优先分析法的出错恢复 ……(359)
§11.4 LR 和 LL 句法分析法的出错恢复 ……(362)
参考文献 ……(367)

第一章 概 论

§1·1 程序设计语言的引入

世界上第一台电子计算机问世至今已有几十年的历史了。在这几十年中，计算机技术有了迅速的发展，电子计算机已被应用到越来越多的领域中去。如今计算机技术的发展水平已成为衡量国家科学技术水平的重要标志。

尽管现代计算机运算速度是人类第一台计算机所不可比拟的，但当前计算机硬件仍然只能理解机器自己的语言——机器指令。计算机的机器指令相当原始，它通过电子线路对高速寄存器和低速内存中值为0和1的位进行操作。从这一点上说，现代计算机和当年第一台电子计算机没有什么本质上的不同。然而，用机器语言进行程序设计，需要对计算机结构有深入的了解。因为人和计算机之间进行信息交流时存在着巨大的鸿沟。人们用自然语言或者数学概念进行思维，用相应符号表达自己的思想，而计算机则依靠存放在存储器中"位"的值自动进行工作。今天，我们依靠许多"程序设计语言"来填补这一人和电子计算机之间存在的鸿沟。人们用结构严密的程序设计语言来表达自己的思想，也就是说，用程序设计语言来进行程序设计。

人们用来和计算机交换信息的语言可以分成汇编语言和高级语言两大部分。汇编语言是一种和计算机的机器语言十分接近的语言；所不同的是，用户可以不必用数字来表示机器指令的操作码和操作数的地址，用户用有助于记忆的符号来表示机器指令的操作码和操作数。例如，用ADD表示加法，SUB表示减法，等等。由于这些符号的含义和功能十分接近，因此用户很容易记住。在

这种语言中，操作数的地址也可以用符号来表示。这样，用户可以比较方便地表达自己的思想，写出的程序也一目了然了。汇编语言的书写格式在很大程度上取决于特定计算机的机器指令，这是一种低级语言。人们在使用低级语言和计算机"对话"时，发觉低级语言仍然未能摆脱机器指令的束缚，这对于人们抽象思维和学术思想的交流十分不便。在这个基础上，高级语言的建立就提到日程上来了。目前已有上百种高级语言在流行，如 FORTRAN, PASCAL, C 等等。这类语言并不依赖于特定计算机的指令组，它们和人们的自然语言或术语十分接近。这就大大提高了人们进行程序设计的效率，也便于人们用这类语言进行学术上的交流。

既然今天的计算机仍然只能理解执行机器自己的语言，而各种各样的程序设计语言只能是人和机器之间进行信息交流的"媒介"，程序设计语言的引入向我们提出了新的问题：必须有一个用机器语言书写的程序，其任务是使机器能够理解用某一程序设计语言书写的用户程序。负担这一工作的程序一般称为"语言处理程序"。语言处理程序可以分为两大类，解释程序和翻译程序。

解释程序是这样一种程序，它接受所输入的用程序设计语言(称为源语言)书写的程序(称为源程序)，然后直接执行源程序或源程序的内部形式。翻译程序则是这样一种程序，它接受所输入的用程序设计语言书写的程序，然后，将它翻译成用另一种语言(称为目标语言)书写的与源程序等价的程序(称为目标程序)。如果源程序是汇编语言，而目标语言是机器语言，则一般将这种翻译程序称为汇编程序。如果源语言是高级语言，而目标语言是低级语言(汇编语言或机器语言)，则这种翻译程序一般称为编译程序。这样，解释程序和编译程序的区别在于，解释程序并不产生目标程序，它直接执行源程序本身或源程序的内部形式。用高级语言书写的程序的执行工作一般分成几步完成：源程序首先被编译成目标程序。如果目标程序是用汇编语言表示的，则还需经过汇编阶段转换成用机器语言表示的目标程序，才能最后将目标程序装入

并执行。

程序设计语言既然是一种语言，我们就会问：某一程序设计语言可以使用哪些单词？这些单词是如何组成合法的程序的？程序的含义如何？等等。这些问题所涉及的一个中心问题是如何规定程序设计语言。人们在这些方面做了大量工作。由于编译程序的设计是建立在形式语言和自动机理论的基础之上，我们将以此为线索来学习编译程序的设计方法。

§1.2 编译程序概貌

编译程序的职能是将用某程序设计语言书写的源程序翻译成与之等价的一串机器指令或汇编指令。这里所说的等价是指目标指令执行了用户程序预定的任务。编译程序的这一项工作实际上相当复杂。为了更好地理解编译程序的构造，我们一般把编译程序分成下面几个部分：

