高等学校教材

C 程序设计

张长海　陈　娟　编著

高等教育出版社

内 容 提 要

本书以国际标准 ISO/IEC 9899 :1999 和国家标准 GB/T 15272 – 94 定义的 C 语言为载体,阐述基本的程序设计方法,并对相关的 C 语言成分进行较严格的介绍。用 BNF 表示 C 语言的语法,引进 PAD 图表示程序逻辑。全书共分十四章,主要内容包括:BNF、PAD 图、程序设计方法、程序开发和结构化程序设计以及 C 语言的各种词法单位、数据类型、语句、函数等。每章都包含大量例题,并附有大量习题,以利于读者提高程序设计能力和学习掌握相关语言概念。

本书最大的特点是以“程序设计”为主线,把重点放在讲述程序设计方法上。摈弃了目前各种程序设计书中流行的以“解释程序设计语言”为主的做法。全书整体结构良好,图文并茂,知识体系新颖完整,概念准确;注重对读者进行程序设计方法及算法的训练,力求体现“结构化程序设计”思想,注重培养和训练读者良好的程序设计风格。

本书可作为高等院校计算机系各专业“高级语言程序设计”、“C 语言程序设计”、“程序设计基础”等课程的教材和参考书,也可供其他专业学生以及从事计算机工作的有关人员阅读参考。

策划编辑 倪文慧 **责任编辑** 武林晓 **市场策划** 陈 振
封面设计 王凌波 **责任印制**

出版发行	高等教育出版社	**购书热线**	010-64054588
社 址	北京市西城区德外大街 4 号	**免费咨询**	800-810-0598
邮政编码	100011	**网 址**	http://www.hep.edu.cn
总 机	010-58581000		http://www.hep.com.cn
经 销	新华书店北京发行所		
印 刷			
开 本	787×1092 1/16	**版 次**	年 月第 1 版
印 张	28.25	**印 次**	年 月第 次印刷
字 数	590 000	**定 价**	30.00 元

本书如有缺页、倒页、脱页等质量问题,请到所购图书销售部门联系调换。

物料号:15116 – 00

前　言

本书适用于“高级语言程序设计”或“程序设计基础”课程。该课程是计算机系的专业基础课,在计算机专业教学中占有重要地位。学好该课程既可以为后续课程打下良好的基础,又会对学生一生的程序设计技术、技巧、风格和习惯产生深远影响。

本书重点在于程序设计,而对 C 语言本身则采取有所取、有所不取的策略。对于那些常用的语言成分,直接讲述与程序设计方法有关的语言成分,详细准确地介绍;对于那些与程序设计方法联系不太紧要,但是还常用的部分,放在最后简单介绍;而对于那些与讲述程序设计方法关系不太大,也不常用的部分则根本不涉及。

本书力图在深度、广度和知识结构上作出合理的安排。试图在既训练学生的编程能力,又培养学生的抽象思维能力上下功夫;使学生既具有较强的编程能力,又能掌握高级语言 C 本身的语法和语义,同时在知识结构、知识面上尽量做到广泛、深入。

本书作者从事计算机教学已经 20 余年,讲授过 10 多门计算机方面的课程。曾十余次为吉林大学计算机系本科生主讲“高级语言程序设计”课。对 C 语言进行了深入研究,仔细研究了国际标准 ISO/IEC 9899:1999 和中华人民共和国国家标准 GB/T 15272 – 94。本书是作者二十余年教学实践的总结。

作为大学本科计算机专业基础课教材,本书具有如下特点:

1. 全书整体结构好,知识体系新颖完整,章节安排合理,并注意由浅入深地介绍程序设计知识。比如有关函数的知识,由浅入深地分三章介绍;有关指针的知识分散到各个章节介绍,免得集中在一章,使学生学起来枯燥乏味,接受困难。

2. 注重对学生进行严格的抽象思维训练。严格按照国际标准 ISO/IEC 9899:1999 和国家标准 GB/T 15272 – 94 介绍 C 语言,并使用 BNF 表示语法,使用自然语言叙述语义。对 C 语言语法、语义的描述严格、细致、准确,并且形式化,为后续课程(例如编译原理)打下了良好的基础。

3. 本书最大的特点是以“程序设计”为主线,重点放在讲述程序设计方法上,摈弃了目前各种程序设计书中流行的以“解释程序设计语言”为主的做法;注重对学生进行程序设计方法及算法的训练,力图做到严格的理论与具体方法、算法有机结合。全书配备有大量例题,一方面,在讲解例题时着重于算法的构造,以便训练学生的编程能力;另一方面,概念的介绍都以例题作引导,从具体实例出发,使概念引进得自然且容易理解。本书绝大部分例题不是单纯地为了解释语言概念,而是从构造算法出发,以训练学生的实际编程能力为目标。

