编译方法

胡笔蕊　杜永建

[illegible]出版社

内 容 简 介

本书旨在介绍最基本的编译原理与方法，主要内容有：文法和形式语言、词法分析与有穷自动机、语法分析、中间语言、符号表、存贮分配、代码生成及优化等。为加强理论联系实际，本书以一个PASCAL语言书写的编译程序（与机型无关）作为附录，并以它为例穿插在各有关章节中。本书除配有习题外，还附有实习题供学生上机实习。

本书选材贯彻少而精，素材编排有利于教学，力求由浅入深、循序渐进、前呼后应、通俗易懂而又不失科学严谨性。

本书内容取舍灵活，可满足社会上各层次读者的需要，可作为工科院校计算机专业（包括本科、专科和成人高校）编译方法课程的教材或参考书。

编译方法
[illegible]杨永建
[illegible]
测绘出版社出版
武汉测绘科技大学印刷[illegible]印刷
新华书店[illegible]店科技发行所发行

开本787×1092 1/16·印张16.5·字数368千字
1988年11月第一版·1988年11月第一次印刷
印数1—5,000册·定价5.00元
ISBN 7—5030—0198—4/TP·1

前　言

本书是计算机专业“编译方法”课程的教材，编者根据计算机编审小组审定的教学大纲及学时数（60—80学时），并总结了多年教学实践经验，将原有讲义和讲稿整理而成。

在选材方面贯彻少而精的原则，既注意介绍些经典编译方法，也精选部分新编译技术，并尽量减少与机器有关的繁琐细节。基本概念讲透，力求使绝大部分学生能掌握最基本的编译原理和方法，同时还照顾到部分学生考研究生的需要。

在素材的编排上，考虑到教与学的方便，力求由浅入深，循序渐进，前呼后应，分散难点，通俗易懂而又不失科学严谨性。

为加强理论联系实际，使学生了解编译程序的概貌和培养其实际动手能力，以一个PASCAL语言书写的编译程序（与机型无关）作为附录，并以它为例穿插在各有关章节中。此外，每章均有习题，并配备了足够数量的实习题供学生上机实习用。

为适应当前计算机专业发展需要，本教材主要以PASCAL程序设计语言为对象介绍编译技术。

为满足当前社会上各层次读者的需要，内容取舍灵活。例如，第二章中“关系”部分，是为那些没有学过离散数学课程的成人高校学生而设置的，对一般院校学生可略去它。对于成人高校、大专班或学时数较少的学校，教学时可删去书中一些独立或较难的章节，而不影响整体。可供选用的章节计有：§2.8、§2.10、§3.3、§3.4、§4.4、§5.1、§5.5、§7.3中二和三、以及§10.3等等。

本教材由武汉测绘科技大学计算机系胡笔蕊、杜永建编写，全书共计十一章，最后两章由杜永建执笔，其余均由胡笔蕊执笔。大部分实习题和习题由武汉测绘科技大学夏启明配置。插图由武汉测绘科技大学王伟绘制。

北京工业大学副教授刘椿年博士在百忙中仔细审阅了全书，并提出了许多宝贵意见，在此谨致真诚的谢意。在出版过程中，得到了武汉测绘科技大学一些同志的大力帮助，也趁此机会一并表示感谢。

由于编者水平有限，书中难免存在一些缺点错误，请广大读者批评指正。

编者　1988.2

目　录

此为试读，需要完整PDF请访问：www.ertongbook.com

第一章 概 述

§1.1 编译程序

当前的电子计算机都是以系统的面貌出现的，它由硬件和软件组成。软件的功能与质量在很大程度上左右着整个计算机系统的功能。软件品种很多，对通用数字计算机来说，主要包括程序设计语言、系统软件与应用软件。操作系统、编译系统、诊断程序等等均属系统软件。

