C语言BNF解译及其程序设计

德　华　编著

陕西科学技术出版社

C语言BNF解译及其程序设计

李德华 编著

张福延 王子平 李家华 胡建鹤 校

陕西科学技术出版社

C语言BNF解译及其程序设计

李德华 编著

陕西科学技术出版社出版

（西安北大街131号）

陕西省新华书店发行　　西北电讯工程学院印刷厂印刷

787×1092毫米 16 开本　　14印张　　330 千字

1986年3月第1版　　1986年3月第1次印刷

印数1—10,000

统一书号：13202·91　　定价：2,50元

前　言

C 语言是目前世界上颇有影响并得到迅速普及的程序设计语言。

随着 UNIX 分时操作系统在计算机领域的广泛流行，近年来 C 语言在软件工程领域里引起人们极大的关注。尽管 C 语言与 UNIX 系统有着依存、互促和并进的密切关系，但 C 语言不是一种专用的程序设计语言。而是同 ALGOL、PASCAL 程序语言一样，是一种通用的计算机程序设计语言。C 语言的通用性和应用范围都比人们所熟悉的 FORTRAN、COBOL等高级语言要好一些，广一些。C语言不仅具有一般高级语言的特征，而且具有语言简洁、表达能力强、语言生成代码质量高以及移植性好等优点，所以 C 语言与 PASCAL 语言一样，已经成为从微机、小型机到大型机上都可使用的语言。C 语言具有丰富的运算符、灵活实用的表达式、较完善的数据结构和先进的控制流。更由于 C 语言的设计思想面向机器底部，所以 C 语言不仅适合于科学计算领域，而且在操作系统程序设计及系统应用程序设计领域更是大有作为。这就是 C 语言越来越受到重视，并得到迅速推广应用的原因所在。

随着微机的开发、普及和应用，学习 C 语言的人也越来越多。作为 C 语言的学习资料，目前较多的是用翻译过来的《C 程序设计语言》一书，这是一部用说明描述而不是用公理定义的 C 语言文本。

《C 语言 BNF 解译及其程序设计》一书，是编者在学习《C 程序设计语言》一书的基础上，在 DUAL-MC-68000-83/40 系统上编制 C 语言程序实践的一个总结。虽然本书用巴科斯——脑尔范式(BNF)解译了 C 语言的语法公式，但出发点决不是从理论上研究 C 语言本身，而是从实用角度出发，以一种通俗的科普读物的形式献给读者。这是本书的宗旨。

本书的语法基础均来自MC-68000 上的 C 语言。文本的语法及例子大部分在MC-68000 上进行了验证。在采用 BNF 方法解释 C 语言的语法过程中，引进了大量的中间语法单元，这些中间语法单元的描述多来自 ALGOL60，ALGOL68 语言文本。因为这两个编译系统在我国流行最早，也是大家比较熟悉的两个文本。引进大量的中间语法单元不仅是由于 C语言过于灵活而使语言语法定义过度自然，更为要紧的是，用汉语描述这些语法定义加深了C 语言的条理性和易读性，使读者更易接受，也给程序的验证带来方便。学习完这本书将会给读者留下一个“汉字 C 语言”的印象。

在本书的编写直至完稿过程中，得到了许多老师、同行的指点和热情支持。在此向赵桂林、周维庭、岳兆峰、金顺根以及许维慧等同志表示衷心地感谢。同时，在编写本书的过程中，编者参阅了部分个人或单位编写的有关 C 语言的资料，这些参考资料在实践和理论上都丰富了编者的知识、开阔了编者的视野，在此，一并表示谢意。

用巴科斯——脑尔范式解释 C 语言仅仅是个尝试，加之本人水平所限，时间又较为仓促，本书一定有不少错误和不足之处，恳切希望读者指正。

编者　1985.10

目　录

前　言

第一章　算法语言程序设计基础……(1)
1.1　程序及其结构 ……(1)
1.2　计算机程序设计 ……(3)
1.3　算法语言的定义方法 ……(4)
1.4　语言规则的推导 ……(5)
1.5　算法语言结构 ……(7)
1.6　程序设计错误分析 ……(8)
第二章　C 语言基础……(10)
2.1　基本符号 ……(10)
2.2　标识符 ……(11)
2.3　名　字……(12)
2.4　常　量……(12)
2.5　符号常量 ……(16)
2.6　基本数据类型 ……(17)
2.7　简单变量 ……(19)
2.8　数组变量 ……(20)
第三章　运算符及表达式……(23)
3.1　最小的 C 语言程序 ……(23)
3.2　C 语言的运算符 ……(26)
3.3　C 语言的表达式 ……(34)
3.4　类型转换及其运算符的优先级 ……(39)
第四章　语句……(41)
4.1　C 语言的语句分类 ……(41)
4.2　选择语句Ⅰ——条件语句 ……(42)
4.3　重复语句Ⅰ——循环语句 ……(47)
4.4　重复语句Ⅱ——前置条件重复语句 ……(50)
4.5　重复语句Ⅲ——do-while 循环语句……(53)
4.6　转移语句 ……(56)
4.7　选择语句Ⅱ——开关语句 ……(59)
4.8　程序举例 ……(63)
第五章　函数……(68)
5.1　子程序、过程、函数 ……(68)
5.2　C 语言的库函数 ……(69)
5.3　函数定义 ……(70)