另外,本书还配备有大量的习题,以便学生做课后练习和进一步提高。

4. 全书自始至终贯穿结构化程序设计思想,所有例题都具有良好的结构和程序设计风格。目的是给学生一个示范,使学生从开始学习程序设计就养成一个良好的程序设计习惯和风格。

5. 本书图文并茂,引进 PAD 图表示程序逻辑。PAD 图的结构比传统的流程图、NS 图等都好,同时也比直接用程序表示算法更直观,易于理解。

全书共十四章,大致分为四部分。

第一部分基本知识,包括第一、二、三章。第一章介绍程序设计基本概念、BNF 和 PAD 图;第二章介绍 C 语言基本符号、单词、数据及其类型;第三章介绍常量、变量、表达式、简单程序、赋值和输入/输出。

第二部分程序设计,包括第四、五、八、九章。第四章简单介绍模块化程序设计思想,引进子程序和函数概念;第五章讲述结构化程序设计的顺序、分支、重复三种程序逻辑,并介绍实现这三种程序逻辑的 C 语言流程控制语句;第八章进一步介绍函数,讲述参数、作用域以及递归程序设计;第九章介绍程序开发和结构化程序设计,包括结构化程序设计原则、程序风格、自顶向下逐步求精的程序设计技术、程序正确性、可移植性、文档以及穷举法和试探法。

第三部分数据组织,包括第六、七、十、十一、十二章。第六章讲述数组;第七章介绍指针;第十章讲述文件及其操作;第十一章讲述对复杂数据的描述,引进结构体和共用体;第十二章讲述动态数据结构及其在程序设计中的应用。

第四部分,包括第十三、十四章。第十三章进一步介绍函数,讲述函数作参数和函数副作用等较深入的问题;第十四章介绍存储类别、位操作、break 和 continue 语句、编译预处理等一些 C 语言独有的特色。

本书的第七、十一、十二、十四章由陈娟执笔,附录四由崔燕提供素材,其余各章节由张长海执笔。全书由张长海统稿。陈娟分别使用 Turbo C 3.0 和 Microsoft Visual C++6.0 调试并通过了所有例题程序。

在本书的编写过程中,作者参阅并引用了国内外诸多同行的文章、著作,在此向他们致意,并恕不一一列举、标明;在本书的成书和出版过程中得到高等教育出版社的大力支持和帮助,作者在此表示感谢。

由于作者学术水平有限,书中难免存在错误和不足,敬请各位读者批评指正,作者表示由衷的感谢。

作 者

2004 年于长春

目　录

第一章　基本知识

现代计算机从出现至今不过 50 多年时间,但其发展速度是任何一种新技术都不可比拟的,目前,计算机已经渗透到各个领域。本书将以 C 语言为背景向大家揭示如何编制计算机程序,即如何使用计算机解决科技、生产、事务处理等方面的问题,介绍程序设计的基本方法、技术和技巧。在具体介绍程序设计之前,先简略介绍一下有关计算机程序设计和程序设计语言的基本知识。

1.1　程序设计语言

一个庞大的计算机系统是怎样有条不紊地工作的呢? 答案是:计算机系统的工作是由事先设计好的程序来控制的。人们首先按自己的需要把让计算机做的工作编写成计算机程序,并把程序送入计算机,然后启动计算机执行程序。计算机的控制器从程序的第一条指令开始,顺序地逐条取出指令进行解释,然后按指令的规定和要求指挥整个计算机系统的工作,从而完成人们设想的要计算机完成的工作。

程序是一个指令序列,也就是用指令序列排成的一个工作顺序、工作步骤。人们平常也使用程序这个名词, 例如运动会程序等。计算机程序是用计算机指令为计算机排定的工作顺序、工作步骤。

为计算机编写程序的过程称为程序设计。

描述程序必须使用一种语言。程序设计语言是指用于编写、描述计算机程序的语言。一般的,人们将程序设计语言分成三类:机器语言、汇编语言和高级语言。

1.1.1　机器语言

机器语言由能被计算机直接执行的机器指令组成,每条机器指令是一串二进制代码。用机器语言编出的程序是一串二进制代码序列。例如计算

$$Y = \begin{cases} X + 15, & \text{若} \quad X < Y \\ X - 15, & \text{若} \quad X \geqslant Y \end{cases}$$

