编 译 原 理

贵州无线电工业学校

黄大胜 编

電子工業出版社

内 容 简 介

本书主要介绍了编译程序的基本概念、基本原理和基本方法。其内容包括：形式语言的基本概念，状态矩阵法的基本原理，词法分析，表达式和语句的编译，数组和过程的处理等。全书详细阐述了编译的全过程，并重点介绍了三种常用的编译方法，即优先数法、递归子程序法和状态矩阵法。叙述由浅入深，循序渐进，突出了中专教材注重实践的教学特点。本书既可作中等专业学校计算机专业的教材，也可供一般工程技术人员、管理干部等参考。

编 译 原 理

黄大胜 编

责任编辑：王惠民

电 子 工 业 出 版 社 出 版

（北京市万寿路）

*

中国铁道出版社印刷厂印刷

新华书店北京发行所发行　　各地新华书店经售

*

开本：787×1092　1/16　　印张：19.875　　字数：459 千字

1985年 6 月第 1 版　　1985年 6 月第 1 次印刷

印数：12000 册　　定价：3.70 元

统一书号：15290·97

目　录

第一章　绪　论……1
1.1　程序语言……1
1.2　编译程序的任务……4
1.3　编译程序的工作……6
习　题……8
第二章　形式语言的基本概念……9
2.1　语言、语法和语义……9
2.2　元语言和BNF 表 示……10
2.3　符号和符号串……12
2.4　文法、终结符和非终结符……15
2.5　文法推导句子……17
2.6　递归文法、短语和句柄……20
2.7　语法树……27
2.8　文法的二义性……33
2.9　句型的分析方法……35
2.10　文法的关系……37
习　题……41
第三章　状态矩阵法基本原理……43
3.1　状态的概念……43
3.2　状态矩阵的结构……45
3.3　状态栈和算法……48
3.4　编译方法……51
3.5　构造状态矩阵的方法……55
3.6　状态矩阵的存放……71
习　题……77
第四章　词法分析……79
4.1　源程序的输入……79
4.2　词法分析的任务……80
4.3　读字符子程序……83
4.4　直接分析的方法……85
4.5　无符号数的处理……89
4.6　整数词和语句标号的识别……96
4.7　专用定义符和标识符的处理……97
4.8　词法分析总结……108
习　题……112
第五章　表达式的处理……113
5.1　概　述……113

5.2 编译表达式的基本思想 ……116
5.3 实现方法 ……122
5.4 优先数法 ……129
5.5 递归子程序法 ……143
5.6 状态矩阵法 ……152
习 题 ……158
第六章 语句的处理 ……159
6.1 赋值语句的处理 ……159
6.2 控制语句的处理 ……164
6.3 循环语句的处理 ……180
6.4 其它语句的处理 ……189
习 题 ……194
第七章 编译程序小结 ……196
7.1 一趟扫描的编译程序 ……196
7.2 多趟扫描的编译程序 ……204
习 题 ……229
第八章 数组和过程的处理 ……231
8.1 数组的处理 ……231
8.2 过程的处理 ……245
8.3 扩充算术表达式的处理 ……268
8.4 读/写语句的处理 ……272
习 题 ……296
第九章 错误处理 ……300
9.1 错误处理综述 ……300
9.2 错误处理方法 ……302
附录Ⅰ ASCII字符码 ……307
附录Ⅱ DJS-130机指令系统 ……308

第一章 绪 论

1.1 程序语言

计算机科学是一门新兴的科学。现在，电子计算机已成为科学技术、经济、军事各个领域中广泛使用的一种强有力的工具，它的出现是二十世纪科学技术方面的一个重要成就。

自从第一台电子计算机ENIAC问世以来，电子计算机技术在各个方面都得到迅猛的发展。计算机的发展水平和应用程度，已成为衡量一个国家科学技术发展水平及其现代化程度的重要标志。

在计算机出现的初期，所谓计算机就是指机器的硬设备。为了使机器能用来解题，人们使用手编程序，即用机器语言来编写程序，而机器语言就是该机器所能接受的代码语言。手编程序大体要分两步进行：首先要分配工作单元，即对计算中需要的一些量和常数分别分配单元。第二步是编制程序。在编程前先要确定程序从内存哪个地址开始。这里，我们用DJS-130机进行说明。

