附录 A　二进制和十六进制

读者很久以前就学过了基本的算术知识，如果没有这些知识后果将难以想象。看到“145”后，你不假思索就知道这是一百四十五。

通常以十进制格式来考虑数字，然而还可以使用其他格式来计数。使用计算机时，最常见的两种计数方式是二进制和十六进制。要理解二进制和十六进制，需要重新审视 145，将其视为一个数的代码，而不是数。

先从较小的开始：考虑数字 3 和“3”之间的关系，数字符号“3”是写在纸上的一个符号，而数字 3 是一个概念。数字符号用来表示数字。

符号三、3、Ⅲ、III、***都可用于表示数字 3 这个概念，由此可以清楚地看出符号和数字概念之间的区别。

在十进制中，使用 10 个符号（0、1、2、3、4、5、6、7、8、9）来表示所有的数。数字 10 如何表示呢？

可以设想使用字母 A 或 IIIIIIIIII 来表示 10。罗马人使用 X 来表示 10。在我们使用的阿拉伯计数方法中，结合使用位置和数字符来表示值。第一位（最右边的那位）为个位，第二位为十位。因此数字十五表示为 15，即 1 个 10 和 5 个 1。

归纳起来，可以得到如下规则：

（1）十进制使用 10 个数字符号：0～9。

（2）各位表示 10 的幂（1、10、100）的倍数。

（3）最大的两位数为 99。推而广之，最大的 n 位数为 10^n-1。因此，最大的三位数为 10^3-1（999）。

A.1　其他进制

人们使用十进制不是偶然的，这是因为人类有十根手指。然而，也可以使用其他进制。根据十进制中的规则，可以这样描述八进制：

（1）八进制使用 8 个符号：数字 0～7。

（2）八进制各位是 8 的幂：1、8、64 等。

（3）最大的 n 位八进制数为 8^n-1。

为区分用不同进制表示的数，可将进制作为下标放在数的后面。十进制数十五可写作 15_{10}。

因此，要用八进制表示 15_{10}，可写作 17_8。注意它也可以读作“十五”因为那仍然是它所表示的数。

为什么是 17 呢？1 表示 1 个 8，7 表示 7 个 1，1 个 8 加 7 个 1 等于 15。请看下面 15 个星号：

**********　*****

自然的做法是将它们分为两组，一组 10 个，另一组 5 个。这可用十进制表示为 15（1 个 10 和 5 个 1）。也可以这样将星号分为两组：

********　*******

即每组分别包含 8 个和 7 个。这可以用八进制表示为 17_8，即是 1 个 8 和 7 个 1。

A.2　不同进制之间的转换

十五这个数可以用十进制表示为 15，用九进制表示为 16_9，用八进制表示为 17_8，用七进制表示为 21_7。为什么是 21_7 呢？因为七进制不使用数字符号 8，为表示十五，需要用 2 个 7 和 1 个 1。

如何归纳上述过程呢？要将十进制数转换为七进制，需考虑每位表示的值：在七进制中，它们是 1、7、49、343 等。为什么是这些值呢？因为它们分别是 7^0、7^1、7^2、7^3 等。

任何数的 0 次幂（如 7^0）都是 1，1 次幂（如 7^1）为这个数本身，2 次幂是该数乘以本身（$7^2=7\times7=49$），3 次幂是该数乘以本身再乘以本身（$7^3=7\times7\times7=343$）。

可以创建一个这样的表：

位	4	3	2	1
幂	7^3	7^2	7^1	7^0
值	343	49	7	1

第一行为位编号，第二行为 7 的幂，第三行为第二行 7 的幂的十进制值。

将十进制数转换为七进制的步骤如下：检查十进制数，决定需要使用多少位才能表示它。例如，如果十进制数为 200，则只需使用三位，因此第 4 位（343）为 0，不必考虑。

为确定有多少个 49，用 49 去除 200。结果为 4，因此第 3 位为 4。余数为 4，它小于 7，因此第 2 位为 0。4 由 4 个 1 组成，因此第 1 位为 4。转换结果为 404_7。

