青少年学科奥林匹克竞赛丛书

信息学(计算机)奥林匹克

中级本

主　编　李竹君
副主编　章维铣
主　审　蒋新儿

江苏省中学生学科奥林匹克竞赛委员会组织编写

南京大学出版社

青少年学科奥林匹克竞赛丛书
江苏省中学生学科奥林匹克竞赛委员会组织编写

信息学(计算机)奥林匹克

中级本

主编　李竹君　副主编　章维铣
主审　蒋新儿

南京大学出版社

丛 书 名　青少年学科奥林匹克竞赛丛书
书　　名　信息学(计算机)奥林匹克(中级本)
主　　编　李竹君　　副主编　章维铣
主　　审　蒋新儿
责任编辑　恽玲聆
责任校对　王金祥
出版发行　南京大学出版社
(南京汉口路22号南京大学校内　邮编210093)
印刷　阜宁印刷厂
经销　全国各地新华书店
开本:787×1092 1/16　印张:14.5　字数:359千
1999年6月第1版　1999年6月第1次印刷
印数　1—15000
定价　19.00元
ISBN　7-305-03413-4/TP·186

前　言

全国青少年信息学(计算机)竞赛活动是经中国科协、教育部批准的一项全国性青少年学科竞赛活动。它吸引了许多青少年学习计算机知识,学习程序设计方法,积极参与各级各类的竞赛。为了使广大读者能较快地掌握有关程序设计的基本方法,学习一些简单的数据结构和算法,我们组织编写了这一本培训教材。

第一章介绍了程序设计中经常处理的最基本的元素——数及其运算基础,从中也可了解计算机对数据的表示方式、运算处理等基本原理。第二章介绍了 PASCAL 语言的基本语句和编写程序的一般方法,使读者特别是已学习过 BASICA 语言的同学能较快地熟悉 PASCAL 语言。第三章介绍了结构化程序设计的基本方法。第四章介绍了一些简单的算法设计方法,以及设计算法时经常采用的策略和技术。第五章以实例的形式来设计或评议一些典型问题的算法设计,并以完整的 PASCAL 程序来描述设计的算法,最后提供上机运行的结果以加深对比和理解,使读者能模仿例题,自己编写求解同类问题的程序。

本书由李竹君主编,第一章由李竹君编写,第二章由陈永财编写,第三章由曹文编写,第四章由章维铣编写,第五章由章维铣和曹文编写。全书由蒋新儿主审统稿。本书在编写过程中,得到了江苏省青少年计算机教育活动中心的王晓敏、赵聆老师的大力帮助和支持,在此表示由衷的感谢。

由于编者的水平关系,以及时间仓促等原因,书中有不当和疏漏之处,在所难免,恳请广大读者批评指正。

编　者

1999.3.26.

第一章　计算机中数的表示及运算

关于记数制我们并不陌生，实际上我们在日常的生活中已接触到了多种记数制，比如：十进制和六十进制。六十进制——六十秒为1分，六十分为1小时，即基数等于60。运算规则是“逢六十进一”，可取的数字符号有0,1,2,…,59共六十个。还可举出许多种记数制，如12支铅笔为“一打”，这是十二进制；七天为一星期，这是七进制。这里，记数制的基数正好等于该数制中数字符号的个数。下面介绍在计算机中常用的数。

1.1　常用数制及其相互转换

1.1.1 十进制数

人们通常使用的是十进制数。十进制数的特点是有十个不同的数字：0,1,2,…,9，并且是“逢十进一”。

在十进制中，各位数字所表示的值不仅与该数字有关，而且与它所在的位置有关。例如在数33中，十位上的3表示3个10，个位上的3表示3个1。每个数位被赋于一定的位值，称为权。十进制中，个、十、百、千……各位的权，依次为1,10,100,1000,…。每个数位的数字所表示的值是这个数字与它相应权的乘积。

对于任意一个十进制数，都可以表示成按权展开的多项式。例如：

$$1804 = 1\times10^3 + 8\times10^2 + 0\times10^1 + 4\times10^0$$

$$48.25 = 4\times10^1 + 8\times10^0 + 2\times10^{-1} + 5\times10^{-2}$$

一般地，对于一个十进制数 $a_n a_{n-1}\cdots a_1 a_0 \cdot a_{-1}\cdots a_{-m}$（这里不考虑其正负），它可表示为：

$$\begin{aligned} &a_n a_{n-1}\cdots a_1 a_0 \cdot a_{-1}\cdots a_{-m} \\ &= a_n 10^n + a_{n-1}10^{n-1} + \cdots + a_1 10^1 + a_{-1}10^{-1} + \cdots + a_{-m}10^{-m} \\ &= \sum(a_i 10^i) \end{aligned}$$