例如，我们要计算一个分段整函数

$$M(I)=\begin{cases}1 & \text{当 } I>0\\ 0 & \text{当 } I=0\\ -1 & \text{当 } I<0\end{cases}$$

为了说明起见，编制程序一般要求画一个框图。解这个问题的程序框图如图1.1所示。

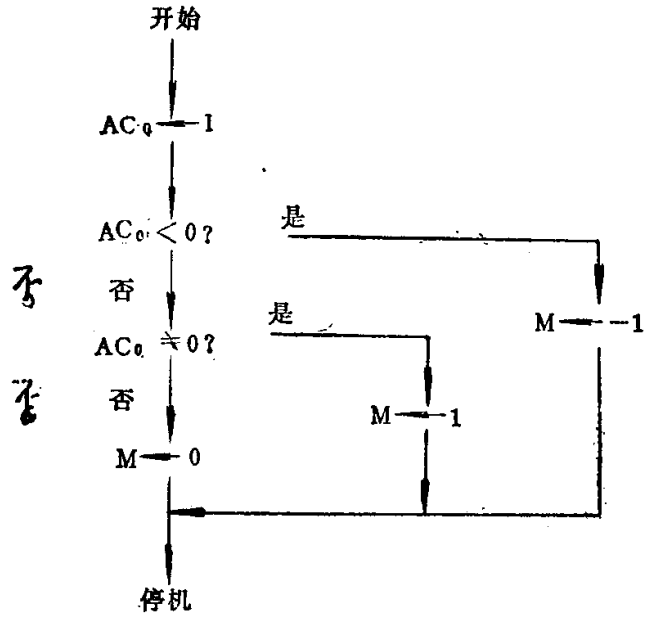

图1.1 例题的程序框图

第一步：分配工作单元。假设量I、M与数－1、0、1分配工作单元如下：

00010 分配给 －1
00011 分配给 0
00012 分配给 1
00013 分配给 I
00014 分配给 M

第二步：编制程序。设程序从单元00100开始编起（用箭头表示），根据框图编出程序（八进制代码）如下：

```
→00100  020013
 00101  101113
 00102  000404
 00103  101005
 00104  000407
 00105  000407
 00106  020010
```

```
00107   040014
00110   000406
00111   020011
00112   040014
00113   000403
00114   020012
00115   040014
00116   063077
```

程序从00100开始执行，如果00013单元（即分配给量I的单元）有值，那么到00116停机时，在00014单元（即分配给M的单元）中就有所需的值。

由此可以看到，直接用机器语言编制程序对内存的依赖性很强，不但常数和变量等需要预先分配单元，在程序中还要用相应的单元地址来引用常数和变量，而且当选定程序开始地址后，程序本身也强烈地依赖于内存地址，例如本例中的转移指令就强烈地依赖于内存地址。此外，如果程序中需要临时工作单元，这些临时工作单元也要分配地址。因此，用户要用机器解题，就要记住该机器的指令代码，同时还要做上述的工作，这不但十分繁杂琐碎而且极易出错。

由于手编程序有这些弱点，因此人们就对它进行改革，产生了符号汇编语言。为了消除程序对机器存储的物理地址的强烈依赖性，人们用符号名来代替各类地址（包括存放各种量的单元的地址和转移指令中所用的地址）。为了增加程序的可读性，采用了一些助记符代替原来指令的操作码。这样，前例中的程序可用汇编语言编写如下：

```
BEGIN: LDA     0，13
       MOVL*   0，0，SNC
       JMP     A
       MOV     0，0，SNR
       JMP     B
       JMP     C
A:     LDA     0，10
       STA     0，14
       JMP     H
B:     LDA     0，11
       STA     0，14
       JMP     H
C:     LDA     0，12
       STA     0，14
H:     HALT
```

程序从BEGIN开始执行，到H停机。

由例可见，在汇编语言程序里，原来出现在手编程序中的操作码和程序开始的物理地址及转移的物理地址都被一些符号所代替。

另外，对于量和数据也可以用符号来代替分配的地址，例如用

CONO表示00010单元，即存放数据－1的单元

CON1表示00011单元，即存放数据0的单元

CON2表示00012单元，即存放数据1的单元

I 表示00013单元，即存放量 I 的单元

M表示00014单元，即存放结果的量M的单元

于是程序可编制如下：

```
BEGIN: LDA      0, I
       MOVL*    0, 0, SNC
       JMP      A
       MOV      0, 0, SNR
       JMP      B
       JMP      C
    A: LDA      0, CONO
       STA      0, M
       JMP      H
    B: LDA      0, CON1
       STA      0, M
       JMP      H
    C: LDA      0, CON2
       STA      0, M
    H: HALT
```

