TURING 高等院校计算机教材系列

C语言程序设计

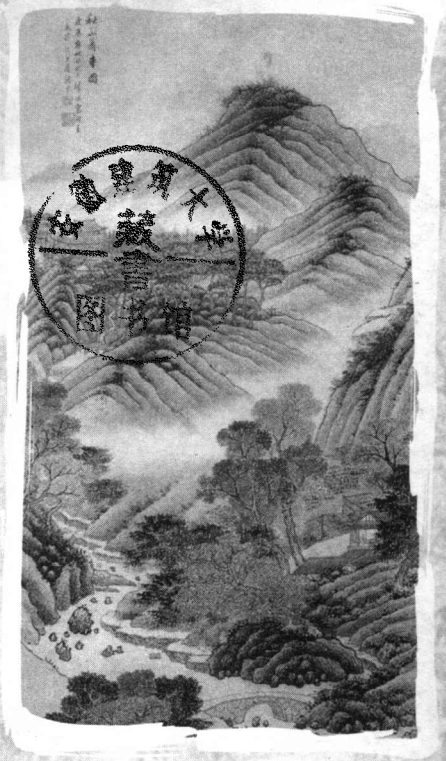
楼永坚　吴鹏　徐恩友　编著

人民邮电出版社
POSTS & TELECOM PRESS

TURING 高等院校计算机教材系列

C语言程序设计

楼永坚　吴鹏　徐恩友　编著



人民邮电出版社
POSTS & TELECOM PRESS

图书在版编目（CIP）数据

C 语言程序设计 / 楼永坚，吴鹏，许恩友编著. —北京：人民邮电出版社，2006.10
（高等院校计算机教材系列）

ISBN 7-115-15096-6

Ⅰ. C… Ⅱ. ①楼… ②吴… ③许… Ⅲ. C 语言—程序设计—高等学校—教材
Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2006）第 089825 号

内 容 提 要

本书是作者在讲授“计算机导论”和“高级语言程序设计”课程的基础上，根据多年的教学经验对授课内容做了深入细致的研究后，依据当前计算机教学系列改革要求整理编著而成的。全书共 12 章，知识覆盖面广（涵盖了 C99 标准），内容由浅入深，包括计算机基础知识、程序设计基础知识、编程语言与 C 概述、数据类型、运算表达式与基本输入/输出、控制语句、数组、指针、函数、结构体与共用体、位运算和文件等，然后在上述知识点的基础上进一步介绍 C 语言的高级应用，包括线性表、栈、队列的概念与应用。鉴于学习程序设计的重要环节是上机，本书最后一章配合教材的内容，提供了 12 个实训。

本书适合作为高等院校计算机专业 C 语言课程的教材，也可以作为大学各专业计算机程序设计入门教学用书，授课内容、习题和实训可根据实际情况进行选用。

高等院校计算机教材系列

C 语言程序设计

◆ 编　　著　楼永坚　吴鹏　徐恩友
责任编辑　杨海玲

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京顺义振华印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：800 × 1000　1/16
印张：18.5
字数：437 千字　　2006 年 10 月第 1 版
印数：1–4 000 册　　2006 年 10 月北京第 1 次印刷

ISBN 7-115-15096-6/TP • 5612

定价：28.00 元

读者服务热线：(010)88593802　印装质量热线：(010)67129223

前　言

C 语言功能丰富，使用灵活，可移植性好，既具有高级语言的优点，又具有低级语言的许多特点，既可以用来编写系统软件，又可用于编写应用软件，是许多计算机专业人员和计算机爱好者学习程序设计语言的首选。如何使教师和学生轻松、愉快地完成 C 语言程序设计课程的教学和学习，具有重要的学术理论价值和社会实践意义。一本好的教材是完成这一使命的第一步。本书旨在教授程序设计基础和 C 语言基础。书中内容广泛，不仅仅在讲一门语言，更重要的是体现了一种编程思想。全书采用了循序渐进的方法，由浅入深地分析关键的知识点，同时给出了大量具有代表性的例子，使学生不仅可以掌握 C 语言程序设计，也为以后的数据结构、编译原理等课程打下良好的基础。

