普通高等学校计算机基础教材

C语言程序设计基础

李 民 钟 钰 秦珀石 主编

武汉理工大学出版社
Wuhan University of Technology Press
WUTP

普通高等学校计算机基础教材

C语言程序设计基础

主　编　李　民　钟　钰　秦珀石
副主编　陈志铭　李　宁　毛　薇
顾治华　袁小玲　汤练兵

武汉理工大学出版社
·武　汉·

内 容 简 介

C语言是目前仍然广泛使用的计算机程序设计语言。它适用范围广、语法简洁、执行效率高，是典型的结构化程序设计语言，也是学习C++面向对象程序设计语言的基础。

本书针对初学者，通过案例和问题引入C语言的语法知识，重点讲解程序设计的基本思想和方法，旨在培养学生的计算思维能力和用计算机解决问题的能力。本书共有9章，涵盖C语言的数据类型、基本控制结构、数组、指针、函数、结构体与共用体、数据文件等内容。

本书可作为高等院校本科学生学习“C语言程序设计”课程的参考用书，也可作为C语言程序设计爱好者的自学用书。

图书在版编目(CIP)数据

C语言程序设计基础/李民，钟钰，秦珀石主编．—武汉：武汉理工大学出版社，2017.9
(2018.8重印)

ISBN 978-7-5629-5626-6

Ⅰ．①C… Ⅱ．①李… ②钟… ③秦… Ⅲ．①C语言-程序设计 Ⅳ．①TP312.9

中国版本图书馆CIP数据核字(2017)第214086号

项目负责人：吴正刚 王兆国　　**责任编辑**：李兰英
责任校对：夏冬琴　　**封面设计**：许伶俐
出版发行：武汉理工大学出版社
地　　址：武汉市洪山区珞狮路122号
邮　　编：430070
网　　址：http://www.wutp.com.cn
经　　销：各地新华书店
印　　刷：荆州市鸿盛印务有限公司
开　　本：787mm×1092mm 1/16
印　　张：17
字　　数：435千字
版　　次：2017年9月第1版
印　　次：2018年8月第2次印刷
印　　数：8001—15000册
定　　价：36.00元

凡购本书，如有缺页、倒页、脱页等印装质量问题，请向出版社发行部调换。

本社购书热线：027－87523148 87664138 87785758 87165708(传真)

前　言

程序设计是普通本科院校学生的计算机基础课程，课程要求学生不仅掌握高级程序设计语言，更重要的是掌握程序设计的基本思想和方法，旨在培养学生的计算思维能力和问题求解能力，为学生在后续课程学习和工作中应用计算机解决实际问题打下基础。C 语言功能丰富、使用灵活方便、程序执行效率高、可移植性好，是目前广泛使用的程序设计语言。它不仅是计算机专业学生必修的专业基础课，也是许多非计算机专业学生必修的基础课程，全国计算机等级考试、信息技术类竞赛等都将 C 语言列入考试范围，学习 C 语言已经成为广大青年学生的广泛需要。

C 语言涉及的概念和规则繁多，使用灵活但容易出错，如何在有限的课时内，有效地组织教学，成为授课教师面临的问题；如何在有限的课时内，有效地学好 C 语言，也是学生们面临的问题。因此，探讨更有效率的教学方法，编撰更符合教学目的的教学参考书，就成为急需解决的问题。本书就是为了满足这个层次的读者需求而编写的。本书编者都具有长期的 C 语言教学实践经验，经过多年的教学内容、教学方法、考核方式的改革实践，形成了以程序设计为主线、以案例应用为驱动、通过问题引入重点介绍程序设计的基本思想和方法。

