第3章 计算机中信息的表示方法

CHAPTER 3

计算机是一个可编程的数据处理机器，本章讨论不同的数据类型以及它们在计算机中是如何表示和存储的。

本章学习目标如下。

- 列出计算机中使用的5种不同的数据类型。
- 描述整数如何以无符号格式表示。
- 描述整数如何以二进制补码格式表示。
- 描述实数如何以浮点格式表示。
- 描述字符如何通过各种不同的编码标准表示。
- 描述音频如何表示。
- 描述图像和图形如何表示。
- 描述视频如何表示。

3.1 数据类型

没有数据，计算机就毫无用处，计算机执行的每个任务都是在以某种方式管理数据，因此，用适当的方式表示和组织数据是非常重要的。

首先，需要辨别“数据”和“信息”这两个术语。虽然它们通常可以互换使用，但进行区分有时还是有必要的，尤其对理解本章内容更是如此。数据是基本数值或事实，而信息是以有助于解决某种问题的方式组织和(或)处理的数据。

例如，计算机中存储的二进制数字01000001，它就是“数据”，但这个数据表示什么意思呢？即它包含的“信息”是什么呢？这就需要根据该数据具体的使用场合来判别。它可能表示的是一个数字，大小为十进制数65；也可能是计算机指令的一部分，表示该指令是一条加法指令；也可能表示的是A这个字符；也可能表示的是图像的一个像素点的灰度，等等。

3.1.1 数据类型简介

计算机发展前期，计算机处理的几乎都是数值和字符数据，但现在它已经成为真正的多媒体设备，可以存储、表示以下各种类型的数据。

1. 数值

使用计算机(尤其是早期)的主要目的是对数值进行计算，如进行算术运算、求解代数或

三角方程、找出微分方程的根等。

2. 字符

计算机利用文字处理程序处理字符，包括字符的存储、对齐、移动、删除等。

3. 音频（声音）

计算机能处理音频数据，我们可以使用计算机播放音乐，并且把声音作为数据输入到计算机中。

4. 图像和图形

计算机可以使用图像处理程序对图像进行创建、收缩、放大、旋转等。

5. 视频

计算机不但能用来播放视频，还能创建在视频中所看到的特技效果。

上述数据最终都被存储为二进制数据，被表示为由 0 和 1 组成的一串数字。本章将依次探讨每种数据类型，介绍它们在计算机中的表示方式。

3.1.2 计算机内部的数据

在开始介绍各种数据类型的表示法之前，要记住二进制的固有特性。1 位（bit）只能是 0 或 1，没有其他的可能，因此，1 位只能表示两种状态之一。例如，如果我们要把食物分成“甜”和“酸”两类，那么只用 1 位二进制数即可。可以规定 0 表示食物是甜的，l 表示食物是酸的。但是，如果要表示更多的分类（如“辣”），1 位二进制数就不能胜任了。

要表示多于两种的状态，就需要多个位。2 位可以表示 4 种状态，因为 2 位可以构成 4 种 0 和 1 的组合，即 00、01、10 和 11。例如，如果要表示一辆汽车采用的是 4 种挡位（停车、发动、倒车和空挡）中的哪一种，只需要 2 位二进制数即可，停车用 00 表示，发动用 01 表示，倒车用 10 表示，空挡用 11 表示。位组合与它们表示的状态很多时候是人为定义的，如果你愿意，也可以用 00 表示倒车。

如果要表示的状态多于 4 种，那就需要两个以上的位。3 位二进制数可以表示 8 种状态，因为 3 位数字可以构成 8 种 0 和 1 的组合。同样地，4 位二进制数可以表示 16 种状态，5 位可以表示 32 种，以此类推。表 3-1 给出了一些位组合。注意，每列中的位组合都是二进制数。

表 3-1 位组合

1 位	2 位	3 位	4 位	5 位
0	00	000	0000	00000
1	01	001	0001	00001
—	10	010	0010	00010
—	11	011	0011	00011
—	—	100	0100	00100
—	—	101	0101	00101
—	—	110	0110	00110
—	—	111	0111	00111
—	—	—	1000	01000
—	—	—	1001	01001
—	—	—	1010	01010
—	—	—	1011	01011

续表

1 位	2 位	3 位	4 位	5 位
—	—	—	1100	01100
—	—	—	1101	01101
—	—	—	1110	01110
—	—	—	1111	01111
—	—	—	—	10000
—	—	—	—	10001
—	—	—	—	10010
—	—	—	—	10011
—	—	—	—	10100
—	—	—	—	10101
—	—	—	—	10110
—	—	—	—	10111
—	—	—	—	11000
—	—	—	—	11001
—	—	—	—	11010
—	—	—	—	11011
—	—	—	—	11100
—	—	—	—	11101
—	—	—	—	11110
—	—	—	—	11111

一般说来，n 位二进制数能表示 2^n 种状态。请注意，每当可用的位数增加一位，可以表示的状态的数量就会多一倍。

反过来，如果想要表示 25 种状态，需要多少位呢？4 位二进制数不够，因为只能表示 16 种状态，因此至少需要 5 位二进制数，它们可以表示 32 种状态。由于我们只需要表示 25 种状态，所以有些位组合没有使用(无意义)。