针对C语言比较难学的情况，作者对全书内容做了精心安排，用通俗易懂的语言和丰富的例题帮助学生理解复杂的概念，着重培养学生的应用能力。全书主要内容分为12章，基本安排如下。

(1) 目前有些高校的计算机科学与技术、计算机通信和软件工程等专业的培养计划中将 C 语言程序设计作为第一学期的专业基础课，而“计算机导论”课或同时开课或被取消，导致 C 语言程序设计课程中的某些内容与导论课中的无法衔接，或者因取消导论课而使得知识体系不完整。所以本书的第 1 章和第 2 章中简单回顾了“计算机导论”课中计算机计算基础、程序设计基础和软件工程基础等方面的内容。同时介绍了编程语言的发展和特点、结构化程序设计方法、算法以及良好的程序设计风格等，使学生对程序设计概念和程序设计语言有一个总体的了解。第 3 章结合简单的实例使学生对 C 程序的组成、结构和开发有一个大致的了解，引导学生进入 C 程序设计的学习。

(2) 根据教学经验，函数的使用是程序设计语言初学者不易理解和掌握的第一个难点，而函数对于程序设计及后续课程的学习又相当重要。所以书中安排第 4 章至第 7 章来描述程序设计中的基本概念与应用，如变量与表达式、控制结构、数组、指针等。在掌握了这些基本概念与简单应用的基础上，第 8 章引入函数的结构与应用，从而避免了过早提出函数使学生往往难以接受的弊端。指针是 C 语言程序设计的典型特性和中心之一，也是学习的最难点，第 7 章介绍指针的基本概念与应用，在使学生接受指针概念及基本用法的基础上，在后续章节中对指针的运用进行了更详细的描述，如指针与函数、指针与结构体等，目的是使学生有一个比较长的时间来学习指针和掌握指针由简到难的应用。因此，围绕数组、指针和函数，在章节安排上区分了基础应用和高级应用，目的是对 C 语言程序设计中的数组、指针和函数这些重点螺旋式地提升难度，提高学生的实际应用和编程能力。

(3) C 语言程序设计是一门实践性很强的课程，最后一章的实训有助于培养学生的实际应用能力。实训中使用 Visual C++ 6.0 作为编程平台，每个练习都结合实际情况，在内容安排上由浅入深、由易渐难，增强学生学习的信心。

(4) 本书最后的附录是对全书内容的补充和总结，介绍了 Visual C++ 6.0 环境下的 C 程序调试与测试和常用 ANSI 库函数的接口。附录 D 则对 C 语言的语法和结构进行了回顾和总结，有利于学生复习。

此外，为了方便教师教学，使用本书的授课老师可以免费获得电子课件（可从 http://www.turingbook.com 下载）。

本书第 1 章至第 5 章以及第 12 章的 12.2~12.5 节和附录 A、附录 B 由楼永坚编写，第 6 章至第 9 章以及第 12 章的 12.6~12.9 节由吴鹏编写，第 10 章和第 11 章以及第 12 章的 12.1 节、12.10 ~ 12.12 节和附录 C、附录 D、附录 E 由徐恩友编写。薛德东同学参加了本书第 6 章至第 9 章程序的调试与修改。全书由楼永坚统稿。