本程序也可编制为：

```
BEGIN: LDA      0, I
       MOVL*    0, 0, SNC
       JMP·+4
       MOV      0, 0, SNR
       JMP·+5
       JMP·+7
       LDA      0, CONO
       STA      0, M
       JMP·+6
       LDA      0, CON1
       STA      0, M
       JMP·+3
       LDA      0, CON2
       STA      0, M
       HALT
```

汇编语言比手编程序前进了一大步，解决了对地址的依赖性，但汇编语言的指令基本上仍和机器指令一一对应，所以仍然依赖于具体的机器。例如，用DJS-130机的汇编语言编写的符号程序拿到DJS-183机上就不能使用，其原因就是机器的指令结构不同。

怎样才能完全甩掉对机器的依赖呢？人们在实践中创立了高级程序设计语言，如BASIC、ALGOL、FORTRAN、COBOL等等。这些语言的表达方式相当接近自然语言的描述，程序员在编程序时，完全用符号名来标识各种量，既不需要知道内存地址的概念，也不需要知道所使用的计算机的具体特性。

例如，用FORTRAN语言编制前述问题的程序（在这里我们省略了某些部分）可示意

如下：

```
      IF(I)10, 20, 30
10    M = - 1
      GOTO      40
20    M = 0
      GOTO      40
30    M = 1
40    STOP
```

由此可见，用高级程序设计语言编写的程序，不但简明清晰、可读性好，而且由于它与具体机器无关，有利于交流和移植。

1.2 编译程序的任务

程序语言从低级的机器语言逐步发展到高级的程序设计语言大大地促进了计算机科学的发展。

计算机只认识如“020013”、“101113”、“000402”等这样的机器语言，但不懂得用汇编语言或高级程序设计语言编写的程序。为了使程序设计语言能被机器接受，必须将以这样的语言书写的程序译成机器语言，为此就需要配制一个“翻译员”。

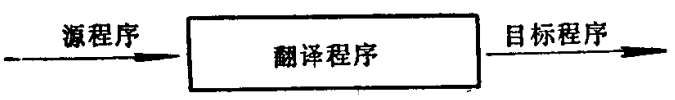

图1.2 翻译程序的作用

用符号语言编写的程序称为源程序。只有把源程序翻译成机器语言程序，计算机才能直接执行，这是“翻译员”要做的工作。由源程序翻译成的机器语言程序也称目标程序。“翻译员”也是一个用机器语言编写的程序，故叫做翻译程序。源程序经过翻译程序变成目标程序的过程如图1.2所示。

翻译程序对高级程序设计语言写的源程序进行检查、分析并把它翻译成机器语言程序。翻译程序大致可分为三类：汇编程序、解释程序和编译程序。这种分法主要是以它们所加工的程序语言为背景的。

1. 汇编程序

汇编程序是一种把汇编语言编成的源程序翻译成某计算机的机器语言的翻译程序。一般说来，符号汇编语言非常接近机器语言，大多数汇编语言是机器语言的符号化，这使得汇编程序能很容易地对它们进行分析加工和产生与之等价的机器语言。例如前例中

```
LDA   0, I              汇编成020013
MOVL* 0, 0, SNC         汇编成101113
JMP *+4                 汇编成000404
```

等等。粗略地说，我们把将助记符换成指令代码的工作称为“代真”，汇编程序的主要工作就是做“代真”。

2. 解释程序

解释程序也是一种翻译程序，每遇到源程序的一个语句，解释程序就将它翻译成机器语言并让机器执行，这种对源程序进行边解释边执行的方式便于实现人机对话。目前，BASIC

语言的翻译一般都是用解释程序来实现的。

3. 编译程序

编译程序是将高级程序设计语言写的源程序翻译成目标程序的程序。这种目标程序可以是机器指令程序，也可以是符号程序。例如把FORTRAN语言编写的源程序

```
IF(I)10, 20, 30
10    M = - 1
 GOTO  40
20    M = 0
 GOTO  40
30    M = 1
40    STOP
```