有时候为了以后扩充或其他原因，即使技术上只需要用最少的位数表示一组状态，也可能多分配一些位。

计算机体系结构一次能够寻址和移动的位数有一个最小值，通常是 2 的幂，如 8、16 或 32。因此，分配给任何类型的数据的最小存储量通常是 2 的幂的倍数。

3.2　数值数据的表示方法

数值是计算机系统最常用的数据类型。对正整数来说，很自然地以它的二进制数来表示和存储。数值的表示还有两个问题需要解决：

(1) 如何处理数字的符号(负数如何表示)？

(2) 对于实数，如何处理小数点？

有多种方法可处理符号问题，本章后面将陆续讨论。对于实数，计算机使用定点和浮点两种不同的表示方法。

数值数据表示的分类如图 3-1 所示。

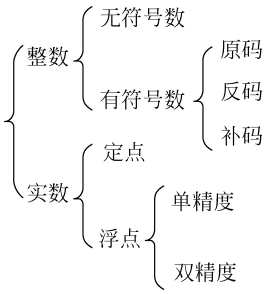

图 3-1　数值数据表示的分类

3.2.1 整数的表示方法

计算机系统中，为了更有效地表示和存储整数，有无符号数和有符号数两种不同的表示整数的方式。

1. 无符号整数表示法

无符号整数就是没有符号的整数，不能表示负数。计算机能表示的无符号整数的最大值是 2^n-1，这里 n 是计算机中分配用于表示无符号整数的二进制位数。例如，$n=8$ 时，能表示的最大无符号整数是 $2^8-1=255$，即表示的范围是 0～255。

1）存储无符号整数

首先将整数转换为二进制数，如果要表示的整数转换为二进制数后不足 n 位，则在二进制整数的左边补 0，使它的总位数为 n。如果位数大于 n，该整数无法存储，会发生溢出，我们随后将讨论这个问题。

【例 3-1】 将十进制数 7 存储在 8 位存储单元中。

解：首先将 7 转换为二进制数$(111)_2$。在左边加 5 个 0 使总位数为 8 位，即$(00000111)_2$，再将该整数存储在存储单元中（因为计算机的存储单元的每一位，要么是 0，要么是 1，不可能为空，前面补 0 不影响数字的大小）。

【例 3-2】 将十进制数 258 存储在 16 位存储单元中。

解：首先将 258 转换为二进制数$(100000010)_2$，在左边加 7 个 0 使总位数达到 16 位，即得到$(0000000100000010)_2$，再将该整数存储在存储单元中。

2）解析无符号整数

有时我们需要将计算机存储设备中的二进制位串显示为一个十进制的无符号整数。

【例 3-3】 当看到内存中的二进制位串 00101011，并且知道它表示的是一个无符号整数，那么它表示的相应的十进制数是多少？

解：使用第 2 章的数值转换方法，二进制整数 00101011 转换为十进制无符号整数 43。

3）溢出

假设计算机的存储单元用 4 位来存储数据，那么它能存储的无符号整数的范围为 0～15。例如，当前存储单元中存储的整数是$(11)_{10}$，又试图再加上 9，就发生了称为溢出的情况。因为 $11+9=20=(10100)_2$，也就是说表示十进制数 20 最少需要 5 位。在存储单元只有 4 位的情况下，计算机会丢掉最左边的位，保留右边的 4 位 0100。这样一来，看到的新的整数是 4（二进制数 0100 等于十进制数 4）而不是 20，就是由于溢出的原因。图 3-2 描述了为什么会发生这种情况。

4）无符号整数的应用

无符号整数表示法因为不存储整数的符号，所有分配的位单元都可以用来存储数字，与后面讲的有符号整数相比，能提高存储的效率。计算机的某些应用中，如果确定不会用到负数，都可以用无符号整数表示。无符号整数可能应用于如下场合。

（1）计数：当我们计数时，不需要负数，可以从 1（有时从 0）开始计数。

（2）地址：有些计算机语言，在一个存储单元中存储了另一个存储单元的地址。地址都是从 0 开始到整个存储器的总字节数的正数，在这里同样也不会用到负数。

（3）其他数据类型：我们后面将讲到的某些其他数据类型（字符、图像、音频和视频）在很多情况下使用的是非负整数。

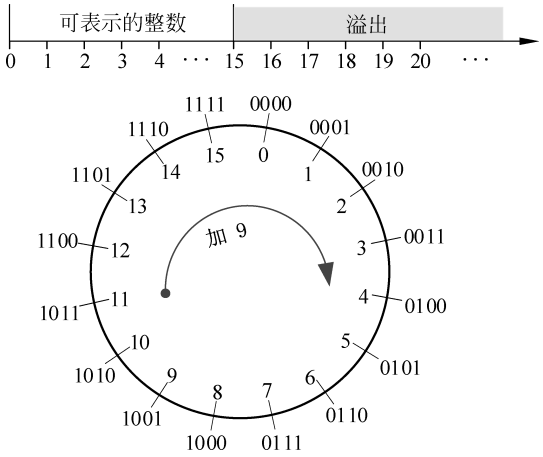

图 3-2　无符号整数的溢出

2. 有符号整数表示法

有符号整数包括负数、零以及正数。计算机中存储的是只有 0 和 1 的二进制串，所以怎么表示负号就成了一个需要解决的问题。

计算机发展过程中，出现了 3 种有符号整数的表示方法，分别是原码、反码以及补码。

1）原码表示法

在这种方法中，n 位二进制数的有效范围 $0\sim2^n-1$ 被分成两个相等的子范围。前一半表示正整数，后一半表示负整数。例如，当存储单元为 4 位时，有效范围是 0000～1111，这个范围被分为两半，0000～0111 以及 1000～1111。原码表示法如图 3-3 所示。

0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	1	2	3	4	5	6	7	−0	−1	−2	−3	−4	−5	−6	−7

图 3-3　原码表示法

原码表示和存储一个整数时，需要分配最左边的 1 个二进制位（有时称为最高位）用于表示符号（0 表示正，1 表示负），剩余的二进制位用来表示数字的绝对值。可存储的数字范围是 $-(2^{n-1}-1)\sim(2^{n-1}-1)$，最大的正数值大约是无符号最大数的一半。例如，在一个 8 位存储单元中（$n=8$），最左边的 1 位分配用于存储符号，其他 7 位表示数字的绝对值，存储范围是 −127～127。注意，原码表示方式中有两个 0：+0 和 −0，即图 3-3 中的 0000 和 1000。

