出 版 说 明

在计算机技术迅猛发展的今天，财经管理学科发生了深刻的变化，新思想、新方法、新技术不断涌现，每一位财经工作者都不得不密切注视着计算机在这一学科领域中应用的进展。

近年来，我们陆续编写出版了一批教科书，收到了很好的效果。多年的教学、科研实践为新书的编著积累了丰富的资料；信息技术的日新月异又不断催促我们将最新的科技成果及时反映在教科书中，以便能尽量地缩短教科书与现实世界的“时差”。为此，我们成立了编委会，组织部分具有丰富的教学经验和较高科研水平的教师，重新编写《计算机财经管理丛书》，期望能通过此套丛书的出版，促进和提高计算机在财经管理方面的应用水平，更好地为社会主义市场经济建设服务。

《计算机财经管理丛书》编委会

1996 年 11 月 19 日于南昌

前　言

C 语言随着 UNIX 系统的成功得到普及推广。C 语言简洁、紧凑，数据类型丰富、控制结构完善，运算符丰富、表达能力强，使用灵活方便、应用面广，目标程序效率高、可移植性好，既具有高级语言的优点、又具有低级语言的许多特点。C 语言特别适合于写系统软件，同时也是一种深受广大计算机应用人员喜爱的通用程序设计语言。

目前，在不少高等院校中，不仅是计算机及其应用类专业开设了 C 语言课程，许多其它专业也开设了 C 语言课程。学习 C 语言已成为广大计算机应用人员及爱好者的迫切要求。

对于计算机及其应用类专业的学生，以前大都是以 PASCAL 语言作为第一语言来学习，现在已有不少高校改用 C 语言作为第一语言来学习；对于广大非计算机类专业的学生来说，更是如此。目前的 C 语言教材大多是按已学过一门计算机高级语言（如 PASCAL 语言）来安排编写的。为适应初学者将 C 语言作为第一门计算机高级语言学习的需要，本书在内容的组织上进行了精心的安排。

本书全面系统地介绍了 C 语言，针对初学者的特点在体系结构上作了精心安排，是一本 C 语言的实用教程。本书主要针对 Turbo C 2.0 进行叙述，内容丰富，概念清楚，结构完整，循序渐进，通俗易懂。本书例题丰富，所有例题均在 Turbo C 2.0 系统上运行通过；每章均附有一定数量的习题，习题形式多样。并结合了财经类专业的特点。

全书共分十九章。第一章由程荃华执笔，第二、九、十、十一、十三章由万常选执笔，第三章由杨小平执笔，第四、十六、十七、十八章由舒蔚执笔，第五、六、七、八章由狄国强执笔，第十二、十四、十五章由勒中坚执笔，第十九章由骆斯文执笔。狄国强、舒蔚对第二、九、十、十一、十三章的初稿进行了审阅、修改。万常选对全书的初稿进行了修改、补充和总纂。

本书配有一张 3.5 英寸软盘，软盘上有本书所有例题的源程序及习题的解答和源程序。需要者请与江西财经大学经济信息管理系联系。

由于作者水平有限，经验不足，加上编写时间仓促，书中肯定会有不少缺点或错误，敬请专家和读者批评指教。

作　者

1996 年 11 月

目　录

第一章　预备知识

§1.1　进位数制与不同基数的数之间的转换

1.1.1　二进制数

进位计数制是一种计数的方法，习惯上人们最熟悉、最常用的是十进制计数法。一个任意的十进制数可以表示为：

$$a_n a_{n-1}\cdots a_1 a_0 . b_1 b_2 \cdots b_m$$

其含义是：

$$a_n \cdot 10^n + a_{n-1} \cdot 10^{n-1} + \cdots + a_1 \cdot 10^1 + a_0 \cdot 10^0 + b_1 \cdot 10^{-1} + b_2 \cdot 10^{-2} + \cdots + b_m \cdot 10^{-m}$$

其中 $a_i(i=0,1,\cdots,n)$和 $b_j(j=1,2,\cdots,m)$均是 0、1、2、3、4、5、6、7、8、9 十个数码中的一个。

十进制数的基数为 10，即其数码的个数为 10，且遵循逢十进一的规则。上式中相应于每位数字的 $10^k(n\geqslant k\geqslant -m)$称为该数字的权，所以每位数字乘以其权所得到的乘积之和即为所表示数的值。例如：

$$8045.79 = 8\times 10^3 + 0\times 10^2 + 4\times 10^1 + 5\times 10^0 + 7\times 10^{-1} + 9\times 10^{-2}$$