5.4 函数调用 ……(74)
5.5 函数说明 ……(78)
5.6 函数的递归 ……(79)
5.7 特殊函数——main( ) ……(80)
第六章 输入与输出 ……(82)
6.1 使用 I/O 库函数的环境 ……(82)
6.2 标准输入和输出 ……(82)
6.3 按格式的输入和输出 ……(85)
第七章 变量说明 ……(94)
7.1 说明符 ……(94)
7.2 C 语言的存贮类说明 ……(94)
7.3 变量的作用域和生存期 ……(97)
7.4 外部变量的定义 ……(97)
7.5 外部变量的说明 ……(102)
7.6 局部变量的定义 ……(103)
7.7 置初值 ……(105)
第八章 指针 ……(108)
8.1 指针的定义 ……(108)
8.2 指针表达式 ……(108)
8.3 指针的运算 ……(110)
8.4 指针与函数参数 ……(113)
8.5 指针与数组 ……(115)
8.6 指针数组 ……(121)
8.7 命令行参数 ……(123)
8.8 指向函数的指针 ……(127)
8.9 程序举例 ……(132)
第九章 结构和联合 ……(138)
9.1 结构说明 ……(138)
9.2 实结构及结构成员的引用 ……(140)
9.3 结构的初始化 ……(143)
9.4 结构与函数 ……(147)
9.5 结构数组 ……(151)
9.6 指向结构的指针 ……(157)
9.7 引用自身的结构 ……(159)
9.8 联 合 ……(165)
9.9 字 段 ……(167)
9.10 类型定义 ……(168)
第十章 文件的使用 ……(169)
10.1 打开文件 ……(169)

10.2 建立文件 ……(171)
10.3 随机存取文件 ……(171)
10.4 使用文件指针读写文件 ……(172)
10.5 使用文件描述字读写文件 ……(173)
10.6 关闭文件——Close ……(175)
第十一章 目标代码的生成、程序调试和程序错误分析 ……(176)
11.1 目标程序的生成 ……(176)
11.2 编译预处理程序的功能 ……(178)
11.3 C 语言程序常见错误分析 ……(183)
附录 A C 语言参考手册 ……(185)
附录 B MC-68000 常用库函数使用说明 ……(212)

第一章　算法语言程序设计基础

本章不涉及C语言本身，只介绍程序设计的基础知识。一、二节介绍程序的概念，并对计算机程序设计方法作以回顾。本章重点是第三节和第四节，这两节介绍了算法语言语法构造原理，对学习用巴科斯——脑尔范式定义的算法语言及后续章节是极其有用的。因此，初学者不要忽略本章，尤其不要忽略第三和第四节。

1.1　程序及其结构

“程序”是一个抽象的概念。我们一般所称的“会议程序”和“工作程序”都是对这一概念的具体应用。我们无论干什么样的工作，从事何种活动，都是在执行某种程序。从工矿企业的生产活动到农村种植庄稼，从科学研究到行政管理，从社会活动到家庭生活，从宏观世界的活动到微观世界的活动，……都离不开程序，都在执行形形色色的程序。因此，“程序”的概念已经渗透到了社会的每个细胞。“程序”具有“计划”或“安排”或“打算”的含义。不仅如此，程序还有着更深刻的含义，这就是“规划”，或“规定”，或反应某种“规律”。例如，一个“会议程序”是一种“安排”；一个家庭的某月收支清单是一种“计划”；一个编织图样的构造方法是一种“规定”；而“一张食谱”反应的则是一种“打算”。那么，以上列举的这些程序与计算机程序有什么联系呢？从宏观上讲，这些程序同计算机程序是类同的，只不过在计算机上的程序“自动化”罢了。也就是说，社会活动和实践是计算机程序的基础。随着计算机技术的普及，几乎所有这些社会活动和实践都将可以用计算机程序来实现或模拟。