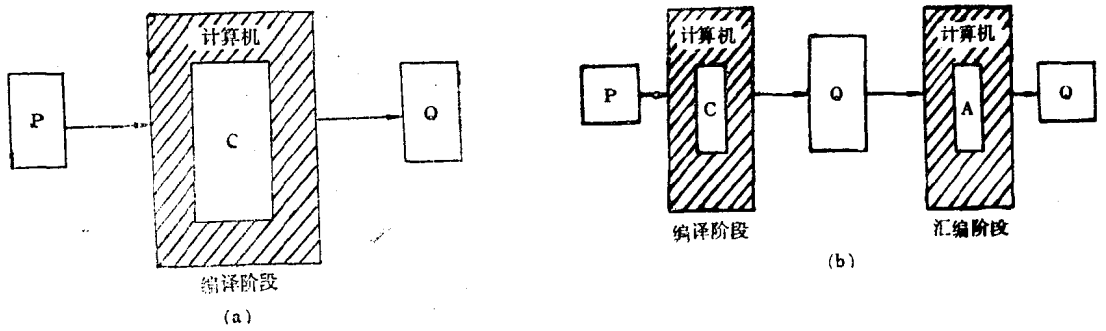
我们知道，程序设计语言是人类用来和计算机系统进行通信、并控制其工作的人工语言，它的品种繁多，就以高级程序设计语言来说，近二十多年来在我国曾广为流行或正在流行的有：ALGOL、FORTRAN、BASIC、PASCAL、COBOL、LISP、PROLOG、C语言以及数据库查询语言dBASE等等。这些语言应用范畴不尽相同，但有一个共同特点，即用它们编制程序比直接用机器语言编制可大大提高效率。可是迄今为止计算机主要是一种逻辑电子装置，它只能接受用二进制数表示的指令和数构成的程序。那么，它怎样接受源程序并完成计算呢？譬如，对于一个简单的赋值语句 y：=a+b*c，计算机怎样识别它，又怎样将它编制成能反映先乘后加的优先运算关系的一组机器指令，然后根据这组指令完成上述运算，并把结果保存在 y 单元中呢？这就有待于翻译。

通常分两种翻译方式：一种为“编译”方式；另一种为“解释”方式。所谓编译方式，是首先把源程序翻译成等价的目标程序，然后再执行此目标程序。而解释方式是边翻译边执行。它们之间的差别主要在于：解释程序不产生将被执行的目标程序，而是直接执行源程序本身。

如果源语言（编写源程序的语言）是ALGOL、PASCAL、FORTRAN、COBOL等这类高级语言，而目标语言是某计算机的汇编语言或机器语言，则这种翻译程序称为编译程序。

如果源语言是某一汇编语言，而目标语言是某计算机的机器语言，则称这种翻译程序为汇编程序。

以下给出按编译方式翻译的示意图（图1.1）。



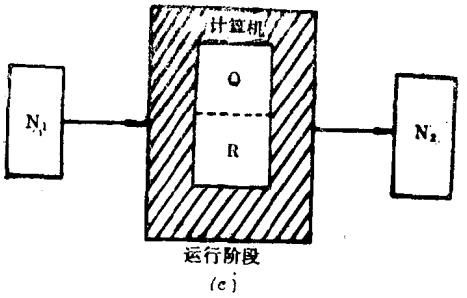


图 1.1

在图1.1中， P 表示源程序（用高级语言写的）；

Q′表示目标程序（由汇编语言构成的）；

Q 表示目标程序（由机器语言构成的）；

C 表示编译程序；

A 表示汇编程序；

R 表示运行（服务）程序；

N_1表示初始数据；

N_2表示计算结果。

本书将主要介绍编译程序。至于解释程序，只在下一节简略地作些介绍，当然，很多编译技术同样也适用于解释程序。

§1.2 解释程序

前面已经讲过，解释程序是这样一种翻译程序，它将源程序作为输入接受并执行之，即边解释边执行。或者说逐行进行翻译，即对每一个语句，依次实现“分析语句”、“检验错误”、“执行规定的操作”等步骤。图1.2是按这种形式翻译的示意图，其中I表示解释程序。