【例 3-4】 用原码表示法将 +28 存储在 8 位存储单元中。

解：先将该整数转换为 7 位的二进制数。然后最左边的 1 位记为 0（下画线标记，下同），一共组成 8 位。

将 28 转换为 7 位的二进制　　0 0 1 1 1 0 0

加符号位并存储　　0　0 0 1 1 1 0 0

【例 3-5】 用原码表示法将 −28 存储在 8 位存储单元中。

解：先将 28 转换为 7 位二进制数。然后最左边的 1 位记为 1，一共组成 8 位。

将 28 转换为 7 位的二进制　　0 0 1 1 1 0 0

加符号位并存储　　1　0 0 1 1 1 0 0

【例 3-6】 将用原码表示法存储的 01001101 转换成十进制整数。

解：因为最左位是 0，所以符号为正。其余位(1001101)转换成十进制数 77，得到整数是 77。

【例 3-7】 将用原码表示法存储的 10100001 转换成十进制整数。

解：因为最左位是 1，所以符号为负。其余位(0100001)转换为十进制数 33，得到的整数是－33。

原码表示法有两个问题。其一，表示 0 的方法有两种，一种是＋0，另一种是－0，这不但浪费空间，而且会引起不必要的麻烦；其二，减法运算不能转换为加法运算。

以 8 位存储单元为例，0＝1－1＝1＋(－1)＝00000001＋10000001＝10000010 ＝－2，这给计算机功能设计增加了复杂性。

基于上面两个原因，现在的计算机中，基本不用原码表示和存储整数了。

2) 反码表示法

反码表示法最高位是符号位(0 表示正，1 表示负)，正数的反码与原码相同，负数的反码是在其原码的基础上，除符号位外各位求反(原来为 0 的变为 1，原来为 1 的变为 0)。0 的反码表示也有两个，即 $(00000000)_2$ 和 $(11111111)_2$。

【例 3-8】 分别将十进制整数 62 和－62 用反码表示法存储在 8 位存储单元中。

解：62 的反码表示与原码表示相同，即 $(00111110)_2$。

－62 的原码表示为 $(10111110)_2$，反码表示时，除了最左侧的符号位，其他位取反，得到 $(11000001)_2$。

3) 补码表示法

当前几乎所有的计算机都使用补码表示法来存储有符号整数。这一方法中，有效范围(0～2^n-1)被分为两个相等的子范围。第一个子范围用来表示非负整数，第二个子范围用于表示负整数。补码能表示的数的范围是 -2^{n-1}～$2^{n-1}-1$。例如，如果 $n=4$，能表示的数的范围是－8～7，0000～0111 表示非负数，1000～1111 表示负数，补码表示法如图 3-4 所示。

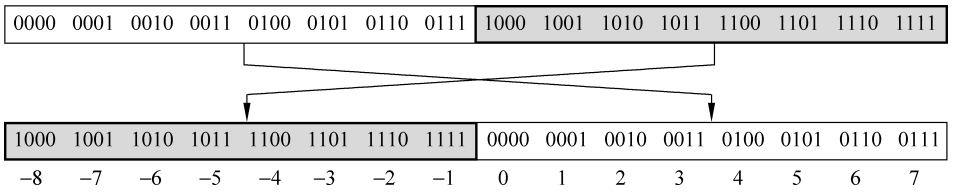

图 3-4 补码表示法

补码表示法的首位(最左位)决定符号，如果最左位是 0，该整数非负，如果最左位是 1，该整数是负数。

补码表示法中，正数的补码与原码相同，负数的补码是在该数反码的最低位加 1。

【例 3-9】 用补码表示法将整数 28 存储在 8 位存储单元中。

解：该整数是正数，其补码表示与原码表示相同，因此在将该整数从十进制转换为二进制数 11100，再在左侧加 3 个 0，使其构成 8 位二进制 00011100。

【例 3-10】 用补码表示法将整数－28 存储在 8 位存储单元中。

解：该整数是负数，因此需要先表示为反码，再加 1。

−28 的 8 位反码：　　11100011

＋　　1

＝　　11100100

【例 3-11】 用补码表示法将整数−28 存储在 16 位存储单元中。

解：该整数是负数，因此需要先表示为反码，再加 1。

−28 的 16 位反码：　　1111111111100011

＋　　1

＝　　1111111111100100

已知一个补码表示的二进制串，求它表示的实际十进制数字，可以按下述方法计算。如果最高位为 0，表明该数为正数，直接将二进制串转换为十进制即可；如果最高位为 1，表明该数为负数，将该二进制串每位都取反，再加 1，得到的二进制串转换为十进制，前面加上负号。

【例 3-12】 已知在 8 位存储单元中存储的补码表示的二进制串 00001101，求它表示的实际十进制整数。

解：最左位是 0，因此符号为正，将该整数转换为十进制即可，$(00001101)_2=(13)_{10}$。

【例 3-13】 已知在 8 位存储单元中存储的补码表示的二进制串 11100110，求它表示的实际的十进制整数。

解：最左位是 1，因此符号为负，在整数转换为十进制前进行补码运算。

　　11100110

取反：　00011001

＋　　1

＝　　00011010

$(00011010)_2=(26)_{10}$，加上负号，得到−26。

4）补码表示法的溢出

同其他表示法一样，补码表示法存储的整数也会溢出。图 3-5 显示了当使用 4 位存储单元存储一个带符号的整数时出现的正负两种情况的溢出。当试图存储一个比 7 大的正整