一个抽象的程序（当然包括计算机程序）有如下特点：

1．程序是有序的。从整体看，或者从相对意义上讲是顺序的。例如，一个会议程序日程安排如下：

a．开幕式
b．大会发言
c．大会讨论
d．会议总结
e．闭幕式

但是，有可能在其内部改变这种顺序。较详细的会议程序安排如下：

六月三日　开幕式
六月四日　大会发言
六月五日　如果天气好，则组织参观，否则会议进行讨论
六月六日　参观或会议讨论
六月七日　会议总结及闭幕式

2．程序的运行总会产生一种效果。一个会议程序的结果是做出某种决策；一个食谱可

能带来美餐。对于计算机程序的执行结果往往是以符号组成的输出形式，或打印，或显示，或绘图。实时控制方面的程序运行效果则是对某一生产过程的即时控制。

3. 程序总是施操作于某些对象，并且在一个程序的前面都附有对这些操作对象的一个说明。例如：制作“鱼香肉丝”的配料清单如下：

原料：

猪肉三两，青笋三两，木耳三钱，酱油三钱，精盐六分，料酒二钱，味精二分，花生油一两，醋二钱，白糖三钱，湿淀粉六钱，豆瓣辣酱三钱，葱末二钱，姜末一钱，蒜末二钱。

这就是一个菜谱程序的原料清单。其中清单中猪肉、青笋、木耳等就是这个菜谱程序的操作对象，清单是对每个操作对象的说明。

对于计算机程序，操作对象称为数据。对于每种数据都必须加以说明，这些说明在一个程序中称为说明部分。

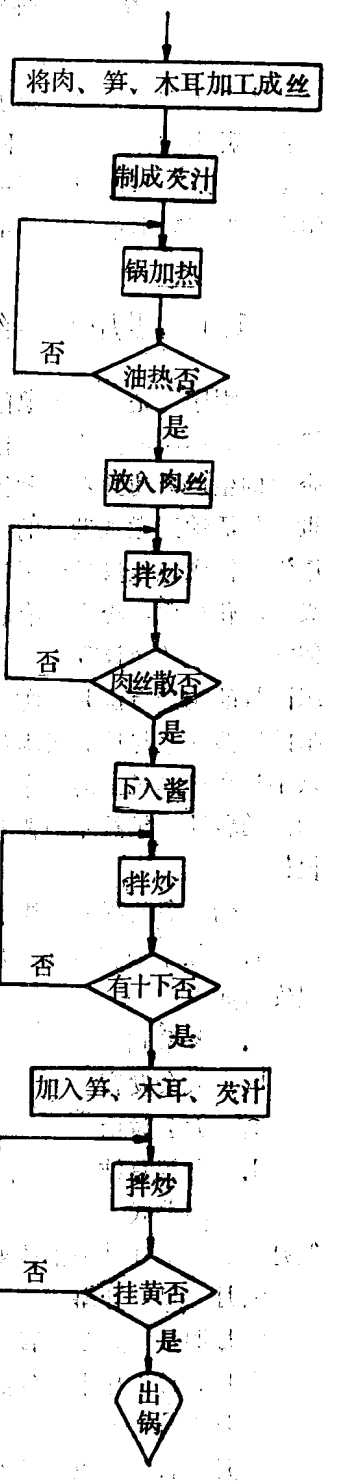
4. 一个程序总是要对程序操作对象的操作方法和步骤进行描述，这就是程序设计者向程序执行者发出的指令序列。在这些指令序列中，有时要求执行者作出判断，在这种情况下，程序的设计者并不知道在一次具体实现中执行者会作什么，但可以建立一个执行者作出判断的标准。同时，在程序中往往有这样的情况，在指令的序列中，一条或一组指令可能需要执行一次以上，即所谓重复。程序设计者对于要重复执行的部分要给出重复的次数或建立一个取决动态程序的标准(或条件)。例如，“鱼香肉丝”的做法如下：

(1) 将猪肉切成长二寸，粗一分的肉丝，且用湿淀粉加清水拌匀浆好，青笋去皮和筋，切成长二寸，粗一分的丝，用精盐腌一腌。将水发木耳切成粗丝。

(2) 将酱油、醋、糖、酒、味精、湿淀粉、葱末、姜末、蒜末加少许清水兑成芡汁。

(3) 将锅加热，下入花生油。油热后，下入浆好的肉丝，用手勺不停拌炒。待肉丝炒散后，下入豆瓣酱炒十下，接着下入青笋丝，木耳丝，再倒入芡汁，待芡汁发出响声并变稠时，用手勺推炒，至汁附在原料上即成。