其中，a_i 是0～9十个数字中的任意一个，m,n为正整数。

这里的10被称为基数，它是相邻数位的权之比。各数位的权是基数10的整数次幂。有时，在一个数的下方注上基数10，来更明确地指出它是一个十进制数。例如：$(13)_{10}$，$(8.2)_{10}$等。

1.1.2 二进制数

在电子计算机中采用的是二进制。二进制数只需两个不同的数字：0和1，并且是“逢二进一”。它的基数是2，各数位的权也是基数的整数次幂。对于二进制数，其整数部分各数位的权，从最低位开始依次是1,2,4,8,…；其小数部分各数位的权，从最高低位开始依次是0.5,0.25,0.125,…。

与十进制数相仿，一个二进制数也可表示成按权展开的多项式。例如：

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

$(10.11)_2 = 1\times 2^1 + 0\times 2^0 + 1\times 2^{-1} + 1\times 2^{-2}$

这里要注意，不要把二进制$(101)_2$读作“一百零一”，而应读成“一零一”。

计算机采用二进制，而不采用十进制，是因为二进制数具有以下一些主要特点：

(1) 二进制数只含有两个数字0和1，因此可用大量存在的具有两个不同的稳定物理状态的元件来表示。例如，可用指示灯的不亮和亮，继电器的断开和接通，晶体管的截止和导通，磁性元件的反向和正向剩磁，脉冲电位的低和高等等，来分别表示二进制数字0和1。计算机中采用具有两个稳定状态的电子或磁性元件表示二进制数，这比十进制的每一位要用具有十个不同的稳定状态的元件来表示，实现起来要容易得多。同时，由于表示二进制数的元件的状态数少，因而数据传送不易出错，工作稳定可靠。

(2) 二进制数的运算规则简单，使得计算机中的运算部件的结构相应变得比较简单。

二进制数的加法和乘法的运算规则，都比十进制数的简单得多。它的加法规则和乘法规则都只有四条：

$0+0=0 \quad 0+1=1 \quad 1+0=1 \quad 1+1=10$

$0\times 0=0 \quad 0\times 1=0 \quad 1\times 0=0 \quad 1\times 1=1$

实际上，二进制数的乘法可通过简单的移位和相加来实现。由于二进制数的运算规则简单，它的算术运算都可通过移位、相加这种简单的操作来实现。这就使计算机的运算部件的结构简单，运行可靠。

(3) 二进制数的两个数字0和1与逻辑代数的逻辑变量取值一样，从而可采用二进制数进行逻辑运算。这样就可以应用逻辑代数作为工具来分析和设计计算机中的逻辑电路，使得逻辑代数成为计算机设计的数学基础。

1.1.3 二进制数与十进制数的相互转换

(1) 二进制数转换成十进制数(乘权求和)：把一个二进制数转换成十进制数，只需将二进制数按权展开求和，即“乘权求和”。

例 1-1：把$(1101.11)_2$转换成十进制数

解：$(1101.11)_2 = 1\times 2^3 + 1\times 2^2 + 1\times 2^0 + 1\times 2^{-1} + 1\times 2^{-2}$

$= 8 + 4 + 0 + 1 + 0.5 + 0.25$

$= 13.75$

(2) 十进制数转换成二进制数(除以2反序取余)：如果一个十进制整数X，已被表示成一个二进制数$(a_n a_{n-1}\cdots a_0)$，那么X可按二进制数的权展开如下：

$X = a_n 2^n + a_{n-1}2^{n-1} + \cdots + a_1 2^1 + a_0 2^0$

由于展开式的前n项均为2的整数倍，因此a_0即为X除以2所得的余数。也就是说，X/2的商为$a_n 2^{n-1} + a_{n-1}2^{n-2} + \cdots + a_1 2^0$，余数为$a_0$。

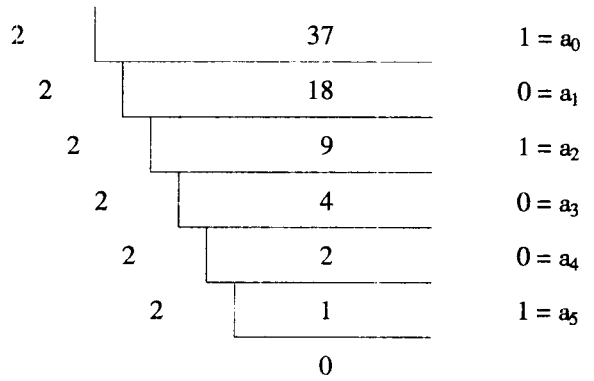
同样，上述商$a_n 2^{n-1} + a_{n-1}2^{n-2} + \cdots + a_1 2^0$再除以2，所得余数是$a_1$。

依此类推，一直除下去，直到商为0为止，这时的余数就是a_n。

用这样的方法就可以得到所求二进制数的各位上的数字$a_0, a_1, \cdots, a_n$。将这些数字逆序排列即得所求的二进制数。我们把这个方法称为“除以2反序取余”。

