程序设计基础

国营二六二厂情报室

西安冶院计算机教研室

一九七八年十一月

目　录

第一章　基础知识

第一节　数的表示和数制转换的方法

为了学习计算机程序编制的需要，我们对电子计算机中的数的表示及其转换应该有一个粗略的了解。

1.1　十进制、二进制、八进制

一个数的表示形式虽然可有多种，但都是代表同一个数，所以其实质是一样的。正如恩格斯所说："一切数的定律都取决于所采用的记数法，而且被这个记数法所决定。"

在日常生活中，人们多半是以十进制（"逢十进一"）来记数的，所以对于十进制数是非常熟悉的。例如：

$$256_{(10)} = 2\times10^2 + 5\times10^1 + 6\times10^0$$

$$2.375_{(10)} = 2\times10^0 + 3\times10^{-1} + 7\times10^{-2} + 5\times10^{-3}$$

因此对于十进制系统中的任何一个数 a 都可以写成以基数"10"为基底的幂运算形式，即

$$a = \sum_{i=-n}^{m} a_i \times 10^i \quad \text{式中 } a_i \in [0, 1, 2, 3, \cdots\cdots 9]$$

一、二进制

二进制是最简单的进位制，每一位非"0"则"1"。它具有最简单的运算法则，可使机器设备节省、结构简单、易于实现，因而绝大多数的数字计算机采用二进制。

在二进制数的系统中，任何一个数 N 都可以写成以基数"2"为基底的幂运算形式：

$$N=\sum_{i=-n}^{m} a_i \times 2^i \quad \text{式中 } a_i \in [0,1]$$

例如：$(111)_2 = 1\times 2^2 + 1\times 2^1 + 1\times 2^0 = (7)_{10}$

$$(0.1011)_2 = 1\times 2^{-1} + 0\times 2^{-2} + 1\times 2^{-3} + 1\times 2^{-4}$$

$$= \left(\frac{11}{16}\right)_{10}$$

$$(11.01)_2 = 1\times 2^1 + 1\times 2^0 + 0\times 2^{-1} + 1\times 2^{-2}$$

$$= \left(\frac{13}{4}\right)_{10}$$

二、八进制

“事物都是一分为二的。”二进制有很多突出的优点，所以在计算机中采用它。但是二进制有一个很大的缺点，就是它表示的数太长，读写和记忆都不方便，尤其当数的位数较长时此缺点要突出地显露出来。

为便于人们读写和记忆，从 $2^3 = 8$ 的简单关系出发，将二进制信息每相邻三位分成一组，组合为八进制的形式。具体对应关系是：

八进制：	0	1	2	3	4	5	6	7
	↕	↕	↕	↕	↕	↕	↕	↕
二进制：	000	001	010	011	100	101	110	111

八进制数是一种“逢八进一”的计算法，八进制数的每一个数字由 0、1、2、3、4、5、6、7 中的一个构成，一个数中不同位置上的数字具有不同的含义。例如：

$(37)_8 = 3\times 8^1 + 7\times 8^0 = (31)_{10}$

$(100)_8 = 1\times 8^2 + 0\times 8^1 + 0\times 8^0 = (64)_{10}$

$$(0.54)_8 = 5\times 8^{-1} + 4\times 8^{-2} = \frac{5}{8} + \frac{4}{64} = \frac{44}{64} = \left(\frac{11}{16}\right)_{10}$$

因此，在八进制系统中，每一个八进制数A都可以写成以基数"8"为基底的幂的运算形式：

$$A = \sum_{i=-n}^{m} Q_i \times 8^i \quad \text{式中} Q_i \in [0, 1, 2 \cdots\cdots 7]$$