把以上三个部分用一个框图来描述，就更清楚了。从框图中可以清楚地看到，有些部分是重复地执行的。设计者给出了某一要重复部分的次数或建立一个重复的标准，当重复次数或建立的重复标准不成立时，则就终止了重复。在框图中，加热、拌炒等都是要重复执行的，这些重复部分的执行条件或次数都要受下面菱形框中注明的条件或次数控制。



5. 一个程序总是有一个名字，用以区别每个程序。程序的名字都出现在这个程序的开始处，在计算机程序中称为程序的首部。

综上所述，程序的一般结构是：

1. 每个程序都有一个名字，位于程序的首部。一个复杂的计算机程序首部还可以包含除程序名字以外的其它项目。

2. 一般的程序都有一个量态的说明，称为程序的说明部分。在计算机程序中的说明部分是对数据属性、量值等方面的描述。

3. 指令序列。指明程序对操作对象的加工顺序和步骤，这是程序的核心。

4. 一个动态的或者是执行过的程序总是产生一种效果。这种效果的必然显示和出现称为程序的输出。任何一个程序都必须有输出功能，没有输出的程序等于这个程序做无用功。

5. 每个程序在最后总要设置一个标记程序结束的信息。

1.2 计算机程序设计

在1.1节中抽象地介绍了一般“程序”的概念和结构，这些都是适合于计算机程序的。

计算机程序设计同计算机的出现是一对双胞胎。它是伴随着计算机的出现和发展形成的一门学科。

从应用角度来看，程序实质上是一种手段，程序的作者借此与执行者通讯。计算机程序就是人们与计算机通讯的手段。通讯就需要语言，而这种语言又不同于一般的自然语言。同任何事物的发展一样，计算机程序设计的编写方法也经历了由低级到高级，从简单到复杂的发展过程。纵观计算机发展历史，大体分为如下三个阶段：

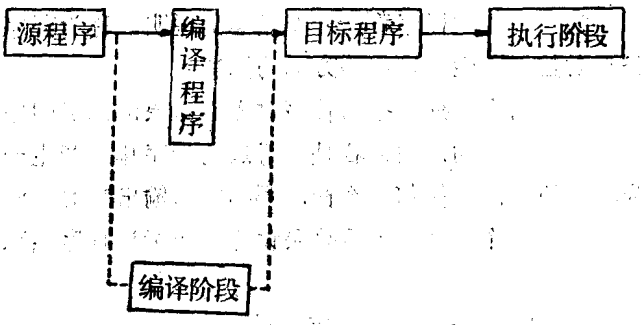
第一阶段，用机器语言编写程序。计算机指令系统是用二进制编码表示的。在计算机刚刚出现的时候，人们只能依赖于特定的机器的指令系统，采用与计算机硬件指令系统完全一致的二进制编码指令来编写程序。这种程序称为机器码指令程序。

第二阶段，用汇编语言编写程序。用机器语言指令编写的程序既繁琐，直观性又差。为此人们用一些简单而又形象的符号来代替机器指令，这些指令又对应于具体机器的二进制指令码，这就是初级的汇编语言。在此基础上，又把一些子程序、存贮器地址也用符号来代替。用汇编语言编写的程序称为汇编程序。用汇编语言编写的程序必须经过加工变成对应的二进制指令码程序，才能在计算机上运行。这种加工过程是由汇编编译程序来完成的。用汇编语言编写程序比用机器语言编写程序的效率显著提高，这是程序设计领域的一大进步。

第三阶段，用算法语言编写程序。无论是机器语言，还是汇编语言程序，都存在直观性差、使用不方便、编写周期长、工作量大等缺点。尤其是机器语言与汇编语言，它们都与具体的计算机有关，因此，编写的程序不便于移植，通用性差。其核心问题是与人们习惯用数学表达方式来描述问题的方法差别较大。于是，人们在汇编语言的基础上，避开具体的机器，用一些符号来描述程序的意图，这就产生了采用数学上的符号及一些特定的字符构成的一种语言，即算法语言。算法语言的产生、应用是计算机领域里的一次革命，它不仅扩大了计算机的应用范围，而且也促进了计算机硬件的发展。目前世界上算法语言有几百种，国内外比较流行的也有几十种，例如ALGOL60，ALGOL68，PL/1，FORTRAN，COBOL，BASIC，PASCAL以及C等，这些算法语言也称为高级语言。用高级语言编写程序具有直观、透明

度好、精确、对硬件适应性好、减少编写程序的工作量和缩短编写程序的周期、使用方便、便于调试和修改、易于掌握等优点。

用算法语言编写成的程序称作源语言程序，简称为源程序。这种源程序同汇编程序一样，是不能被计算机直接接受的。它必须经过加工，成为等价的、能被电子计算机直接接受的、由机器指令所表示的程序，这种程序称为目标程序。把源程序翻译成目标程序的程序称为编译程序，也称编译器。把源程序翻译成目标程序的过程称为编译过程。