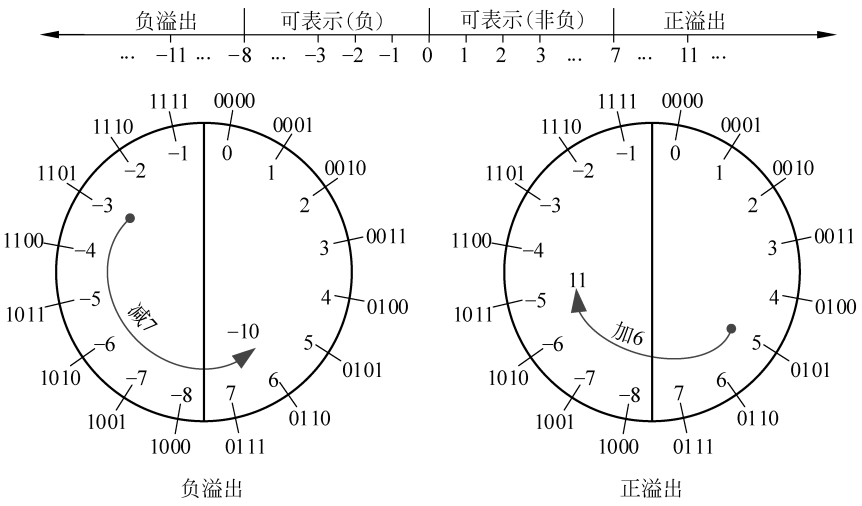

图 3-5　补码表示法的溢出

数时，出现正溢出。例如，存储单元保存的是整数 5，想再加上 6，我们期望的结果是 11，但实际得到的值为－5。从图中可以形象地看到，从 5 开始顺时针走 6 个单位，就停在－5。当我们试图存储一个比－8 还小的负整数时，出现负溢出。例如，整数－3 再减去 7，我们期望的结果是－10，但实际得到的值为＋6。从图中可以形象地看到，从－3 开始逆时针走 7 个单位，就停在＋6 了。

补码表示法的优点如下。

(1) 与原码和反码不同，补码不区分＋0 和－0，只有一种 0 的表示。

(2) 补码表示法的符号位可以直接参与运算，而且减法能转化为加法运算，在不溢出的情况下，能得到正确的结果。

例如，$0=1-1=1+(-1)=0001_{【补码】}+1111_{【补码】}=0000_{【补码】}=0$。

正因为有上述优点，所以现在的计算机基本都采用补码表示和存储有符号整数。

3.2.2 实数的表示方法

实数具有整数部分和小数部分。例如，104.32、0.999999、357.0 和 3.14159 都是十进制实数。

1. 实数的浮点表示法

数 N 的浮点形式可写成：$N=\pm M\times B^{E}$，其中，M 代表尾数，B 代表基，十进制数的基就是 10，二进制数的基是 2，E 代表阶码(指数)。

例如，十进制数 74250000.00 可以写成 7.425×10^{7}，可以看到它包含 3 部分，其中符号为＋(正号隐含)，尾数部分为 7.425，指数为 7；－0.000000000232 可以写成 -2.32×10^{-10}，其中符号为－，尾数为 2.32，指数为－10。

用浮点格式表示二进制数字$(1010010000000)_2$，使用类似十进制的方法，小数点前只保留一位数字，可以表示为 1.01001×2^{12}，其中符号为＋(隐含)，指数为 12(这里为了方便理解，写成十进制数 12，实际情况下它以二进制数字 1100 存储)，尾数为 1.01001。

2. 规范化

为了使表示法的固定部分统一，科学记数法(用于十进制)和浮点表示法(用于二进制)都在小数点左边使用了唯一的非零数码，称为规范化。

＋	2^6	×	1.0001110101
＋	6		0001110101
↑	↑		↑
符号	指数		尾数

图 3-6 规范化表示

$(1000111.0101)_2$ 可以表示成如图 3-6 所示的规范化形式。

规范化以后，在计算机中表示时，需要考虑符号、指数和尾数这 3 部分的存储。因为所有的浮点数在规范化以后，尾数部分都是 1.XXXX 的形式，所以尾数部分小数点和左边的 1 不需要存储。

(1) 符号：符号可以用一个二进制位存储，0 表示正数，1 表示负数。

(2) 指数：指数(2 的幂)可以为正，也可以为负，用余码表示法(后面讨论)表示。

(3) 尾数：尾数是指小数点右边的二进制数，它定义了该数的精度。尾数是作为无符号整数存储的，但需要记住，它不是整数。强调这一点是因为如果在尾数的左边插入多余的零，表示的值将改变，而在一个真正的整数中，如果在数字的左边插入多余的零，值是不会改变的。

3. 余码系统

浮点数的指数部分是有符号数，尽管可以用补码表示法存储，但现在计算机通常都使用一种称为余码的表示法表示指数。该表示法将原来的指数（可能是正数，也可能是负数）加一个正整数（称为偏移量），指数都变成了正数，可以用无符号数存储。这个偏移量的值是 $2^{m-1}-1$，m 是内存单元存储指数的位数。余码表示法的优点在于，当对指数进行比较或运算时不需要考虑正负。