1.2 数制转换方法

以上右我们已经知边，对于同一个数可以用不同的记数法来表示，下右我们将进一步讨论它们之间的转换关系：

一、二进制与八进制的转换关系

由上右的介绍；我们可以看到八进制的一位数字恰与二进制的三位数目相对应，从而八进制化成二进制可以采用"一位拉三位"的方法。例如：

$$113.54(8) = 001001011.110100(2)$$

相反，二进制数化成八进制数时，应以小数点为基准，往左往右都是"三位并一位"，例如：

$$1110.01111(2) = 16.36(8)$$

二、八进制与十进制的转换关系

此转换关系需分整数、小数进行讨论

(1) 整数的互化

设N是整数，它在十进制中的表示是：

$$a_1 a_2 \cdots\cdots a_n (10)$$

即 $N = a_1\times 10^{n-1} + a_2\times 10^{n-2} + \cdots\cdots + a_{n-1}\times 10^1 + a_n\times 10^0$

它在八进制中的表示是：

$b_1 b_2 \cdots\cdots b_m\ (8)$

即 $x = b_1 \times 8^{m-1} + b_2 \times 8^{m-2} + \cdots\cdots + b_{m-1} \times 8^1 + b_m \times 8^0$

应当有：

$$a_1 \times 10^{n-1} + a_2 \times 10^{n-2} + \cdots\cdots + a_{n-1} \times 10^1 + a_n$$

$$= b_1 \times 8^{m-1} + b_2 \times 8^{m-2} + \cdots\cdots + b_{m-1} \times 8^1 + b_m$$

如果已知 $a_1 a_2 \cdots\cdots a_n$ 求 $b_1 b_2 \cdots\cdots b_m$，就是把十进制数转换成八进制数。反过来，如果已知 $b_1 b_2 \cdots\cdots b_m$ 求 $a_1 a_2 \cdots\cdots a_n$，就是将八进制数转换成十进制数。

为了把整数 x 由十进制化成八进制，只要用 8 除以整数，得到的余数是八进制数的个位，所得的商再用 8 除，得到的余数是八进制数的第二位，所得商再用 8 除……，直到所得的商比 8 小时为止，这个商数就是八进制的最高位。这个法则叫“除 8 取余法”。

例 1.1，把十进制数 3453 化成八进制数

$$\begin{array}{r|l} 8 & 3453 \\ 8 & 431 \quad \cdots\cdots 5 \\ 8 & 53 \quad \cdots\cdots 7 \\ & 6 \quad \cdots\cdots 5 \end{array}$$

$3453_{(10)} = 6575_{(8)}$

由八进制化为十进制，只需把上面的过程反过来就可以了，即将八进数的最高位乘以 8，加上第二位，再乘以 8……，直到加上最后一位。

例 1.2，把八进制 6575 化成十进制数

$6575_{(8)} = [(6 \times 8 + 5) \times 8 + 7] \times 8 + 5 = 3453_{(10)}$

也可用另一方法来化：

$$6575(8) = 6\times 8^3 + 5\times 8^2 + 7\times 8^1 + 5\times 8^0 = 3453(10)$$

(2) 小数的互化

例1.3，将十进制小数0.825化成八进制小数

假设转换结果是 $0.b_1 b_2 \cdots\cdots b_n \cdots\cdots$，则

$$0.825(10) = b_1\times 8^{-1} + 8^{-2} + \cdots\cdots + b_n \times 8^{-n} \cdots\cdots$$

等式两端乘8得

$$6 + 0.6(10) = b_1 + b_2 \times 8^{-1} + \cdots\cdots + b_n \times 8^{-n+1} + \cdots\cdots$$

两端的整数下分应相等，所以 $b_1 = 6$。同时有

$$0.6(10) = b_2 \times 8^{-1} + \cdots\cdots + b_n \times 8^{-n+1} + \cdots\cdots$$

等式两边再乘以8得

$$4 + 0.8(10) = b_2 + b_3 \times 8^{-1} + \cdots\cdots b_n \times 8^{-n+2} + \cdots\cdots$$

得到 $b_2 = 4$，并且有

$$0.8(10) = b_3 \times 8^{-1} + \cdots\cdots + b_n \times 8^{-n+2} + \cdots\cdots$$