位	4	3	2	1
幂	7^3	7^2	7^1	7^0
值	343	49	7	1
200 的七进制表示	0	4	0	4
十进制值	0	$4\times49=196$	0	$4\times1=4$

在这个例子中，第 3 位的 4 表示十进制值 196，第 1 位的 4 表示十进制值 4。$196+4=200$，因此 $404_7=200_{10}$。

再来看一个例子，将十进制数 968 转换为六进制。在六进制中，各位表示的值如下：

位	5	4	3	2	1
幂	6^4	6^3	6^2	6^1	6^0
值	1296	216	36	6	1

为将十进制数 968 转换为六进制，从第 5 位开始。968 大于 1296 吗？否，因此第 5 位为 0。968 除以 216 的商为 4，余数为 104，因此第 4 位为 4。104 除以 36 的商为 2，余数为 32，因此第 3 位为 2。32 除以 6 的商为 5，余数为 2。因此结果为 4252_6。

位	5	4	3	2	1
幂	6^4	6^3	6^2	6^1	6^0
值	1296	216	36	6	1
968 的六进制表示	0	4	2	5	2
十进制值	0	$4\times216=864$	$2\times36=72$	$5\times6=30$	$2\times1=2$

A.2.1　二进制

二进制是这种概念终极扩展，它只使用两个符号：0 和 1。各位表示的值如下：

位	8	7	6	5	4	3	2	1
幂	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
值	128	64	32	16	8	4	2	1

要将十进制数 88 转换为二进制，可采取同样的步骤：88 小于 128，因此第 8 位为 0。

88 除以 64 的商为 1，余数为 24，因此第 7 位为 1。24 小于 32，因此第 6 位为 0。

24 除以 16 的商为 1，余数为 8，因此第 5 位为 1。8 除以 8 的商为 1，余数为 0，因此第 4 位为 1，其他位全为 0。

位	8	7	6	5	4	3	2	1
幂	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
值	128	64	32	16	8	4	2	1
88 的二进制表示	0	1	0	1	1	0	0	0
十进制值	0	64	0	16	8	0	0	0

为检验结果是否正确，对其做如下计算：

```
1 × 64 =  64
0 × 32 =  0
1 × 16 =  16
1 × 8  =  8
0 × 4  =  0
0 × 2  =  0
0 × 1  =  0
          88
```

A.2.2 为什么使用二进制

二进制与计算机要表示的东西相对应，因此二进制在编程中非常重要。计算机实际上根本不知道什么字母、数字符号、指令或程序。计算机的内核是电路，在某个给定的结点要么电压很高，要么很低。

为使逻辑清晰，工程师不用相对量表示电压，而是用两个量来表示：电压足够高和电压不够；他们不说“够”和“不够”，而是将其简化为“是”和“否”。“是”和“否”（或“真”和“假”）可以用 1 和 0 表示。按照约定，1 表示“是”（“真”），但这只是一种约定，也可以约定 1 表示“否”（“假”）。

有了这种认识上的飞跃后，二进制的威力就非常明显了：使用 1 和 0，可以表示每一段电路的真实状态（通电或断电）。计算机只知道“是”和“否”，“是”用 1 表示，“否”用 0 表示。

A.2.3 位、字节和半字节

决定用 1 和 0 表示真和假后，二进制位就变得非常重要。由于早期的计算机每次发送 8 位，因此很自然地用 8 位数字来编写代码，8 位叫一个字节。

注意：4 位被称为半字节。

8 个二进制位可以表示 256 个不同的值。为什么呢？如果 8 位都为 1，则值为 255（128 + 64 + 32 + 16 + 8 + 4 + 1）；如果 8 位都为 0，则值为 0。0～255 有 256 种可能的状态。