4. IEEE 标准

IEEE 754 定义了几种存储浮点数的标准，这里只讨论单精度和双精度这两种最常用的格式（如图 3-7 所示）。方框上方的数是每一项的位数。

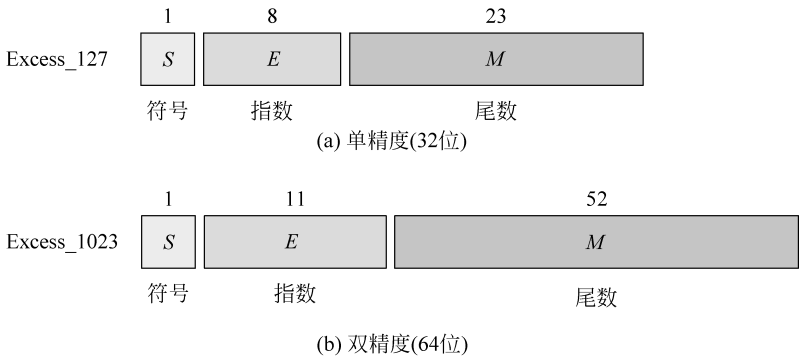

图 3-7　IEEE 单精度和双精度表示

单精度格式采用 32 位存储一个浮点数，符号占用 1 位（0 为正，1 为负），指数占用 8 位（余码使用的偏移量为 $2^{8-1}-1=127$），尾数占用 23 位（无符号数）。

双精度格式采用 64 位存储一个浮点数，符号占用 1 位（0 为正，1 为负），指数占用 11 位（余码使用偏移量为 1023），尾数占用 52 位。

一个实数可以按以下方法存储为 IEEE 标准浮点数格式。

（1）在 S 中存储符号（0 或 1）。

（2）将数字转换为二进制。

（3）规范化。

（4）找到 E 和 M 的值。

（5）连接 S、E 和 M。

【例 3-14】 写出十进制数 6.75 的单精度表示。

解：

（1）符号为正，所以 $S=0$。

（2）十进制转换为二进制：$(6.75)_{10}=(110.11)_2$。

（3）规范化：$(110.11)_2=(1.1011)\times 2^2$。

（4）指数部分 E 使用余码，所以 $E=2+127=129=(10000001)_2$。

尾数 $M=1011$，需要在 M 的右边增加 19 个 0 使之成为 23 位。

（5）连接 S、E 和 M。

0　　10000001　　10110000000000000000000

S　　E　　M

将3部分合并，存储在计算机中的二进制数字是01000000110110000000000000000000。

【例3-15】 写出十进制数－161.875的单精度表示。

解：

(1) 符号为负，所以 $S=1$。

(2) 十进制转换为二进制：$(161.875)_{10}=(10100001.111)_2$。

(3) 规范化：$(10100001.111)_2=(1.0100001111)\times 2^7$。

(4) 指数 $E=7+127=134=(10000110)_2$，尾数 $M=(0100001111)_2$。

(5) 连接 S、E 和 M。

1	10000110	01000011110000000000000
S	E	M

存储在计算机中的二进制数字是11000011001000011110000000000000。

【例3-16】 写出十进制数－0.0234375的单精度表示。

解：

(1) 符号为负，所以 $S=1$。

(2) 十进制转换为二进制：$0.0234375=(0.0000011)_2$。

(3) 规范化：$0.0000011=1.1\times 2^{-6}$。

(4) 指数 $E=-6+127=121=(01111001)_2$，尾数 $M=(1)_2$。

(5) 连接 S、E 和 M。

1	01111001	10000000000000000000000
S	E	M

存储在计算机中的二进制数字是10111100110000000000000000000000。

3.3 字符的表示方法

字符(Character)是文字与符号的总称，包括文字、图形符号、数学符号等。一组字符的集合就是字符集(Charset)。字符集常常和一种具体的语言文字对应起来，该文字中的所有字符或大部分常用字符就构成了该文字的字符集，如英文字符集、汉字字符集、日文汉字字符集等。

我们在计算机屏幕上看到的字符，不管是英文字符还是汉字字符，在计算机中是怎么表示和存储的呢？我们已经知道，计算机实质上只能存储二进制(0和1)序列串，计算机要处理各种字符，就需要将每个字符和一个二进制序列串对应起来，这种对应关系就是字符编码(Character Encoding)。

制定编码首先要确定字符集，并将字符集内的字符排序，然后和二进制数字对应起来。根据字符集内字符的多少，需要确定用几位二进制来编码，二者是对数关系。如果需要编码的字符有 n 个，那么至少需要 $\mathrm{lb}n$ 位二进制来编码。

每种编码都限定了一个明确的字符集合，叫作被编码过的字符集(Coded Character Set)，这是字符集的另外一个含义，通常所说的字符集大多是这个含义。下面介绍几种常见的字符集。

1. ASCII

ASCII(American Standard Code for Information Interchange)是目前计算机中使用最广泛的字符集及编码,由美国国家标准局(American National Standards Institute,ANSI)制定。它被国际标准化组织(International Organization for Standardization,ISO)认定为国际标准,称为 ISO 646 标准。

基本的 ASCII 字符集共有 128 个字符,其中有 96 个可打印字符,包括常用的字母、数字、标点符号等,另外还有 32 个控制字符。标准 ASCII 使用 7 个二进制位对字符进行编码。例如,大写字母 A～Z 的 ASCII 编码依次从 65(二进制为 1000001)到 90,小写字母 a～z 的 ASCII 编码依次从 97 到 122,相应的大小写字母之间差 32。数字字符(0,1,2,…,9)对应的 ASCII 编码从 48 到 57。具体 ASCII 码表请参考附录。

虽然标准 ASCII 码是 7 位编码,但由于计算机基本处理单位为字节(1 字节＝8 位),所以一般仍以一个字节存放一个 ASCII 字符。每一个字节中多余出来的一位(最高位)在计算机内部通常保持为 0(在数据传输时可用作奇偶校验位)。

2. 汉字编码

1) GB2312

ASCII 编码只有 8 位,即使将最高位利用起来,总共也只能表示 $2^8=256$ 个字符。为了满足在计算机中使用汉字的需要,中国国家标准总局发布了一系列的汉字字符集国家标准编码,统称为 GB 码,或国标码。其中最有影响的是于 1980 年发布的《信息交换用汉字编码字符集 基本集》,简称为 GB2312。GB2312 编码通行于中国大陆,新加坡也采用此编码。