例 1－2：把 37 转换成二进制数

解：我们把解题过程写成如下格式：



在这里，我们把 37 除以 2 的商写在 37 的下面，余数 1 写在 37 的右面，然后对所得的商 18，继续用 2 来除，直至商为 0。上述 $a_5, a_4, \cdots, a_0$ 依次为 1,0,0,1,0,1。

于是得：$37 = (100101)_2$

(3)十进制小数转换成二进制小数(乘 2 取整)：一个十进制纯小数转换成二进制纯小数，其方法如下：先用 2 乘这个十进制纯小数，然后去掉乘积中的整数部分；用 2 乘剩下的小数部分，再去掉乘积中的整数部分，如此下去，直到乘积的小数部分为 0 或者已得到所要求的精确度为止。把上面每次乘积的整数部分依次排列起来，就是所要求的二进制小数。我们把这个方法简称为“乘 2 取整”。下面的例子将进一步说明这个方法。

例 1－3：把 0.6875 转换成二进制小数

解：设 0.6875 转换成二进制小数 $0.a_{-1}a_{-2}\cdots a_{-m}$，其中 a_{-i} 是 0 或 1($1 \leqslant i \leqslant m$)，那么有：

$$0.6875 = (0.a_{-1}a_{-2}\cdots a_{-m}) = a_{-1}2^{-1} + a_{-2}2^{-2} + \cdots + a_{-m}2^{-m}$$

转换过程，可写成如下形式：

$$\begin{array}{l} 0.6875 \\ \underline{\times) \quad 2} \\ a_{-1} = 1 \cdots\cdots 1.3750 \\ \underline{\times) \quad 2} \\ a_{-2} = 0 \cdots\cdots 0.7500 \\ \underline{\times) \quad 2} \\ a_{-3} = 1 \cdots\cdots 1.5000 \\ \underline{\times) \quad 2} \\ a_{-4} = 1 \cdots\cdots 1.0000 \end{array}$$

这里，我们把 0.6875 乘以 2 的乘积 1.3750 写在 0.6875 的下面。接着，对上述乘积的小数部分 0.3750 继续乘以 2，直到乘积中的小数部分为 0，或者达到所要求的精度为止。最后，从上到下依次记下左侧各乘积的整数部分，即为所求。于是得 $0.6875 = (0.1011)_2$。

要注意，一个有限的十进制小数并非一定能够转换成一个有限的二进制小数，即上述过程的乘积的小数部分可能永远不等于 0，这时，我们可按要求进行到某一精确度为止。

如果一个十进制数既有整数部分，又有小数部分，则可将整数部分和小数部分分别进行转换，然后再把两部分结果合并起来。

1.1.4 八进制数

由于二进制数的特点，电子计算机中采用二进制。但是，一个数值的二进制表示，所需位数较多，造成读写不方便。为此，在有关计算机的讨论中，人们还经常使用八进制和十六进制。

八进制数的基数为 8，有八个数字：0，1，…，7，并且是“逢八进一”。

由于八进制数的基数 8 是二进制的基数 2 的 3 次幂，即 $2^3 = 8$，所以一位八进制数相当于三位二进制数。这样使得八进制数与二进制数的相互转换十分方便。

八进制数转换成二进制数时，只要把八进制数的每一位改成等值的三位二进制数，即“一位变三位”。

例 1－4：把 $(56.103)_8$ 转换成二进制数

解：因为

5	6	1	0	3
↓	↓	↓	↓	↓
101	110	001	000	011

所以 $(56.103)_8 = (101110.001000011)_2$

二进制数转换成八进制数时，从小数点开始向两边以三位为一段（不足补 0），每段改成等值的一位八进制数即可，即“三位变一位”。

例 1－5：把 $(11101.1101)_2$ 转换成八进制数

解：因为

011	101	110	100
↓	↓	↓	↓
3	5	6	4

所以 $(11101.1101)_2 = (35.64)_8$

八进制数与十进制数互相转换的方法，和二进制数与十进制数互相转换的方法完全类似。

例 1－6：把十进制数 314.15 转换成八进制数

解：对整数部分 314，用除以 8 取余

8	314	2
8	39	7
8	4	4
	0	

对小数部分 0.15，用乘 8 取整

$$
\begin{array}{r}
0.1500 \\
\times) \quad 8 \\
\hline
1\cdots\cdots 1.2000 \\
\times) \quad 8 \\
\hline
1\cdots\cdots 1.6000 \\
\times) \quad 8 \\
\hline
4\cdots\cdots 4.8000 \\
\times) \quad 8 \\
\hline
6\cdots\cdots 6.4000
\end{array}
$$

得 $314.15 = (472.1146\cdots)_8$

例 1-7：把$(17.26)_8$转换成十进制数

解：用乘权求和的方法

$$
\begin{aligned}
(17.26)_8 &= 1\times 8^1 + 7\times 8^0 + 2\times 8^{-1} + 6\times 8^{-2} \\
&= 1\times 8 + 7\times 1 + 2\times 0.125 + 6\times 0.015625 \\
&= 15.34375
\end{aligned}
$$

