工业和信息化普通高等教育“十二五”规划教材立项项目
21世纪高等教育计算机规划教材

COMPUTER

C语言程序设计（第3版）

The C Programming Language

安俊秀　主编
于华 董妍汝 闫俊伢　副主编
张永奎　主审

- 以 Visual C++ 6.0 为开发平台
- 选用典型例题，示例由简到难
- 注重立体化建设，提供丰富的教学资源

人民邮电出版社
POSTS & TELECOM PRESS

工业和信息化普通高等教育“十二五”规划教材立项项目
21世纪高等教育计算机规划教材

COMPUTER

C语言程序设计（第3版）

The C Programming Language

安俊秀　主编
于华 董妍汝 闫俊伢　副主编
张永奎　主审

人 民 邮 电 出 版 社
北　京

图书在版编目（CIP）数据

C语言程序设计 / 安俊秀主编. -- 3版. -- 北京 : 人民邮电出版社, 2014.9
21世纪高等教育计算机规划教材
ISBN 978-7-115-36278-0

Ⅰ. ①C… Ⅱ. ①安… Ⅲ. ①C语言－程序设计－高等学校－教材 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2014)第156817号

内 容 提 要

本书严格遵循C语言标准，全面、系统、深入浅出地阐述了C语言的基本概念、语法和语义，以及用C语言进行程序设计的方法和技术。全书共三篇，第一篇为“C语言程序设计基础知识”，第二篇为“C语言高级编程技术”，第三篇为“C语言综合应用与实践”。第一篇强调对基本概念的理解和掌握，主要讲解C语言的语法规则、C语言的基本控制结构、数组、函数等知识；第二篇强调在理解和掌握的基础上运用高级编程技术的方法编写程序的能力，该篇主要介绍了指针、结构体、共用体、文件等相关知识；第三篇重在培养学生综合编程能力及程序编码的规范性，主要介绍了学生成绩管理系统的设计和开发。

本书内容丰富，可读性强，内容的编排尽量符合初学者的要求，在实例的选择上从易到难，并且能够解决一些实际问题。配套的实验指导书《C语言趣味实验》可以强化学生熟练和巩固所学知识。

本书可作为大学本科计算机和相关专业的“C程序设计”教材，也适合作为C语言初学者的入门读物和自学教程。

◆ 主　　编　安俊秀
副 主 编　于　华　董妍汝　闫俊伢
主　　审　张永奎
责任编辑　邹文波
责任印制　彭志环　杨林杰

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路11号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京圣夫亚美印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：21　　2014年9月第3版
字数：553千字　　2014年9月北京第1次印刷

定价：45.00元

读者服务热线：(010)81055256　印装质量热线：(010)81055316
反盗版热线：(010)81055315

第 3 版前言

《C 语言程序设计（第 3 版）》是初学者学习 C 语言的良好教程。本书以循序渐进的方式介绍了 C 语言编程方面的知识，提供了丰富的实例和大量的练习。与《C 语言程序设计（第 2 版）》相比，本书在教学内容和顺序上进行了如下调整。

1. 书中所有内容和程序均以 Visual C++ 6.0 为基础，去掉了原有以 Turbo C 2.0 为调试基础的内容。

2. 调整了部分内容的顺序。如考虑到输入/输出函数的频繁使用，对其的详细介绍从原来的第 5 章调整到第 4 章，将第 13 章位运算调整到了第 3 章基本数据类型、运算符与表达式中，对 C 语言的运算符在第 3 章进行统一讲解，考虑到初学者的适应能力，对位运算的内容进行了简化。

3. 增大了例题量，选用典型例题，例题由简到难。

4. 为了降低初学者的学习难度，第 8 章函数之前的例题改为由一个主函数实现，之后章节的例题以主函数调用用户自定义函数的方式实现。

5. 为了巩固各个章节的知识点，为本书配套了实验指导书。

全书共 14 章，分为三篇。第一篇为“C 语言程序设计基础知识”，主要讲解 C 语言的语法规则、C 语言的基本控制结构、数组、函数等知识。第二篇为“C 语言高级编程技术”，这一部分强调对基本概念的理解和掌握，并在此基础上运用高级编程技术的方法，锻炼、培养学生进行较大规模、比较复杂的应用程序的编程能力。第三篇为“C 语言综合应用与实践”，以实际案例为主线，引入软件工程的思想，介绍软件开发的方法，培养学生分析问题和解决实际问题的能力。