源程序、编译程序、目标程序三者的关系是：源程序经过编译程序的加工，成为目标程序。目标程序在计算机上可直接地运行，而源程序在计算机上只是间接运行。右图表示这三者的关系。

在高级语言问世后，用机器指令和汇编语言来编写程序显然是不太合适了，但这两种方法都并没有因此而消声匿迹。在一些程序的最关键、最重要的部分仍用机器指令，尤其是用汇编语言来编写。例如，操作系统的核心部分往往是用汇编语言，甚至用机器指令来编写。由汇编语言编写的程序生成的代码质量最高，这一点任何高级语言都是无法与之相比的。这就是机器语言，尤其是汇编语言仍有生命力的原因。在目前流行的微处理机上，用汇编语言编写的程序仍是构成软件的主要部分之一。编写高质量的程序仍用汇编语言来完成。

1.3 算法语言的定义方法

语言是人与人之间进行思想交流的工具和手段。同样，算法语言是人与计算机通信的工具和手段。语言同任何事物一样都有着特定的内在联系，这就是人们通常所说的规则。这种规则在语言中称之为语法。掌握一种语言，就必须学习和研究其语法，学习算法语言也是如此。

“语法”这个概念，对人们并不陌生。在语言学中，每种语言都有自己的语法。例如，汉语有汉语语法，英语有英语语法，语言的语法反应了一组规则。这些规则一部分称为词法规则，另一部分称为语法规则，又称产生规则。词法规则本质上定义了某种语言可取的一张字符表，而语言的语法规则则规定如何从这张字符表中构造更大的结构。这种结构称为语法单位。算法语言的语法单位有：表达式、语句、子程序、函数、过程和程序。对于一个语言来讲，不仅需要给出它的词法、语法，而且还要定义语义，也就是字符表中每个符号和语法单位的含义。在定义了一个有限的字符表，规定了语法和语义以后，就可以构造一种语言了。

为了传授某种语言，往往需要一种介质。例如，对中国人来讲学习英语要借助于汉语来描述。对于法国人来讲学习英语要借助于法语来描述。这种用以描述某些其它语言的语言称为元语言。巴科斯—脑尔范式(简称 BNF)的元语言有三个：

“::=”读为“定义作”

“|”读为“或”

“〈”与“〉”表示其中被括起来某一语法单元中的一个。

用 BNF 定义算法语言有三种方法：

1． 枚举法。把被定义的语法单元中可能出现的所有情况都一一列举出来的方法称为枚举法。例如

〈非 0 数字〉::=1｜2｜3｜4｜5｜6｜7｜8｜9

语法单元〈非0数字〉的定义就是把所有出现的情况都一一列举出来，这就表示“非0数字”可以是 1，2，3，……9 这九个数字中的任一个。

2． 并置方法。用二个以上语法单位来定义某一新的语法单位，称为并置法。例如

〈指针变量〉::=*〈标识符〉

指针变量的定义就是由“*”与“标识符”并置来定义的。

3． 递归法。被定义的语法单位同时又可以在定义符中出现。例如

〈字符串〉::=〈字符〉｜〈字符串〉〈字符〉

该定义中“字符串”不仅出现在定义式的左边，又出现在定义式的右边，而这种定义可以一直递归下去。引伸的含义是：字符串是任意长的一串字符。

用巴科斯—脑尔范式定义算法语言，最基本的方法就是以上三种，但这三种方法不是孤立的，定义一个语法单元可同时用到枚举、并置和递归或几种方法的组合。

熟悉和掌握算法语言的定义方法是非常重要的。这是学习程序语言的基础。就如同要学会操作一部机器必须从原理上掌握它一样重要。

1.4 语法规则的推导

用高级语言编写程序，有两个问题要解决。一是如何根据语言文本定义的规则去编写程序；二是如何依据语言文本的规则去验证程序编写的正确性。这是初学者颇为关心的问题。

一个语法规则可写成如下形式：

$$U::=X;$$

其中U 是符号，X 是非空有穷符号串。一般地称U 为规则的左部，X 称为规则的右部。规则的非空有穷集合称为文法，用G[z]表示。下面给定文法 G[〈十进制数〉]的一组规则是：