1.1.5 十六进制数

十六进制数的基数为16，有十六个数字：0，1，…，9，A，B，C，D，E，F，并且是“逢十六进一”。

由于 $2^4 = 16$，所以一位十六进制数相当于四位二进制数。这样，仿照八进制数与二进制数的相互转换，十六进制数与二进制数的相互转换也十分方便。

例 1-8：把$(3AD.B8)_{16}$转换成二进制数

解：可用“一位变四位”的方法

3	A	D	B	8
↓	↓	↓	↓	↓
0011	1010	1101	1011	1000

所以$(3AD.B8)_{16} = (1110101101.10111)_2$

例 1-9：把$(1111100111.111111)_2$转换成十六进制数解：可用“四位变一位”的方法

0011	1110	0111	1111	1100
↓	↓	↓	↓	↓
3	E	7	F	C

所以$(1111100111.111111)_2 = (3E7.FC)_{16}$

至于十六进制数与十进制数之间相互转换的方法，完全与八进制数与十进制数之间的转换类似，同学们可以自己进行练习。

为便于区别和书写，通常在数的后面加上字母 B(Binary)表示二进制数，加上 Q(Octal，为了区别数字 0，不用字母 O，而用 Q)表示八进制数，加上 H(Hexadecimal)表示十六进制数，如 1001B，703Q，93H 等。

表 1-1　列出了一些数的二、八、十和十六进制形式

二进制	八进制	十进制	十六进制	二进制	八进制	十进制	十六进制
0000	0	0	0	1001	11	9	9
0001	1	1	1	1010	12	10	A
0010	2	2	2	1011	13	11	B
0011	3	3	3	1100	14	12	C
0100	4	4	4	1101	15	13	D
0101	5	5	5	1110	16	14	E
0110	6	6	6	1111	17	15	F
0111	7	7	7	10000	20	16	10
1000	10	8	8	10001	21	17	11

1.2　计算机中数的表示

在计算机中，所有的数据、指令以及一些符号等都是用特定的二进制代码表示的。

我们把一个数在计算机内被表示的二进制形式称为机器数，该数称为这个机器数的真值。机器数具有下列特点：

(1)由于计算机设备上的限制和操作上的便利，机器数有固定的位数。它所表示的数受到固定位数的限制，具有一定的范围，超过这个范围就会产生“溢出”。例如，一个 8 位机器数，所能表示的无符号整数的最大值是“11111111”，即十进制数 255。如果超过这个值，就会产生“溢出”。

(2)机器数把其真值的符号数字化。通常是用机器数中规定的符号位(一般是最高位)取 0 或 1，来分别表示其真值的正或负。例如，一个 8 位机器数，其最高位是符号位，那么在定点整数原码表示情况下，对于 00101110 和 10010011，其真值分别为十进制数 +46 和 -19。

(3)机器数中，采用定点或浮点方式来表示小数点的位置。关于数的定点和浮点表示形式，详见本节稍后的进一步说明。

1.2.1 原码、反码和补码

机器数的形式是人们规定的，原码和补码是最常见的机器数形式或称数的编码方式。为简单起见，这里仅介绍整数的原码和补码。

1.*原码*

整数 X 的原码是指：其符号位的 0 或 1 表示 X 的正或负，其数值部分就是 X 绝对值的二进制表示。通常用$[X]_{原}$表示 X 的原码。

例如，假设机器数的位数是 8，其中最高位是符号位，其余是数值部分，那么：

$$[+17]_{原}=00010001 \qquad [-39]_{原}=10100111$$

值得注意的是，由于$[+0]_{原}=00000000$，$[-0]_{原}=10000000$，所以数 0 的原码不唯一，有“正零”和“负零”之分。

2.*反码*

在反码表示法中，正数的表示方式与原码相同；负数的补码是把其原码除符号位外的各

位取反(即0变1,1变0)。通常,用$[X]_{反}$表示X的反码。

例如:$[+45]_{反}=[+45]_{原}=00101101$

由于$[-32]_{原}=10100000$,所以$[-32]_{补}=11011111$。

3.补码

在进行减法运算时,数的原码表示显得很不方便,于是就引进了数的补码表示。

整数的补码可用下列简单的方法求得:正数的补码与其原码相同;负数的补码是在其反码的最低有效位上加1。通常,用$[X]_{补}$表示X的补码。

例如:$[+14]_{补}=[+14]_{原}=00001110$。

由于$[-36]_{原}=10100100$,而$[-36]_{反}=11011011$,所以$[-36]_{补}=11011100$。

数0的补码表示是唯一的,即$[0]_{补}=[+0]_{补}=[-0]_{补}=00000000$。

利用公式$[X]_{补}+[\pm Y]_{补}=[X\pm Y]_{补}$,可以把加法和减法统一成加法。这里要注意的是:把符号位和其他位上的数一样运算,如果符号位上有进位,则把这个进位1舍去不要(这里不考虑"溢出"问题)。