全书共 9 章，主要包括以下内容：第 1 章是概述，主要介绍数据在计算机中的表示以及计算机程序设计语言。为配合本科教学的培养方案改革，在本章增加了计算机基础知识部分的二进制以及数值数据编码部分内容，作为课程中有关数据知识的理论基础。第 2 章～第 4 章讲解程序结构的三种基本结构，在 C 语言基本数据类型及其运算、基本输入输出操作的基础上，介绍顺序结构、选择结构和循环结构的程序设计方法，通过案例学习简单算法的设计和程序编写，掌握用程序解决问题的基本方法。第 5 章和第 6 章学习基本构造类型(数组)和指针的用法。第 7 章～第 9 章通过学习函数掌握模块化编程方法，以及结构体类型、文件在编程中的使用，全面掌握用 C 语言解决复杂问题的编程方法，进一步培养分析问题和解决问题的能力。附录则提供 ASCII 码表、运算符、常用函数表等，以便在学习过程中随时查阅。书中的每个例题均在 CodeBlocks 下调试通过，并给出了运行结果，引导读者从模仿到改写，再到独立编写程序，加深对 C 语言编程方法的了解。

与本书配套的《C 语言程序设计实验指导书》精心设计了相应的实验案例，循序渐进地引导学生熟悉实验环境，学会编写简单程序，掌握程序调试的基本方法，从而掌握程序设计的基本思想和方法，同时还提供与教材配套的习题，以帮助学生巩固所学知识。

本书由李民、钟钰、秦珀石担任主编，由陈志铭、李宁、毛薇、顾治华、袁小玲、汤练兵担任副主编。第 1 章由吕曦编写，第 2 章由李宁编写，第 3 章由毛薇编写，第 4 章由钟钰编写，第 5 章由李民、孙骏编写，第 6 章由顾治华编写，第 7 章由袁小玲编写，第 8 章由秦珀石编写，第 9 章由汤练兵编写，附录部分由陈志铭整理编写。全书由李民、陈志铭、钟钰、吕曦、汤练

兵、段翠苹、毛薇、秦珀石分章校订，李民、钟钰、秦珀石、陈志铭统稿校订。在本书的编写过程中，郑敬、郭羽成、段翠苹、吴利军、吕淑琴、李岫、张伟、李捷、凌咏红、周彩兰、鄢红国、魏敏、何九周、黄靖等老师提出了不少建议，在此深表谢意。

由于编者水平有限，书中难免存在不妥与疏漏之处，敬请广大读者批评指正。

编　者

2017 年 5 月

目　录

1 计算机基础与C语言简介

1.1 冯·诺依曼式计算机与二进制

1.1.1 冯·诺依曼式计算机

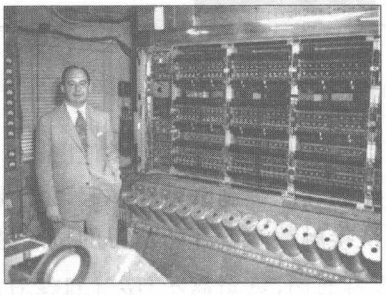

图 1-1 冯·诺依曼与 ENIAC 合影

1945 年 3 月，冯·诺依曼(图 1-1)在共同讨论的基础上起草了“存储程序通用电子计算机方案”——EDVAC (electronic discrete variable automatic computer)。这对后来计算机的设计有决定性的影响，从第一台电子计算机 ENIAC(ENIAC 并不是冯·诺依曼体系)之后到当前最先进的计算机依然采用的是冯·诺依曼体系结构。