翻译成相应的代码程序（机器语言程序）

```
020013
101113
000404
101005
000405
000407
020010
040014
000406
020011
040014
000403
020012
040014
063077
```

或者翻译成符号程序

```
LDA   0, 13
MOVL* 0, 0, SNC
JMP.+ 4
MOV   0, 0, SNR
JMP.+ 5
JMP.+ 7
LDA   0, 10
STA   0, 14
JMP.+ 6
LDA   0, 11
STA   0, 14
JMP.+ 3
LDA   0, 12
STA   0, 14
HALT
```

但符号程序还要经过汇编程序“代真”成为代码程序，机器才能执行，即要进行两级翻译。

源程序是编译程序的输入，而目标程序是编译程序的输出。因此，编译程序的任务就是将源程序翻译成与之等价的机器能够直接执行的目标程序（见图1.3）。

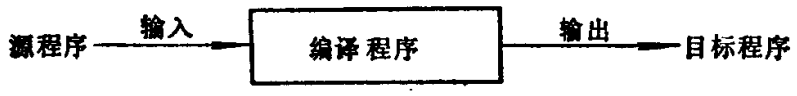

图 1.3 编译程序的任务

一批代码输入，经过一趟翻译而生成新的一批代码输出，这批新代码就叫做目标程序。编译程序可以采用一趟翻译或者若干趟翻译。一趟翻译就叫做一趟扫描，每趟扫描把上一趟扫描的结果作为输入，经过翻译生成新的代码输出（即目标程序）。如果编译程序采用多趟扫描的话，则本趟扫描把上一趟扫描生成的目标程序作为输入，而输出新的目标程序作为下一趟扫描输入，中间生成的目标程序机器也是不能直接执行的。

一般来说，目标程序还要加上必要的运行系统（例如一些服务性程序）才可对输入数据进行计算或处理，以得到所需的结果。这个过程示于图1.4。

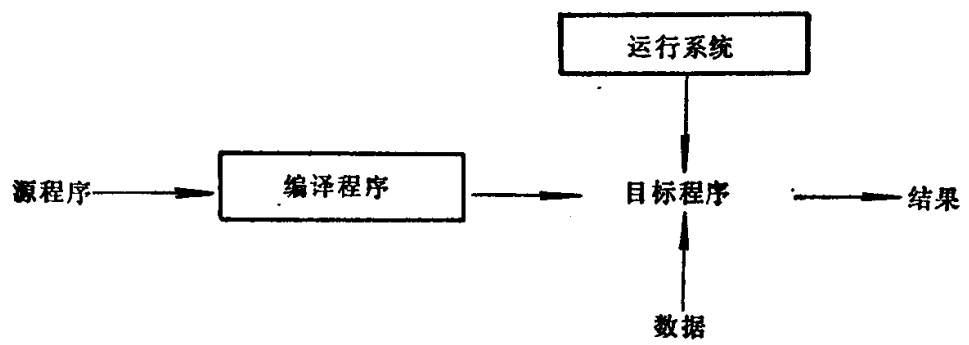

图 1.4 编译及运行

源程序翻译成等价的目标程序这一阶段称为编译阶段，而计算机执行目标程序的阶段称为运行阶段。在运行过程中，协助目标程序运行的子程序的集合称为运行系统。编译程序和运行系统综合在一起称为编译系统。

1.3 编译程序的工作

编译程序要把源程序翻译成机器能接受的目标程序，它该怎样做呢？既然是翻译工作，编译程序就要从前到后逐个扫描源程序的符号，在扫描过程中进行各种分析和加工处理。

编译程序的任务可用一趟扫描完成，也可以用多趟扫描完成。从源程序的第一个字符开始，逐个向后读，一直读到源程序的末尾，一边读一边完成一定的任务，这一过程就叫做一趟扫描。

只要扫描一趟即可把源程序翻译成最终可在机器上运行的目标程序的编译程序，称为一趟扫描编译程序（见图1.5）。

编译程序进行多趟扫描时，中间将产生几种中间语言（中间语言即目标语言，为区别于

最终的目标语言，我们称为中间语言）。前面扫描处理的输出作为后面扫描的输入，经若干趟扫描后才产生出可在机器上运行的目标程序的编译程序，称为多趟扫描的编译程序（见图1.6）。