本书由安俊秀任主编，于华、董妍汝、闫俊伢任副主编。成都信息工程学院安俊秀编写第 2 章、第 5 章、第 10 章，山西大学商务学院赵宇兰编写第 1 章、第 13 章，董妍汝编写第 3 章、第 9 章，闫俊伢编写第 6 章、第 12 章，杨丽英编写第 7 章，郭燕萍编写第 8 章，于华编写第 11 章、第 14 章、附录 1、附录 2，于华、董妍汝、赵宇兰编写附录 3，山西医科大学满晰编写第 4 章。

全书由张永奎、安俊秀统稿，张永奎审校。

由于编者水平有限，书中难免存在疏漏之处，敬请读者批评指正。

编　者

2014 年 6 月

目 录

第三篇　C 语言综合应用与实践

第一篇　C 语言程序设计基础知识

第 1 章
C 语言程序设计预备知识

计算机是以硬件系统为基础，能够通过各种系统软件和应用软件对信息进行自动处理的电子装置。这里的“信息”包括数值、文字、符号、语言、图形和图像等，不论哪种信息在计算机内部最终都必须以数字化编码的形式来表示和处理。程序设计语言就是对这些信息进行处理的软件工具。学习 C 语言之前，除了要了解计算机的工作原理之外，更重要的是要了解各种信息在计算机中的表示形式，只有了解了这些底层细节，才会理解计算机系统和计算机语言对数据做了什么，才会知道如何最大限度地利用可用的硬件资源编写出高效的程序代码。

本章重点介绍计算机内各种信息的常用编码方案，这些内容是进行程序设计的基石，是学习 C 语言必须掌握的基础知识。

1.1　计算机系统组成及工作原理简介

一个完整的计算机系统由硬件和软件两大部分组成。硬件是构成计算机系统的物理装置，是看得见摸得着的设备实体。软件是计算机程序及相关的技术文档资料。软件依赖硬件的物质条件，而硬件则需要在软件的支配下才能有效地工作。

1.1.1　硬件系统的组成及其工作原理

计算机问世 70 多年来，尽管计算机硬件系统在性能指标、运算速度、工作方式、应用领域和价格等方面存在差异，但是它们的基本体系结构却是相同的，都是冯·诺依曼机。冯·诺依曼机由微处理器、存储器及 I/O 接口等大规模或超大规模集成电路芯片组成，各部分之间通过“总线”连接在一起，实现信息的交换。图 1-1 给出了计算机硬件系统的基本组成图。

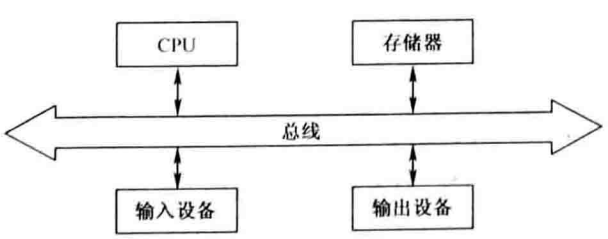

图 1-1　计算机硬件系统的基本组成

1. 微处理器（CPU）

运算器和控制器被集成在同一块微处理器芯片上，统称为微处理器或 CPU 芯片。微处理器是计算机硬件系统的核心，其重要性好比大脑对于人一样。它是计算机的运算和控制中心，负责处理、计算计算机内部的所有数据。

运算器是对二进制数据进行加工和处理的逻辑部件。因为计算机内部是依靠数字电路来存储和计算的，电路的开关状态正对应二进制的 0 和 1。运算器根据器件的物理状态表示和处理二进制数，不仅能够非常容易地实现基本的算术运算和逻辑运算，而且具有高的可靠性。

控制器是计算机的“神经中枢”，是协调指挥计算机各部件和谐工作的元件。它能够综合有关的逻辑条件与时间条件，并按照主频的节拍产生相应的微控制信号，以指挥计算机各部件按照指令功能的要求自动执行指定的操作。