用 Pentium 机器语言可编出如下程序片段(设程序从 100 号单元开始,X 、Y 分别占用 116 、118 号单元):

```
1010 1001 0001 0110 0000 0001
```

```
0011 1100 0001 1000 0000 0001
0111 1100 0000 0101
0010 1101 0001 0101 0000 0000
1110 1010 0000 0011
0000 0101 0001 0101 0000 0000
1010 0011 0001 1000 0000 0001
 ⋮    ⋮    ⋮    ⋮    ⋮    ⋮    ⋮
0000 0000 0000 0000
0000 0000 0000 0000
```

用机器语言编程序显然十分困难,编出的程序不但容易出错,调试极为困难,而且程序本身也极不好读。基于上述原因,人们引进了汇编语言。

1.1.2　汇编语言

汇编语言是符号化了的机器语言。也就是引进一些助记符表示机器指令中的操作码、地址等。完成 1.1.1 节例子中同样计算的 Pentium 汇编语言程序片段如下:

```
     MOV   AX ,X
     CMP   AX ,Y
     JL    S1
     SUB   AX ,15
     JMP   S2
S1:  ADD   AX ,15
S2:  MOV   Y ,AX
          ⋮
     X     DW ?
     Y     DW ?
```

汇编语言程序显然比机器语言程序进了一步,它比较好读、好懂,写起来也显然比二进制代码程序方便得多。其原因在于用符号助记符代替了单调的二进制代码。

虽然汇编语言比机器语言前进了一步,但是它仍然十分烦琐,并且仍然依赖于具体的计算机,程序不便于移植,因此人们又进一步引进了高级语言。

1.1.3　高级语言

高级语言程序不依赖于具体的计算机,它以较接近于自然语言或专业语言的方式描述操作。例如,使用 C 语言完成 1.1.1 节例子中同样的计算,可用如下语句:

```
if ( X < Y )
   Y = X + 15;
```

```
else
    Y = X - 15;
```

显然,这种程序十分好读,它几乎类似于英语句子。这种程序编写起来自然很轻松,而且还具有通用性,可以在不同机器上运行,十分便于移植。

1.1.4 程序的执行

尽管用汇编语言和高级语言编写程序比用机器语言方便得多,但不幸的是,计算机只认识二进制代码(机器语言),既不懂汇编语言的符号代码,也不懂高级语言近似自然语言的语句。为了使计算机懂得汇编语言及高级语言程序,人们设计了翻译器。由翻译器把用汇编语言或高级语言编写的程序(称为源程序)翻译成等价的机器语言程序(称为目标程序)。翻译器也是一个程序,人们称汇编语言的翻译器为汇编程序,高级语言的翻译器为编译程序。

如图 1.1 所示,用汇编语言或高级语言编写程序解题的过程是:首先用汇编语言或高级语言编写出程序;然后把源程序录入计算机;接着启动翻译器,由翻译器将相应的汇编语言程序或高级语言程序翻译成机器语言程序;接着再启动连接程序,由连接程序将机器语言程序连接生成计算机可执行的程序;最后再把可执行程序送入计算机,并启动计算机执行,完成人们要计算机做的工作。

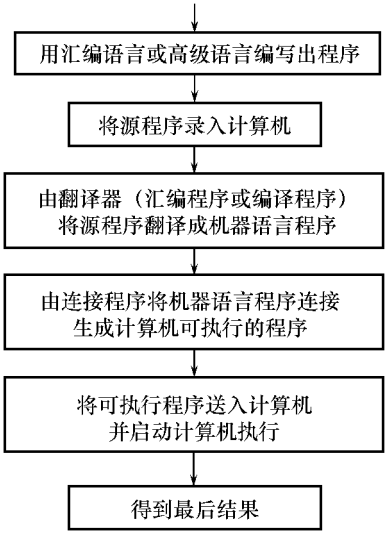

图 1.1 汇编语言、高级语言解题过程

1.2 C 语言简况

本书以 C 语言为背景讲述程序设计,因此有必要简单介绍一下 C 语言,包括它出现的历史背景、它的主要特性以及优点和缺点。

20 世纪 70 年代初,C 语言在美国贝尔实验室诞生。它的前身可以追朔到 ALGOL 60、CPL 、BCPL 和 B 语言。

1960 年公布的 ALGOL 60 语言称为算法语言,是一种面向算法的高级程序设计语言,在程序设计语言理论、编译理论、形式语言理论方面起到里程碑的作用。它的优点是语法严谨且形式化,并完全脱离具体计算机硬件;缺点是不适于编写计算机系统程序。1963 年,英国剑桥大学设计了 CPL 语言,与 ALGOL 60 相比,它更接近硬件,但是规模较大。1967 年,Martin