如此一直继续下去，从而得到

$$0.825(10) = 0.64631463 1\cdots\cdots(8)$$

上述过程称为“乘8取整法”。

要指出的是，十进制的有限小数在八进中可能表示为无限循环小数，这时常取其有限位作为近似值。

三 十进制数转换成二进制数

将十进制数化为二进制数，有两种方法。一种是直接把十进制数化为二进制数，这方法同上面讲过的十进制数转换为八进

制数的化法相类似。即用“除二取余法”把十进制整数化为二进制整数，用“乘二取整法”把十进制小数化为二进制小数。另外一种化法是，先把十进制数化为八进制数，再由八进制“拆开”成二进制数，这种方法比较简便。下面分别用例子说明这两种方法。

例1.4，　把十进制数137化成二进制数

用直接方法：

2	137	余数
2	68	……1
2	34	……0
2	17	……0
2	8	……1
2	4	……0
2	2	……0
2	1	……0
	0	……1

于是得到：

$$137(_{10}) = 10001001(_2)$$

用十进制→八进制→二进制方法：

$$137(_{10}) = 211(_8) = 10001001(_2)$$

例1.5，把十进制数0.625化为二进制数

用直接方法：

$$\begin{array}{r} 0.625 \\ \times\quad 2 \\ \hline 1.250 \quad\cdots\cdots 1 \\ \times\quad 2 \\ \hline 0.500 \quad\cdots\cdots 0 \\ \times\quad 2 \\ \hline 1.000 \quad\cdots\cdots 1 \end{array}$$

于是得到：

$$0.625(_{10}) = 0.101(_{2})$$

用十进制→八进制→二进方法

$$0.625(_{10}) = 0.5(_{8}) = 0.101(_{2})$$

例1.6，把十进制数13.24化为二进制数

用十进制→八进制→二进制方法：

$$13.24(_{10}) = 15.1727(_{8}) = 1101.001111010111(_{2})$$

习 题 一

习题1.1 将下列二进制数化成八进制数：

1、 0101100010011100
2、 0000100111111100
3、 1111101100000000
4、 1010101010010000
5、 0001100010100101

习题1.2 将下列八进制数化成二进制数：

1、 063077
2、 177633
3、 143000
4、 060213
5、 006300
6、 051113

7、 000663

习题1.3 将下列十进制数翻译成八进制数：

1、 $\sqrt{2}=1.4142$　　2、 $\pi=3.14159$

3、 $e=2.71828$　　4、 $\lg 2=0.30102$

习题1.4 将下列八进制数翻译成十进制数：

1、 12　　2、 144　　3、 1750　　4、 23420

第二节 逻辑代数基础

逻辑代数是把统一物从具体事物矛盾着的二个方面抽象出来赋予变量的记号，（例如A，B，C，……）并称之为逻辑变量（或简称变量），且用数学的方法来研究统一物矛盾的转化及其矛盾运动的规律。在逻辑代数中，把统一物从具体的事物中抽象出来之后，将其矛盾着的两个状态分别用二个最简单的数字0与1来标记，也就是说逻辑变量的变化范围仅仅是0与1二个数。

2.1 逻辑代数的基本运算

逻辑代数的基本运算是分析逻辑线路和数字电子计算机解题所不可缺少的工具。

一、逻辑非

逻辑非是一种否定法则，否定在这里是指统一物由一个状态向着它的对立面转化的法则。这个法则写成形式记号就是 $\bar{A}$，读成非A。例如我们用A表示代表电灯状态的逻辑变量，则当A代表电灯亮时，$\bar{A}$就代表电灯不亮，其中$\bar{A}$就表示电灯由亮到不亮的转化。反之，当A表示电灯不亮时，$\bar{A}$就代表电灯亮。按否定