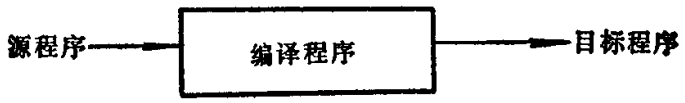

图1.5　一趟扫描编译程序

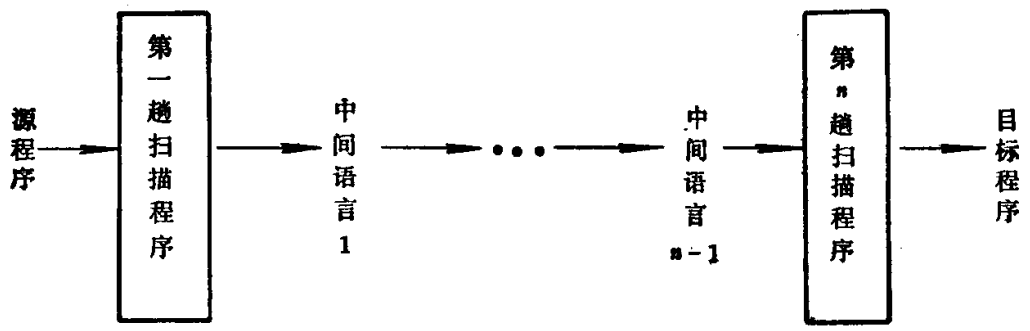

图1.6　多趟扫描编译程序

两种编译程序各有优缺点。一趟扫描编译程序的优点是编译速度快，编译程序的总长度短，但编译程序复杂；多趟扫描编译程序的优点是各趟所要完成的功能明确而单纯，编译程序较容易且便于分工，但编译程序的总长度长。

无论是一趟扫描或多趟扫描，一般来说，编译程序大致可分为词法分析、语法分析及代码生成等几个部分，下面粗略地介绍编译程序的这几个部分及其作用。

1. 词法分析

词法分析的主要功能是把一个一个字符拼成有意义的语法单位（单词）。换句话说，就是把字符拼成单词。它的作法是对源程序的字符从左到右进行扫描，在拼成单词的同时也进行词法检查，一旦发现不符合词法规定之处，机器就打印出错信息。只有做了必要的修改后，才能继续进行词法分析，直到结束为止。

2. 语法分析

语法分析的主要功能是把单词拼成能够表达一个完整语义的句法单位（即句子）。在这一过程中，一旦发现源程序中有不适合语法要求的地方，就打印出错信息。只有做了必要的修改之后，这种分析过程才能继续进行。一直到全部源程序都分析完时才能进行下一步的工作。

3. 语义分析及代码生成

完成上述两个步骤之后，为了把源程序翻译成目标程序，还要造各种表格，并把源程序变成机器的内部表示。这时，要检查句子本身的语义是否正确。只有语义正确，才能把源程序的机内表示翻译成目标程序，这就是代码程序的生成过程。如果发现语义有错误，就打印出错信息，待改正之后，才能把目标程序编制出来。为了提高目标程序的运行效率，有时还要进行优化工作。

编译程序的框图如图1.7所示。

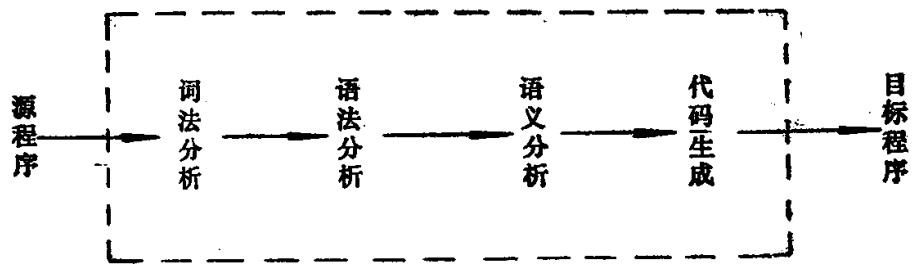

图1.7　编译程序框图

编译程序的几部分工作不是截然分开的，而是有机地有节奏地进行，对源程序进行检查、分析和翻译工作，最终目的是要生成与源程序等价的机器可直接执行的统一的目标程序。

习　　题

1. 什么叫做源程序？什么叫做目标程序？
2. 什么叫做翻译程序？翻译程序可分为哪三类？
3. 分别用FORTRAN语言、汇编语言（或机器代码语言）表达如下过程：