冯·诺依曼的思想可概括为以下三点：

(1) 计算机硬件由运算器、控制器、存储器、输入设备和输出设备五个基本部件组成。

(2) 计算机采用二进制来表示指令和数据。

(3) 计算机采用“存储程序”方式自动逐条取出指令并执行程序。

冯·诺依曼式计算机硬件之间的相互关系如图 1-2 所示。

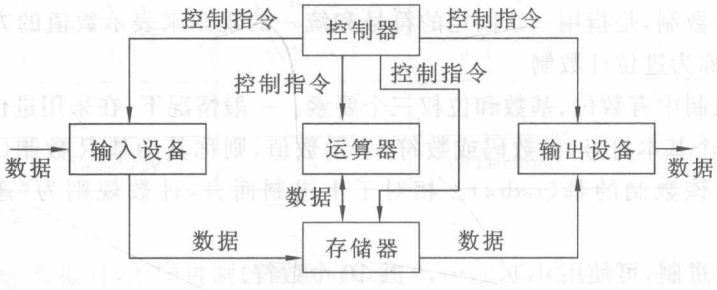

图 1-2 计算机硬件之间的关系图

从图 1-2 中可以看出，冯·诺依曼式计算机的核心部件是存储器，所有的数据都需要经过存储器才能参与运算，在此硬件基础之上，冯·诺依曼提出了计算机如下的运行方式：

(1) 把程序指令逐条取入控制器。

(2) 在控制器输入命令的作用下，把需要的原始数据通过输入设备送入计算机。

(3) 向存储器和运算器发出存数、取数和运算命令，并把计算结果存放在存储器内。

(4) 在控制器发出的取数和输出命令的作用下，通过输出设备输出计算结果。

计算机一经启动，就能按照程序指定的逻辑顺序从存储器中读取指令并逐条执行，自动完成指令规定的操作。

冯·诺依曼(John von Neumann，1903—1957 年)，20 世纪最重要的数学家之一，在现代计算机、博弈论、核武器和生化武器等诸多领域内有杰出建树的最伟大的科学全才之一，被后人称为“计算机之父”和“博弈论之父”，原籍匈牙利，布达佩斯大学数学博士，先后执教于柏林大学和汉堡大学，1930 年前往美国，后入美国籍，历任普林斯顿大学、普林斯顿高级研究所教授，美国原子能委员会会员，美国国家科学院院士。

1.1.2 二进制

由于在物理上实现类似图 1-3 的方波图是一件很容易的事情，于是这种易实现的二态自然就和二进制联系在一起了。要明白二进制就要先了解数制的概念，其实数制是一个我们从接触数学就开始学习的概念，数学中最常见的十进制也是一种数制。

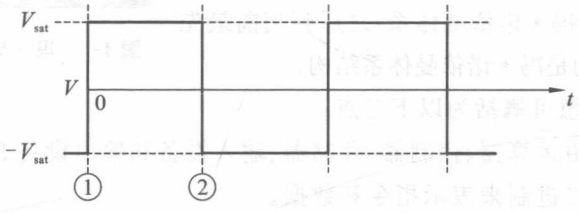

图 1-3 方波图

(1) 数制

数制也称计数制，是指用一组固定的符号和统一的规则来表示数值的方法。按进位的方法进行计数，称为进位计数制。

在进位计数制中有数位、基数和位权三个要素。一般情况下，在采用进位计数的数字系统中，如果用 R 个基本符号(即数码或数符)表示数值，则称其为基 R 数制(radix-r number system)，R 称为该数制的基(radix)。相对于十进制而言，计数规则为“逢 R 进一，借一当 R”。

R=10 为十进制，可使用 0，1，2，…，9 共 10 个数符。

R=2 为二进制，可使用 0，1 共 2 个数符。

R=8 为八进制，可使用 0，1，2，…，7 共 8 个数符。

R=16 为十六进制，可使用 0，1，2，…，9，A，B，C，D，E，F 共 16 个数符。

数位是指数码在一个数中所处的位置(记为 n，n 为整数)。

位权是指在某种进位计数制中，每个数位上的数码所代表的数值的大小，等于在这个数位上的数码乘上一个固定的数值，这个固定的数值就是这种进位计数制中该数位上的位权。

如数101.1，在不同进位计数制中所表示的数的大小不同，见图1-4。

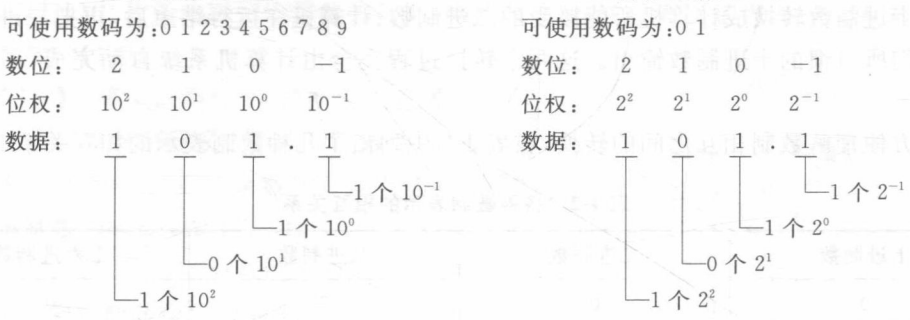

图1-4　不同进位制中数的表示

在R(R≥2，R为整数)进位计数制中，任意数N(不考虑其正负)的通用表达式为：

$$N=A_nA_{n-1}\cdots A_1A_0.A_{-1}A_{-2}\cdots A_{-m}$$

或

$$N=A_n\times R^n+A_{n-1}\times R^{n-1}+\cdots+A_0\times R^0+A_{-1}\times R^{-1}+\cdots+A_{-m}\times R^{-m}$$