法则的转化原理，把$\bar{A}$看作一个新的变量再对$\bar{A}$施行否定法则就行：

$$\overline{(\bar{A})} = \bar{\bar{A}} = A$$

也就是说记号$\bar{\bar{A}}$标志着向着$\bar{A}$的对立面A的转化法则。

如果电灯的二个状态用数0和1分别表示，则代表电灯状态的逻辑变量A可以取值0和1。

在逻辑代数中，常数仅仅只有0与1两个，施行逻辑非运算就有：

$$\bar{0} = 1 \qquad \bar{1} = 0$$

二、逻辑加与逻辑乘

当若干个条件同时作用于一个统一物时，促使该统一物矛盾转化的方法，有二类：一类是在若干个条件中只要有一个条件成立时（或者这个，或者那个都可以），统一物的矛盾就转化了（其中有一个以上的条件成立或者全下成立也是一样）。另一类是在若干个条件中，必须使其中所有的条件都同时成立，统一物的矛盾才能转化。例如种子发芽的过程是统一物矛盾转化的实例，我们知道，欲使种子发芽必须同时具备一定的空气、一定的温度和一定的湿度这三个基本条件，缺少一个种子就不能发芽。

上述二类法则，在理论上前者就称为“或”法则，后者就称为“与”法则。

(1) “或”法则

“或”法则类似于算术中的加法，故“或”法则称之谓逻辑加，逻辑加的形式记号用$\vee$表示（读成或）。

设有三个逻辑变量A、B、C，其中A、B是自变量，C是因变量，且三个变量之间有如下关系：

自变量		因变量
A	B	C
0	0	0
1	0	1
0	1	1
1	1	1

我们可以用形式记号描述成：

$$C = A \vee B$$

读成C等于A或B。

逻辑加 $C = A \vee B$ 的分析表达式可记成：

$$C = \begin{cases} 0 & \text{当A与B同时等于0时} \\ 1 & \text{当A与B中只要有一个等于1时} \end{cases}$$

逻辑加运算对于常数0与1，则只有如下的特性：

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

对于具有二个以上自变量的逻辑加运算也完全类似。

(2) "与"法则

"与"法则类似于算术中的乘法，故"与"法则又称为逻辑乘，并用运算记号∧表示（读成与）。

当变量A、B、C有如下关系时：

自变量		因变量
A	B	C
0	0	0
1	0	0
0	1	0
1	1	1

我们说C等于A和B的逻辑乘，且记成

$$C = A \wedge B$$

读成C等于A与B。

逻辑乘 $C = A \wedge B$ 的分析表达式可记成：

$$C = \begin{cases} 0 & \text{当自变量（A与B）中只要有一个是0} \\ 1 & \text{当自变量（A与B）都等于1} \end{cases}$$

对于二个以上自变量的逻辑乘运算与二个变量的逻辑乘运算完全类似。

逻辑乘运算对于常数0与1，则只有如下特性：

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

三、逻辑异

逻辑代数在实际使用过程中，经常会遇到关于它们的组合公式的出现，尤其是逻辑公式：$(A \wedge \bar{B}) \vee (\bar{A} \wedge B)$ 更为有用，习惯上称它为逻辑异，逻辑异运算既有算术加法的功能又有"或"法则的功能故用形式记号$\forall$来表示，并且两个逻辑变量A与B的

逻辑异运算可写成：

$$A \veebar B = (\bar{A} \wedge B) \vee (\bar{A} \wedge B)$$

如果

$$C = A \veebar B$$

则关于 C 的分析表达式便是：

$$C = \begin{cases} 0 & \text{当变量 A 与 B 的值相同时} \\ 1 & \text{当变量 A 与 B 的值相异时} \end{cases}$$

逻辑公式：$C = A \veebar B$ 的函数列表表示法是：

自变量		因变量
A	B	C
0	0	0
1	0	1
0	1	1
1	1	0

例 1.7 设 A、B 是两个逻辑变量，试证明：

一 $A \vee B = A + \bar{A} \wedge B$（式中“+”是算术加）

证：因为 A 是逻辑变量，所以 A 有二种状态。

当 $A = 0$ 时，$A \vee B = B$，$\bar{A} \wedge B = 1 \wedge B = B$

得 $A + \bar{A} \wedge B = 0 + B = B = A \vee B$

当 $A = 1$ 时，$A \vee B = 1$，而 $\bar{A} \wedge B = 0 \wedge B = 0$

得 $A + \bar{A} \wedge B = 1 + 0 = 1 = A \vee B$.

因此，不论 A 是什么状态，都成立

$$A \vee B = A + \bar{A} \wedge B$$

证毕