GB2312 是一个简体中文字符集,编码用两个字节(16 位二进制)表示一个汉字,所以理论上最多可以表示 $2^{16}=65536$ 个汉字。由于兼容等方面的限制,实际上 GB2312 由 6763 个常用汉字和 682 个全角的非汉字字符组成。

为了与 ASCII 兼容,GB2312 字符在进行存储时,通过将原来的每个字节的第 8 位(最左位)设置为 1,同西文加以区别。如果第 8 位为 0,表示西文字符,否则表示 GB2312 中的字符。例如,汉字“啊”在计算机中存储的是十六进制 B0A1,即二进制数 1011000010100001,注意每个字节最高位(下画线处)为 1。

2) Big5

在中国台湾、中国香港与中国澳门地区,使用的是繁体中文,而 GB2312 并不支持繁体汉字。在使用繁体中文字符集的地区,1984 年制定了一种繁体中文编码方案,被称为大五码,英文写作 Big5。

Big5 包括 13053 个繁体汉字,808 个标点符号、希腊字母及特殊符号。因为 Big5 的字符编码范围同 GB2312 的编码范围存在冲突,所以在一个文档中不能同时支持两种字符集的字符。

3) GBK

GB2312 的出现基本满足了汉字的计算机处理需要,但对于人名、古汉语等方面出现的罕用字,GB2312 不能处理,这导致了后来 GBK 及 GB18030 汉字字符集的出现。GBK 编码标准兼容 GB2312,共收录汉字 21003 个、符号 883 个,并提供 1894 个造字码位,将简、繁体字融于一体。

3. Unicode

为了将全世界常用文字都包括进来，计算机制造商联合共同设计了一种名为 Unicode 的编码，它为每种语言中的每个字符设定了统一并且唯一的二进制编码，以满足跨语言、跨平台进行字符转换、处理的要求。Unicode 目前已经相当普及。

Unicode 的实现方式与编码方式不同。一个字符的 Unicode 编码是确定的，但是在实际传输过程中，由于不同系统平台，以及出于节省空间的目的，对 Unicode 编码的实现方式有所不同。Unicode 的实现方式称为 Unicode 转换格式（Unicode Translation Format，UTF），常用的有 UTF-8 和 UTF-16。

3.4 音频的表示方法

音频是声音或音乐的表现，本质上与上面讨论过的数值和字符不同。不同字符个数是可数的，而音频是不可数的，是随时间变化的模拟数据。在计算机中存储和处理音频信号，必须对其进行数字化，方法是按照一定的频率（时间间隔）对声音信号的幅值进行采样，然后对得到的一系列数据进行量化与二进制编码处理，即可将模拟声音信号转换为相应的二进制序列。这种数字化后的声音信息就能被计算机存储、传输和处理。当需要计算机播放数字化的音频时，会将数字信号还原为模拟信号播放，这样就可以听到声音了。

1. 采样

我们不可能记录一段音频信号的所有幅值，只能记录其中的一些。每隔一段时间在模拟声音波形上取一个幅值的过程称为采样，可以记录这些采样值来表现模拟信号。图 3-8 显示了在 1s 音频信号上采样 10 次的状况。

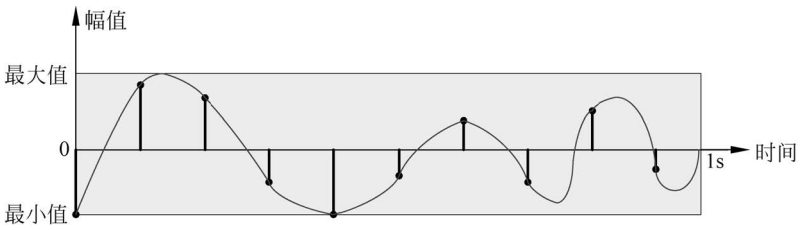

图 3-8 音频信号的采样

每秒钟需要采样多少次才能还原出原始音频信号呢？通常每秒 40 000 个左右样本的采样率就能很好地还原出音频信号。采样率低于这个值，人耳听到的还原声音会失真。较高的采样率当然会更保真，生成的声音质量更好，但到达某种程度后，再提高采样率，人耳已分辨不出差别了，这意味着白白浪费数据，多占用存储空间。

2. 量化

采样得到的值是实数，这意味着可能要为每一秒的样本存储几万个实数值。为了减少存储量，每个样本使用一个整数表示更合适。量化是将样本的值截取为最接近的整数值。例如，实际的采样值为 17.2，就可截取为 17；如果采样值为 17.7，就可截取为 18。

3. 编码

量化后的样本值需要被编码。一些系统的样本取值有正有负，另一些系统通过把曲线移动到正的区间从而只有正值。换言之，一些系统使用有符号整数表示样本，而另一些系统

使用无符号整数表示。有符号整数可以用补码或原码表示。

每个样本编码时，系统需要决定分配多少位来表示它。早先仅有 8 位分配给声音样本，现在每个样本用 16、24 甚至 32 位表示都较常见。另外，编码时会在数据中加入一些用于纠错、同步和控制的数据。

4. 声音编码标准

当今音频编码的主流标准是 MP3(MPEG Layer 3 的简写)，该标准是 MPEG 音频压缩编码标准的一部分。它采用的方案是每秒采样 44100 个样本，每个样本用 16 位编码，再使用信息压缩方法进行压缩，压缩后再存储。

3.5　图像和图形的表示方法

图像(Image)和图形(Graph)这两个术语有时会混用，严格意义上讲，图像是按照一个一个像素点(光栅图格式)存储的，而图形是按照几何形状描述(矢量图格式)存储的。