除十进制数之外，人们还采用了其它的计数方法，例如计时用的分、秒就是按 60 进制计数的。对于任何一个基数为 r 的 r 进制数的值可以表示为：

$$a_n \cdot r^n + a_{n-1} \cdot r^{n-1} + \cdots + a_1 \cdot r^1 + a_0 \cdot r^0 + b_1 \cdot r^{-1} + b_2 \cdot r^{-2} + \cdots + b_m \cdot r^{-m}$$

其中 a_i 和 b_j 可以是 0、1、…、r－1 中的任一个数码，$r^k(n\geqslant k\geqslant -m)$为各位数相应的权。

计算机中为便于存储及计算的物理实现，采用了二进制数。二进制数的基数为 2，只有 0 和 1 两个数码，并且遵循逢二进一的规则，它的各位权是 $2^k(n\geqslant k\geqslant -m)$，因此二进制数 $a_n a_{n-1}\cdots a_2 a_0 . b_1 b_2 \cdots b_m$ 的值是：

$$a_n \cdot 2^n + a_{n-1} \cdot 2^{n-1} + \cdots + a_1 \cdot 2^1 + a_0 \cdot 2^0 + b_1 \cdot 2^{-1} + b_2 \cdot 2^{-2} + \cdots + b_m \cdot 2^{-m}$$

其中 a_i 和 b_j 为均 0、1 两个数码中的一个。例如：

$$(1101.11)_2 = 1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 + 1\cdot 2^{-1} + 1\cdot 2^{-2} = (13.75)_{10}$$

其中数的下标(2 或 10)表示该数的基数 r，即二进制数 1101.11 与十进制数 13.75 等值。

n 位二进制整数可以表示 2^n 个数。例如 3 位二进制整数可以表示 0～7 这 8 个十进制整数，4 位二进制整数可以表示 0～15 这 16 个十进制数。一些常用的十进制数与二进制数的对应关系如表 1－1 所示。

为了便于阅读和书写，人们经常使用八进制数或十六进制数来表示十进制数。它们的基数和数码表示如表 1－2 所示。

按同样的方法，我们可以很容易地掌握八进制数和十六进制数的表示方法。我们从表 1－1 和表 1－2 可以看出：

八进制数是以 8 作为基数，其数码的个数为 8，且遵循逢八进一的规则。而十六进制数则是以 16 作为基数，其数码的个数为 16，且遵循逢十六进一的规则。例如：$(415)_{10}$可以表示为$(637)_8$或$(19F)_{16}$。

在计算机数制的表示方面，通常是用数字后面跟一个英文字母来表示该数的数制。其中十进制数用 D(Decimal)、二进制数用 B(Binary)、八进制数用 O(Octal)、十六进制数用 H(Hexadecimal)来表示。例如：12D，1101101B，07AH。

十进制数	二进制数	八进制数	十六进制	十进制数	二进制数	八进制数	十六进制
0	00000000	0	0	12	00001100	14	C
1	00000001	1	1	13	00001101	15	D
2	00000010	2	2	14	00001110	16	E
3	00000011	3	3	15	00001111	17	F
4	00000100	4	4	16	00010000	20	10
5	00000101	5	5	31	00011111	37	1F
6	00000110	6	6	32	00100000	40	20
7	00000111	7	7	63	00111111	77	3F
8	00001000	10	8	64	01000000	100	40
9	00001001	11	9	127	01111111	177	7F
10	00001010	12	A	128	10000000	200	80
11	00001011	13	B	255	11111111	377	FF

表 1－1　几种常用的进位计数制所表示的数之间的对照

进位计数制	基数	数码
二进制数	2	0，1
八进制数	8	0，1，2，3，4，5，6，7
十进制数	10	0，1，2，3，4，5，6，7，8，9
十六进制数	16	0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F

表 1－2　几种常用的进位计数制的基数和数码

1.1.2　二进制数与十进制数之间的转换

一、二进制数转换为十进制数

各位二进制数码乘以与其对应的权之和即为与该二进制数相对应的十进制数。

例如：

$$1011.101B = 1\cdot 2^3+0\cdot 2^2+1\cdot 2^1+1\cdot 2^0+1\cdot 2^{-1}+0\cdot 2^{-2}+1\cdot 2^{-3}$$
$$= 11.625D$$

二、十进制数转换为二进制数

十进制数转换为二进制数的方法很多，这里只介绍比较简单的降幂法和除 2 取余、乘 2 取整法两种。

（一）降幂法

首先写出要转换的十进制数，其次写出所有小于此数的各位二进制数的权值，然后用要转换的十进制数减去与它最相近的二进制权值，如够减则减去并在相应位记以 1；若不够减则在相应位上记以 0 并跳过此位；如此不断反复，直到该数为 0 为止（若出现不会为 0 的情况，则根据保留位数 N 取前 N 位，以后类似情况均采用此法）。