由于时间仓促及编者水平有限，谬误不足之处在所难免，敬请读者批评指正。

作者联系方式：

楼永坚：louyjhz@hziee.edu.cn

吴鹏：ewupeng@gmail.com

徐恩友：xuenyou@163.com

编　　者

2006 年 8 月于杭州电子科技大学

目　录

第 1 章

基 础 知 识

1.1 计算机运算基础

由于计算机内部采用二进制计数系统，二进制是计算机运算的基础，因此，了解二进制的特点、与其他数制之间的转换关系以及信息编码等概念，是学习计算机程序设计的前提。

1.1.1 进位计数制

进位计数制是指用一组特定的数字符号按照一定的进位规则来表示数的计数方法。使用任何一种计数制都必须了解两个重要概念：基数和位权。

1. 基数

进位计数制中所使用的不同基本符号的个数称为该进位计数制的基数。例如，十进制共有 10 个基本符号（ 0, 1, 2,…, 9 ），其基数为 10；二进制共有 2 个基本符号（ 0, 1 ），其基数为 2。

2. 位权

位权是进位计数制中的一种因子，是一个乘方值，乘方的底数为进位计数制的基数（本例中为 10），而指数由各位数字在数中的位置来决定。位权的概念可用以下实例来说明。十进制数 3643.76 可表示为 $3643.76 = 3\times10^3+6\times10^2+4\times10^1+3\times10^0+7\times10^{-1}+6\times10^{-2}$。在这个数中，有些相同的数字处在不同的位置，因而所代表的数值的大小也不同，各位数字所代表的数值的大小由位权来决定。在这个十进制数中，从左至右各位数字的位权分别为 10^3、10^2、10^1、10^0、10^{-1}、10^{-2}。

因此，任意进制的数都可以表示为它的各位数字与位权乘积之和。假设有一个 R 进制的数 P 共有 m 位整数和 n 位小数，每位数字用 D_i（$-n\leqslant i\leqslant m-1$）表示，即 $P=D_{m-1}D_{m-2}\cdots D_1D_0D_{-1}\cdots D_{-n}$，它可展开为：

$$P = D_{m-1}\times R^{m-1}+D_{m-2}\times R^{m-2}+\cdots+D_0\times R^0+D_{-1}\times R^{-1}+\cdots+D_{-n}\times R^{-n}$$

此多项式的值即为 R 进制的数 P。

在各种进位计数制中，十进制是人们最熟悉的。但是计算机内部使用二进制，八进制和十六进制则可看成是二进制的压缩形式（见表 1-1）。十六进制数的表示需要特别说明。十六进制共有 16 个基本符号（其中包括 6 个英文字母），依次为 0、1、2、3、4、5、6、7、8、9、A、B、C、

D、E、F。例如，32A、CD8B、4F7等都是十六进制数。为了避免各种进位计数制的数在使用时产生混淆，在给出一个数时，应指明它的进位计数制，通常用下标10、2、8、16或者字母D、B、O、H分别表示十进制、二进制、八进制和十六进制，如$(1124)_{10}$、$(11011)_2$、$(374)_8$、$(4FE)_{16}$、$(1124)_D$、$(11011)_B$、$(374)_O$、$(4FE)_H$等。

表1-1 四种进位制数的对应关系

十进制	二进制	八进制	十六进制
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

计算机内部采用二进制主要有以下几方面的原因。

- 容易表示，电路简单可靠。二进制数只有“0”和“1”两个基本符号，易于用两种对立的物理状态表示。例如，可用电灯开关的“闭合”状态表示“1”，用“断开”状态表示“0”；晶体管的导通表示“1”，截止表示“0”。一切有两种对立稳定状态的器件都可以表示二进制的“0”和“1”。而十进制数有10个基本符号（0, 1, 2,…, 9），要用10种状态才能表示，实现起来很困难。
- 运算简单。二进制数的算术运算特别简单，加法和乘法仅各有3条运算规则（0+0=0，0+1=1，1+1=10和0×0=0，0×1=0，1×1=1），运算时不易出错。
- 逻辑性强。二进制数的“1”和“0”正好可与逻辑值“真”和“假”相对应，这样就为计算机进行逻辑运算提供了方便。算术运算和逻辑运算是计算机的基本运算，采用二进制可以简单方便地进行这两类运算。

