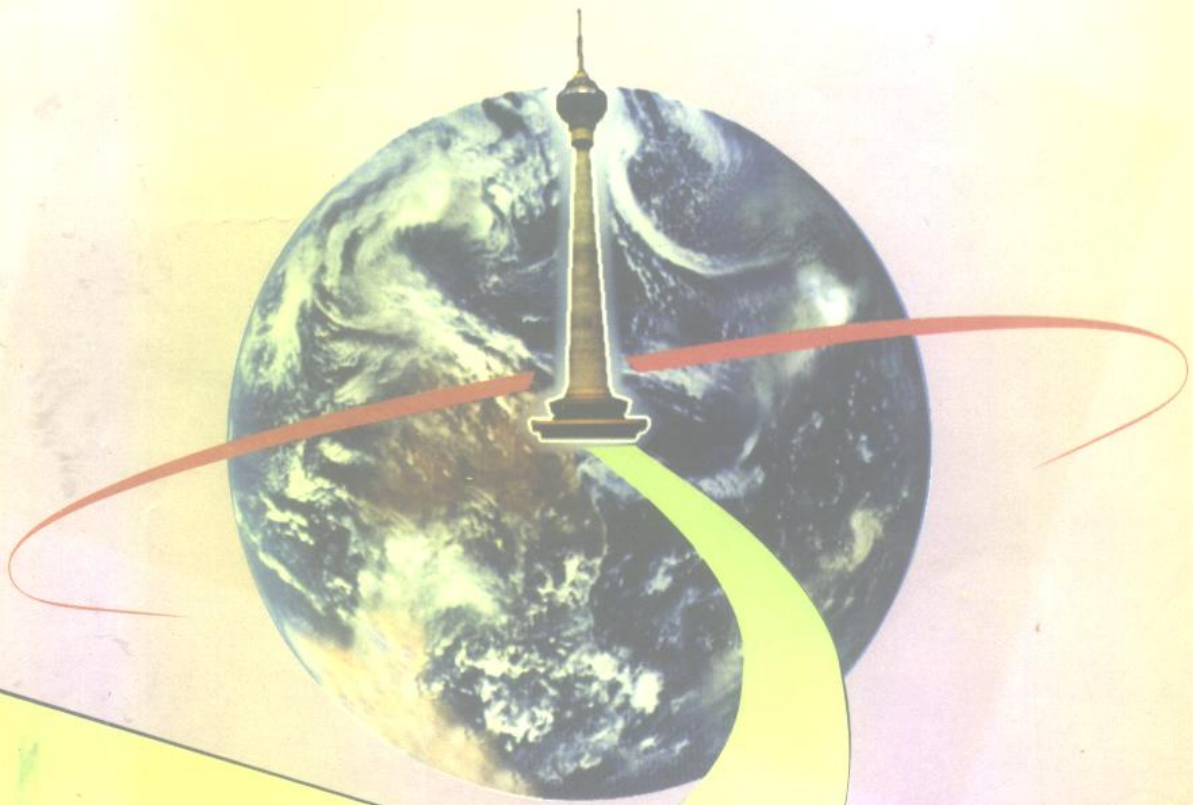
ZXFD 全国非计算机专业等级考试、
自学考试辅导丛书

新编高级C语言程序设计自学辅导

本丛书编委会



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

全国非计算机专业等级考试、自学考试辅导丛书 ·第四册·

新编高级C语言程序设计

自学辅导

本丛书编委会

電子工業出版社

内 容 提 要

本书是全国非计算机专业等级考试、自学考试辅导丛书的第四册，全书以通俗、浅显的文字介绍了C语言的基础知识和程序设计技术。全书突出基本技能的培养，并配有作业及答案。书中收集了初学者易出的错误，并给予答疑辅导。本书适合于参加各类计算机等级考试的读者自学使用，亦可作为计算机基础教育的入门教材。