$$M = 1 + 2 + \cdots\cdots + 100$$

并指出编译程序的任务是什么？

4. 什么叫做编译程序？试指出编译程序的几个部分及其作用。

第二章　形式语言的基本概念

2.1　语言、语法和语义

我们学过汉字及其标点符号，这些汉字和标点符号就是汉语的基本字符（或称基本符号），这些基本字符是我们所能识别的。但一个基本字符，有的有一定的意义，有的没有意义，只有由若干个基本字符组合在一起才有一定的意义。是否随便把一些基本字符或基本字符组组合在一起我们都能看得懂呢？不是的！所谓看得懂就是指一句话是否符合一定的规则并具有一定的意义。

〔例2.1〕我吃苹果。

这一句话的意思大家都明白，即是说它符合我们汉语的语法。在汉语中有这么一条语法规则：

〈主语〉〈谓语〉〈宾语〉〈句号〉

其中，〈主语〉可由〈名词〉组成，〈谓语〉可由〈动词〉组成，〈宾语〉可由〈名词〉组成。这一语法要求是：先是〈主语〉，随后是〈谓语〉，接着是〈宾语〉，最后由一个〈句号〉“.”结束。这里的〈名词〉由“我”、“苹果”组成（注意，“苹果”由两个基本汉字组成），〈动词〉由“吃”组成。我们可以用树结构把这个句子表示成如图2.1所示的样子。

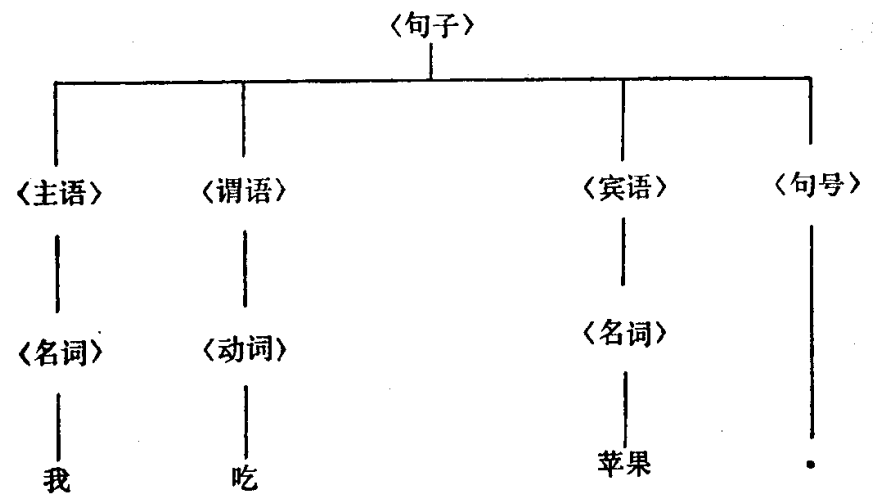

图2.1　语法树

这种用树结构来表示语法的图我们称之为语法树。语法树把句子分解成各个组成部分，并用它来描述句子的语法（或结构）。每个组成部分就是一个单词，而能够由别的单词构成的单词称为语法单位，我们用“〈”和“〉”把语法单位括起来，以便和基本字符区分开。

显然，汉语中有很多语法规则，按照一定的语法规则可以构成句子。具有一定规则的语法单位的集合称为语法，具有一定语法要求的基本字符集合称为句子，语言是由句子组成的集合。

所谓语言的语法，就是指该语言的规则。如果一个句子符合该语言的某一条规则，就说这个句子符合该语言的语法要求。例如

我吃苹果。

这个句子，符合汉语中的一条规则

〈主语〉〈谓语〉〈宾语〉〈句号〉

也即符合汉语的这一条语法。如果某一个句子不符合该语言的所有规则，则说该句子不符合语法。

所谓语言的语义，就是指一个句子不但符合语法要求，而且要有一定的意义，即是说这句话有意义。然而，一个句子符合语法要求，却不一定就有意义。例如句子

苹果吃我。

显然也是符合规则

〈主语〉〈谓语〉〈宾语〉〈句号〉

但这个句子没有意义。

计算机算法语言所能识别的符号称为该算法语言的基本字符，每一种算法语言都有一定的基本字符集。一个或若干个基本字符组成一个单词，一些单词又可以构成相应的语法单位，这些语法单位按照一定的规则（即语法）构成句子（在算法语言中称为语句），很多句子构成一种算法语言的程序。例如FORTRAN程序就是由很多语句组成的，它的每个语句由若干个部分组成，每个部分又按一定的语法要求放在一定的位置上。