1.1.2 数制转换

1. 十进制与*R*进制的相互转换

(1) *R*进制数（*R*为进位计数制的基数，可为二、八或十六）转换为十进制数。转换规则是：

将各位数字与位权相乘求和，所得和数即为转换结果。例如：

$$(10110.1)_2 = 2^4+2^2+2^1+2^{-1}$$
$$= (22.5)_{10}$$
$$(456.45)_8 = 4\times8^2+5\times8^1+6\times8^0+4\times8^{-1}+5\times8^{-2}$$
$$=(302.578125)_{10}$$
$$(2AF)_{16} = 2\times16^2+A\times16^1+F\times16^0$$
$$= 2\times16^2+10\times16+15\times1$$
$$=(687)_{10}$$

(2) 十进制数转换为 R 进制数

- 十进制整数转换为 R 进制整数。转换规则是：“除基数取余”，即用十进制数反复地除以基数，记下每次得到的余数，直至商为 0。将所得余数按最后一个余数到第一个余数的顺序依次排列起来即为转换结果。例如，将 83 转换成二进制数，逐次除 2 取余：

```
2 | 83 ---- 1
  2 | 41 ---- 1
    2 | 20 ---- 0
      2 | 10 ---- 0
        2 | 5 ---- 1
          2 | 2 ---- 0
            2 | 1 ---- 1
```

可得$(83)_{10}=(1010011)_2$。

- 十进制小数转换成 R 进制小数。转换规则是：“乘基数取整”，即用十进制小数乘以基数，得到一个乘积，将乘积的整数部分取出来，将乘积的小数部分再乘以基数。重复以上过程，直至乘积的小数部分为 0 或满足转换精度要求为止。最后将每次取得的整数按第一个整数到最后一个整数的顺序依次排列起来即为转换结果。例如，将 0.8125 转换为二进制小数，逐次乘 2 取整：

```
  0 .8125
×     2
-------
  1 .625
×     2
-------
  1 .25
×     2
-------
  0 .5
×     2
-------
  1 .0
```

可得 $(0.8125)_{10}=(0.1101)_2$。

在本例中，能够精确转换，没有丝毫误差。特别要注意的是，由于计算机的精度有限并非所有的十进制小数都能完全准确地转换成对应的二进制小数。此时可以在满足所要求的精度的条件下用 0 舍 1 入的方法进行处理（类似于十进制中的四舍五入的方法）。显然，在转换过程中，做的乘法次数越多，结果就越精确。例如，将 0.335 转换为二进制小数，精确到 0.001。

$$
\begin{array}{r}
0 \quad .335 \\
\times \quad\quad 2 \\
\hline
0 \quad .67 \\
\times \quad\quad 2 \\
\hline
1 \quad .34 \\
\times \quad\quad 2 \\
\hline
0 \quad .68 \\
\times \quad\quad 2 \\
\hline
1 \quad .536
\end{array}
$$

可得 $(0.335)_{10} = (0.0101\cdots)_2 \approx (0.011)_2$。

- 一个十进制数既有整数部分，又包括小数部分，要将其转换成 R 进制数的转换规则是：将该十进制数的整数部分和小数部分分别进行转换，然后将两个转换结果拼接起来。例如，将$(124.625)_{10}$转换成二进制数，因为$(124)_{10} = (1111100)_2$，$(0.625)_{10} = (0.101)_2$，所以$(124.625)_{10} = (1111100.101)_2$。

以上介绍了十进制数与 R 进制数（在此主要是指二进制、八进制及十六进制数）的相互转换方法。为便于记，可简单归纳为：R 至十，位权展开求和；十至 R 用连除连乘法，并特别注意转换结果的排列规则（除基数取余是“先余为低，后余为高”；乘基数取整是“先整为高，后整为低”）。