例1-10:已知X=6,Y=2,求X-Y

解:$[X]_{补}=00000110$　　$[-Y]_{补}=11111110$

$$\begin{array}{r} 00000110 \\ +)\ 11111110 \\ \hline 100000100 \end{array}$$

所以,$[X-Y]_{补}=00000100$,即X-Y=4

例1-11:已知X=-19,Y=-30,求X+Y

解:$[X]_{补}=11101101$　　$[-Y]_{补}=11100010$

$$\begin{array}{r} 11101101 \\ +)\ 11100010 \\ \hline 111001111 \end{array}$$

所以,$[X+Y]_{补}=11001111$,即X+Y=-49

从上面例子可见,加法和减法统一成了加法。再由于乘除可通过移位和加减来实现。于是就使四则算术运算在计算机中能转化成对补码进行简单的移位和相加,从而大大简化了计算机运算部件的电路设计。

1.2.2 数的定点和浮点表示

数的补码表示解决了带符号数的运算问题。至于小数点的处理,计算机中通常用定点表示法或浮点表示法来解决。

1.定点表示法

定点表示法是把小数点约定在机器数的某一固定的位置上。如果小数点约定在符号位和数值的最高位之间,这时所有参加运算的数的绝对值小于1,即为定点纯小数。

例如:$[X]_{补}=01010000$,这时X=0.625

↑(小数点位置)

如果小数点约定在数值的最低位之后,这时所有参加运算的数都是整数,即为定点整数。

例如：$[X]_{补}=11010000$，这时 $X=-48$

↑（小数点位置）

由于 $[M]_{补}=01111111$，即 $M=+(2^7-1)=+127$

$[N]_{补}=10000000$，即 $N=-2^7=-128$

所以 8 位定点整数（补码表示）的范围是 $-128\sim+127$。

相仿可得，16 位定点整数（补码表示）的范围是 $-32768\sim+32767$。

定点数在使用时，所有原始数据事先都要按比例化成纯小数或整数，运算结果又要按比例转换成实际值。

定点表示法所能表示的数值范围非常有限，计算机在进行定点数运算时，容易产生结果超出表示范围而发生溢出错误。

2.浮点表示法

任何一个二进制数 N，都可写成 $2^e\times t$ 的形式，即 $N=2^e\times t$。例如，1010.11 可写成 $2^{100}\times 0.101011$，即 $1010.11=2^{100}\times 0.101011$。这里，e 称为 N 的阶码，是一个二进制整数；t 称为 N 的尾数，是一个二进制纯小数。

机器数用阶码和尾数两部分表示，称为浮点表示法。一般规定，阶码是定点整数，尾数是定点纯小数。它们可采用原码、补码或其他编码表示。

例如，一个数 X 用 8 位机器数浮点表示如下，其中前三位表示阶符和阶码值，后五位表示尾符和尾数值，它们都用原码表示：

1	10	0	1100
阶符	阶码	尾符	尾值

那么 $X=0.0011000B=2^{-2}\times 0.75=0.1875$。

浮点表示中，尾数的大小和正负决定了所表示的数的有效数字和正负，阶码的大小和正负决定了小数点的位置，因此机器数小数点的位置随阶码的变化而浮动。为了使运算中不丢失有效数字，提高运算精度，计算机中的浮点表示，通常采用改变阶码来达到规格化数的表示。这里，规格化数要求尾数值的最高位必须是 1。

如果在 16 位机器数浮点表示中，取前六位为阶码（其中第一位是阶符），后十位为尾数（其中第一位是尾符），它们都用原码表示，那么它能表示的最大数和最小数分别为：

0111110111111111　　即 $2^{31}(1-2^9)\approx 2.14329\times 10^9$

0111111111111111　　即 $-2^{31}(1-2^9)\approx -2.14329\times 10^9$

可见它表示数的范围约为 $-2.14329\times 10^9\sim 2.14329\times 10^9$，比相同位数机器数定点表示的范围大得多。

在数的浮点表示和运算中，也会产生溢出的问题。当一个数的阶码大于机器所能表示的最大阶码时，产生上溢出。这时，机器一般不再继续运算，而是转入出错处理。当一个数的阶码小于机器所能表示的最小阶码时，产生下溢出。这时，机器一般自动把该数作 0 处理。浮点表示法表示数的范围大，但浮点数的运算规则复杂，运算速度相对来说较慢。在有些计算机中，用协处理器来进行浮点数运算，提高了复杂的数值计算的处理速度。