2. 存储器

存储器是计算机的“记忆系统”，是存放程序和数据的逻辑部件。根据作用不同，存储器分为内存储器（简称内存）和外存储器（简称外存）。内存是 CPU 能根据地址直接寻址的存储空间，它用来存放当前正在使用的或者随时要使用的程序或数据。其特点是速度快、容量小，价格较高。外存（如硬盘）用来存放内存的副本和暂时不用的程序或数据。当需要处理外存中的程序或数据时，必须通过输入/输出指令，将其调入内存中才能被 CPU 执行处理。外存的存取速度比内存慢，但容量比内存大得多，且无需电力，并且可以永久保存信息。

3. 输入设备/输出设备

输入设备与输出设备是实现人机交互的主要部件。输入设备用来接收用户输入的原始程序或数据，并将它们转变为计算机能识别的二进制数据存入到内存中，其功能类似于人的“眼”和“耳”——既能看又能听。输出设备用来将计算机处理的结果转变为人们能接受的形式输出，功能类似于人的“手”和“嘴”——既能写又能说。目前常用的输入设备有键盘、鼠标、触摸屏、光笔、画笔、图形板、摄像机、图文扫描仪和图文传真机等，输出设备有显示器、打印机、绘图仪和音箱等。

冯·诺依曼体系结构的设计思想可以简单地概括为“指令存储，顺序执行”。按照冯·诺依曼存储程序的原理，计算机系统的工作工程就是反复进行“取指令→分析指令→执行指令”的过程，具体过程如图 1-2 所示，即在控制器的控制下，计算机通过输入设备输入程序或数据，并自动存放在存储器中，然后控制器通过地址访问存储器，逐条取出指令、分析指令，并根据指令产生的控制信号控制其他部件执行这条指令中规定的任务。这一过程周而复始，实现了程序的自动执行。

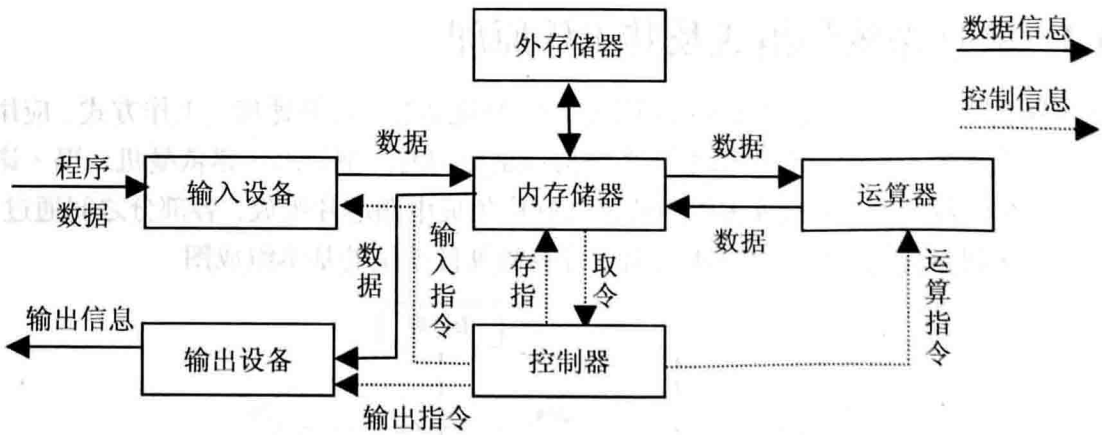

图 1-2　计算机硬件系统的工作原理

1.1.2　软件系统的组成

只有硬件的计算机（裸机）是无法运行的，还需要软件的支持。所谓软件是指使计算机正常运行需要的程序及相关技术文档的总称，其中更为重要的是程序，在不严格的情况下，通常认为程序就是软件。如今，软件技术已经渗透到生活、生产的各个领域，即便人们不了解计算机的内部结构和原理，也可以灵活地使用计算机为人们工作、服务。计算机能够协助人类完成各种各样的任务，正是依赖于各种用途的软件。

根据软件用途，计算机的软件分为系统软件和应用软件。

系统软件是为了计算机能正常、高效工作所配备的各种管理和监控程序，在系统一级上提供服务。主要分为两类：一类是面向计算机的软件，如操作系统、各种服务性程序；另一类是面向用户的软件，如编程语言处理程序，以及为适应事务处理的需要而开发的数据库管理系统。