2. 二进制数与八进制、十六进制数之间的转换

由于二进制数与八进制、十六进制数的特殊关系（8 和 16 都是 2 的整数次幂，即 $8 = 2^3$，$16 = 2^4$），所以由二进制转换成八进制、十六进制，或者做反方向的转换，都非常简单。每一位八进制数可用 3 位二进制数表示，每位十六进制数可用 4 位二进制数表示，这是以上三种数制相互转换的要点。

(1) 二进制数转换为八进制数

二进制数转换成八进制数的转换规则是：“三位并一位”，即以小数点为基准，整数部分从右至左，每三位一组，最左一组不足三位时补 0，小数部分从左至右，每三位一组，最右一组不足三位时补 0，然后，每组改成等值的一位八进制数。例如，将$(10010001.0011)_2$转换成八进制数。首先分组：

10 010 001.001 1

小数点的左边，有一组“10”不足三位，应该补一位 0，即应补为“010”；小数点的右边，有一组“1”不足三位，应该补两位 0，即应补为“100”。补 0 后的分组情况为：

010 010 001.001 100

即得

$$(10010001.0011)_2 = (221.14)_8$$

(2) 八进制数转换为二进制数

八进制数转换成二进制数的转换规则是："一位拆三位"，即把每一位八进制数写成等值的三位二进制数，然后按权连接。例如，将$(576.35)_8$转换成二进制数。将八进制数的每位数依次用三位二进制数代替，即得

$$(576.35)_8 = (101111110.011101)_2$$

(3) 二进制数转换为十六进制数

二进制数转换成十六进制数的转换规则是："四位并一位"，即以小数点为基准，整数部分从右至左，每四位一组，最左一组不足四位时补 0，小数部分从左至右，每四位一组，最右一组不足四位时补 0，然后，每组改成等值的一位十六进制数。例如，将$(10110001.0011)_2$转换成十六进制数。首先以小数点为中心，分别向左右两个方向每四位划分成一组：

1011 0001.0011

然后，每四位用一个相应十六进制数码代替，即得

$$(10110001.0011)_2 = (B1.3)_{16}$$

(4) 十六进制数转换为二进制数

十六进制数转换成二进制数的转换规则是："一位拆四位"，即把每一位十六进制数写成等值的四位二进制数，然后按权连接。例如，将$(576.35)_{16}$转换成二进制数。将十六进制数的每位数依次用四位二进制数代替，即得

$$(576.35)_{16} = (010101110110.00110101)_2$$

八进制数与十六进制数之间的转换可以通过多种途径，请学习者自己思考。

1.1.3　码制

在计算机内部表示数，要考虑数的长度、符号以及小数点的表示等问题。由于二进制数的每一位数字（0 或 1）是用电子器件的两种稳定状态来表示的，因此，二进制位（bit）是最小信息单位，一个数的长度按二进制位数来计算。计算机内部最常用的信息单位是字节（byte）（1 字节 = 8 位），字节也是计算机存储容量的单位，一个数的长度可用字节数来表示。

1. 机器数

把数本身（指数值部分）及符号一起数字化了的数称为机器数，机器数是二进制数在计算机内部的表示形式。

机器数具有以下几个特点。

(1) 有固定的位数。在一台计算机中，所存储和运算的二进制数的长度（最大位数）是固定的（这由所用双稳态器件的数目来确定），但不同的计算机存储和运算的二进制数的长度可能是不同的。

(2) 把其真值的符号数字化。一个数在使用时是有符号的，而计算机对正负号不能识别，因此，数的符号在机器内部要做变换，用专门的符号位表示，符号位放在最高数值位的前面，用“0”表示正，用“1”表示负。

(3) 依靠格式上的约定表示小数点的位置。

2. 机器数的不同表示方法

在计算机中，一个数可以采用原码、补码或反码表示。一个正数的原码、补码和反码是相同的，而负数则不同。

(1) 原码。原码表示的规则是：最高位（最左边一位）表示数的符号（“0”表示正号，“1”表示负号），其余各位表示数的大小。数 0 的原码不唯一，有“正零”和“负零”之分。