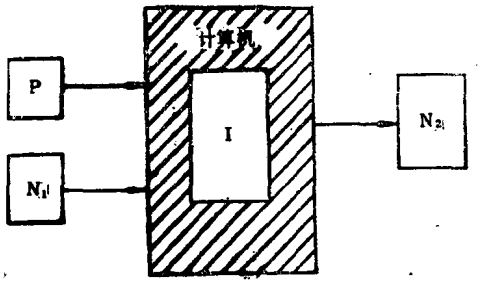


图 1.2

解释程序的主要优点之一是易于为用户提供调试功能。用户对运行中的程序始终有控制权。解释程序对源程序的语法分析及出错处理都很及时，修改调试都很方便。

由于小型通用语言BASIC是一种行结构、会话型语言，所以常采用解释方式进行翻译。以下是一个熟知的简单的BASIC源程序(排序)，我们来看看它在 IBM PC 机上解释运行的情况，从中可体会到解释方式的方便之处。

```
05  REM THIS IS A SORTING PROGRAM
10  INPUT M
15  DIM A(M)
20  FOR I=0  TO  M
25  INPUT  A(I)
30  NEXT  I
35  FOR  I=0  TO  M-1
40  FOR  J=I+1  TO  M
45  IF  A(I)>=A(J)  THEN  65
50  LET  K=A(J)
55  LET  A(J)=A(I)
60  LET  A(I)=K
65  NEXT  J
70  NEXT  I
75  FOR  I=0  TO M
80  PRINT A(I)
85  NEXT  I
90  END
```

当用户从终端输入完毕上述程序，并发布RUN运行命令后，解释程序便立即逐行进行翻译。在解释执行语句10时，它将向用户索取数据M（屏幕显示“？”），当用户应答（如13↵）后，便继续解释执行语句15。如果用户原先将语句15误输为：

```
15  DIM  A(M
```

解释程序此时将报告如下错误信息：

```
syntax error in 15
OK
15  DIM  A(M
```

用户可立即修改（移动光标，在M之后加“)”）。如果用户漏输了语句65（或70），则解释执行到语句35时，将再次报告出错信息：

```
FOR  without  NEXT  in  35
```

提醒用户，缺少与FOR匹配的NEXT。这时，用户可立即增设语句：

```
65  NEXT  J
```

如果数组A中的个别数据输入有误，如A(3)=-6.2被误为-2.6，用户可运用键盘运算

```
LET  A(3)=-6.2
```

改正它，解释程序将立即执行此键盘运算，然后用户可发布命令GOTO 35，重新排序。

由上可见，使用BASIC解释程序交互式地调试BASIC源程序是非常方便的。不仅BASIC语言如此，象LISP、PROLOG等语言，也常采用解释方式翻译其源程序，甚至象FORTRAN这样的高级语言，也有采用解释方式进行翻译的，如84年在郑州研制的多用户FORTRAN交互系统。目前国内在微型计算机上广为流行的dBASE-Ⅱ、Ⅲ大都为解释型的。

虽然用解释方式实现的交互系统，具有良好的会话性、分时性和独占性，但和编译程序相比，其执行效率低。因为解释程序（以BASIC为例）是逐行进行翻译，若此语句在一个循环体内（例如在一个FOR－NEXT循环中），每次执行此语句都必须重复上述翻译过程。此外，BASIC程序中的变量存放在一张变量表中，在运行该程序时，每当遇到一个变量，便要从头开始检索这张变量表。类似地，BASIC程序本身的存放与一张标号表相关，运行期间，每当遇到转向语句（例如 GOTO 或 GOSUB 语句），为了确定转向位置，也需要从头开始检索标号表。而编译程序是一次性翻译，即所有翻译工作均在运行之前实现，因此不会出现上述重复翻译以及检索变量表和标号表的现象。此外，对初次编译的源程序来说，编译程序可向用户报告它检测到的一切错误。由于上述种种原因，象ALGOL、FORTRAN、PASCAL 等这类高级程序设计语言，一般都采用编译的方式进行翻译。对于BASIC、PROLOG 、dBASE 等语言，既有解释型的也有编译型的翻译程序。或者给用户创造一个更强有力的编译环境，即让解释程序和编译程序相互补充。IBM/PC BASIC 编译程序就具有此优点，它被设计为能够支持大部分解释过的BASIC程序。因此，用户可以将BASIC解释程序作为调试程序的工具，然后再用BASIC编译程序编译原来那些程序以提高程序执行速度；用户也可以先使用 BASIC 编译程序对源程序进行初次编译，扫描出全部语法错误并纠正后，再用BASIC 解释程序测试源程序逻辑上的正确性。总之，用户可灵活运用此有力的编译环境。