全国非计算机专业等级考试、自学考试辅导丛书
新编高级C语言程序设计自学辅导
本丛书编委会
责任编辑：杨丽娟
*
电子工业出版社出版
北京市海淀区万寿路173信箱(100036)
电子工业出版社发行　各地新华书店经销
北京科技印刷厂印刷
*
开本 787×1092 毫米　1/16　印张 13.75　字数 335 千字
1996 年 7 月第一版　1996 年 7 月北京第一次印刷
印数：6000　册　定价：16.00 元
ISBN 7-5053-3648-7/TP・1508

前　言

随着科学技术的迅猛发展，计算机已成为各个学科领域不可缺少的应用工具，计算机知识和应用能力已成为当代大学生知识和能力结构的一个重要组成部分，也是我国教育培养跨世纪人才最突出的需要加强的环节之一。目前在高校中普遍开展的计算机知识和应用能力等级考试正有效地推动这一目标的实现。1993 年 12 月国家教委考试中心颁布的在全国进行计算机应用能力认证考试文件，这必将进一步推动全社会学习计算机、使用计算机的热潮。与此有关的教材和参考资料的需求与日俱增。

到目前为止，各省、自治区、直辖市都举办了计算机等级考试，此类的书为数不少，本书的出版正是在充分吸收先期出版的同类图书的优点、克服存在的弊病基础上，推陈出新，更上一层楼。

为了达到这个目的，我们在编撰过程中特别注意了以下三个问题。

(1)适当放低起点，但不降低总的要求，充分贯彻《国家教育委员会全国计算机等级考试大纲》的要求，兼顾目前存在的多种等级考试的要求，循序渐进，深入浅出，对基本内容讲透、讲够，对于易出的错误，给予明确指出，同时对《大纲》进行适当扩充，以保证该书具有一定的实用性与超前性。

(2)注意考试科目的基本知识讲授，在掌握基础知识的同时，适当地进行基本技能的训练，而不以“习题”和“模拟试题”为主。我们认为扎扎实实的掌握基本技能，完全可以达到有关考试的要求。

(3)本书适合于读者自学，也适合于有关专业进行课堂教学，每道习题均有答案，习题本着精辟、典型的原则进行收录。

(4)本书本着实用、广谱的原则，兼容各级各类考试的要求，适合于以下几种考试：

- 各省、市、自治区组织的非计算机专业计算机等级考试
- 国家教委考试中心的非计算机专业计算机能力考试
- 国家教委考试中心的计算机专业的计算机水平和资格考试

全国非计算机专业等级考试、自学考试辅导丛书编委会

本书阅读方法

◁**必学/了解**

这部分内容是基础性的和稍具理论性的，了解它们是为了学习后续内容的需要。

◁**必学/重点**

这部分内容是实质性的，要求准确地理解，熟练地掌握，读者阅读时可多加推敲。

◁**自学答疑**

这部分的内容收集了初学者易产生的一些概念错误、操作错，它是本书的画龙点睛之处，对于缩短学习周期，提高学习效率有重要指导意义，应认真对待。

◁**选　　学**

这部分的内容超出了考试大纲的要求，若读者在实际应用中遇到这些问题或有空余的时间，不妨一看。

◁**编程技巧**

这部分是一些极具实用性的技巧，当你“山穷水尽疑无路”时，它也许能让你有“柳暗花明又一村”的感觉。

◁**作　　业**

这部分收集了一些典型的习题，要求认真完成。

◁**答　　案**

本部分是“作业”部分的答案，也许你的答案和书中的答案不一样，这很正常，也许你的答案更先进(向你祝贺)，也许我们的答案有误(欢迎指正)。

目　录

第一章　高级编程基础

【导读提要】C 语言高级编程的书已为数不少，但何谓“高级编程”，本书作者认为：C 语言中有关位运算、结构类型、存储类型和文件的程序设计可以称为“高级编程”，因为它不仅需要有优良编程素质、掌握基本语法和基本概念，更需要计算机硬件方面的知识，尤其是二进制、数的编码、存储器等方面的知识。为了平稳地由 C 语言初级编程跨入 C 语言高级编程，本章介绍二进制的基本知识与数的编码知识。