应用软件是为解决用户的特殊需要而开发的特定程序。这类软件通常与具体的应用相关，可以应用于科学计算、数据处理、商业经营、经济管理、工程控制、企业管理等各个领域。应用软件种类非常多，包括办公软件、电子商务软件、通信软件、行业软件和游戏软件等。随着软件技术的发展，这类软件将越来越多。初学 C 语言的读者的主要任务是学习如何编写应用软件。

1.2　进位计数制及其转换

什么是进位计数制？进位计数制是利用固定的数字符号和统一的规则来表示数值的方法，可使用的数字符号的个数称为基数，基数为 n，即可称 n 进位制，简称 n 进制。

在日常生活中，常见的数值计算都是十进制，如“10 角等于 1 元”、“10 两等于 1 斤”，这种数制系统使用阿拉伯数字 0～9 组成的数字序列来表示数值。除此之外，还会遇到其他进制，如“每小时 60 分钟”是六十进制，“一年 12 个月”是十二进制，古衡器流行时期的“半斤八两”是指十六进制。在计算机系统中，由于数字电路采用电压的高低或通断表示两种状态（计算机内部采用 0、1 表示），所以计算机内部的所有信息都是采用二进制表示的。

1.2.1　计算机中数制的表示

1. 十进制表示

十进制（基数为 10）是使用阿拉伯数字字串来表示数值的方法。它包括“十进位”和“位值制”两条规则。其中，“十进位”是指按“满十进一”、“借一当十”的原则进行计数。“位值制”是指每一个数码符号根据它在这个数中所处的位置来决定其实际数值，即各数位的位权是以 10 为底的幂次方。

按照“十进位”与“位值制”的规则，任意一个十进制数 D，可表示成如下形式：

$$(D)_{10} = D_{n-1} \times 10^{n-1} + \cdots + D_1 \times 10^1 + D_0 \times 10^0 + D_{-1} \times 10^{-1} + \cdots + D_{-m} \times 10^{-m}$$

其中，$(D)_{10}$ 为十进制数的下标表示法；D 为数位上的数码，其取值范围为 0～9；n 为整数位个数，m 为小数位个数；10 为基数；10^i 是十进制数的位权（i 为 $n-1$、…、1、0、-1、…、$-m$）。

如十进制序列 $(123.45)_{10}$ 可以表示为：

$$(123.45)_{10} = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$$

2. 二进制表示

二进制（基数为 2）是指使用“0”和“1”两个基本符号来表示数值，按照“二进位”规则进行计数的方法。二进制是计算机信息表示和信息处理的基础。

与十进制数“形象”类似，任意一个二进制数 B，按照位权 2^i 展开，可以表示成如下形式：

$$(B)_2 = B_{n-1} \times 2^{n-1} + \cdots + B_1 \times 2^1 + B_0 \times 2^0 + B_{-1} \times 2^{-1} + \cdots + B_{-m+1} \times 2^{-m+1} + B_{-m} \times 2^{-m}$$

如二进制序列 $(1001.1)_2$ 可以表示为：

$$(1001.1)_2 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1}$$

3. 十六进制表示

由于二进制书写冗长、易错、难记，为了提高程序的可读性，程序员一般在程序源文件中使用二进制的“短格式”——十六进制。十六进制（基数为 16）是使用阿拉伯数字 0～9 和 A～F（依次对应于十进制数 10～15）来表示数值，并按照“十六进位”的原则进行计数的方法，如 123_{16}、$ABFC_{16}$、8976_{16}、$20FA.BC_{16}$ 都是合法的十六进制数。

同样，可以将十六进制序列 $(20A5.BC)_{16}$ 按位权 16^i 展开，表示成如下形式：

$$(20A5.BC)_{16} = 2 \times 16^3 + A \times 16^1 + 5 \times 16^0 + B \times 16^{-1} + C \times 16^{-2}$$

4. 八进制表示

八进制（基数为 8）表示法在早期的计算机系统中应用非常广泛，适用于 12 位和 36 位的计算机系统。它是使用阿拉伯数字 0～7 来表示数值的方法，权为 8^i。如：123_8、-456_8、10007_8、177777_8 都是合法的八进制数。

八进制序列 $(177777)_8$ 按位权展开，可以表示成如下形式：

$$(177777)_8 = 1 \times 8^5 + 7 \times 8^4 + 7 \times 8^3 + 7 \times 8^2 + 7 \times 8^1 + 7 \times 8^0$$