式中，R为基数；A_i($i\in[-m,n]$)为该进位制使用的某个数码，$0\leqslant A_i\leqslant R-1$；n+1为小数点左边数码个数；m为小数点右边数码个数。

在计算机中，只能是二进制，但是二进制表示一个数比较冗长，且不易读写，因此在表示计算机信息时，常用八进制和十六进制。表1-1所示的是计算机信息表示中常用的几种进位数制。

表1-1　计算机中常用的几种进制数的表示

数制	二进制	八进制	十进制	十六进制
规则	逢二进一	逢八进一	逢十进一	逢十六进一
基数	R=2	R=8	R=10	R=16
数符	0,1	0,1,…,7	0,1,…,9	0,1,…,9,A,B,C,D,E,F
位权	2^i	8^i	10^i	16^i
形式表示	B(Binary)	O(Octal)	D(Decimal)	H(Hexadecimal)

注：其中i为整数。

在具体表示数据时，不同进制的数据可以用不同的数字下标或后缀字母表示。如：

① 一个八进制数573.4可记作$(573.4)_8$或573.4O；

② 一个二进制数110.01可记作$(110.01)_2$或110.01B；

③ 一个十六进制数8A6.D可记作$(8A6.D)_{16}$或8A6.DH。

④ 一个十进制数962.5可记作$(962.5)_{10}$或962.5D，或直接记作962.5。

(2) 数制的相互转换

将数由一种数制转换成另一种数制称为数制间的转换。由于计算机采用二进制，但用

计算机解决实际问题时对数值的输入输出通常使用十进制，这就有一个十进制向二进制转换或由二进制向十进制转换的过程。也就是说，在使用计算机进行数据处理时首先必须把输入的十进制数转换成计算机所能接受的二进制数；计算机在运行结束后，再把二进制数转换为人们所习惯的十进制数输出。这两个转换过程完全由计算机系统自动完成，不需人们参与。

为方便理解数制相互之间的转换，在表 1-2 中列出了几种数制表示的相互关系。

表 1-2　各种数制表示的相互关系

十进制数	二进制数	八进制数	十六进制数
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

① 十进制数转换成 R 进制数

从十进制数转换成其他进制数，需要把整数部分和小数部分分别进行处理。首先我们讨论十进制数转换成二进制数的方法。

● 十进制整数转换成 R 进制整数

设十进制整数 $N_{(10)}=a_n\cdots a_1a_{0(R)}$，由 R 进制整数按位展开公式的形式

$$\begin{aligned}N_{(10)}=a_n\cdots a_1a_{0(R)}&=(a_nR^n+\cdots+a_1R+a_0)_{(R)}\\&=((\cdots((0+a_n)R+\cdots+a_2)R+a_1)R+a_0)_{(R)}\end{aligned}\tag{1-1}$$

将式(1-1)除以 R，商为整数$(\cdots((0+a_n)R+a_{n-1})R+\cdots+a_2)R+a_1$，余数为 a_0；所得商再除以 R，商为整数$(\cdots((0+a_n)R+a_{n-1})R+\cdots a_3)R+a_2$，余数为 a_1；以此类推，直至商为 0，余数为 a_n。