2.2 元语言和BNF表示

在语言中，所有的句子都具有某种语法结构，计算机算法语言尤其是这样。通常，我们希望用少数几条形式规则来描述出一个语言的全部语法结构，从而能够建立各种复杂的句子。为此，人们在实践中总结出一种元语言。

用一种语言去描述另一种语言（也可能是其本身）时，我们把前者称为元语言，后者称为目标语言。元语言是一种用以描述某些其它语言的语言。例如，我们用汉语描述英语时，汉语就是元语言，而英语则是目标语言；在英语课中，英语又作为描述它自己的元语言。在前面的语法树中，语法树是元语言，而“我吃苹果。”则是这种元语言所描述的目标语言的一个句子。实际上这个语法树所描述的目标语言只包括这一个句子。

为了描述算法语言，目前普遍采用的元语言是BNF表示法。在这个表示法中，符号::=表示“由…组成的”的缩写，语法单位用“〈”和“〉”括起来，符号::=的左边只能是一个语法单位，右边是若干个单词，这些单词又可能是语法单位。::=读作“左边的语法单位是由右边的单词（或语法单位）组合成的”，或“右边的单词（或语法单位）是由左边的语法单位产生的”。

〔例2.2〕 〔例2.1〕的BNF表示法可示意如下：

〈句子〉::=〈主语〉〈谓语〉〈宾语〉〈句号〉

〈主语〉::=〈名词〉

〈谓语〉::=〈动词〉
〈宾语〉::=〈名词〉
〈名词〉::=我
〈名词〉::=苹果
〈动词〉::=吃
〈句号〉::=。

左边的同一个语法单位有两个以上表示时，可以合起来书写，中间用竖线|（读作“或”）隔开。例如

〈名词〉::=我
〈名词〉::=苹果

可以合成

〈名词〉::=我|苹果

读作“〈名词〉是由我或苹果组合成的”，或者“我或苹果是由〈名词〉产生的”。于是，前面例子的BNF表示法又可以示意如下：

〈句子〉::=〈主语〉〈谓语〉〈宾语〉〈句号〉
〈主语〉::=〈名词〉
〈谓语〉::=〈动词〉
〈宾语〉::=〈名词〉
〈名词〉::=我|苹果
〈动词〉::=吃
〈句号〉::=。

每一条BNF表示就是一条语法规则。一般来说，语言有若干条语法规则。我们可以使用BNF表示法来表达更为复杂的语法结构。

为了便于表达更为复杂的语法结构，我们可以配合使用另外的一些表示方法。

（1）花括号表示法

在FORTRAN语言中，标识符最多由六个字母数字字符组成，但第一个必须是字母。我们可以用BNF表示如下：

〈FORTRAN标识符〉::=〈字母〉|〈字母〉〈字母数字串〉

〈字母数字串〉::=〈字母数字〉|〈字母数字〉〈字母数字〉|〈字母数字〉〈字母数字〉〈字母数字〉|〈字母数字〉〈字母数字〉〈字母数字〉〈字母数字〉|〈字母数字〉〈字母数字〉〈字母数字〉〈字母数字〉〈字母数字〉