(1)—(2)〈数〉::=〈数字〉｜〈数字串〉

(3)—(4)〈数字串〉::=〈非零数字〉〈数字〉｜〈数字串〉〈数字〉

(5)—(6)〈数字〉::=0｜〈非零数字〉

(7)—(15)〈非零数字〉::=1|2|3|4|5|6|7|8|9

在给定的文法(有机联系的规则的集合)中，规则左部出现的那些符号称为非终结符号，规则左部不能出现的那些符号称为终结符号。这就是说，在文法中，非终结符号一定要在规则的左边出现，也可以出现在规则的右边；终结符只能出现在规则的右边。如果用 VN 表示非终结符号的集合，用 VT 表示终结符号的集合，对于文法 G[〈十进制数〉]则有：

VN={〈非零数字〉，〈数字〉，〈数字串〉，〈十进制数〉}

VT={0,1,2,3,4,5,6,7,8,9}

如果用 V 表示文法中所有符号的集合，则有：

$$V=VN\cup VT$$

语法规则的推导，是指在给定的文法G[z]中，对于每组规则，当规则的左部可以用本文法中另一组规则右部来代替时，则进行替换。若替换后的规则的左部中还能被本文法中相应规则的右部来代替，则仍进行上述的替换，直到全部替换成终结符号为止。这就是语法规则的推导。

下面用例子来说明这一推导过程。

[例1-1]

已知：文法G[〈十进制数〉]的规则(前面已述)

求：(1) 23的推导，(2) 101的推导，(3) 0 的推导，(4) 8 的推导。

解：

(1)

V	W	所用规则
〈十进制数〉	⇒〈数字串〉	(2)
〈数字串〉	⇒〈非零数字〉〈数字〉	(3)
〈非零数字〉〈数字〉	⇒2〈数字〉	(8)
2〈数字〉	⇒2〈非零数字〉	(6)
2〈非零数字〉	⇒23	(9)

则23的推导为：

〈十进制数〉⇒〈数字串〉⇒〈非零数字〉〈数字〉⇒2〈数字〉⇒2〈非零数字〉⇒23

(2)

V	W	所用规则
〈十进制数〉	⇒〈数字串〉	(2)
〈数字串〉	⇒〈数字串〉〈数字〉	(4)
〈数字串〉〈数字〉	⇒〈非零数字〉〈数字〉〈数字〉	(3)
〈非零数字〉〈数字〉〈数字〉	⇒1〈数字〉〈数字〉	(7)
1〈数字〉〈数字〉	⇒10〈数字〉	(5)
10〈数字〉	⇒10〈非零数字〉	(6)
10〈非零数字〉	⇒101	(7)

则101的推导为：

〈十进制数〉⇒〈数字串〉⇒〈数字串〉〈数字〉⇒〈非零数字〉〈数字〉〈数字〉⇒1〈数字〉〈数字〉⇒10〈数字〉⇒10〈非零数字〉⇒101

(3)

V	W	所用规则
〈十进制数〉	⇒〈数字〉	(1)
〈数字〉	⇒0	(5)

则0 的推导为：

〈十进制数〉⇒〈数字〉⇒0

(4)

V	W	所用规则
〈十进制数〉	⇒〈数字〉	(1)
〈数字〉	⇒〈非零数字〉	(6)
〈非零数字〉	⇒8	(14)

则8 的推导为：

〈十进制数〉⇒〈数字〉⇒〈非零数字〉⇒8

利用上述推导方法，不难验证080不属于本文法定义的十进制数。可以进一步推论出：

凡是以数字0打头的数字串都不是本文法定义的十进制数。因此，可以利用文法的推导方法来验证一个语法单元的书写的正确性。这是掌握语法推导的另一个意义。

[例1-2] 有一文法G[〈标识符〉]有如下规则组成：

〈标识符〉::=a|b|c|〈标识符〉a|〈标识符〉b|〈标识符〉1

求：a，aaa，abc1的推导，并求出文法G[〈标识符〉]的VN，VT。

解：a的推导为：

〈标识符〉⇒a；

aaa的推导为：

〈标识符〉⇒〈标识符〉a⇒〈标识符〉aa⇒aaa；

abc1的推导为：

〈标识符〉⇒〈标识符〉1⇒〈标识符〉c1⇒〈标识符〉bc1⇒abc1；

VN={〈标识符〉a，〈标识符〉b，〈标识符〉c，〈标识符〉1}

NT={a,b,c}。

1.5 算法语言结构

目前，世界上流行的高级计算机程序设计语言有几百种。但无论哪种语言都必须有两个基本的组成部分，一个是对所要实现的动作的描述，另一个是对这些动作所要操作的数据的描述。动作是由通常所说的"语句"描述的，而数据是由通常所说的"说明"或"定义"描述的。

一个程序设计语言必须提供数据类型。在流行的程序设计语言中，数据类型分为两大类，一类是简单类型，一类是构造类型。

简单类型的变量的值是以数字或字符为标记的类型。简单类型包含有整数型、实数型、字符型、布尔型。整数型指示或标识具有整数值的变量，而具有小数值的变量是用实数型来指明的。字符型标识特定的计算机设备上可用字符集中的字符变量。布尔型变量取逻辑值，即true(真)和false(假)。