Richard 对 CPL 进行了简化,推出了 BCPL,可以称其为基本 CPL 语言。1970 年,美国贝尔实验室的 Ken Thompson 又对 BCPL 做了进一步的简化,设计出简单且很接近计算机硬件的 B 语言(取 BCPL 的第一个字母),并用 B 语言编写了 UNIX 操作系统,但是 B 语言又过于简单了。

1972 年,贝尔实验室的 Dennis Ritchie 在 B 语言的基础上设计并实现了 C 语言(取 BCPL 的第二个字母),C 语言既保持了 B 语言的优点,又克服了 B 语言的缺点。1973 年,Ken Thompson 和 Dennis Ritchie 用 C 语言改写了 UNIX,从此,C 语言和 UNIX 紧密地联系在一起。C 语言和 UNIX 的突出优点引起计算机界的广泛重视,UNIX 日益广泛地被使用,C 语言也得到迅速推广。

C 语言的标准化工作从 1982 年开始,当时美国国家标准协会(American National Standards Institute,ANSI)认识到标准化将有助于 C 语言在商业化编程中的普及,因此成立了以 Jim Brodie 为主席的一个委员会(X3J11),该委员会工作的结果是制定了一个 C 语言标准,并在 1989 年被正式采用(即美国国家标准 X3.159-1989),称这个标准为"ANSI C"。

国际标准化组织(International Organization for Standardization,ISO)考虑到编程工作是国际化的,C 的标准化工作应该列为 ISO 的工作日程,因此成立了一个以 P. J. Plauger 为组长的工作小组:ISO/IEC JTC1/SC22/WG14。该小组只做了少量的编辑性修改,即把 ANSI C 变成了国际标准:ISO/IEC 9899:1990,此后 ISO/IEC 标准又被 ANSI 采用。人们把这个公共标准称为"标准 C 语言",简称"C89"。

1995 年,WG14 小组对 C89 做了两处技术修改和一个补充,称这个版本为"C95"。

从 1995 年开始,WG14 开始对 C 进行更大的修订,最终于 1999 年完成并获得批准,新标准的标准号为 ISO/IEC 9899:1999,称该新标准为"C99"。

我国于 1994 年 12 月 4 日公布了"中华人民共和国国家标准 GB/T 15272-94 程序设计语言 C",该标准实际上是 C89 的翻版。

C 的优点可以概括为以下几点:

(1) 语言简洁、紧凑,使用方便、灵活;

(2) C 本身是模块式,便于集体分工合作开发大型程序;

(3) 运算符丰富;

(4) 数据结构丰富;

(5) 具有结构化控制结构;

(6) 与计算机硬件联系紧密,可以直接访问计算机内存,具有位操作;

(7) 生成目标代码质量高。

同时 C 也有许多缺点:

(1) 语法不严格;

(2) 类型机制不严密,比如字符类型与整数类型没有区别,不检查下标超界;

(3) 程序设计自由度太大,不利于保证程序的正确性;

(4) 若程序与计算机硬件联系太密切,则可移植性不好;
(5) 有些语言成分太复杂,比如运算符;
(6) 语言本身不能保证程序设计的结构化。

1.3 程序设计语言的形式描述

1.3.1 语法、语义

一个程序设计语言是一个记号系统,如同自然语言一样,由语法和语义两方面组成。

1. 语法

语法是一组规则。它描述程序的结构形式及构成规律。使用这组规则可以产生和形成一个合法的程序,即语法上正确的程序,人们上机时看到的编译能通过的程序。为了表示方便及叙述严谨,本书将采用 BNF(巴科斯 - 脑尔范式,简称巴科斯范式)来描述 C 的语法。

2. 语义

语义也是一组规则,它定义一个程序的意义。例如怎样理解语句

```
for i:=1 step b until 10 do
        begin b:= b + 1;
              a[i]:= 0;
        end
```