§1.3 编译程序的组成

一. 编译程序的组成部分

为了完成源程序的翻译工作，编译程序要做的事情很多，所以它一般都比较庞大复杂。由于采用的编译方法和扫描（对源程序从头到尾扫视一次）遍数不同，同时支撑环境也往往不同，所以各种编译程序有着千差万别，但每一个编译程序一般都需要做以下五个方面的工作：词法分析、语法分析、语义分析、代码生成、代码优化等。完成这五个方面工作的程序分别称为：词法分析程序、语法分析程序、语义分析程序、代码生成程序和代码优化程序。这些程序便是编译程序的主要组成部分，在此分别对它们做些简要说明，以后各章节将分别给出更详尽的论述。

1．词法分析程序

词法分析程序也称扫描程序，它的主要任务是从构成源程序的字符串中识别出一个个具有独立意义的最小语法单位，即所谓单词（如BEGIN、IF、一个标识符、一个常数、一个运算符等），有时还指出单词的一些属性，为语法分析打下基础。

例如，以下是某源程序中由37个字符（含空格）组成的一个字符串。

IF a1＞0 THEN b1:=(c1＋3)＊d1 ELSE b2:=0

词法分析程序将从此字符串中依次识别出具有独立意义的18个单词：

IF	a1	＞	0	THEN	b1	:=	(	c1	＋
3	)	＊	d1	ELSE	b2	:=	0		

此为试读，需要完整PDF请访问：www.ertongbook.com

一般地，词法分析程序工作方式如图1.3所示。

由图 1.3 可见，由左到右从源程序中逐个读取字符并拼写成单词时，可以检查出书写中的词法错误，这项工作由出错处理程序完成。在进行词法分析时，有时可能要与一些表格发生联系，由表格处理程序完成这项工作。

2．语法分析程序

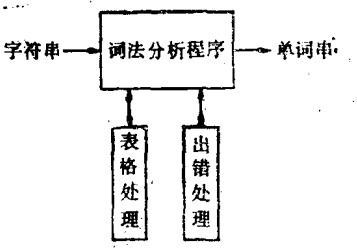


图 1.3

语法分析程序是编译程序的主要组成部分，它的主要任务是根据语言的语法规定进行语法分析，由单词符号串中识别出各种语法成分，判定当前正在编译何种说明和语句，并将控制转到相应的处理程序去工作。通过这种分析，确定整个单词符号串是否构成一个语法上正确的程序，若发现有不合语法规则的地方，语法分析程序便将出错性质、出错地点等信息通知用户，或对某些错误自动进行修正。

例如，语法分析程序将从如下单词符号串

$$w := (a1 + b1) * c1$$

中识别出这是一个〈赋值语句〉语法成份，而赋值等号右端为一个〈表达式〉语法成份。

又例如，对于单词符号串

FOR co : = 1 TO cy DO

se : = 3/(se + 1

语法分析程序不仅识别出此〈循环〉语法成份（其中还有〈赋值语句〉、〈表达式〉等语法成份），还将指出其中的语法错误（括号不匹配）。

语法分析过程中，将建立一些表格，也需要在一些表格中查找信息。此外，对于检查出的源程序中的语法错误，一方面向用户提供出错信息，另一方面对某些错误自动进行修正，使语法分析过程得以继续进行。上述工作分别由表格管理程序和出错处理程序完成，与词法分析部分相比，这些程序要复杂得多。

3．语义分析程序

语义分析程序由许多子程序组成，是编译程序具体处理各种说明和语句的加工程序。它是根据各种语法成份的语义编写的，是完成语义解释的具体体现。例如，对如下条件赋值语句：

w : = IF u THEN (a + b) * c ELSE (a − b) * c

加工程序将完成此语法成份的语义解释：根据布尔表达式 u 的值是“真”或“假”，决定把表达式 (a + b) * c 或 (a − b) * c 赋给变量w，同时，还要做些语义检查，如检查赋值语句中左部〈变量〉和右部〈表达式〉中的类型是否一致。