1.2.3 ASCII 码和 BCD 码

1.ASCII 码

在人们通常接触和处理的信息中，相当一部分是用字符或字符的组合来表示的。例如，在文字信息处理中，一篇英文或中文的文章，就是由英文字母符号或汉字符号及其他一些字符组成的；在数据处理中，需要把数字信息输入计算机，这些数字信息就是由一些数字字符组成的。总之，我们这里的字符是指字母、数字以及其他一些可打印显示的符号。例如，26个英文字母的大写和小写：A，B，…，Z，a，b，…，z；10个数字：0，1，…，9；以及 +，-，*，/，<，>，…等等。

计算机和外部设备之间进行通讯联系时，还需要一些控制功能符号。例如，控制打印机的走纸符、换行符等，使铃响的响铃符等等。通常把这些控制功能符看作特殊的字符。

在计算机内部，上述字符必须用一种二进制代码来表示。目前，在国际上广泛采用的是美国标准信息交换代码(American Standard Codefor Information Interechange)，简称 ASCII 码。

ASCII 码是 7 位二进制编码，它可以表示 $2^7 = 128$ 个字符(见表 1 - 2)。

表 1 - 2 由 16 行 8 列组成，行用一个字节的低四位 b_4，b_3，b_2，b_1 来编码，列用一个字节的高三位 b_7，b_6，b_5 来编码，最高位不用。

例如字符“A”在表中的第 4 列第 1 行，所以它的二进制的 ASCII 编码是 1000001。字符“X”在表中的第 5 列第 8 行，所以它的二进制的 ASCII 编码是 1011000。

一个字符的 ASCII 码可用二进制表示，也可用八进制、十六进制或十进制等形式来书写表示。例如换行符“LF”的 ASCII 码可用二进制表示为 0001010B，花括号“{”的 ASCII 码可用十进制表示为 173，数字“0”～“9”的 ASCII 码用十六进制表示为 30H～39H，等等。

表 1 - 2　　ASCII 码表

十六进制(高位)					0	1	2	3	4	5	6	7
十六进制低位	二进制低四位				高三位							
	b4	b3	b2	b1	b7 b6 b5							
					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
0	0	0	0	0	^@	^P	sp	0	@	P	`	p
1	0	0	0	1	^A	^Q	!	1	A	Q	a	q
2	0	0	1	0	^B	^R	"	2	B	R	b	r
3	0	0	1	1	^C	^S	#	3	C	S	c	s
4	0	1	0	0	^D	^T	$	4	D	T	d	t
5	0	1	0	1	^E	^U	%	5	E	U	e	u
6	0	1	1	0	^F	^V	&	6	F	V	f	v
7	0	1	1	1	^G	^W	'	7	G	W	g	w
8	1	0	0	0	^H	^X	(	8	H	X	h	x
9	1	0	0	1	^I	^Y	)	9	I	Y	i	y
A	1	0	1	0	^J	^Z	*	:	J	Z	j	z
B	1	0	1	1	^K	^[	+	;	K	[	k	{
C	1	1	0	0	^L	^\	,	<	L	\	l	|
D	1	1	0	1	^M	^]	-	=	M	]	m	}
E	1	1	1	0	^N	^^	.	>	N	^	n	~
F	1	1	1	1	^O	^_	/	?	O	-	o	del

从表中不难发现，只要记住字母“A”、“a”和数字“0”的 ASCII 码，就容易推算出所有英文大、小写字母和数字的 ASCII 码。

为了检测电气干扰或设备故障引起的传送错误，计算机在进行字符处理和信息交换时，一般在 7 位 ASCII 码的左边再附加一个奇偶校验位，组成 8 位代码进行存储和传送。在偶校验系统中，用附加的偶校验位使所有字符的 8 位代码都有偶数个 1。若传送过程中，发现某个字符的 8 位代码中 1 的个数为奇数，即发生了传送错误，要求重新传送。相仿，在奇校验系统中，用附加的奇校验位使所有字符的 8 位代码都有奇数个 1，传送后进行奇数个 1 的特征检验。

例如，字母“R”和“S”的 ASCII 码为 1010010B 和 1010011B。采用偶校验，它们对应的 8 位代码分别是 11010010B 和 01010011B；采用奇校验，分别是 01010010B 和 11010011B。

由于标准的 7 位 ASCII 码所能表示的字符较少，不能满足有些信息处理的需要，于是在 ASCII 码的基础上又设计了一种扩充的 ASCII 码。扩充的 ASCII 码是一种 8 位二进制编码，它可以表示 256 个字符。

2. BCD 码

十进制数在键盘输入和打印、显示输出时，往往是将各个数字以 ASCII 码来表示的。但是它在计算机内运算时，是以二进制形式进行的。为了便于转换，人们设计了一些二进制编码来表示十进制数，称为二－十进制码，即 BCD 码(Binary Coded Decimal)。

BCD 码是用四位二进制代码来表示一位十进制数。从 16 个四位二进制代码 0000～1111 中，只须选择其中 10 个作为十进制数的 10 个数字的代码就可以了。这样就有多种 BCD 码：如 8421 码、2421 码、余 3 码和格雷码等(见表 1－3)。