例如，假设机器数的位数是 8，则

$$[+73]_{原}=01001001 \qquad [-73]_{原}=11001001$$

$$[+127]_{原}=01111111 \qquad [-127]_{原}=11111111$$

(2) 补码。补码的作用是能把减法运算化成加法运算，从而减少计算机中的设备和计算机的运算时间。在介绍补码之前，先讲解一下模的概念。“模”是指一个计量系统的计数范围，如一天 24 小时、一周 7 天、一年 12 个月等，而计算机中的一个存储单元也可认为是一个有计量范围的计量工具，n 位计算机的模为 2^n，计量范围为 0~2^n-1。任何有模的计量器，均可化减法运算为加法运算。

对于 n 位计算机来说，数 X 的补码定义为：

$$[X]_{补}=\begin{cases}X & (0\leqslant X<2^n-1)\\ 2^n+X & (-2^n-1\leqslant X<0)\end{cases}$$

正数的补码是其本身，负数的补码是真值与模数相加。数 0 的补码表示是唯一的。

$$[0]_{补}=[+0]_{补}=[-0]_{补}=00000000$$

求负数的补码的简便方法是：符号位取 1，其余各位按其真值取反，然后在它的末位加 1。

将一个补码值转换成真值的方法是：若符号位为 0，则符号位后的二进制数就是真值，且为正；若符号位为 1，则将符号位后的二进制数逐位取反，再在末位加 1，所得结果为真值，且为负。

(3) 反码。对于 n 位计算机来说，数 X 的反码定义为：

$$[X]_{反}=\begin{cases}X & (0\leqslant X<2^n-1)\\ 2^n+X-1 & (-2^n-1\leqslant X<0)\end{cases}$$

正数的反码是其本身，负数的补码保持其原码的符号位不变，其他各位取反。

(4) 移码。对于 n 位计算机来说，数 X 的移码定义为：

$$[X]_{移}=2^{n-1}+X \qquad (-2^{n-1}\leqslant X<2^{n-1})$$

无论 X 为正数还是负数，其移码均加 2^{n-1}。

移码在计算机中主要用于表示浮点数中的阶。

1.1.4　定点数与浮点数

根据小数点的位置是否固定，机器数又分为定点数和浮点数两种表示方法。小数点是隐含的（即小数点本身不占一个二进制位）。

1. 定点数

定点数是将小数点固定在数中某个约定的位置，通常有以下两种约定。

(1) 定点整数：小数点位置固定在最低数值位的后面，用来表示纯整数。例如，$(120)_{10}=(1111000)_2$，若计算机使用的定点数长度为 1 字节（8 位，最高位为符号位），则该数的机内表示为 01111000。

(2) 定点小数：小数点固定在符号位与最高数值位之间，用来表示纯小数。例如，$(-0.625)_{10}=(-0.101)_2$，定点数长度仍为 8 位，则该数的机内表示为 11010000。

定点数的运算规则比较简单，但不适宜对数值范围变化比较大的数据进行运算。

2. 浮点数

如果机器数采用浮点表示，则小数点的位置是不固定的，可以浮动。为理解小数点浮动的概念，下面用数的指数表示形式为例来说明。设有十进制数 54.768，该数可以等价地表示为以下各种形式：54.768×10^0，5476.8×10^{-2}，0.54768×10^2，等等。不难看出，在以上各种表示中，小数点可以向右或向左浮动（表示数扩大或缩小若干倍），只要指数做相应的增减，数值保持不变。任何一个二进制数 N 都可以表示为下面的指数形式：

$$N=2^i\times S$$

i 称为 N 的阶码，为整数，可正可负，决定小数点的位置；S 称为 N 的尾数。

尾数决定了数的精度（有效位数），阶码决定了小数点在尾数中的位置，从而决定了数的取值范围。浮点数运算方式比较复杂，但表示数的范围大。对于绝对值很大或很小的数，用浮点数表示非常方便。各种高级语言中的实型数据在机器内部一般都是用浮点数表示的。