类似地，在语义分析过程中，需要建表查表；需要向用户报告语义错误信息，这些工作分别由表格管理程序和出错处理程序完成。

语法分析和语义分析是不同的概念，但在实际完成编译工作时，两者又是紧密相结合的。如前所述，当从某单词符号串中识别出相应语法成份并判断语法无误后，便调相应的语义分析程序进行处理。

4．代码生成程序

代码生成程序是将源程序翻译成汇编语言或机器语言的程序。例如，将赋值语句 y：=a+b*c 翻译为如下汇编语言程序段：

```
CLA  b        MUL  c
ADD  a        STO  y
```

一般地，在生成目标代码前，往往先将源程序加工成某种中间语言形式，如四元组。对于上述赋值语句，可先加工为如下四元组：

(*, b, c, T1)
(+, a, T1. T2)
(:=, T2, , y)

然后根据此四元组，生成目标代码。当然，在生成目标代码之前，还要进行许多准备工作，如对变量分配运行时的存贮单元等等。

5．代码优化程序

代码优化程序的任务是使编译程序能产生更为有效的目标程序，即尽量压缩目标程序运行时间和所占存贮空间。

因为编译程序需要根据某给定语言的语法规则，对不同的源程序进行统一的处理，因此较多地考虑了这些源程序的共性，不大可能照顾到它们各自的特点。因此，一般说来，不考虑优化工作的编译程序，它所产生的目标程序的质量往往较差。

二．编译程序的结构

由上段所述，编译程序一般要完成上述五个方面的工作，编译程序本身由相应的五个部分组成。图1.4是一种典型的编译程序工作过程。

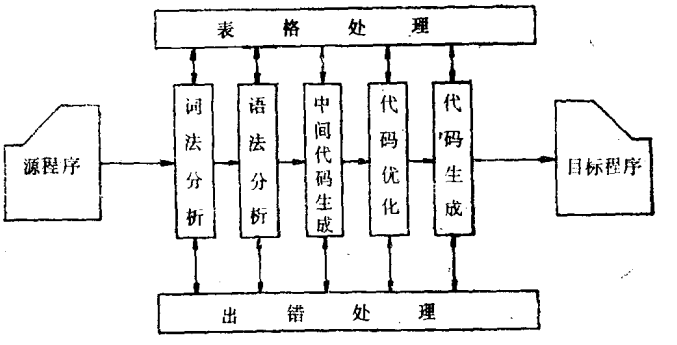


图 1.4

图 1.4 表示编译程序各部分之间的逻辑关系，而不一定是时间上的执行顺序。有些编译程序可能恰好按图1.4中的顺序执行这些逻辑过程(例如 IBM PC DOS PASCAL 编译程序)，另一些编译程序可能按平行、互锁方式执行它们，总之，采用哪种执行方式，将随语种、机型等等因素而不同。

有些编译程序采用多遍扫描方式，有些编译程序采用单遍扫描方式。所谓“遍”(pass)或“趟”，即处理数据的一个完整周期。在编译程序中，往往把编译的几个不同的阶段组合

成遍。就一遍来说，仅有一个入口（输入），一个出口（输出），入口即待加工的中间语言程序，出口即加工后的中间语言程序。多遍扫描的编译程序，其每一遍扫描的输入都是其前一遍扫描的输出。前述的IBM PC DOS PASCAL是一个多遍扫描的编译程序。70年代初在我国广为流行的TQ—16机ALGOL编译程序为三遍扫描。其中第一遍专为语法分析，一次便可检测出源程序中绝大部分语法错误，这种集中在一遍中检测出全部静态错误的方法，是很受用户欢迎的。此编译程序的第二遍扫描生成中间语言程序，并进行存贮分配；最后一遍扫描生成可供运行的目标程序。在单遍扫描的编译程序中，词法分析程序往往作为语法分析的一个子程序，当语法分析需要一个新的单词时，便调用词法分析程序。单遍扫描编译程序也有不少例子。本书将要介绍的PL/0语言（见附录）是一个很小的教学用语言，它便采用单遍扫描的编译方式。其编译程序的结构以语法分析程序为核心，词法分析程序和代码生成程序都作为独立的子程序。当语法分析需要单词时，便调用词法分析程序，而当经过分析确认语法正确需要产生相应目标程序时，便调用相应的代码生成程序。图1.5是PL/0编译程序的结构图，也可看作单遍扫描的编译程序的例子。