表 1－3　常用 BCD 码

十进制数字	8421 码	2421 码	余 3 码	格雷码
0	0000	0000	0011	0000
1	0001	0001	0100	0001
2	0010	0010	0101	0011
3	0011	0011	0110	0010
4	0100	0100	0111	0110
5	0101	0101	1000	1110
6	0110	0110	1001	1010
7	0111	0111	1010	1000
8	1000	1110	1011	1100
9	1001	1111	1100	0100

由于 8421 码最常用，因此通常就把它称为 BCD 码。它是用四位二进制代码来表示一位十进制数字，其特点是二进制代码本身的值就是它所对应的十进制数字的值。这样，一个十进制数就可用其各位数字所对应的一组 BCD 码来表示。

例如，十进制数 315 可用 BCD 码表示为 001100010101。

要注意的是，用 BCD 码表示的数，形式上像二进制数，但不是真正的二进制数。例如，

315 转换成二进制数是 100111011B。

在计算机中，可以把用 BCD 码表示的十进制数转化成真正的二进制数后，再进行运算，但也可以直接对用 BCD 码表示的十进制数进行运算。这种运算是以二进制数进行的，但需要进行“十进制调整”，以符合“逢十进一”的十进制数的运算规则。

1.3 逻辑运算基础(*选学)

我们知道，由于二进制数所具有的特点，在数字式电子计算机中，所有信息都是以二进制形式表示的。它们通过计算机内具有各种逻辑功能的逻辑部件，进行存储、传送、运算等操作。这些逻辑部件实际上是由一些基本逻辑门电路组合而成的逻辑电路。借助于逻辑代数这种数学工具，能对逻辑电路进行有效的分析和综合；从而就使逻辑代数成为计算机硬件电路设计的有力工具。

同时，我们知道计算机有逻辑运算和逻辑判断的能力，即计算机能对逻辑数据进行逻辑操作。正因为这样，我们在程序设计中，就可以使用一些能进行逻辑运算和逻辑判断的语句，编制能自动实现不同解题要求的程序。这里就涉及到如何利用逻辑代数，使得由逻辑常量和逻辑变量等构成的逻辑表达式能正确的描述问题。

总之，为了对计算机的基本组成和工作过程能有一个比较具体而深入的了解，同时也为了今后进行计算机程序设计中正确合理地使用逻辑表达式，掌握逻辑代数的基本知识是很有必要的。

1.3.1 逻辑命题

所谓逻辑命题是指一个有具体意义，并能判定真或假的判断语句。例如“2 大于 1”，“15 是偶数”，“A 等于 7”等都是逻辑命题。逻辑命题所作的判断可以是正确的，也可以是错误的，即它的值可以取“真”，也可以取“假”。在上述逻辑命题中，“2 大于 1”的值为“真”，“15 是偶数”的值为“假”，“A 等于 7”的值取“真”或“假”，则视 A 的数值而确定。

我们把“真”和“假”这两个不变的逻辑值，称为逻辑常量；把只能取“真”和“假”这两个逻辑值的变量，称为逻辑变量。

我们知道，在计算机中，任何数据，包括逻辑类型的数据，都是用二进制形式来表示的。通常，我们用“1”表示“真”，用“0”表示“假”。

逻辑代数就是以代数形式研究逻辑变量及其之间关系的性质和应用的科学。逻辑代数最早由英国数学家布尔(G.Boole)提出，所以它也被称为布尔代数。同时，逻辑常量和逻辑变量也分别可称为布尔常量和布尔变量。

1.3.2 基本逻辑运算

像普通代数中有代数运算那样，逻辑代数中有逻辑运算，也称布尔运算。

1. 逻辑非

逻辑非(即逻辑否定)是一个一元运算。对逻辑变量 A 进行逻辑非运算，其结果是得到一个与 A 取值相反的逻辑值。通常用上划线“¯”(读作“非”)表示逻辑非运算。有时也用“-”或“NOT”来作为逻辑非的运算符，即把 A 写成 - A 或 NOT A。

2. 逻辑与

逻辑与(即逻辑乘)是一个二元运算。当且仅当两个逻辑变量 A 和 B 都取值为 1 时，A 和 B 逻辑与的运算结果为 1。通常用“∧”(读作“并且”或“与”)表示逻辑与运算。

当 A=0,B=0 时，A∧B=0；当 A=0,B=1 时，A∧B=0

当 A=1,B=0 时，A∧B=0；当 A=1,B=1 时，A∧B=1

有时也用“·”(甚至省略它)或“AND”来作为逻辑与的运算符，即把 A∧B 写成 A·B 或 AB 或 A AND B。

3. 逻辑或

逻辑或(即逻辑加)也是一个二元运算。当且仅当两个逻辑变量 A 和 B 都取值为 0 时，A 和 B 逻辑或的运算结果为 0。通常用“∨”(读作“或”)表示逻辑或运算。