十进制、二进制、八进制与十六进制的特征对照表如表 1-1 所示。

表 1-1　　二进制、八进制、十进制与十六进制的特征对照表

进　制	数　码	进 位 规 则	数的地址表示法
十进制	0、1、2、3、4、5、6、7、8、9	满十进一	$(4096)_{10}$
二进制	0、1	满二进一	$(1011)_2$
八进制	0、1、2、3、4、5、6、7	满八进一	$(6777)_8$
十六进制	0～9、A、B、C、D、E、F	满十六进一	$(AF8E)_{16}$

1.2.2 非十进制数和十进制数的转换

非十进制数转换成十进制数采用“按权相加”法，即将非十进制数按位权写成加权系数展开式，然后按十进制加法规则进行求和。

【例 1-1】 将 $(1011.01)_2$、$(FD5)_{16}$ 和 $(475)_8$ 转换成十进制数。

解：$(1011.01)_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} = (11.25)_{10}$

$(FD5)_{16} = 15 \times 16^2 + 13 \times 16^1 + 5 \times 16^0 = (4053)_{10}$

$(475)_8 = 4 \times 8^2 + 7 \times 8^1 + 5 \times 8^0 = (317)_{10}$

十进制数转换成非十进制数时，由于整数部分与小数部分转换算法不同，需要分别进行。整数部分采用“除基取余”法，直至商为 0，先得到的余数为低位，后得到的余数为高位。小数部分采用“乘基取整”法，直至乘积为整数或达到控制精度。

【例 1-2】　将十进制数 100 和 0.625 分别转换成二进制数、八进制数与十六进制数。

解：

```
2 | 100
2 |  50  … 0
2 |  25  … 0
2 |  12  … 1
2 |   6  … 0
2 |   3  … 0
2 |   1  … 1
      0  … 1
```

```
8 | 100
8 |  12  … 4
8 |   1  … 4
      0  … 1
```

```
16 | 100
16 |   6  … 4
       0  … 6
```

所以（100）$_{10}$=（1100100）$_2$，（100）$_{10}$=（144）$_8$，（100）$_{10}$=（64）$_{16}$

```
0.625
x     2
1.250  … 1
x     2
0.500  … 0
x     2
1.000  … 1
```

```
0.625
x     8
5.000  … 5
```

```
0.625
x    16
10.000  … A
```

同样，（0.625）$_{10}$=（0.101）$_2$，（0.625）$_{10}$=（0.5）$_8$，（0.625）$_{10}$=（0.A）$_{16}$

1.2.3　二进制数、八进制数和十六进制数的转换

八进制与十六进制之所以流行不仅在于它们易于在程序中更紧凑地表示数值，还在于八进制与十六进制的基数恰好是 2 的幂，所以这两种数制与二进制之间的转换十分容易，只需要记住几条简单的规则，如表 1-2 所示，就能实现二进制数与八进制数、十六进制数之间的快速转换。

表 1-2　　常用数制对照表

十进制	二进制	八进制	十六进制	十进制	二进制	八进制	十六进制
0	0000	0	0	9	1001	11	9
1	0001	1	1	10	1010	12	A
2	0010	2	2	11	1011	13	B
3	0011	3	3	12	1100	14	C
4	0100	4	4	13	1101	15	D
5	0101	5	5	14	1110	16	E
6	0110	6	6	15	1111	17	F
7	0111	7	7	16	10000	20	10
8	1000	10	8	17	10001	21	11

1. 二进制数与八进制数的相互换算

二进制数转换成八进制数采用“三合一”法，即将 3 位二进制数转换成 1 位八进制数。具体转换过程是：将二进制数从小数点位置向左和向右按 3 位一组进行划分，向左不足 3 位按在数的最左侧补零的方法处理，向右不足 3 位按在数的最右侧补零的方法处理，然后按照“常用数制对照表”将 3 位一组的二进制数分别转换成相应的八进制数。