【例 1.1】A＝119.625D，转换为二进制数。

小于 A 的二进制权为：	64	32	16	8	4	2	1		0.5	0.25	0.125
对应的二进制数是：	1	1	1	0	1	1	1	.	1	0	1

计算过程如下：

$119.625-2^6=119.625-64=55.625$ $(a_6=1)$

$55.625-2^5=55.625-32=23.625$ $(a_5=1)$

$23.625-2^4=23.625-16=7.625$ $(a_4=1)$

$7.625<2^3$ $(a_3=0)$

$7.625-2^2=7.625-4=3.625$ $(a_2=1)$

$3.625-2^1=3.625-2=1.625$ $(a_1=1)$

$1.625-2^0=1.625-1=0.625$ $(a_0=1)$

$0.625-2^{-1}=0.625-0.5=0.125$ $(b_1=1)$

$0.125<2^{-2}$ $(b_2=0)$

$0.125-2^{-3}=0.125-0.125=0$ $(b_3=1)$

所以，A＝119.625D＝1110111.101B。

（二）除 2 取余乘 2 取整法

把要转换的十进制数的整数部分不断除以 2，并记下余数，直到商为 0 为止。最后将所得余数从第一个至最后一个按照从右至左的顺序排列，得到的即为所求十进制整数对应的二进制数。

【例 1.2】A＝118D，转换为二进制数。

$118/2=59$ ——— 余 0 $(a_0=0)$

$59/2=29$ ——— 余 1 $(a_1=1)$

$29/2=14$ ——— 余 1 $(a_2=1)$

$14/2=7$ ——— 余 0 $(a_3=0)$

$$7/2=3 \quad \cdots \text{余}\ 1 \quad (a_4=1)$$

$$3/2=1 \quad \cdots \text{余}\ 1 \quad (a_5=1)$$

$$1/2=0 \quad \cdots \text{余}\ 1 \quad (a_6=1)$$

所以，A＝119D＝1110110B。

对于被转换的十进制数的小数部分则应不断乘以 2，并记下其整数部分，直到结果的小数部分为 0 为止。最后将所得整数从第一个至最后一个按从左至右的顺序排列，得到的即为所求十进制小数对应的二进制数。

【例 1.3】A＝0.6875D，转换为二进制数。

$$0.6875\times2=1.375 \quad \cdots \text{整数部分为}\ 1 \quad (b_1=1)$$

$$0.375\times2=0.75 \quad \cdots \text{整数部分为}\ 0 \quad (b_2=0)$$

$$0.75\times2=1.5 \quad \cdots \text{整数部分为}\ 1 \quad (b_3=1)$$

$$0.5\times2=1.0 \quad \cdots \text{整数部分为}\ 1 \quad (b_4=1)$$

所以，A＝0.6875D＝0.1011B。

1.1.3 二进制数与八进制数、十六进制数之间的转换

一、二进制数与八进制数之间的转换

由于八进制数的基数为 8，又 $8=2^3$，即每位八进制数可以用三位二进制数来表示，因此只要把八进制数中的每一位用 3 位二进制数表示，就形成了相应的二进制数。

【例 1.4】

2	1	6	7	.	5
010	001	110	111	.	101

即 $(2167.5)_8=(10001110111.101)_2$。

反之，要将二进制数转换为八进制数，只要把二进制数的整数部分从低位至高位，小数部分从高位至低位每 3 位组成一组，直接用八进制数来表示就可以了。

【例 1.5】

1	001	101	011	110	.	010	100
1	1	5	3	6	.	2	4

即 $(1001101011110.0101)_2=(11536.24)_8$。

二、二进制数与十六进制数之间的转换

二进制数与十六进制数之间的转换方法同上面介绍的相类似，不同之处在于：由于十六进制数的基数是 16，所以每位十六进制数可以用四位二进制数表示($16=2^4$)。

【例 1.6】

B	2	9	D	.	8
1011	0010	1001	1101	.	1000

即 B29D.8H＝1011001010011101.1B。

【例 1.7】

10	0101	1111	0110	.	1100	0110
2	5	F	6	.	C	6

即 10010111110110.1100011B＝25F6.C6H。

上面讲述的是二进制数与其它进位数制之间的转换，至于十进制数与八进制数和十六进

制数之间的转换跟十进制数与二进制数之间的转换相类似，这里就不再赘述。

§1.2 二进制数的运算

利用二进制数也可以进行跟十进制数相类似的加、减、乘、除四则运算。

1. 加法规则：

0+0=0　　0+1=0　　1+0=1　　1+1=0(进位1)