浮点数在计算机中的表示形式为：

阶码符号（if）	阶码（i）	数符（St）	尾数（S）

假设有 16 位虚拟机，阶码用 5 位表示，尾数用 9 位表示，阶码符号和数符各占 1 位，写出 $(34.625)_{10}$ 的浮点数表示。

先将 $(34.625)_{10}$ 表示为指数形式，即 $N=(34.625)_{10}=(100010.101)_2=2^6\times(0.100010101)_2$。用 16 位二进制数码来表示这个浮点数：

0	00110	0	100010101

将浮点数的尾数表示为纯小数的形式并使尾数中小数点后第1位数字为1称为浮点数的规格化。用规格化的浮点数进行运算，可提高运算精度。

1.1.5 信息编码

计算机只能识别1和0，因此在计算机内表示的数字、字母、符号等都要以二进制数码的组合来代表，这就是二进制编码。编码是采用少量的基本符号，选用一定的组合原则，以表示大量复杂多样的信息的技术。根据不同的用途，有各种各样的编码方案，较常用的有ASCII码、BCD码、汉字编码等。

1. ASCII码

ASCII码是美国国家标准信息交换码（American National Standard Code for Information Interchange）的简称，是目前国际上使用最广泛的字符编码。ASCII码的编码规则为：每个字符用7位二进制数（$d_6d_5d_4d_3d_2d_1d_0$）来表示，7位二进制共有128种状态（2^7=128），可表示128个字符，7位编码的取值范围为0000000～1111111。在计算机内部，每个字符的ASCII码用1个字节（8位）来存放。字节的最高位（d_7）为校验位，通常用“0”来填充，后7位（$d_6d_5d_4d_3d_2d_1d_0$）为编码值。7位编码的ASCII码字符集包括128个字符，称为标准ASCII码字符集。详细的ASCII码表参见附录A。

2. 二-十进制编码（BCD码）

由于人们日常使用的是十进制，而机器内部使用的是二进制，所以，需要把十制数表示成二进制码。一位十进制数字用4位二进制码来表示可以有多种方法，但常用的是BCD码。4位二进制数表示2^4（即16）种状态，只取前10种状态来表示0～9。从左到右每位二进制数的权分别为8，4，2，1，因此又叫8421码。

BCD码有十个不同的码，即0000，0001，0010，0011，0100，0101，0110，0111，1000，1001，且它是逢“十”进位的，所以是十进制数。但它的每位是用二进制编码来表示的，因此称为二进制编码的十进制（Binary Coded Decimal）。

BCD码十分直观，可以很容易实现与十进制的转换。例如，可以方便地确认

$$(0010\ 1001\ 0101\ 1000.0110\ 0011)_{BCD}$$

代表的十进制数是2958.63。

在每个BCD码上加上0011（即3）称为余3码。

把BCD码重新排序，使相邻两个BCD码只有一位不同，称为格雷码。

3. 汉字编码

随着我国国际地位的不断提高，以及计算机应用在我国和其他地区的日益普及，汉语在国际事务和全球信息交流中的作用越来越大，对汉字的计算机处理已成为当今文字信息处理中的重要内容。要在计算机中处理汉字，必须解决以下几个问题：首先是汉字的输入，即如何把结构复杂的方块汉字输入到计算机中去，这是汉字处理的关键；其次是汉字的表示和存储，即汉字在计算机内部如何表示和存储？如何与西文兼容？最后是汉字的输出，即如何将汉字的处理结果从计算机中输出？为此，必须将汉字代码化，即对汉字进行编码。对应于汉字处理过程中的输入、内部