【例 1-3】　将二进制数（1010101111.110101）$_2$转换成八进制数。

解：001　　010　　101　　111　.　110　　101　　二进制数

1　　2　　5　　7　.　6　　5　　八进制数

所以（1010101111.110101）$_2$=（1257.65）$_8$

八进制数转换成二进制数也容易，采用“一拉三”法，按照表 1-2 将 1 位八进制数用 3 位二进制数替换即可。

【例 1-4】　将八进制数（667.25）$_8$转换成二进制数。

解：6　　6　　7　.　2　　5　　八进制数

110　　110　　111　.　010　　101　　二进制数

所以（667.25）$_8$=（110110111.010101）$_2$

2. 二进制数与十六进制数的相互换算

二进制数转换成十六进制数采用“四合一”法，按照表 1-2 将 4 位一组的二进制数转换为 1 位十六进制数。十六进制数转换成二进制数采用“一拉四”法，将 1 位十六进制数用 4 位二进制数替换即可。

【例 1-5】　将二进制数（1010101111.110101）$_2$转换成十六进制数。

解：0010　　1010　　1111　.　1101　　0100　　二进制数

2　　A　　F　　.　D　　4　　十六进制数

所以（1010101111.110101）$_2$=（2AF.D4）$_{16}$

1.3　机器数的表示形式及其表示范围

计算机最主要的功能是处理各种数值与非数值数据，如整数、实数值、字符、字符串等，在计算机内部，这些数据必须经过二进制编码后才能被存储和处理。为了确保编写的程序能够有效地处理各种不同类型的数据，我们必须清楚计算机物理上是如何表示这些数据的。

1.3.1　真值与机器数

计算机可以直接识别和处理用 0、1 两种状态的二进制形式的数据，却无法按人们的日常书写习惯来表示数值的正、负符号。那么，计算机是如何处理这种带符号数的呢？在计算机内部，数值的符号也是采用二进制代码 0 和 1 来存储和处理的，用 0 表示“+”，1 表示“-”。这种采用二进制表示形式，连同数的正、负符号一起代码化的数据称为机器数。与机器数对应的实际数据则为该机器数的真值。

机器数可分为无符号数和带符号数。无符号数表示正数，机器数的所有二进制位都用来表示数值。对于带符号数，机器数的最高位为符号位，其余位表示数值。若机器字长为 8 位，无符号数 44 和带符号数-44 的机器数如图 1-3 所示。

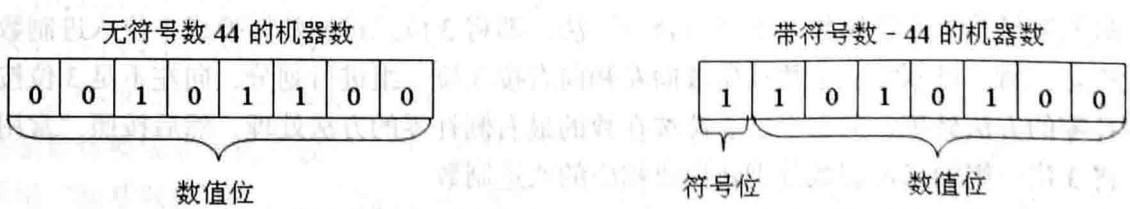

图 1-3　无符号数 44 与带符号数-44 的机器数

根据小数点位置固定与否，机器数又有定点数和浮点数两种表示法。定点数表示法在早期计算机中比较常见，但是由于定点二进制数的表数范围非常有限，人们又设计了浮点数表示法。由于浮点数的小数点是非固定的，为了简化浮点数的操作，一般需要将浮点数规范化处理为 $\pm 1.bbb\cdots b \times 2^{\pm E}$ 的形式。1985 年通过的 IEEE754 标准规定了 32 位的单精度和 64 位的双精度两种浮点格式。标准的 32 位的单精度浮点数的组织形式由符号位（1 位）、指数位（8 位）和有效数位（23 位）三部分组成。64 位的双精度浮点格式与 32 位标准浮点形式遵循相似的规则，只不过它的有效数位为 52 位，指数位为 11 位，有更高的精度和更大的表数范围。

1.3.2　数的原码、反码和补码

前面提到，对于带符号数，其符号也由二进制代码 0 和 1 来存储和处理，那么计算机具体是如何存储带符号数的？又是如何实现带符号数的运算操作呢？为了妥善处理好这些问题，带符号的机器数可采用原码、反码和补码等不同的编码方法。这些编码又称为码制。