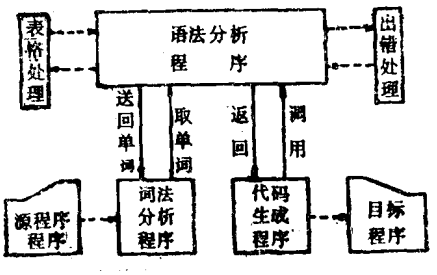


图 1.5

采用多遍扫描的方式，往往可以节省内存空间、提高目标程序的质量、缩短产生编译程序的周期等等，但也将使各遍扫描之间有一些重复性的工作，并延长源程序的编译时间。具体实现一个编译程序时，究竟采用几遍扫描的方式，要根据许多因素作出选择。这些因素包括：

• 源语言的结构（有些源语言本身隐含着至少要求分两遍扫描编译，如变量说明可放在使用之后，那么，对整个输入程序作出处理之前不可能生成代码）。

• 所选机型的可用内存的大小（将编译程序分为若干遍，因而可重复使用内存，达到节省内存的目的）。

• 设计目标的技术指标（如允许的编译速度、目标程序运行速度以及应具有的调试功能）。

• 参加研制编译程序人员的数量和素质。

至此，本节仅简略地介绍了编译程序的各组成部分，以后将分章较为详细地介绍其中每一部分。

§1.4 BNF范式和语法图

为使源程序能被正确翻译，产生等价的目标程序，源语言的使用者和实现者都应遵守关于源语言的共同约定。因此，每种程序设计语言都应有它自己的语法规则。使用者可查阅这些语法规则，以解决他书写程序的形式、结构等细节问题；实现者则根据这些语法规则，决定他的翻译程序需要完成的一些工作，以及确定允许用户输入程序的结构等问题。

巴科斯范式便是描述语法规则的一种表示方法。巴科斯范式也叫巴科斯—瑙尔范式，有时简称BNF（即Backus—Normal Form的缩写），它是由Backus为了在ALGOL－60报告中描述ALGOL首先提出的。采用这种形式体系的方式定义语法规则，可以用简洁的公式把各种

语法规则严格而清晰地描述出来。例如，要描述源语言的逻辑值，因逻辑值只有两个值，true 或 false，因此描述逻辑值的巴科斯范式为：

〈逻辑值〉::=true | false

其中，逻辑值用尖括号括起来，表示它是需要加以说明的一种语法上的概念（通常叫做语法成份或者非终结符号），“::=”可读作“可以是”，竖线“|”可读作“或者是”。上面整个式子可读作：逻辑值可以是 true 或者 false。

以上这种描述语法成份的方法，称为枚举的方法——把它的各种情况全部列出来。因为〈逻辑值〉只有 true 和 false两种情况，可以采用这种方法。对于另外一些语法成份，有时仅有这种方法就不行了。例如〈整数〉，因为它有无穷多种情况，就不能再用枚举的方法描述它了。整数是一个一个的数字构造出来的，它的构造方法可以用两句话来描述：

（1）单个的数字都是整数；

（2）整数再接上一个数字仍然是整数。

前一句话写成巴科斯范式就是：

〈整数〉::=〈数字〉

〈数字〉::=0|1|2|3|4|5|6|7|8|9

后一句话写成巴科斯范式就是：

〈整数〉::=〈整数〉〈数字〉

将以上式子合起来，便得到了关于〈整数〉的完整的描述了。

〈整数〉::=〈数字〉|〈整数〉〈数字〉

由第一个定义可知“一个数字是整数”，由第二个定义可知“两个或多个数字拼起来也是整数”。在上式中，〈整数〉在式子两边都出现，就是说，在说明〈整数〉这个概念时，又用到了它自己，这种自己说明自己的方法，叫做递归说明。使用递归说明的方法，可以描述许多复杂的语法现象。