因而，将一个十进制整数转换为R进制数的转换规则为：除以R取余数，直到商为0时结束。所得余数序列，先得到的余数为低位，后得到的余数为高位。

将式(1-1)中R^i按十进制运算法则计算，则式(1-1)的逆过程就是将R进制整数转换为十进制整数的过程。

【例1-1】 将十进制数37转换成二进制数。

在转换过程中用基数2连续除整数商，并取余数。直到商为0时停止，然后将余数从下往上按顺序书写，得到二进制数100101(图1-5)，即：

$$(37)_{10}=(100101)_2$$

图1-5　十进制整数转换成二进制数

● 十进制小数转换成R进制小数

设十进制纯小数$M_{(10)}=0.a_{-1}\cdots a_{-m(R)}$，由R进制小数按位展开公式的形式

$$M_{(10)}=0.a_{-1}\cdots a_{-m(R)}=(a_{-1}R^{-1}+a_{-2}R^{-2}+\cdots+a_{-m}R^{-m})_{(R)} \tag{1-2}$$
$$=((a_{-1}+(a_{-2}+\cdots+(a_{-m})\cdot 1/R\cdots)\cdot 1/R)\cdot 1/R)_{(R)}$$

将式(1-2)乘以R，得整数部分为a_{-1}，小数部分为$(a_{-2}+\cdots+(a_{-m})\cdot 1/R\cdots)\cdot 1/R)$；小数部分再乘以R，得整数部分为$a_{-2}$，小数部分为$(\cdots+(a_{-m})\cdot 1/R\cdots)$；以此类推，直至小数部分为0或转换到所要求的精度为止。

因而，将一个十进制小数转换为R进制数的转换规则为：乘以R取整数，直到余数为0时或达到精度时结束。所得整数序列，先得到的整数为高位，后得到的整数为低位。

将式(1-2)中R^i按十进制运算法则计算，则式(1-2)的逆过程就是将R进制纯小数转换为十进制纯小数的过程。

【例1-2】 把十进制数0.6875及0.78转换成二进制小数。

把0.6875乘以2所得乘积1.3750写在0.6875的下面。接着，对上述乘积的小数部分0.375继续乘以2，直到乘积中的小数部分为0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，如图1-6(a)所示。于是得：

$$0.6875=(0.1011)_2$$

把0.78乘以2所得乘积1.56写在0.78的下面。接着，对上述乘积的小数部分0.56继续乘以2，直到达到所要求的精度(假设精确到小数点后6位)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，如图1-6(b)所示。于是得：

$$0.78=(0.110001)_2$$

由此可见，十进制数转换成二进制数时，任何整数可以精确地转换为对应的二进制数。但小数有可能不能精确地转换为对应的二进制数。不能精确转换的小数是密集的，能精确转换的小数是稀疏的。这是产生计算机运算误差的主要原因之一。

综合整数和小数的转换方法，可以将任意十进制数转换成R进制数。

② R进制数转换成十进制数

二进制、八进制、十六进制数都可转换成等量的十进制数。将该数按其基数的指数形式写出多项式，然后用十进制运算法则计算就可以很容易完成这一转换。

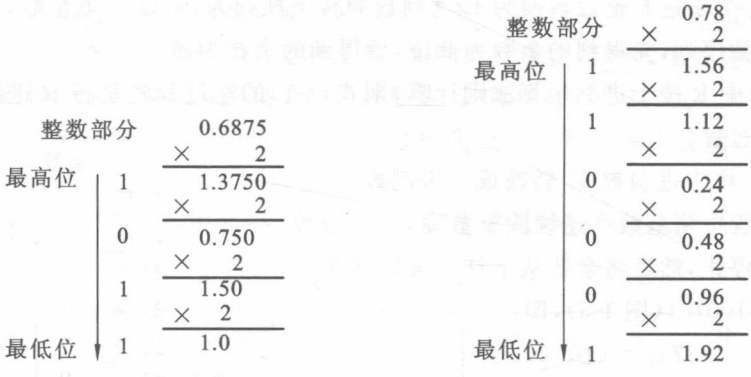

图 1-6　十进制纯小数转换成二进制小数

(a)0.6875 的转换;(b)0.78 的转换

例如:

$$(1101.1)_2=(1\times2^3+1\times2^2+0\times2^1+1\times2^0+1\times2^{-1})_{10}$$
$$=(13.5)_{10}$$
$$(50.6)_8=(5\times8^1+0\times8^0+6\times8^{-1})_{10}$$
$$=(40.75)_{10}$$
$$(4B.E1)_{16}=(4\times16^1+11\times16^0+14\times16^{-1}+1\times16^{-2})_{10}$$
$$=(75.87890625)_{10}$$