本章中要求熟练掌握的内容有：

- 二进制的概念
- 数的编码
- 真值与原码
- 化原码为补码
- 化原码为反码

§1.1　数的进制

一、十进制数与二进制数　　◁必学/了解

人们习惯于使用十进制数，逢十进一，借一当十，那么为什么在计算机中却偏偏采用二进制呢？这是因为电子元器件最容易实现的是电路的通断、电位的高低、电极的正负，而在逻辑学中也常常用到二值逻辑，这都是因为两状态的系统具有稳定性（非此即彼），以及抗干扰性等。为了保证在计算机中进行数据传送、运行中不产生差错，和减少计算机硬件的成本，都必须采用二进制。

二进制数只有“0”和“1”两个数字，而且由低位向高位进位时逢二进一。

像 101，110，110.011 等都是二进制数，但是以上三个数也可以认为是十进制数，为了表示它们的区别，可以给这些数字加上括号和下标，标明是几进制的数，如 $(101)_2$，$(110)_2$ 表示二进制的数；$(110)_{10}$，110 表示十进制的数。

一个十进制数 525，在十进制中说它是 5 个百，2 个十，5 个一的和，也就是：

$$525 = 500 + 20 + 5 = 5 \times 100 + 2 \times 10 + 5 \times 1 = 5 \times 10^2 + 2 \times 10^1 + 5 \times 10^0$$

所以任意的一个十进制数都可以表示成：

$$N = d_m \times 10^m + d_{m-1} \times 10^{m-1} + \cdots + d_0 \times 10^0 + d_{-1} \times 10^{-1}$$

$$+ d_{-2} \times 10^{-2} + \cdots + d_{-n} \times 10^{-n} = \sum_{i=-n}^{m} d_i 10^i \qquad (n,m \geqslant 0) \qquad (1\text{-}1)$$

上式中，$\sum$ 是求和符号；d_i 表示各个位上的数字；m 表示 10 的次幂。

这里我们把10叫做权，把式（1-1）叫做十进制数的按权展开式。它实际上表明了每一位上可取的数字的个数，如10进制每位上可以有0，1，2，…，9十个数字；二进制每一位上可以有0与1两个数字。于是，对于任意r进制数，可能出现的数字是0，1，2，…，$r-1$，共r个。把式（1-1）中的10用r来代替：

$$N = d_m \times r^m + d_{m-1} \times r^{m-1} + \cdots + d_0 \times r^0 + d_{-1} \times r^{-1}$$

$$+ d_{-2} \times r^{-2} + \cdots + d_{-n} \times r^{-n} = \sum_{i=-n}^{m} d_i r^i \qquad (m \geqslant 0, n \geqslant 0) \qquad (1\text{-}2)$$

式（1-2）是任意进制的按权展开式。取式中$r=2$，那么每一位上可取的数字就只有0和1，这就是计算机中广泛使用的二进制数。对于二进制数我们可以把式（1-2）写成：

$$N_2 = b_m \times 2^m + b_{m-1} \times 2^{m-1} + \cdots b_0 \times 2^0 + b_{-1} \times 2^{-1}$$

$$+ b_{-2} \times b^{-2} + \cdots + b_{-n} \times b^{-n} = \sum_{i=-n}^{m} b_i 2^i \qquad (m, n \geqslant 0) \qquad (1\text{-}3)$$

那么，上面提到的几个二进制数可以表示成：

$(101)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$　$(110)_2 = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

每一个十进制数都能找到相对应的二进制数，一些简单数字的二进制和十进制对照如表1-1所示。

表1-1　十进制与二进制对照表

十进制	10	9	8	7	6	5	4	3	2	1	0	0.5	0.25	0.125	0.0625
二进制	1010	1001	1000	111	110	101	100	11	10	1	0	0.1	0.01	0.001	0.0001