大家熟知的〈标识符〉这种语法成份，它用巴科斯范式描述为：

〈标识符〉::=〈字母〉|〈标识符〉〈字母〉|〈标识符〉〈数字〉

而

〈字母〉::=A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z|

〈数字〉::=0|1|2|3|4|5|6|7|8|9

这样便刻划出了〈标识符〉是以字母开始的一列字母和数字的任意组合这种特点。

根据上述BNF范式定义，很容易判断一个字符串是否为〈标识符〉。例如，字符串 a4y 是一个〈标识符〉，而字符串 4ay 则不是。因为根据上述定义可以推导出 a4y，而无法推导出 4ay。以下是字符串 a4y 的推导过程：

〈标识符〉⟹〈标识符〉〈字母〉

⟹〈标识符〉y

⟹〈标识符〉〈数字〉y

⟹〈标识符〉4y

⟹〈字母〉4y

⟹a4y

除了用BNF范式描述源语言的语法规则以外，还可以采用语法图的方式。象PASCAL 语言的语法规则，常采用语法图定义，例如〈标识符〉和〈无符号数〉的语法图定义如图1.6所示。

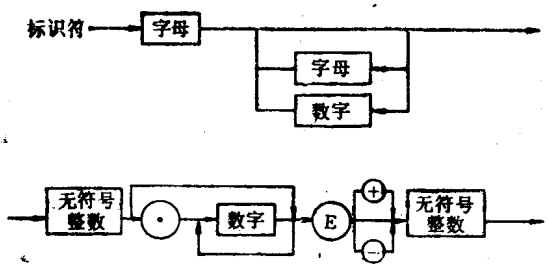


图 1.6

在以后各章节中，这两种语法定义都将出现，但主要还是采用BNF范式定义源语言的语法。

第二章　文法和形式语言简介

§2.1　引言

若要构造程序设计语言的编译程序，则首先要对程序设计语言本身有较为精确的描述。而关于程序设计语言的描述，将涉及语法、语义和语用三个方面。所谓语法，它涉及语言的构成规律，即程序的结构或形式；而语义是指语言所代表的含义；语用则涉及到实际应用。

例如对于赋值语句 y：=a+b*x，其非形式描述为：

·语法　赋值语句由一个变量，后随一个符号“：=”，再在后面跟一个表达式所构成。

·语义　赋值语句的执行过程是：先对该语句右部的表达式求值，然后把所得结果与语句左部变量相结合，并取代该变量原有的值。

·语用　赋值语句可用来计算和保存表达式的值（可能在程序中不止一处要用到这个值）。

以上这种非形式的描述，不够清晰和准确，因此，关于语言的形式化描述便成了一个很值得探讨的课题。所谓形式化方法，简单地说，便是用一整套带有严格规定的符号体系来描述问题的理论或方法。很早以来，便对语法形式化开展了许多工作并取得不少进展。而语义形式化的工作开展得晚一些。在早期的 ALOGL－60 程序设计语言中，第一次明确区分了语言的语法和语义，并用 BNF 范式（见§1.4）成功地实现了语法的形式描述。ALGOL－60 是通过“形式的语法说明和非形式的语义说明”定义的，实际上，把其中一些语义说明（如在一个分程序首部的标识符只能被说明一次）理解为语法规则更合理（即严格说来，语法的形式定义只是局部的）。

到目前为止，形式语法的规范比形式语义的规范要成功得多，但近二十多年来，形式语义的研究取得了巨大进展。语言的形式定义（形式语法和形式语义）已成为程序设计语言的必要组成部分，它对程序设计语言的设计、使用和实施都产生了深刻的影响。对于语用，是否应进行形式化，目前还不是很清楚。针对本书的任务，我们仅限于讨论语法和语义方面的问题，并着重以形式语法为基础，讨论程序设计语言的一些编译方法。

本章是全书的基础，将系统地介绍什么是程序设计语言的文法、文法和语言的关系、文法的分类以及语法树和二义性等等。这些都是以后各章节所需要的一些最基本的概念。作为上述内容的预备知识，本章还将简要地介绍一些基本概念：集合和关系（§2.2和§2.3）。如果读者已熟悉这方面的知识，便可略去阅读这两节。

§2.2　集　合

一．集合

此为试读，需要完整PDF请访问：www.ertongbook.com