③ 八进制数、十六进制数与二进制数之间的转换

● 八进制数与二进制数之间的转换

由于 $2^3=8$,$2^4=16$,所以一位八进制数相当于三位二进制数,一位十六进制数相当于四位二进制数,这样使得八进制数、十六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,只要将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 1-3】 将$(1234.567)_8$转换成二进制数。

1　2　3　4　.　5　6　7

↓　↓　↓　↓　↓　↓　↓　↓

001　010　011　100　.　101　110　111

即得:$(1234.567)_8=(1010011100.101110111)_2$

将二进制数转换成八进制数时,以小数点为界,整数部分从右往左,每三位一组,最左边不足三位时,左边添 0 补足至三位;小数部分从左往右,每三位一组,最右边不足三位时,右边添 0 补足至三位;然后将每组的三位二进制数用相应的八进制数表示出来。即“三位变一位”。

【例 1-4】 将$(1011010101.1111)_2$转换成八进制数。

001　011　010　101　.　111　100

↓　↓　↓　↓　↓　↓　↓

1　3　2　5　.　7　4

即得：$(1011010101.1111)_2=(1325.74)_8$

类似八进制和二进制之间的转换方法，用“一位变四位”可将十六进制数转换成二进制数，用“四位变一位”可将二进制数转换成十六进制数。

【例 1-5】 将$(29C.1A)_{16}$转换成二进制数。

```
  2     9     C     .     1     A
  ↓     ↓     ↓     ↓     ↓     ↓
0010  1001  1100  .  0001  1010
```

即得：$(29C.1A)_{16}=(1010011100.0001101)_2$

【例 1-6】 将$(1011010101.1111)_2$转换成十六进制数。

```
0010  1101  0101  .  1111
  ↓     ↓     ↓     ↓     ↓
  2     D     5     .     F
```

即得：$(1011010101.1111)_2=(2D5.F)_{16}$

(3) 数值数据的编码

我们经常遇到的数是有正负之分的，一般用“+”或“-”符号简单地附加在数上，以体现数的正负区别。而在计算机中，数的符号与一般的符号表示法不同，并且在某些情况下同数值位一道参加运算操作。为了妥善地处理好这些问题，就产生了把符号位和数字位一起编码来表示相应的数的各种表示方法，如原码、反码、补码、移码等。下面我们将研究符号数在计算机中的表示法。为了简便起见，我们将以整数为例，并设机器字长为8位。

① 真值和机器码

在计算机中，数值数据的符号也被“数字化”了。符号数在计算机中的一种简单表示方法就是：用最高位存放符号，正号用“0”表示，负号用“1”表示。例如：

+53的二进制值为：+110101　　机器数表示为：00110101

-53的二进制值为：-110101　　机器数表示为：10110101

为了区别原来的数与它在计算机中的表示形式，通常，称表示一个数值数据的机内编码为机器数，而它所代表的实际值称为机器数的真值。

② 原码

在上面提到的符号数的表示方法，即正数的符号位为0，负数的符号位为1，其他位用二进制数表示数的绝对值，这是一种最简单的表示方法，即为原码表示法。如：

+102的二进制真值为：+1100110　　原码为：01100110

-102的二进制真值为：-1100110　　原码为：11100110

可见两个符号相异，绝对值相同的数的原码，除了符号位以外，其他位都是一样的。

原码简单易懂，而且与真值的转换方便。但若是两个异号数相加(或两个同号数相减)，就要做减法。做减法就会有借位的问题，很不方便。为了简化运算逻辑电路，加快运算速度，将加法运算与减法运算统一起来，就引进了反码和补码。

③ 反码

正数的反码与其原码相同，负数的反码为其原码除符号位外的各位按位取反(即0改为1，1改为0)。例如：

+79 的二进制真值为：+1001111　　原码为：01001111　　反码为：01001111

−79 的二进制真值为：−1001111　　原码为：11001111　　反码为：10110000

可以看出，负数的反码与负数的原码有很大的区别。反码通常只用作求补码过程中的中间形式。可以验证，一个数的反码的反码就是其原码。

④ 补码

正数的补码与其原码相同，负数的补码为其反码在最低位加 1。例如：

+23 的原码为：00010111　　反码为：00010111　　补码为：00010111

−23 的原码为：10010111　　反码为：11101000　　补码为：11101001

同样可以验证，一个数的补码的补码就是其原码。

引入补码后，加减法运算都可以统一用加法运算来实现，符号位也被当作数值参与运算，且两数和的补码等于两数补码的和。因此，在许多计算机系统中都采用补码来表示带符号的数。例如：