A.2.4 什么是 KB

2^{10}（1 024）近似等于 10^3（1 000）。这种巧合有助于记忆，因此计算机科学家基于千的前缀为 kilo，将 2^{10} 个字节称为 1KB。

同样，由于 1 024*1 024（1 048 576）接近与 100 万，因此被称为 1MB；1 024MB 被称为 1GB（10 亿字节）。最后，1 024GB 被称为 1TB（万亿字节）。

A.2.5 二进制数

计算机用 1 和 0 对其要执行的任何操作进行编码。机器指令被编码成一系列的 1 和 0，由电路进行解释。任意一系列 1 和 0 都可以由计算机科学家翻译为数字，但认为这些数字有内在含义是错误的。

例如，Intel 8086 芯片组将位模式 1001 0101 解释为一条指令。你当然可以将其解释为十进制数 149，但是这个数本身是没有意义的。

有时，这些数为指令，有时为值，有时为编码。一种重要的标准编码集为 ASCII。在 ASCII 编码集中，每个字母和标点符号都用一个 7 位的二进制数表示。例如，小写字母 a 用 0110 0001 表示。这不是一个数，虽然可以将其解释为十进制数字 97（64 + 32 + 1）。因此，人们说字母 a 用 ASCII 码 97 表示；实际情况是，

97 的二进制表示 0110 0001 是字母 a 的编码，而十进制值 97 适合人类使用。

A.3　十六进制

由于二进制数难以阅读，于是人们想出了一种更为简单的表示方式。将二进制转换为十进制涉及大量的计算，但从二进制转换为十六进制非常简单，因为有一种非常好的快捷方法。

为理解这一点，必须先理解十六进制。十六进制使用十六个符号：0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F。最后 6 个是任选的，选择 A~F 是因为它们易于在键盘上表示。在十六进制中，各位表示的值如下：

位	4	3	2	1
幂	16^3	16^2	16^1	16^0
值	4096	256	16	1

要将十六进制数转换为十进制，可以使用乘法。十六进制数 F8C 对应的十进制数为：

```
F × 256 = 15 × 256 = 3840
8 × 16 =      128
C × 1 = 12 × 1 =   12
3980
```

要将十六进制数 FC 转换为二进制，最好先将其转换为十进制，然后再转换为二进制

```
F × 16 = 15 × 16 = 240
C × 1 = 12 × 1 =   12
252
```

将十进制数 252 转换为二进制需要使用如下表格：

位	9	8	7	6	5	4	3	2	1
幂	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
值	256	128	64	32	16	8	4	2	1

252 小于 256。因此第 9 位为 0；252 除以 128 的商为 1，余数为 124，因此第 8 位为 1；124 除以 64 的商为 1，余数为 60，因此第 7 位为 1；60 除以 32 的商为 1，余数为 28，因此第 6 位为 1；28 除以 16 的商为 1，余数为 12，因此第 5 位为 1；12 除以 8 的商为 1，余数为 4，因此第 4 位为 1；4 除以 4 的商为 1，余数为 0，因此第 3 位为 1，其他位全为 0。因此转换得到的二进制数为 1111 1100。

如果将该二进制数分成二组，每组 4 位，便可以将其奇妙地转换为十六进制。

右边一组为 1100。对应的十进制数为 12，十六进制数为 C。

左边一组为 1111，对应的十进制数为 15，十六进制数为 F。

因此 1111 1100 对应的十六进制数为 FC，这种总是管用。可以将任何长度的二进制数划分成每 4 位一组，分别将每组转换为十六进制，再将这些十六进制数组合在一起，就得到了十六进制结果。下面是一个大得多的二进制数：

```
1011 0001 1101 0111
```

为验证上述快捷转换方法，首先将其转换为十进制。

在二进制中，每位表示的值依次翻倍。第 1 位为 1，第 2 位为 2，然后是 4、8、16，依此类推。

将上述二进制数转换为十进制的计算过程如下：

```
1×1       1
1×2       2
1×4       4
0×8       0
1×16      16
0×32      0
1×64      64
1×128     128
```

```
1×256       256
0×512       0
0×1024      0
0×2048      0
1×4096      4096
1×8192      8192
0×16384     0
1×32768     32768
十进制值    45527
```