构造类型的产生是计算机在非数值领域中应用的结果。构造类型是由简单类型定义的更复杂的数据结构。构造类型有数组结构、记录结构和集合结构等。

数组结构是由同一类型数据组成的有序结构。这一类型可以是简单类型，也可以是复杂类型。数组的每个元素用数组名与其序号表示，该序号称为下标值。从逻辑上说，一个数组是同一类型数据所组成的某种n维矩阵结构。那么，表示数组的每个元素的下标可以是一维、二维、…n维的。

记录结构是其各成分不一定属于同一类型的一种数据结构。记录结构的每个元素(成分)不能用一个可计算的值来表示，它们通常用标识符来表示。数组结构与记录结构都是一种随机存贮结构。

一个文件结构由属于同一类型的成分的序列所组成，这个序列定义了各成分的自然次序。算法语言系统之所以建立文件结构是为了方便用户程序与计算机外部设备的联系。文件的内容可以是一个程序，或一组数据。

构造类型又分为静态构造类型和动态构造类型。如静态数组与动态数组，静态记录与动态记录等。静态与动态是对存贮管理而言。静态类型的存贮分配是在编译状态下进行，并在整个

执行过程中保持不变。动态类型的存贮分配是在程序运行状态下进行的，并在执行过程中有可能改变。有的程序语言，例如C语言，就没有动态数据结构。

指针类型是用以指向或定位其它的数据项的数据结构。最简单的指针仅仅表示所指数据的机器地址。因此，指针意味着对操作数据的"间接"。引进指针类型变量比用其它方法获得的代码更紧凑和更有效，甚至有时指针变量是表达计算的唯一方法。最后指出的是指针类型也隶属于简单类型，因为指针是指示某个变量的地址。

数据是所表示变量的值。在源程序中出现的每个变量都必须通过变量说明将其引进，变量说明是对变量的属性的各种说明。一个变量属性包含有变量的长度、数据类型、作用域和生存期。变量类型本质上定义了那个变量可取值的集合。

语句是对数据操作程序的描述。对数据操作的动词是语言系统定义的运算符集。运算符集可分为算术运算、逻辑运算、关系运算及特殊运算。对于数值数据，如整数、实数，可施行+、-、*、/等算术运算；对于布尔型或位串型数据可进行∧、∨、～等逻辑运算；关系运算几乎可施于任何数据类型的数据。

运算符与操作数的有机的结合构成了表达式。表达式是语句最直接的成分。

不同的算法语言文本含有不同形式和功能的各种语句。从功能上讲，语句大体可分为执行性语句和说明性语句。说明性语句用于定义各种不同数据类型的变量或运算，执行性语句用于描述程序的动作。执行性语句又可分为赋值语句，输入/输出语句和控制语句。赋值语句是任何语言文本都不可缺少的一种语句，其功能是把一个计算值赋给一个变量(或一个变量的分量)。控制语句是最重要的语句，其功能是控制程序的流程。控制语句一般包含有循环语句、条件语句、转向语句等。

函数和过程是算法语言又一组成部分，引进函数和过程的目的之一是扩展算法语言的功能。函数或过程可以用一个标识符来引进或调用。这个标识符类似于一个语句。当然要对引进的函数或过程进行描述，这种描述称为函数或过程说明。

1.6 程序设计错误分析

用算法语言编写的程序经过编译器翻译成由机器码组成的目标程序后才能在计算机上执行。对于用户，编译程序是不透明的，也就是说，用户不必了解编译程序的细节，而完全有理由把编译程序同计算机系统看成一个整体，我们把一个编译程序和一台机器的组合称为虚拟机。一个源程序就是在这种虚拟机上运行，完成编译和执行两个过程。第一次使用虚拟机是完成编译，该过程的输入信息是源程序，结果是由源程序翻译成的目标程序。然后再次使用计算机，用目标程序做它的运行对象，原始数据作为输入数据，结果就是我们需要的输出数据。

尽管有关编译程序的知识对一个普通用户来讲是否了解关系不大，但了解它对程序设计过程中出现的错误的分析还是有益的。

从编译程序工作的动态特征来划分，编译程序分为词法分析、语法分析、编排中间代码、优化中间代码和目标代码生成五个过程。

词法分析过程的任务是：输入源程序，对构成源程序的字符串进行扫描和分析，识别出一个个单词，如基本符号、标识符、常数、运算符和定界符等，并进行词法检查。在有的编译器中，源程序的修改、删除、插入等也在该过程中进行。