1. 原码

原码又称符号绝对值码，是一种比较直观的机器数表示法。原码与真值的区别仅仅是将真值的符号数字化，即用 0 表示“+”，1 表示“-”。

【例 1-6】　若机器字长为 16 位，分别给出十进制数 +84 和 -45 的原码表示。

解：由于 $(84)_{10}=(1010100)_2$，将“+”数字化为 0，则 $[+84]_{原}$ = 0 000000001010100；$(45)_{10}=(101101)_2$，将“-”数字化为 1，则 $[-45]_{原}$ = 1 000000000101101。

原码表示数简单易懂，易于同真值进行转换，实现乘除运算规则简单。由于符号位要单独操作，实现加减运算很不方便，逻辑电路实现也较为复杂。

2. 反码

反码是机器数运算过程的中间表示形式。在反码表示中，正数的反码与原码相同；负数的反码等于其原码（除符号位）按位取反（即 0 变为 1，1 变为 0）。

如例 1-6 中，$[+84]_{反}$ = 0 000000001010100，$[-45]_{反}$ = 1 111111111010010。

3. 补码

在所有的算术运算中，加减运算要占到 80%以上，因此，能否方便地进行正、负数加减运算直接影响计算机的运行效率。为了克服原码实现加减运算不方便的局限性，从而引入了补码表示法。目前，现代计算机系统中广泛采用补码表示法来表示和存储数值。

在补码表示中，正数的补码与其原码相同，即 $[X]_{补}=[X]_{原}$；负数的补码等于其原码（除符号位）按位取反，末位加 1，即 $[X]_{补}=[X]_{反}+1$。

如例 1-6 中，$[+84]_{补}$ = 0 000000001010100，$[-45]_{反}$ = 1 111111111010011。

【例 1-7】　若机器字长为 16 位，分别给出+1，−1，+32767，−32767 的原码、反码和补码。

解：$[+1]_{原}=[+1]_{反}=[+1]_{补}$ = 0 000000000000001

$[-1]_{原}$ = 1 000000000000001　　$[-1]_{反}$ = 1 111111111111110

$[-1]_{补}$ = 1 111111111111111

$[+32767]_{原}=[+32767]_{反}=[+32767]_{补}$ = 0 111111111111111

$[-32767]_{原}$ = 1 111111111111111　　$[-32767]_{反}$ = 1 000000000000000

$[-32767]_{补}$ = 1 000000000000001

反之，已知一个数的补码求原码，如果补码的符号位为“0”，表示该数为正数，补码等于该数的原码。如果补码的符号位为“1”，表示该数为负数，求原码的操作是，将补码的各位（除符

号位）取反，末位加 1。

【例 1-8】 一个数的补码为 1 111111111111001，试求该补码对应的真值。

解：由于符号位为“1”，表示负数，数值位按位取反后为 000000000000110，再加 1，则该数的原码为 1 000000000000111，真值为 - 7。

1.3.3 无符号整数与带符号整数

计算机中的整数可分为无符号整数和带符号整数。在某些情况下，要处理的整数全是正数时，此时保留符号位毫无意义。如果将符号位也作为数值位处理，可形成无符号整数。如 8 位二进制补码 10101010，如果表示带符号整数，则最高位 1 作为符号位处理，其真值为-86；如果表示无符号整数，则 $1\times2^7+1\times2^5+1\times2^3+1\times2^1$，即 170。

无论是无符号整数还是带符号整数，其表示的范围均受机器字长的限制。若机器字长为 n 位，则最多可以表示 2^n 个不同的数值。如果表示无符号整数，其表示范围为[0，2^n-1]，如果用来表示带符号整数，则需要将这 2^n 个不同的组合在负数与非负数之间进行“平分”，其表示范围为[-2^{n-1}，$2^{n-1}-1$]。表 1-3 列出了 8 位、16 位、32 位无符号整数和带符号整数的取值范围。

表 1-3　　数的表示范围

机器字长 n（位）	能表示的值的个数	无符号整数的取值范围	带符号整数的取值范围
8	256	0～255	−128～127
16	65536	0～65535	−32768～32767
32	4294967296	0～4294967295	−2147483648～2147483647

ANSI（American National Standard Institute）标准规定 C 编译系统中的带符号整数的最小取值范围是[−32768，32767]，无符号整数的最小取值范围是[0，65535]。如果带符号整数与无符号整数的数值超出了它们所能表示的范围，就会产生“溢出”。如在 Turbo C 和 Turbo C++中，带符号整数 i=32767，而 i+1=−32768，为什么一个正整数 32767 加 1 后变成了一个负数呢？这是由于在运算过程中，数值位的数据溢出到符号位造成的，如图 1-4 所示。