3.5.1　图像的表示方法

1. 像素

照片等自然场景的图像在计算机中是按光栅图(也称为位图)存储的。将图像在行和列的方向均匀地划分为若干个小格，每个小格称为一个像素，一幅图像的尺寸可以用像素点来衡量。像素点通常都很小，为了能直观地理解像素的概念，将图像的一部分放大，如图 3-9 所示，从放大的图像中看到的每一个小方块就是一个像素。

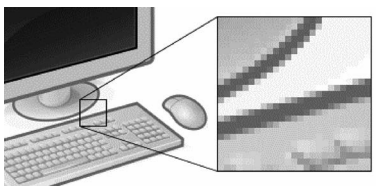

图 3-9　图像是由像素构成的

数码相机等图像数字化设备在拍照时，将连续的模拟图像信号转换为离散的数字信号，也就是像素表示的数字图像，图 3-10 展示了图像数字化过程，输出数字图像的每个小格就是一个像素。

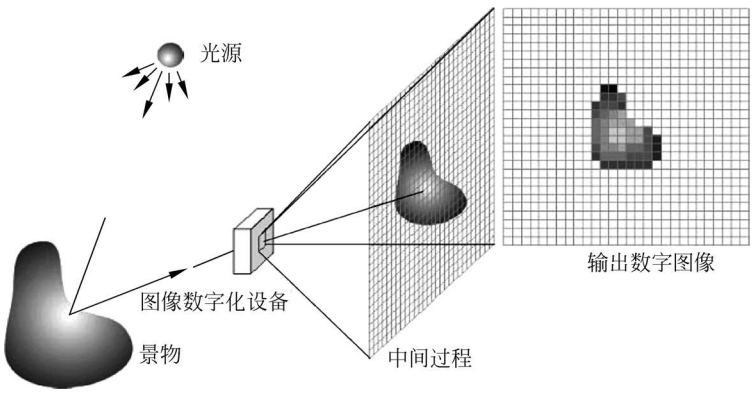

图 3-10　图像数字化过程

图像中像素点的个数称为分辨率，用“水平像素点数×垂直像素点数”表示。图像的分辨率越高，构成图像的像素点就越多，能表示的细节就越多，图像就越清晰；反之，分辨率越低，图像就越模糊。

2. 图像表示

存储图像本质上就是存储图像每个像素点的信息。根据色彩信息可将图像分为黑白图像、灰度图像和彩色图像。

1）黑白图像

黑白图像(有时也称二值图像、单色图像)只有黑和白两种颜色，因此构成它的每个像素只需要 1 位就能表示(通常用 0 表示黑色，1 表示白色)。一幅宽为 400 像素，高为 300 像素的图像需要 400×300×1＝120000b＝15000B 来存储。

2）灰度图像

灰度图像的每个像素可以由纯黑，深黑，深灰，…，浅灰，纯白构成，为了表示不同的灰度层次，通常需要用 1B(8b)表示一个像素，这样可以表示 2^8(0～255)种不同的状态。0 表示纯黑，1～254 表示从深到浅的不同灰度，255 表示纯白。黑白图像和灰度图像的例子如图 3-11 所示。

(a) 黑白图像　　(b) 灰度图像

图 3-11　黑白图像与灰度图像

3）彩色图像

彩色图像的表示分为真彩色和索引色。

(1) 真彩色

我们知道，任何颜色都可以用红、绿、蓝 3 种颜色混合得到。真彩色使用 24 位编码一个像素，在该技术中，红、绿、蓝三原色(Red Green Blue，RGB)每一种都用 8 位表示。因为 8 位可以表示 0～255 的数，所以每种颜色都由为 0～255 的 3 组数字表示。真彩色模式可以编码 2^{24} 即 16777216 种颜色。表 3-2 显示了真彩色的一些颜色。

表 3-2　真彩色的一些颜色

颜色	R	G	B
黑色	0	0	0
红色	255	0	0
绿色	0	255	0
蓝色	0	0	255
黄色	255	255	0
青色	0	255	255
洋红	255	0	255
白色	255	255	255

(2) 索引色

真彩色模式使用了超过 1600 万种颜色，许多应用程序其实不需要如此大的颜色范围，索引色(或调色板色)模式仅使用其中的一部分。在该模式中，每个应用程序从大的色彩集中选择一些颜色(通常是 256 种)并对其建立索引。对选中的颜色赋一个 0～255 的值。这就好比艺术家可能在他们的画室用到很多种颜色，但一次仅用到调色板中的一些。索引色的使用减少了存储一个像素所需要的位数。索引色模式通常使用 256 个索引，需要用 8 位存储一个像素。

3. 图像格式

常见的图像格式有 BMP、JPG(JPEG)、PNG、GIF、PCX 等。BMP 是一种与硬件设备无关的图像文件格式，使用非常广泛，它不进行任何压缩，因此，BMP 文件所占用的空间很大。JPEG 使用真彩色模式，通过压缩图像来减少存储量。GIF 标准使用索引色模式。

3.5.2　图形的表示方法

图像像素表示有两个缺点，其一是文件尺寸太大，其二是重新调整图像大小不方便。放大位图图像意味着扩大像素，放大后的图像看上去很粗糙。图形(矢量图)编码方法并不存储每个像素的值，而是把一个图分解成几何图形的组合，如线段、矩形或圆形。每个几何形状由数学公式表达，线段可以由它端点的坐标描述，圆可以由圆心坐标和半径长度描述。矢量图是由定义如何绘制这些形状的一系列命令构成的。