语法分析过程。语法分析是编译程序核心部分之一，其主要功能是依据语言的语法规则进行语法检查。源程序中的大部分错误能在此部分得到提示。

编排中间代码。这一部分是编译系统中一个过渡过程。这一过程主要构造一些表格和生成中间代码，这些中间代码是介于基本符号和目的代码中间的“记号”系统。这为代码优化和翻译目的代码提供了物质基础。

优化中间代码。优化的任务是对中间代码进行加工转换，以期产生更为高效的目的代码。

目的代码生成过程。这是编译程序的最后一个阶段，也称翻译阶段。该过程的功能是将优化的中间代码翻译成用机器指令表示的目的代码，即产生目标程序。

编译程序的静态结构可以按照这五个阶段的任务划分成模块，按块进行编译程序的设计，其对应关系如下：

词法分析过程——→扫描器
语法分析过程——→语法分析部分
编排中间代码——→编排部分
优化中间代码——→优化部分
目标代码生成——→翻译部分

在编译程序的五个部分中，都含有出错处理功能。也就是说，源语言程序的错误可以通过五个过程告诉用户，当用户得到错误报警信息后，首先要查找错误的出处，分析错误的原因。

源程序在被翻译和执行过程中，程序设计的主要错误有四类：

1. 能在翻译过程中识别出的错误，称为编译时错误。编译程序有一个很重要的功能是对源程序进行语法检查，尽可能给出源程序错误的位置、性质和其他信息。

2. 目标程序执行过程中识别出的错误，称为运行时错误。例如：对一个开平方程序的运算对象是一个负数，除法中除数为0等等。这类错误可在执行中识别出来。

3. 在编译和执行过程中检查不出的错误。这类错误将不易查找，一般的方法是通过一个正确的实验数据来验证程序运行后产生的结果。实验数据应该是清晰、简单且一目了然。这样可通过静态的或动态的方式来查找错误的出处。

4. 错误不限于程序，也可能在数据中出现。这是因为输入数据与程序所需要的数据不一致。

程序设计错误的检查是程序员的一项基本功，对于一个程序设计者要具有思维的连贯性、逻辑的严密性和细致的工作作风，要培养检查程序错误的能力。不仅如此，程序设计错误的检查应该从一种技巧走向一种科学的方法，这是程序设计人员所要考虑和研究的任务之一。

第二章 C语言基础

2.1 基本符号

任何一种高级算法语言都有一张基本词汇表，基本词汇表中的每个词称为基本符号。在算法语言中，基本符号是终结符号，而基本词汇表是该语言中所有终结符号的集合。利用基本词汇表，按照给定的语法规则，就可以进一步构造语言中的其它成分，如标识符、表达式、语句以及程序等。在源程序中不允许出现给定词汇表中没有定义的符号。

C语言的基本符号分为字母、数字、关键字、定义符、分隔符和字符文字六个部分。关键字也可称保留字。

1. 语法定义

〈基本符号〉::=〈字母〉|〈数字〉|〈关键字〉|〈定义符〉|〈分隔符〉|〈字符文字〉

〈字母〉::=〈大写英文字母〉|〈小写英文字母〉|〈下底线〉

〈大写英文字母〉::=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

〈小写英文字母〉::=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

〈下底线〉::=_

〈数字〉::=0|〈非零数字〉

〈非零数字〉::=1|2|3|4|5|6|7|8|9

〈关键字〉::=do|if|for|int|char|else|long|goto|case|auto|float|union|break|while|entry|short|extern|double|struct|static|return|sizeof|switch|typedef|default|register|unsigned|continue

〈定义符〉::=(|)|{|}|,|“|”|.|#|'|&|&&|||||=|?|:|<(|>)|[|]|\|;|->|!|+|-|*|/|%|^|~|==|!=|<|<=|>|>=|;

〈分隔符〉::=〈空白〉|〈制表符〉|〈新行〉|〈注解〉

〈空白〉::=〈空〉

〈制表符〉::=\t

〈新行〉::=\n

〈注解〉::=/*…*/

〈字符文字〉::=非\, “, '|\b|\n|\r|\t|\\|\“|\'|\f

2. 语法说明

(1) 关于字母。在ASCII字符集中，英文的大写与小写字母具有不同的值。为此，C语言规定，英文的大写与小写字母表示的意义不同。这一点与PASCAL语言规定是不同的。在PASCAL语言中，字母的大写与小写意义是相同的。同时，C语言还把下底线“_”字符也定义为字母。这样，C语言共有53个字母。

(2) 在C语言中，每个关键字都有固定的含义。因此，在一般情况下都不要把关键字作