的意义。这就是语义要描述的问题。本书将采用自然语言来描述 C 语言的语义。

1.3.2 BNF

为了描述 C 语言的语法,下边讲述有关形式语言方面的一些知识。

在自然语言中, 语言是句子的集合,句子是语言的基本成分。在程序设计语言中,句子就是程序。

在自然语言中, 句子由单词构成,单词又由字母构成。在程序设计语言中,程序由词法单位构成,词法单位又是由字符构成的。

我们的目的是给出一种表示方法,用它精确地表示一个语言,使得凡是按此方法表示的符号串都是相应语言的句子,反之,任意给一个该语言的句子都能用相应的方法表示出来。巴科斯范式(BNF)表示法是描述语言的强有力的工具。下面介绍巴科斯范式表示法。

文法

考虑英语句子:

The big elephant ate the peanut

由英语知识,可以把它图解成图 1.2 所示的形式。这种图解式把一个英语句子分解成它的各个组成部分,并以此来描述一个英语句子的语法。

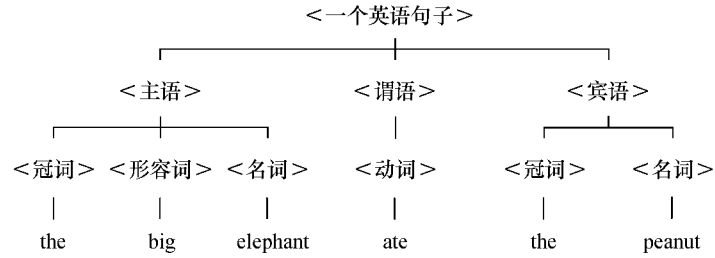

图 1.2　一个英语句子的图解式

由图 1.2 的图解式可以看出:一个英语句子是由主语后跟随谓语,再后跟随宾语组成的;主语是由冠词后跟随形容词,再后跟随名词组成的;等等。可以用符号"::="表示"由……组成的",并以书写顺序表示跟随,则可以把上述"一个英语句子"的构成规则写成下述形式:

G:<一个英语句子> ::= <主语><谓语><宾语>
<主 语> ::= <冠词><形容词><名词>
<冠 词> ::= the
<形容词> ::= big
<谓 语> ::= <动词>
<动 词> ::= ate
<宾 语> ::= <冠词><名词>
<名 词> ::= peanut
<名 词> ::= elephant

在上述表示法中,每一行称为一条规则式(或规则);在每条规则式中,符号"::="表示它左端的符号由它右端的符号串组成,或者说左端的符号产生右端的符号串。非形式地,人们称这种描述形式为"文法","G"是给该文法起的名字,标在第一条规则式前。称规则式左端的符号为非终极符;只出现在文法规则式右端的符号为终极符;第一条规则式的左端符号为文法的开始符。从文法的开始符出发不断用右端符号串替换左端符号,直至所有符号都不能再替换(全部为终极符)为止,得到的符号串称该文法的句子。文法的所有句子构成的集合称为该文法定义的语言。<一个英语句子>是文法的开始符,该文法的句子有:

The big elephant ate the peanut

The big peanut ate the elephant

The big peanut ate the peanut

The big elephant ate the elephant

这些句子构成的集合称为文法 G 定义的语言。当然,有些句子的意义是不对的,称语义上是错误的,但在形式上或语法上都是正确的。上述描述一个语言语法的方法称为巴科斯范式(简称 BNF)表示法,它是由 J. W. Backus 和 P. Naur 于 1960 年在 ALGOL 60 标准报告中首先提出来的。下边举几个具体文法的例子。

例 1.1 无符号整数集合的文法如下:

G(整数): N ::= D
N ::= ND
D ::= 0
D ::= 1
D ::= 2
D ::= 3
D ::= 4
D ::= 5
D ::= 6
D ::= 7
D ::= 8
D ::= 9

用符号“|”表示“或者”,则描述同一左端符号的不同规则式就可以简写。上述整数文法可简写成:

G(整数): N ::= D | ND
D ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

例 1.2 下述文法 G1 定义集合 $\{0^n1^n \mid _{n\geqslant 1}\}$

G1: S ::= 01 | 0S1

递归定义

注意前述两个例子,其中有规则:

N ::= ND

S ::= 0S1

这种在定义一个符号时又利用该符号本身(例 N, S)的现象称为递归。递归是定义无穷集合的一种常用手段。在实际应用中,递归定义的例子很多。

例 1.3 标识符可以如下定义:

<标识符> ::= <字母> | <标识符> <字母> | <标识符> <数字>

表示:一个字母是标识符;一个标识符后边跟一个字母也是标识符;一个标识符后边跟一个数字还是标识符。即标识符是由字母开头的后边跟任意多个字母或数字组成的字符串。

例 1.4 标识符表可以如下定义:

<标识符表> ::= <标识符> | <标识符表>, <标识符>

表示:一个标识符是标识符表;一个标识符表后边跟一个逗号和一个标识符还是标识符表。

例 1.5　整数表达式可以定义成:

<表达式> ::= <运算分量> | <运算分量> <运算符> <表达式>
<运算分量> ::= <整数> | (<表达式>)
<运算符> ::= + | - | × | /

表示:一个运算分量是表达式;一个运算分量后边跟一个运算符,再跟一个表达式还是表达式。而运算分量是一个整数或一个括号括起来的表达式;运算符包括:+、-、×、/。

1.3.3　文法的其他表示法

除使用 BNF 表示法表示文法外,在文献中还使用其他一些方法表示文法,它们与 BNF 具有相同的表达能力。下边介绍几种其他的文法表示法。

1. 花括号

花括号{ }的定义如下:

{ β } 表示 ε 或 β 或 ββ 或 βββ 或 …

其中,ε 表示空符号串;β 是任意符号串。即花括号"{"和"}"括起来的部分表示这部分可以重复 0 次或任意有限次。这样

{ a } 表示 ε 或 a 或 aa 或 aaa 或 …

{ ,a } 表示 ε 或,a 或,a,a 或,a,a,a 或 …

在标准 BNF 表示法中,为了表示任意长度的串,是通过递归来定义的,例如,假定有非形式化规则式

A ::= a,a,a,…,a

用标准 BNF 表示为

A ::= a | A,a

将花括号用于 BNF 表示法中,上式可以表示为

A ::= a{ ,a }

这里就没有递归了。可以将花括号看成循环标志,表示花括号括起来的部分要循环出现若干次,直到满足需要为止。

用这种表示法,整数的文法可以写成

N ::= D{ D }

D ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

标识符的文法可以写成

<标识符> ::= <字母>{ <字母> | <数字> }

还可以在花括号的闭括号后标以上下角标,下角标表示花括号中的符号串至少重复的

次数,上角标表示花括号中的符号串至多重复的次数。例如,FORTRAN 标识符只有前 6 个字符有效,则其文法可以定义成:

$$<\text{标识符}> ::= <\text{字母}>\{<\text{字母}>\mid <\text{数字}>\}_0^5$$

2. 方括号

方括号"["和"]"用来表示可选择的符号串,定义如下:

[β] 表示 ε 或 β

即方括号"["和"]"括起来的部分表示这部分可以出现一次或不出现。例如,标准 BNF 规则式

$$E ::= T \mid T + E$$

可表示为

$$E ::= T [+ E]$$

花括号"{ "和" }"及方括号"["和"]" 都是元语言符号,其地位与 " ::= " 是一样的,它们在规则式右端出现,表示被括起来的符号串的重复,而不属于符号串的一部分。

3. 语法图

语法图以图的方式表示语法,例如,前述标识符的语法可以用语法图表示,如图 1.3 所示。整数的语法图如图 1.4 所示。语法图的描述方法不是形式化的。语法图可理解成沿箭头方向,从入口到出口,能在该图中通过的任意一个字符序列都是该语言的句子。

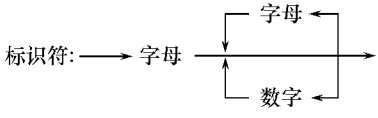

图 1.3 标识符

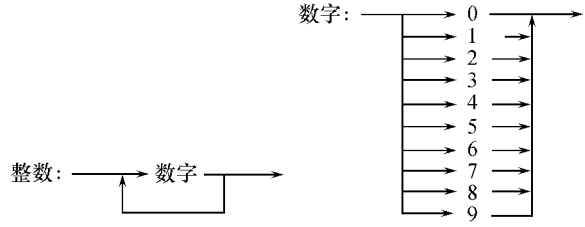

图 1.4 整数

4. 替代符号

由于 C 语言本身使用 BNF 的元语言符号:

" | "、"["、"] "、"{ "、" } "

本书避免使用这些符号作为元语言符号,而是采取如下措施:

(1) 以全角黑竖直符号" ┃ "代替" | ";

(2) 以全角空心方括号"〚"、"〛"分别代替方括号"["、"]";

(3) 不使用花括号"{ "、"}"。

又由于印刷及书写上的原因,本书以全角右箭头" → "替代符号" ::= "。

另外,为了清晰和避免混淆,本书把所有非终极符都用全角尖括号" < "、" > "括起来。