2. 乘法规则：

0×0=0　　0×1=0　　1×0=0　　1×1=1

【例1.8】

$$\begin{array}{r} 01101011 \\ +10111001 \\ \hline 100100100 \end{array}$$

即 01101011+10111001=100100100

【例1.9】

$$\begin{array}{r} 10110001 \\ -01011011 \\ \hline 01010110 \end{array}$$

即 10110001−01011011=01010110

【例1.10】

$$\begin{array}{r} 11010011 \\ \times\quad 1101 \\ \hline 11010011 \\ 00000000 \\ 11010011 \\ +\quad 11010011 \\ \hline 101010110111 \end{array}$$

即 11010011×1101=101010110111

【例1.11】

$$\begin{array}{r} 10.1 \\ 1101\overline{)100000.1} \\ -\ 1101 \\ \hline 110.1 \\ -110.1 \\ \hline 0 \end{array}$$

即 100000.1÷1101=10.1

§1.3 计算机中数和字符的表示

1.3.1 数的原码、反码和补码表示

计算机中的数是用二进制来表示的，数的符号也是用二进制来表示的。把一个数连同其符号在机器中的表示加以数值化，这样的数称为机器数。一般用最高有效位来表示数的符号，正数用0表示，负数用1表示。机器数可以用不同的码制来表示，常用的有原码、反码和补码。下面分别针对整数和定点小数来讲述其对应的编码方法。

一、定点小数的编码方法

定点小数是指小数点固定在最高数据位的左边的纯小数。用二进制数表示时，小数点的左边表示符号，右边表示该小数的数值，因此在书写时小数点不必明确指出。

(一)原码表示法

1. 定义

若X是定点小数，则：

$$[X]_{原}=\begin{cases}X & 0.5\leqslant X<1\\ 1-X & -1<X\leqslant -0.5\end{cases}$$

2. 举例

$X=+0.1101$　　$[X]_{原}=01101$

$X=-0.1101$　　$[X]_{原}=11101$

通过对定义的分析可以看出，一个定点小数的原码表示实际上就是符号位(0 或 1)后跟小数部分数值位的组合。

(二)反码表示法

1. 定义

若 X 是 n 位(不包括符号位)定点小数，则：

$$[X]_{反}=\begin{cases}X & 0.5\leqslant X<1\\ (2-2^{-n})+X & -1<X\leqslant -0.5\end{cases}$$

2. 举例

$X=+0.1101$　　$[X]_{反}=01101$

$X=-0.1101$　　$[X]_{反}=10010$

从一个数的原码求反码很方便。如果 X 为正数，则$[X]_{反}=[X]_{原}$；如果 X 为负数，则$[X]_{原}$除符号位不变外，每位全求反，就可得到$[X]_{反}$。

(三)补码表示法

1. 定义

若 X 为 n 位(不包括符号位)定点小数，则：

$$[X]_{补}=\begin{cases}X & 0.5\leqslant X<1\\ 2+X & -1<X\leqslant -0.5\end{cases}$$

2. 举例

$X=+0.1101$　　$[X]_{补}=01101$

$X=-0.1101$　　$[X]_{补}=10011$

一个正数的补码与它的原码相同。一个负数的补码表示还可以用下面比较简单的办法来写出：先写出该负数反码表示，然后在其反码的末位(最低位)加 1，就可以得到该负数的补码表示。

【例 1.14】写出 $A=-0.1110011$ 的补码表示。

-0.1110011 的原码为　　11110011

反码为　　10001100

反码末位加 1 后为　　10001101

即$[A]_{补}=10001101$。

在这里，我们顺便说明一下用补码表示数时的符号扩展问题。所谓符号扩展是指一个数从位数较少扩展至位数较多(如从 8 位扩展至 16 位，或从 16 位扩展至 32 位)时应注意的问题。对于用补码表示的数，正数的符号扩展应在前面补 0，而负数的符号扩展应在前面补 1。例如，设机器的字长为 8 位，则：

$[(+35)_{10}]_{补}=00100011$，$[(-35)_{10}]_{补}=11011101$；

如果要把它们从 8 位扩展至 16 位，则：

$[(+35)_{10}]_{补}=0000000000100011$，$[(-35)_{10}]_{补}=1111111111011101$。

二、整数的编码方法

整数可以认为是小数点固定在数值最低位右边的一种数据，其最高位与定点小数一样用作数的符号位。整数的原码、反码和补码这三种编码方法与定点小数的编码方法相类似。下面讨论的整数 X 都是数值位数为 n(不包括符号位)。