〈字母数字〉::=〈字母〉|〈数字〉

〈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

上述表示显然是比较麻烦的我们可改用形如

$$\left\{\ \right\}_n^m$$

的花括号来表达，其中m≥n，表示花括号内的内容可以出现n次直至m次，即下角标n是出现的最小次数，上角标m是出现的最大次数。于是，FORTRAN标识符可以表示成

〈FORTRAN标识符〉::=〈字母〉{〈字母数字〉}$_0^5$

〈字母数字〉::=〈字母〉|〈数字〉

下角标0表示不出现，上角标5表示重复出现最多五次。

注意，上角标m为有穷多次时可以不写，即

$$\{\ \}_0$$

表示花括号内的内容可以不出现或可以出现有穷多次。显然

$$\{\ \}_0^0$$

没有什么意义。

（2）提因子表示法

若我们用U、A、B、C表示四个语法单位，有如下规则：

$$U::=AB|AC$$

它读作“U由AB组合成或由AC组合成”，显然，右部有相同的语法单位A，用类似数学中提公因子的方法把公共的语法单位A提出来，则可以表达成

$$U::=A(B|C)$$

如果规则为

$$U::=BA|CA$$

则也可以表达为

$$U::=(B|C)A$$

把公共的语法单位（或几个语法单位）作为因子提出来，插进圆括号，这种表示法叫做提因子表示法。当然圆括号可以嵌套任意层。

注意，上述花括号和圆括号都是作为元语言的符号（一般称为元符号）来使用的。如果花括号特别是圆括号作为所定义的语言（即目标语言）的符号，则会引起一些问题。在此情况下，必须给出注明，或者用一定的记号给予标注。对于符号::=和|也同样处理。

2.3 符号和符号串

汉语中的汉字或标点符号，统称为基本字符，它们构成了汉语的基本字符表。由一个或若干个基本汉字构成一个字汇，这些字汇构成了一个基本字汇表。我们知道，汉语是由句子组成的，而句子又是由单字和标点符号组成的序列。

例如整数语言和偶整数语言，它的基本字符是数字0，1，2，3，4，5，6，7，8，9。由这十个基本字符组成的序列可以构成整数语言，而使其构成偶整数语言，必须每个序列的最后一位数字是0，2，4，6或8。

基本字汇表是元素的非空有穷集合。我们把基本字汇表中的元素称为符号。注意，这里的符号可以是基本字符或由基本字符构成的字汇，我们用大写字母来表示符号。由基本字汇表中的符号组成的任何有穷序列一般称为符号串，我们用小写字母x、y、z等表示符号串。因为符号或符号串均用字母来表示，故一般基本字汇表称为字母表。

〔例2.3〕 字母表={A，B，C}的符号有A、B和C，相应的一些符号串有A、B、C、AB和AACA等。

因此，符号串形式定义为由某个有穷非空集合中的符号所组成的任何有穷序列。

在符号串中，我们要注意符号的出现顺序，出现顺序不同就是不同的符号串。例如，符

号串AB就不同于BA，符号串ABCA与AABC也是不同的。

符号串x所包含的符号个数称为符号串x的长度，记作 |x| 。例如

$$|a| = 1, \quad |abb| = 3$$

符号串的联结xy，就是把后面的符号串y写在其前面的符号串x之后而得到的符号串。例如，如果

$$x = ab$$
$$y = bc$$

那么

$$xy = abbc$$

或

$$yx = bcab$$

联结形式定义为对两个符号串的一种形式运算：被运算的两个符号串按书写的先后次序，而不改变其本身的结构，形式上串在一起组成一个新的符号串。

不包含任何符号串称为空符号串，我们用∧表示空符号串。例如，x为空符号串，通常表示为

$$x = \wedge$$

显然，对于任意符号串x来说，我们可以得到

$$\wedge x = x \wedge = x$$

特别需要指出的是空符号串的长度为 0，即 $|\wedge| = 0$ 。

如果z＝xy是一符号串，那么x是z的头，而y是z的尾。如果y是非空的（即y不是∧），那么x是固有头；同样，如果x是非空的，那么y是固有尾。

对于符号串的方幂x^n定义为

$$\underbrace{xx\cdots x}_{n\text{次}}$$

特别是，x^0是空符号串∧，即$x^0 = \wedge$，又，$x^1 = x$，$x^2 = xx$，$x^3 = xxx$，等等。

〔例2.4〕 设x＝a，Z＝abb，试求出zx、xz、z^0、z^1、z^2、z^3。

根据定义，我们有

$$zx = abba$$
$$xz = aabb$$
$$z^0 = \wedge$$
$$z^1 = abb$$
$$z^2 = abbabb$$
$$z^3 = abbabbabb$$

〔例2.5〕*设符号串

$$x = abb$$

试求出符号x的长度、头、固有头、尾和固有尾。

根据定义，我们有

$$|x| = 3$$