要将这个十进制数转换为十六进制，需要使用如下表格：

位	5	4	3	2	1
幂	16^4	16^3	16^2	16^1	16^0
十进制值	65536	4096	256	16	1

45527 小于 65536，因此可以从第 4 位开始。45527 除以 4096 的商为 11，余数为 471；471 除以 256 的商为 1，余数为 215；215 除以 16 的商为 13，余数为 7。因此对应的十六进制数为 B1D7。

检验如下：

```
B (11) × 4096 =      45056
1 × 256 =              256
D (13) × 16 =          208
7 × 1 =                  7
总计                 45527
```

快捷方法是，将原来的二进制数 1011000111010111 分为 4 位一组：1011 0001 1101 0111。然后计算每组对应的十六进制数：

```
1011 =
1 × 1 =         1
1 × 2 =         2
0 × 4 =         0
1 × 8 =         8
总计            11
十六进制位      B

0001 =
1 × 1 =         1
0 × 2 =         0
0 × 4 =         0
0 × 8 =         0
总计            1
十六进制位      1

1101 =
1 × 1 =         1
0 × 2 =         0
1 × 4 =         4
1 × 8 =         8
总计            13
十六进制位      D

0111 =
1 × 1 =         1
1 × 2 =         2
1 × 4 =         4
```

```
0 × 8 =          C
总计             7
十六进制位       7

十六进制数    B1D7
```

就像变魔术一样，从二进制转换为十六进制的简捷方法得到的结果与更复杂的方法相同。

在高级编程中，程序员非常频繁地使用十六进制，但是在很长的时间内，读者不使用十六进制也能高效地进行编程。

注意：一种使用十六进制的常见情形是处理颜色值。在 C++程序和其他领域（如 HTML）都如此。

附录 B　C++关键字

关键字是语言保留给编译器使用的。不能将关键字用作类、变量或函数的名称。

asm	false	sizeof
auto	float	static
bool	for	static_cast
break	friend	struct
case	goto	switch
catch	if	template
char	inline	this
class	int	throw
const	long	true
const_cast	mutable	try
continue	namespace	typedef
default	new	typeid
delete	operator	typename
do	private	union
double	protected	unsigned
dynamic_cast	public	using
else	register	virtual
enum	reinterpret_cast	void
explicit	return	volatile
export	short	wchar_t
extern	signed	while

除上述关键字外，下述单词也被保留：

and	compl	or_eq
and_eq	not	xor
bitand	not_eq	xor_eq
bitor	or	

附录 C　运算符优先级

了解运算符有优先级很重要，但并不一定要记住运算符的优先级。

优先级是程序执行公式中运算的顺序。如果一个运算符的优先级高于另一个，则先执行它指定的计算。

优先级高的运算符比优先级低的运算符结合更紧密，因此，先执行优先级高的运算符指定的运算。在表 C.1 中，等级越低优先级越高。

表 C.1　　运算符优先级

等　级	名　称	运　算　符
1	作用于解析运算符	::
2	成员选择、下标、函数调用、后缀递增和后缀递减	., ->, (), [], ++, --
3	sizeof、前缀递增和递减、求补、逻辑非、单目加和减、取址和解除引用、new、new[]、delete、delete[]、强制类型转换、sizeof( )	++, --, ~, !, +, -, &, ( )
4	用于指针的成员选择	.*, ->*
5	乘、除、求模	*, /, %
6	加、减	+, -
7	移位（左移和右移）	<<, >>
8	不等关系	<, <=, >, >=
9	相等关系	==, !=
10	按位“与”	&
11	按位“异或”	^
12	按位“或”	|
13	逻辑“与”	&&
14	逻辑“或”	||
15	条件运算符	?:
16	赋值运算符	=, *=, /=, %=, +=, -=, <<=, >>=, &=, |=, ^=
17	逗号运算符	,