1．词法分析程序；2．句法和语义分析程序；

3．代码优化程序；4．代码生成程序。

各部分之间关系如图 1.1 所示。

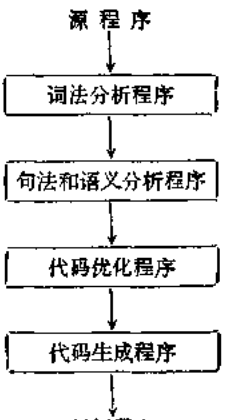

图 1.1

词法分析程序是编译程序的第一部分，它的输入就是由源程序中字符所组成的一串符号。词法分析程序将源程序的这种外部形式转换成更适合编译程序其余几个部分处理的内部形式。具体说来，词法分析程序具有如下功能：

(1) 识别程序中意义独立的最小词法单位—单词；

(2) 删除无用的空格、回车和其它与输入介质有关的的符号；

(3) 删除程序员为了提高程序可读性所加的注解；

(4) 如果发现错误的话，报告出错。

观察下述输入符号串：

$$\text{IF B1} = 25 \text{ THEN GOTO } L_2$$

在程序设计语言中，关键字(如 BEGIN、IF、WHILE 等)都各自有独立的含义。用作变量、过程和标号名的标识符是一个整体，代表数值常数的符号串也是一个整体。经过词法分析程序处理后，上述符号串被分解成 7 个单词：

(关键字, IF)，　(标识符, B1)，

(特殊符, =)，　(常量, 25)，

(关键字, THEN)，　(关键字, GOTO)

(标识符, L_2)

一般说来，经过词法分析程序处理，每个单词包含两个基本信息：单词的类别和单词的值。单词类别指出该单词属于哪一类，而单词值则把这一个单词和同类中其它单词区分开来。一个程序设计语言的单词究竟分成几类，由编译程序设计者决定。例如，我们在上面把单词分成四大类：标识符，关键字，常量和特殊符号。这样，IF 就属于关键字，其值则指出它是 IF 这一关键字。B1 是标识符，其值可能是该标识符在标识符表的入口地址。25属于常量，其值可能是此常量 25 在常量表的入口地址，等等。

由于经过词法分析程序处理后，每个单词都是一个意义独立的单位，其所含的信息量个数固定，因此输入的源程序就转换成长度固定、分类明确的单词串。这种大小固定，意义明确的单词串对

于下面句法和语义分析程序的处理是十分有益的。

前面已经说过，程序设计语言是结构严格的语言。它有一组有规定书写格式的文法规则。句法和语义分析程序接受词法分析程序的输出，即单词串，然后检查其是否符合语言的文法规则。一旦句法分析程序分解出其中一个文法结构，该结构的语义分析程序就进行相应的语义检查，如果需要的话，就输出相应的内部代码（称为中间代码）。这种内部代码可以理解成假想计算机的指令，其执行次序反映了用户程序的原始含义。例如，当句法和语义分析程序处理赋值语句

$$A := B + 3*6$$

的相应单词串时，它就会依次检查各运算符两边的运算对象类型是否相同。这是由于每一个运算符只能以特定的数据类型中的元素为其运算对象。如果运算对象不是所预期的类型，它就会报告出错。否则的话，它输出相应的内部代码。下面是此赋值语句的一种内部代码的表示形式：

$$(*, C_3, C_6, T_1)$$
$$(+, i_B, T_1, T_2)$$
$$(:=, T_2, i_A)$$

其中，T_1、T_2 是编译程序所生成的临时变量。这样的词法分析程序输出的7个单词经过句法和语义分析程序的加工变成三条意义明确的中间代码。这三条中间代码分别表示“将常量3和常量6的乘积送入临时变量 T_1”，“将变量B和临时变量 T_1 的和送入临时变量 T_2”以及“将临时变量 T_2 的值送入变量 A”。这一次序恰好反映原先输入的赋值语句的含义。句法和语义分析程序是编译程序中的关键部分。

中间代码可以直接由代码生成程序翻译成对应的机器指令（或汇编指令）。然而，我们亦可在句法语义分析程序和代码生成程序之间插入代码优化程序这一部分。代码优化程序对所输入的中间代码进行改动，将它修改成一种更有效的形式。常量合并就

是代码优化程序所作的工作之一，即在编译时刻计算原先在运行时刻所要进行的常量运算。上述赋值语句 A:=B+3*6 中，子表达式 3*6 可以由 18 来代替，A:=B+18。这样做完全不改变原赋值语句的含义。下面是代码优化程序所输出的中间代码：

$$(+, i_B, C_{18}, T_1)$$
$$(:=, T_1, i_A)。$$

最后，代码生成程序负责将所输入的中间代码串翻译成对应的机器指令。例如，对于上述未经优化的中间代码，代码生成程序可能生成：

```
LOAD    0, d3(3)
LOAD    1, d6(3)
MPY     0, 1
LOAD    2, dB(3)
ADD     2, 1
STORE   1, dA(3)
```