当 A=0,B=0 时，A∨B=0；当 A=0,B=1 时，A∨B=1

当 A=1,B=0 时，A∨B=1；当 A=1,B=1 时，A∨B=1

有时也用“+”或“OR”来作为逻辑或的运算符，即把 A∨B 写成 A+B 或 A OR B。

从以上基本逻辑运算的定义，显然有：

$\bar{0}=1$　　$\bar{1}=0$

$0\wedge 0=0$　　$0\wedge 1=0$　　$1\wedge 0=0$　　$1\wedge 1=1$

$0\vee 0=0$　　$0\vee 1=1$　　$1\vee 0=1$　　$1\vee 1=1$

1.3.3 逻辑表达式

把逻辑常量、逻辑变量、逻辑运算符、括号等适当地连接起来，可组成逻辑表达式。例如，A+BC,(A+B)(A+C),A·B+A·B 等，都是逻辑表达式，这里的 A、B、C 都是逻辑变量。如同代数表达式中要规定代数运算的优先级别一样，在逻辑表达式中，规定逻辑运算的优先级别是按括号，逻辑非，逻辑与，逻辑或的次序递减。

我们把逻辑表达式 $A\bar{B}+\bar{A}B$ 表示的逻辑变量 A 和 B 之间所进行的复合逻辑运算，称为异或。通常用“⊕”表示异或运算。换句话说，逻辑变量 A 和 B 的异或 A⊕B，定义为 $A\bar{B}+\bar{A}B$，即 $A\oplus B=A\bar{B}+\bar{A}B$。

有时也用“EOR”(Exclusive OR)来作为异或的运算符，即把 A⊕B 写成 A EOR B。

1.3.4 逻辑运算中的基本性质

在逻辑代数中，逻辑运算具有下列一些基本性质。这些性质都能用真值表来加以证明。

① 交换律：$A+B=B+A$　　$A\cdot B=B\cdot A$

② 结合律：$A+(B+C)=(A+B)+C$　　$A(BC)=(AB)C$

③ 分配律：$A(B+C)=AB+AC$　　$A+BC=(A+B)(A+C)$

④ 反演律：$\overline{A+B}=\bar{A}\cdot\bar{B}$　　$\overline{A\cdot B}=\bar{B}+\bar{A}$

⑤ 重叠律：$A+A=A$　　$A\cdot A=A$

⑥ 对合律：$\bar{\bar{A}}=A$

⑦ 吸收律：$A+AB=A$　　$A\cdot(A+B)=A$

$A+\bar{A}B=A+B$　　$A\cdot(\bar{A}+B)=A\cdot B$

⑧ 互补律：$A+\bar{A}=1$　　$A\cdot\bar{A}=0$

⑨ 0—1 律：$0+A=A$　　$0\cdot A=0$

$1+A=1$　　$1\cdot A=A$

利用上述性质，可把比较复杂的逻辑表达式，等价地转化成比较简单的形式。

例如：$AB + \overline{A}C + BC$

$= AB + \overline{A}C + 1 \cdot BC$ （0—1 律）

$= AB + \overline{A}C + (A + \overline{A})BC$ （互补律）

$= AB + \overline{A}C + ABC + \overline{A}BC$ （分配律）

$= AB + ABC + \overline{A}C + ACB$ （交换律）

$= (AB + ABC) + (\overline{A}C + \overline{A}CB)$ （结合律）

$= AB + \overline{A}C$ （吸收律）

这种化简方法需要一定技巧，只有通过大量实践才能熟练掌握。

习　题

1.把下列十进制数转化成二进制数（精确到小数点后四位）：

（1）215　（2）235　（3）75.52　（4）101.01

2.把下列十进制数转化成八进制数和十六进制数：

（1）149　（2）327　（3）3175　（4）1112

3.把下列二进制数转化成十进制数、八进制数和十六进制数：

（1）101101.101　（2）110100　（3）11010111

4.把下列十六进制数转化成二进制数、八进制数和十进制数：

（1）ABCD　（2）E1F　（3）7CD2　（4）101

5.分别用反码和补码表示下列各数：

（1）12　（2）－33　（3）110　（4）－101

6.将下列字符用 ASCII 码表示：

（1）127　（2）1＋1＝2　（3）OK　（4）China

7.把下列十进制数转化成 8421BCD 码：

（1）127　（2）33　（3）1100　（4）101

8.用逻辑运算计算下列各式：

（1）$10101101 \vee 10001111$　（2）$10011111 \wedge 01101111$

（3）$101101 \oplus 101011$　（4）$101101 \oplus 101101$

9.化简下列逻辑表达式：

（1）$AB\overline{C} + ABC + \overline{A}B + \overline{B}$　（2）$ABC + AB\overline{C} + A\overline{B}C + A\overline{BC}$