二、二进制数的运算　　◁必学/了解

因为二进制数只有0、1两个数字，所以它的四则运算特别简单，运算规则如表1-2（a）与表1-2（b）所示。

表1-2（a）　加法

+	0	1
0	0	1
1	1	10

表1-2（b）　乘法

×	0	1
0	0	0
1	0	1

对于加法运算，遵循“逢二进一”的法则；作减法时，要遵循“借一当二”的法则。如图1-1所示。

```
111+101=1100        111-010=101        1011-101=110

   111                 111                1011
+) 101              -) 010             -) 101
-------             -------            -------
  1100                101                110
```

（a）加法　　（b）减法　　（c）减法

图1-1　二进制数运算

三、十六进制数　　◁必学/了解

按照式（1-2）取 $r=16$，就得到 16 进制数的展开形式。但是十六进制数要有十六个数字，而常用的阿拉伯数字只有 0～9 十个数字，为此，需要引入 A～F 来表示其余的 5 个数字，参见表 1-3。

表 1-3　十进制与十六进制数的对应关系

十进制	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
十六进制	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

十六进制数与二进制数的对应关系见表 1-4。

表 1-4　十六进制数与二进制数的对应关系

十六进制	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
二进制	0	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111

十六进制可以用括号加上下标来表示，如 $(3FD)_{16}$，$(068E)_{16}$等。

数的表示法总结　　◁自学答疑

① 十进制数也用数字后加上英文“d”或“D”来表示，如 126= $(126)_{10}$=126D=126d

② 二进制数则以其后加上“B”或“b”来表示，如 $(11000)_2$=11000B=11000b

③ 十六进制数可以在数字后加上“H”或“h”来表示，如 $(3FD)_{16}$=3FDH=3FDh

十进制、十六进制、二进制和八进制数的书写表示法见表 1-5。

表 1-5　各种进制数表示方法函数

数　制	常规表示示范	C 语言中的表示方式
十进制	126，$(126)_{10}$，126D，126d	126，1.26e2
二进制	$(1011)_2$，1011B，1011b	无
八进制	$(247)_8$，247O，247o	0247
十六进制	$(101f)_{16}$，101fH，101fh	0x101f

§1.2　数的编码

C 语言是为了描述系统而设计的，它最早用于编写 UNIX 操作系统，因此，它具有一定的汇编语言功能，本章的位运算以及第三章的地址运算都属于汇编语言的功能。

所谓位运算是指进行二进制位的运算。在系统软件的编制中常要用到位运算。

一、位的概念　　◁必学/了解

大多数计算机系统的内存储器是由众多的存储单元构成的，在微机中，每个存储单元是 1 个字节，它由 8 位二进制数构成，可以表示 2^8=256 种信息，各位的编号从 0～7，最

左边的位（第7位）是最高位，最右边的位（第0位）是最低位。由于二进制本身的特点，各位上的数字不是1，便是0，见图1-2。

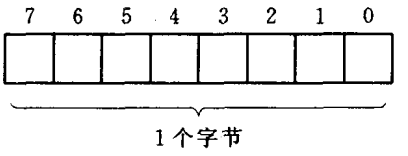


图1-2 一个存储单元的各位

本章中的位就是指上述提到的二进制位，本章中的位运算就是指对这些二进制的位进行逻辑运算、移位运算等操作。

二、数的编码 ◁必学/了解

我们早已知道，数在计算机中是以二进制表示的，然而，并不是像读者所想像的那样简单与直接，计算机中的数的确是以二进制来表示的，但是并不是简单地以它本身的数值的二进制形式来直接表示，而要进行一定的编码，以方便计算机进行处理，常用的编码有原码、反码、补码三种。

三、真值与原码 ◁必学/了解

我们将一个十进制数的二进制表示称为这个十进制数的真值。它代表了这个十进制数本身的数值。表1-6列出了一些数的真值。

表1-6 真值举例

数	二进制表示	真　　值（16位）
−7	−111	−0000000000000111
−1	−1	−0000000000000001
0	0	0000000000000000
1	1	0000000000000001
⋮	⋮	⋮
7	111	000000000000111
⋮	⋮	⋮
15	1111	0000000000001111
⋮	⋮	⋮
255	11111111	0000000011111111
⋮	⋮	⋮
4095	111111111111	0000111111111111
⋮	⋮	⋮
65535	1111111111111111	1111111111111111