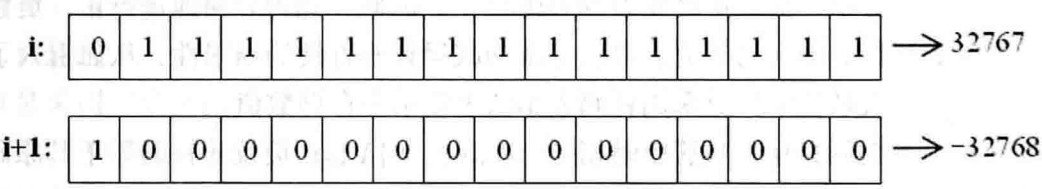

图 1-4　数值位的数据溢出

从图 1-4 可以看出：整型变量 i 在内存中的二进制补码为 0 111111111111111，如果将其加 1，则变为 1 000000000000000，该二进制序列正是−32768 的补码形式。因此，32767 加 1 得不到预想的结果 32768。值得注意的是，当产生溢出时，C 编译系统并不会报错。这是非常危险的。如果 C 程序初学者对无符号整数和带符号整数的取值范围没有充分的认识，那么这种由于溢出而产生的错误将被堂而皇之地潜伏下来，直到最终得出一个离预想相差甚远的结果。

1.3.4 字符的表示法

“字符”一词是指人或者机器可读的符号。一般来说，一个字符就是任何一个可以使用键盘键入或者可以显示在视频显示器上的符号。字符数据包括字母、标点符号、数字、空格、制表符、

回车、控制字符，以及其他特殊符号。这些信息应用到计算机中时，也需要转换成二进制序列后才能进行存储。那么它们是如何进行编码的呢？

目前计算机中用得最广泛的字符集及其编码是由美国国家标准局（ANSI）制定的 ASCII 码（American Standard Code for Information Interchange，美国标准信息交换码）。ASCII 码适用于所有拉丁文字，它用 7 位二进制数进行编码，可以表示 128 个不同的字符。其中，ASCII 码 0～32 号及第 127 号（共 34 个）表示控制字符或通信专用字符，如控制符：LF（换行）、CR（回车）、FF（换页）、DEL（删除）等；第 48～57 号表示 0～9 十个阿拉伯数字字符；第 65～90 号表示 26 个大写英文字母，第 97～122 号表示 26 个小写英文字母，其余的表示一些标点符号、运算符号等。常用字符与 ASCIT 的对照表如附录 1 所示。

由于 ASCII 码的最高位没有使用上，所以人们又造出了 ASCII 扩展码，它的值为 128～255。目前许多基于 x86 的系统都支持 ASCII 扩展码，它用以存放英文的制表符、部分音标字符及其他符号。最早的文字排版系统都会使用到 ASCII 扩展码，以完成表格的输出和打印。

习　题　1

1. 简述计算机系统的基本组成及其工作原理。

2. 将下列各数转换成十进制数。

$(1011.001)_2$，$(127.75)_8$，$(A1.D4)_{16}$

3. 将下列各数转换成二进制数、八进制数和十六进制数（无法精确表示时，二进制取 6 位小数，八进制和十六进制数取 2 位小数）。

$(25.34)_{10}$，$(125.25)_{10}$，$(258)_{10}$，$(783.8275)_{10}$

4. 设机器字长为 16，分别写出下列各值的原码、反码和补码。

$(127)_{10}$，$(-127)_{10}$，$(-128)_{10}$，$(-46)_{10}$，$(32767)_{10}$，$(-32768)_{10}$

5. 已知 X 的补码，写出其原码与真值。

（1）$[X]_{补}=01010011$　　（2）$[X]_{补}=10001001$

（3）$[X]_{补}=11111111$　　（4）$[X]_{补}=11000000$

6. 设机器字长为 16 位，表示带符号整数，数值位 15 位，符号位 1 位，试分析所能表示的最大整数与最小整数分别是多少。

7. 设机器字长为 32 位，浮点表示时，数符 1 位；阶码 8 位，用补码表示；尾数 23 位，用补码表示。试分析规格化数所能表示的数的范围。