$$102-79=102+(-79)=23$$

用原码相减：01100110−01001111＝00010111

用补码相加：01100110＋10110001＝00010111

由于一个字节只有 8 位，所以高位自然丢失，可见用原码相减和用补码相加所得的结果是相同的，都是 23 的补码 00010111。

当然，在不同的计算机中还有其他形式的编码。

1.2　机器语言、汇编语言和高级计算机语言

1.2.1　机器语言

机器语言是用二进制代码表示的计算机能直接识别和执行的一种机器指令的集合，它是计算机的设计者通过计算机的硬件结构赋予计算机的操作功能。机器语言是计算机唯一能够直接识别和运行的语言，不同型号的计算机其机器语言是不相通的，按照一种计算机的机器指令编制的程序，不能在另一种计算机上执行。

一条指令就是机器语言的一个语句，它是一组有意义的二进制代码，指令的基本格式包括操作码字段和地址码字段，其中操作码指明了指令的操作性质及功能，地址码则给出了操作数或操作数的地址。例如下面是某种 CPU 的几条机器指令：

机器指令	含义
0000000000000010000	LOAD A,16
0000001000000000001	LOAD B,1

可以看出，机器指令对于普通人来说是非常难以理解的，如果没有相关 CPU 的技术说明书，甚至计算机专业从业者也很难知道其含义，因此机器语言是无法作为广泛使用的计算机程序设计语言的。

1.2.2　汇编语言

汇编语言(assembly language)是一种用于电子计算机、微处理器、微控制器或其他可编程器件的低级语言，亦称为符号语言。在汇编语言中，用助记符(mnemonics)代替机器指令的操作码，用地址符号(symbol)或标号(label)代替指令或操作数的地址。在不同的设备中，汇编语言对应着不同的机器语言指令集，通过汇编过程转换成机器指令。一般来说，特定的汇编语言和特定的机器语言指令集是一一对应的，不同平台之间不可直接移植。

例如下面是某汇编语言的部分指令代码：

汇编指令	含义
LOAD A,16	将16存入寄存器A中
LOAD B,1	将1存入寄存器B中
ADD A,B	将A+ B的结果存入A中

汇编指令所用的助记符(LOAD、ADD)等，已经是普通人可以理解的英文单词了，这比机器语言向前进了一大步，但是由于不同的CPU会有不同的汇编指令集，所以在编写程序时仍然需要相应CPU的技术资料才能进行，这对于非计算机专业的使用者来说仍然要求太高，所以汇编语言也不是广泛使用的编程语言。

1.2.3　高级计算机语言

高级计算机语言是以人类的日常语言为基础的一种编程语言，它使用一般人易于接受的文字来表示，从而使程序员编写更容易，亦使程序有较高的可读性，非计算机专业的人员也可以通过一段时间的学习，掌握编写程序的基本方法。由于基本脱离了机器的硬件系统，使用高级计算机语言编写的程序可以不做改动，或做少量改动就可以在不同体系结构的计算机上运行。使用高级计算机语言编写的程序称之为源程序。

例如下面是某高级计算机语言所编写的部分代码：

高级计算机语言指令	含义
A= 16	将16存入变量A中
B= 1	将1存入变量B中
A= A+ B	将A+ B的结果存入A中

从上面的例子可以看出，高级计算机语言与人类所使用的语言已经没有什么差别了，这种易用性正是我们希望计算机程序语言所具有的特点。高级程序语言的出现，才真正使计算机编程进入普通人也可以方便学习和使用的范围，也真正促使计算机应用于各行各业。

1.2.4　三种语言的关系

机器语言是计算机唯一能识别的语言，但却是普通人无法理解的；汇编语言是少部分专业人士可以理解，但是多数人不会使用，计算机无法识别；高级计算机语言是多数人可以理解和使用的，但是计算机无法识别，这种关系如图1-7所示：