可见，用真值表示的数只能是正数，对于负数，要用“−”号标明，这势必造成用计算机表示数时的不便，故引入了原码表示法。

原码表示法中，最高位代表符号位，用“1”表示负数，用“0”表示正数，余下的数位用来表示真值的绝对值。原码表示的例子见表1-7。

表 1-7　原码示例

数	真　　值（16 位）	原　　码（16 位）
−7	−0000000000000111	1000000000000111
⋮	⋮	⋮
−0	−0000000000000000	1000000000000000
+0	+0000000000000000	0000000000000000
⋮	⋮	⋮
7	+0000000000000111	0000000000000111

可见，数字零存在着两种表示方法（+0 与−0）。

四、反　　码　　◁必学/了解

若采用反码表示，则对应的原码应按照以下方法进行转换：

① 如果真值为正，则它的反码与原码同。

② 如果真值为负，则反码的符号位为 1，其余各位就是对原码取反（即原码的 1 变为 0，原码的 0 变为 1），具体的例子见表 1-8，同样，0 的表示也不唯一。

表 1-8　真值、原码、反码对照示例

数	真　　值	原　　码	反　　码
−7	−0000000000000111	1000000000000111	1111111111111000
⋮	⋮	⋮	
−0	−0000000000000000	1000000000000000	1111111111111111
+0	+0000000000000000	0000000000000000	0000000000000000
⋮	⋮	⋮	
+7	+0000000000000111	0000000000000111	0000000000000111

五、补　　码　　◁必学/了解

1. 为什么要引入补码

补码具有许多独特的优点，它可以变减法运算为加法运算，使得计算时步骤统一，速度提高，其次，在这种系统下的“0”只有唯一的一种表示方法。这就是现代的计算机系统中大多采用补码表示的原因。因篇幅所限，本书只介绍补码的表示法，至于为什么补码系统能起到上述作用，请查阅有关资料，本书从略。

2. 补码的规定

① 正数的原码、补码、反码均相同；

② 计算负数的补码时，先置符号位为 1，再对剩余原码的位数逐位取反，之后对整个数加 1。补码的示例见表 1-9。

表 1-9 补码举例

数	原　码	反　码	补　码
−7	1000000000000111	111111111111000	1111111111111001
⋮	⋮	⋮	⋮
−1	1000000000000001	1111111111111110	1111111111111111
−0	1000000000000000	1111111111111111	0000000000000000
+1	0000000000000001	0000000000000001	0000000000000001
⋮	⋮	⋮	⋮
+7	0000000000000111	0000000000000111	0000000000000111

在微机上以8位为一字节的存储单元中，采用补码系统，则可以存放的最小整数为−128，最大整数为+127。若采用两个字节来表示一个整数，则可表示的最小整数为−32768最大整数为+32767，见表1-10的分析（以8位为例）。

表 1-10 补码的表示范围

数域	数　值	补　码
零	0	00000000
负整数	−1	11111111
	−2	11111110
	−3	11111101
	−4	11111100
	⋮	⋮（往下不断减1）
	−127	10000001
	−128	10000000
正整数	1	00000001
	2	00000010
	⋮	⋮（往下不断加1）
	126	01111110
	127	01111111