这里假定变量 A、B 都是整型变量，上述指令都是定点指令，上述指令组中 d_3、d_6、d_A 和 d_B 都是 3、6、A 和 B 的存贮单元地址相对于某一地址（假定该地址已在 3 号寄存器 r_3 中）的位移。对于上述经过优化处理的中间代码，代码生成程序只需生成

```
LOAD    0, d18(3)
LOAD    1, dB(3)
ADD     0, 1
STORE   1, dA(3)
```

很显然，无论从所占用的存贮空间和执行时间上来看，后者的目标程序都要比前者好得多。

编译程序在完成其任务过程中，还有一些其它工作需要做，包括符号表管理和出错恢复。编译程序中符号表登录了源程序中出现的每一个标识符及其属性。这样，在整个编译阶段，我们都可以了解某标识符的属性，如：标识符是如何说明的（如 REAL，ARR-

AY 等)，如果为数组，数组维数是多少，所需存贮单元数，所分配的内存单元的地址(或形式地址)，这些都是标识符的属性。出错恢复程序一般由句法和语义分析程序调用。一旦句法分析程序发觉源程序有错，无法继续其正常工作时，出错恢复程序就开始工作，其任务是诊断源程序错误性质，并做某种恢复工作。例如，删去或插入某些单词，以使句法分析程序可以继续工作。

第二章　形式语言基础

§2.1　文法的形式定义

我们对英文句子的分析都十分熟悉。英文句子由某字符集构造而成，该字符集中的元素包括字母、数字、空格和标点符号。这些字符首先组成英文单字，然后根据文法规则再组成英文句子。完全类似，计算机程序也是从语言的字符集构造而成的。这些字符先组成单词(意义独立的最小词法单位)，然后再进一步根据文法规则组成句子。与英文不同的是，程序设计语言的文法规则的结构相当严格。人们对此作了深入的研究，已得到许多有用的结果。

用程序设计语言书写的程序一般都是由一些基本符号组成的。下面是一些常用的基本符号：

字母：A, B, C, …, X, Y, Z

数字：0, 1, …, 9

其它符号：+, −, *, …, :, =

在这些基本符号基础之上，组成如 BEGIN、IF、ELSE 等关键字，程序员自己使用的标识符和常量，并且进一步组成用户的程序。

字母表是一个由字符所组成的有限集合。一个程序设计语言的字母表在该语言定义时确定。程序设计语言的字母表可以是键盘字符所组成集合的子集，每一个通常意义下的字母、数字和特殊符号都是字母表的字母。例如，FORTRAN 的字母表由 26 个英文字母、10 个数字及特殊符号"+ − *／·, ( ) = $/ :"共 48 个字符组成，这样 FORTRAN 程序就可以看成是一串由这些字符所组成的字符串。然而一般说来，程序设计语言的字母表中元素由若干个

字符组成。例如，在 ALGOL 语言中，字符":"和"="组成的赋值号为":=",程序员自己定义的标识符和常数、语言的关键字等等都是字母表中的元素。字母表一般用 V 或 Σ 来表示。

定义 2.1 字母表 V 上的符号串的递归定义如下：

1. 空串——一个不含有任何元素的符号串，是字母表 V 上的符号串。空串用记号"ϵ"表示。

2. 字母表 V 中任何一个元素 $a\epsilon V$，是字母表 V 上的符号串。

3. 设 x、y 是 V 上的两个符号串，则它们的并置——记以 xy，将符号串 y 的元素依次写在 x 后面所得的符号串，也是字母表 V 上的符号串。

符号串 x 中元素的个数称为符号串 x 的长度，记以 $|x|$。设 $z=xy$ 是一个符号串，则 x 是 z 的头(或称前缀)，而 y 是 z 的尾(或称后缀)。若 $|x|=1$，则称 x 是 z 的头符号。同样，若 $|y|=1$，则 y 是 z 的尾符号。注意，空串 ϵ 是任何一个符号串的头和尾。

例 2.1：设 $V=\{0, 1\}$，则

$$\epsilon, 0, 1, 00, 01, \cdots$$

是 V 上的字符串。若 $x=0110$，则 $|x|=4$，且

$$\epsilon, 0, 01, 011, 0110$$

都是 x 的头。而

$$0, 10, 110, 0110 \text{ 和 } \epsilon$$

都是 x 的尾。0 是 x 的头符号和尾符号。

定义 2.2 设 A 和 B 是两个由符号串组成的集合。符号串集合 A 和 B 的积 $A\cdot B$ 定义为

$$A\cdot B=\{xy|x\epsilon A, y\epsilon B\},$$

符号串集合 A 与 B 的和 $A\cup B$ 定义为

$$A\cup B=\{x|x\epsilon A \text{ 或 } x\epsilon B\}。$$

若规定 $V^0=\{\epsilon\}$，V^n 递归定义为 $V^n=V\cdot V^{n-1}$(n 为大于 0 的整数)，V 上所有符号串所组成的集合可以表示成下面诸集合的