(一)原码表示法

1. 定义

$$[X]_{原}=\begin{cases}X & 0\leqslant X<2^n\\ 2^{n+1}+X & -2^n<X<0\end{cases}$$

2. 举例

$X=+1011$　　$[X]_{原}=01011$

$X=-1011$　　$[X]_{原}=11011$

整数的原码表示实际上是符号位(0 或 1)与数值位的组合。

(二)反码表示法

1. 定义

$$[X]_{反}=\begin{cases}X & 0\leqslant X<2^n\\ (2^{n+1}-1)+X & -2^n<X<0\end{cases}$$

2. 举例

$X=+1011$　　$[X]_{反}=01011$

$X=-1011$　　$[X]_{反}=10100$

正整数的反码与它的原码相同；负整数的反码表示实际上是符号位为 1，数值位按原码各位求反。

(三)补码表示法

1. 定义

$$[X]_{补}=\begin{cases}X & 0\leqslant X<2^n\\ 2^{n+1}+X & -2^n\leqslant X<0\end{cases}$$

2. 举例

$X=+1011$　　$[X]_{补}=01011$

$X=-1011$　　$[X]_{补}=10101$

正整数的补码与它的原码相同；负整数的补码表示实际上是在其反码的末位加上 1。

三、浮点数的编码方法

浮点数是指小数点不固定(在数据中的位置可以左右移动)的数据，通常表示成：

$N=M\cdot R^E$　(其中 M 称为浮点数的尾数，R 称为阶的基数，E 称为阶的阶码。)

对于尾数 M，通常用定点小数的形式表示，它给出了有效数字的位数，因此决定了浮点数的精度。对于阶码 E，通常用整数形式表示，它指出了小数点在数据中的位置，因此决定了浮点数的表示范围。对于基数 R，计算机中一般规定为 2(也可用 8 或 16)，不需在浮点数中明确表示出来。浮点数也必须有符号位，实际上一个浮点数在计算机通常表示成下面的格式：

其中 M_h 是尾数的符号位，占一位；E 是阶码，常用补码表示，占 m 位（含一位阶码符号位，阶码数值实际上只有 m－1 位）；M 是尾数，也用补码表示，占 n 位。因此对于用二进制数表示的浮点数，其总长度为 1＋m＋n 位。一般情况下，用 32 位表示的浮点数，符号位占一位，阶码占 8 位，尾数占 23 位，数的表示范围约为 $\pm 1.7\times 10^{\pm 38}$，其精度约为十进制的 7 位有效数字。因为许多实数的位数往往超过 7 位，若用 7 位有效数字表示，多余的位数就必须舍去，因此在计算机中对实数的表示往往有误差。

计算机将符合下面两种情况之一的所有浮点数均看成零值（也称机器零）：

1. 浮点数的尾数为零，阶码不论为何值。

2. 浮点数的尾数不论为何值，阶码的值（负整数）太小以至不能用 m 位数表示出来。

1.3.2 字符的表示

计算机中处理的信息并不全是数，有时也需要处理字符或字符串。例如，从键盘输入的信息或从打印机输出的信息一般是以字符方式输入输出的。因此，计算机必须能表示字符。字符包括：

字母：A、B、C、…、Z、a、b、c、…z；

数字：0、1、…、9；

专用字符：＋、－、＊、/、＃、SP（space 空格）、……；

非打印字符：BEL（Bell 响铃）、LF（Line Feed 换行）、CR（Carriage Return 回车）、……。

这些字符在机器里一般是用八个二进制位的编码来表示，通常用一个字节来保存这 8 个二进制位。目前世界上使用最为广泛的编码方案是 ASCII（American Standard Code for Information Interchange 美国信息交换标准代码）。ASCII 编码方案规定 8 个二进制位的最高一位（一般为 0）用作校验位，余下的 7 位可以组成 128 个编码，分别对应于 128 个不同的字符。其中从 0 至 31 及 127 是控制字符，不能显示和打印，从 32 至 126 则可显示和打印。一些常用字符的 ASCII 码值请见本书的附录一。

习　题　一

1. 把下列各十进制数分别转化为所对应的二进制数、八进制数和十六进制数。

(1)1017　(2)137.625　(3)$25\frac{3}{8}$

2. 将下列各数分别转换为所对应的十进制数。

(1)$(11010111.10011)_2$　(2)$(620.75)_8$　(3)$(7A039D.2E)_{16}$

3. 求下列二进制数表达式的值。

(1)100.011＋11011.01　(2)110010.1－111.011　(3)1011.01×1100.1

(4)1101011.101÷101.1