作　业　◁必　做

一、选　择

（　）1. 对于r进制数，每一位上的数字可以有几个？

(A) r　　(B)　　(C) r/2　　(D) r+1

（　）2. 在计算机领域，常常用到的数制有：

(A) 二进制　　(B) 八进制

(C) 十六进制　　(D) 四进制

（　）3. 在计算机中采用二进制，是因为：

（A）这样可以降低硬件成本

（B）两个状态的系统具有稳定性

（C）二进制的运算法则简单

（D）十进制无法在计算机中实现

二、填　表

表 1-11

数	原码（8 位）	补码（8 位）	反码（8 位）
−127			
−64			
32			
15			
0			

答　案　　◁仅供参考

一、选　择

1. A　2. ABC　3. ABC

二、填　表

表 1-12

数	原码（8 位）	补码（8 位）	反码（8 位）
−127	11111111	10000001	10000000
−64	11000000	11000000	10111111
32	00100000	00100000	00100000
15	00001111	00001111	00001111
0	00000000 10000000	00000000	00000000 11111111

第二章　位运算

【导读提要】位运算是初学C语言的读者的一大障碍，究其原因是：第一，不知计算机中的几种基本数值编码，故第一章先介绍了此方面的知识；第二，对“位逻辑”运算概念模糊，应知道“位逻辑”运算是基于“位”的逻辑运算。在介绍了有关基础知识之后，本章最后还介绍了数制转换的理论与编程。本章属较高的要求，故不配作业、习题。

本章中要求熟练掌握的内容有：

- 位运算的概念、种类
- 按位与运算
- 按位或运算
- 按位非运算
- 按位异或运算

本章的难点在于：

- 数制转换理论
- 数制转换的编程实践
- “&&”与“&”的区别
- “||”与“|”的区别

§2.1　位逻辑运算

一、位运算总述　　◁必学/了解

C语言提供如表2-1所列出的位运算符。

表2-1　位逻辑运算与移位运算

类别	运算符	含　义
位逻辑运算符	&	按位与
	|	按位或
	^	按位异或
	~	取反
移运位算	<<	左移
	>>	右移

【说明】

① 位运算符中，除取反运算以外，均为二元运算符，即要求算符两侧各有一个运算量。

② 运算量只能是整型或字符型的数据，不能为实型或结构类型的数据。

③ 本节中先介绍逻辑运算，下一节中再介绍移位运算。

二、“按位与”运算　　<必学/了解

参加运算的两个运算量，如果它们对应的位都为1，则该位的结果值为1，否则该位的结果值为0（见表2-2）。

【例2.1-1】计算3&5。

表达式3&5的值不等于8，因为这是按位与运算，应先把3和5以补码表示，再进行按位与运算（见表2-3），结果是1的补码，因此，3&5的值为1。

表2-2　按位与运算规则

&	0	1
0	0	0
1	0	1

表2-3　位运算示例

数	补　码
3	0000000000000011
5	0000000000000101
&运算结果	0000000000000001

三、“按位与”运算的特殊用途　　◁必学/了解

1. 清零

如果想将一个存储单元清零，即使其全部二进位为0，可按这样的方法计算：

① 找一个数，它的补码形式中各位的值符合如下条件：原来的数中为1的位，新数中相应位为0（注意，并不要求原数为0的位上，新数相应位为1，此时，新数相应位可以是0或1）。

② 对二者进行&运算。

【例2.1-2】清零的算法。

设原数为0000000000101011，另找一个数，设它为0000000010010100，它符合上述条件，即在原数为1的位置上，它的位值均为0。将两个数进行&运算（如表2-4所示）。

2. 取一个数中某些字节

对于一个整数a（占2个字节），如要想得到其中的低字节，只需将a与特定的一个数按位与即可。

【例2.1-3】取指定数的低字节。

设a的值为0x2cac，b为0x00ff则计算过程见表2-5所示，因为其中c=a&b，故c只取a的低字节，高字节为0。如果想取两个字节中的高字节，只需令c=a&b，b=0xff00，这时，计算过程如表2-6所示。

表2-4　清零运算示例

数	补　码
原数	0000000000101011
使其清零的数	0000000010010100
&运算结果	0000000000000000

表2-5　取指定位示例

数	十进制	十六进制	补　码
a	11436	0x2cac	0010110010101100
b	255	x00ff	0000000011111111
&运算结果		0x00ac	0000000010101100