当要显示或打印矢量图时，将图像的尺寸作为输入传给系统，系统重新设计图像的大小，并用相应的公式画出图像。

例如，考虑半径为 r 的圆形，程序需要绘制该圆的主要信息如下。

(1) 圆的半径 r。

(2) 圆心的位置(坐标)。

(3) 绘制的线型和颜色。

(4) 填充的类型和颜色。

当该圆的大小改变时，程序改变半径的值并重新计算这些信息以便再绘制一个圆。改变图像大小不会改变绘图的质量。

矢量图不适合存储细微精妙的照片图像，它适合存储采用几何元素创建的图形，如 TrueType 和 PostScript 字体、计算机辅助设计、工程绘图等。

3.6　视频的表示方法

视频是图像（称为帧）的时间推移的表示形式，视频由一系列连续放映的帧组成。所以，如果知道如何将一幅图像存储在计算机中，也就知道如何存储视频了。每一幅图像或帧按照位图模式储存，这些图像组合起来就可表示视频。需要注意的是，视频通常需要进行压缩后再存储，否则存储容量太大。

常见的视频格式有 MPEG、AVI、MOV、WMV、MKV、RMVB 等。

※读者可观看本书配套视频 4：计算机数据表示总结。


微课视频

3.7 小结

计算机操作的数据包括数值、字符、声音、图像、图形、视频。由于计算机只能操作二进制数值，所以所有类型的数据都必须表示为二进制形式。

无符号整数由它们对应的二进制数表示，有符号整数常用补码表示。实数通常按IEEE 754标准表示，由符号、尾数和指数3部分构成。字符根据使用的字符集不同，有不同的编码标准，常用的有西文ASCII编码、几种汉字编码，以及能表示各种文字的Unicode编码。模拟的声音信号经过采样、量化、编码后被表示为数字化的音频。图像通过像素表示，图形通过几何形状表示。视频由一系列图像构成。

计算机本质上只能存储二进制数，也就是0和1组成的序列串。这些序列串在不同的应用场合表达的含义不同。例如，二进制串10111100110000000000000000000000，如果出现在无符号整数运算里，它表示十进制数3166699520；如果出现在有符号整数的运算里，它表示十进制数－1128267776；如果出现在浮点运算里，它表示－0.0234375；如果出现在字符串处理里，根据字符编码不同，它表示若干个字符；如果出现在图像文件中，它表示几个像素点；如果出现在声音文件中，它表示某个音符。

3.8 习题

1. 判断题

(1) 计算机用模拟形式表示信息。（　　）

(2) 计算机系统内部使用二进制表示信息。（　　）

(3) 4个二进制位可以表示32种状态。（　　）

(4) 使用原码表示法时，0有两种表示方法。（　　）

(5) 整数数值最常用的是补码表示法。（　　）

(6) 当为计算结果分配的位容不下计算出的值时，将发生溢出。（　　）

(7) 在ASCII字符集中，大写字母和小写字母没有区别。（　　）

(8) Unicode字符集包括ASCII字符集中的所有字符。（　　）

(9) 音频信号的数字化需要进行采样。（　　）

(10) MP3音频格式是一种压缩格式。（　　）

(11) RGB值用3个数字值表示一种颜色。（　　）

(12) 图像格式只有BMP、GIF和JPEG 3种。（　　）

2. 单项选择题

(1) 一个字节包含_____位。

A. 2　　B. 4　　C. 8　　D. 16

(2) 在一个有64种符号的集合中，要表示所有的符号，每个符号最少需要的位长度为_____位。

A. 4　　B. 5　　C. 6　　D. 7

(3) 10位可以表示_____种符号。

A. 128　　B. 256　　C. 512　　D. 1024

(4) 假如 E 的 ASCII 码是 1000101,那么 e 的 ASCII 码是_____。

A. 1000110　　B. 1000111　　C. 0000110　　D. 1100101

(5) 使用 16 位编码的字符集是_____。

A. ANSI　　B. Unicode　　C. GB 2312　　D. 扩展 ASCII 码

(6) 图形(Graph)在计算机中通常使用_____方法来表示。

A. 位图　　B. 矢量图　　C. 补码系统　　D. 答案 A 和 B

(7) 在计算机中表示图像的_____表示方法中,每个像素用一位表示。

A. 位图　　B. 矢量图　　C. 量化　　D. 二值图像

(8) 当我们存储音乐到计算机中时,音频信号必须要_____。

A. 采样　　B. 量化　　C. 编码　　D. 以上全部

(9) 在数值的_____表示法中,如果最左边一位为 0,其表示的十进制数是非负的。

A. 补码　　B. 浮点　　C. 余码　　D. 答案 A 和 B

(10) 在数值的_____表示法中,如果最左边一位为 1,其表示的十进制数是负的。

A. 补码　　B. 浮点　　C. 余码　　D. 答案 A 和 B

(11) _____数字表示方法能存储小数部分。

A. 无符号整数　　B. 补码　　C. 余码　　D. 以上都不是

(12) 浮点表示实数时,计算机存储_____。

A. 符号　　B. 指数　　C. 尾数　　D. 以上全部

(13) 存储于计算机中的数字的小数部分的精度由_____定义。

A. 符号　　B. 指数　　C. 尾数　　D. 以上全部

3. 问答题

(1) 请给出下面整数的原码、反码与补码的表示(8 位二进制)。

A. 127　　B. −127　　C. 33　　D. −100

(2) 将下列 8 位二进制补码表示的数用十进制表示出来。

A. 10000000　　B. 11001110　　C. 11111111　　D. 01001011

(3) 浮点数表示时为什么需要规范化?

(4) 计算机中数值信息(包括整数和浮点数)是怎么表示的?

(5) 计算机中字符信息是怎么表示的?常用的字符集有哪些,它们各有什么特点?

(6) 计算机中音频信息是怎么表示的?

(7) 计算机中图像(位图)是怎么表示的?有哪些图像格式?

(8) 计算机中图形是怎么表示的?