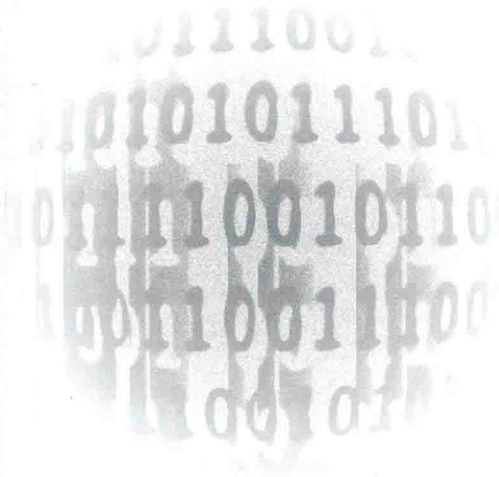
工业和信息化普通高等教育“十二五”规划教材立项项目
21世纪高等学校计算机规划教材
21st Century University Planned Textbooks of Computer Science

C语言程序设计

C Language Programming

赵山林 高媛 主编



高校系列

人民邮电出版社
POSTS & TELECOM PRESS

C语言程序设计

C Language Programming

工业和信息化普通高等教育“十二五”规划教材立项项目
21世纪高等学校计算机规划教材
21st Century University Planned Textbooks of Computer Science

C语言程序设计

C Language Programming

赵山林 高媛 主编

高校系列

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 / 赵山林，高媛主编. — 北京 : 人民邮电出版社，2012.10
21世纪高等学校计算机规划教材
ISBN 978-7-115-28793-9

Ⅰ. ①C… Ⅱ. ①赵… ②高… 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(2012)第208204号

内 容 提 要

本书是“零基础”读者的 C 语言教材。全书按照基本类型数据、结构化程序设计、模块化函数设计、指针、数组、字符串、结构体、文件的顺序阐述了 C 语言语法和句法规则，归纳了分支、循环算法设计思路和函数设计方法，并配以大量实例阐述了程序设计方法。本书习题包括选择题、填空题、编程题 3 种题型，覆盖范围广。

本书可作为高等院校计算机及相关专业的教材，也可作为自学参考书。

21 世纪高等学校计算机规划教材

C 语言程序设计

◆ 主　　编　赵山林　高　媛
责任编辑　邹文波

◆ 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　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
三河市海波印务有限公司印刷

◆ 开本：787×1092　1/16
印张：21.25　　2012 年 10 月第 1 版
字数：558 千字　　2012 年 10 月河北第 1 次印刷

ISBN 978-7-115-28793-9

定价：39.80 元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154

前言

以计算机和网络技术为代表的信息技术以前所未有的深度和广度渗透于社会和生活的各个领域，正在改变着人们传统的工作和生活方式，推动着社会的发展。人类社会的信息化是社会发展进程中的一次革命，大学生是信息社会中未来最重要的主体力量，是推动国家经济健康持续发展的生力军。程序设计能力是培养计算思维、提高信息能力最有效的手段。

C 语言既具备高级语言结构化的特点，还具备机器语言直接访问计算机硬件的特点，是高效的程序设计语言。C 语言解决实际问题的方法更接近于计算机思维，能有效地处理各种信息。C 语言广泛应用于操作系统、大型应用软件、嵌入式系统和工业控制等软件的编写，是应用最为广泛的程序设计语言。

本书为了让“零基础”读者能较好地掌握 C 语言规则和程序设计方法，首先阐述了学习 C 语言必备的计算机基础知识和程序设计基础知识，然后从认知角度出发，合理组织了 C 语言的各个知识点，详细地阐述了 C 语言语法和句法规则及程序设计方法。本教材充分体现了以下特点。

1. 通俗易懂。本书是“零基础”读者的 C 语言教材，在阐述语法和句法规则时避免了枯燥的、难以理解的概念和理论，以实例和解释的方式阐述规则，使读者易于学习和掌握。

2. 内容组织合理。本书打破了传统的 C 语言教材内容组织方法，从认知角度出发，合理编排了各知识点。按照基本类型数据、结构化程序设计、模块化函数设计、指针、数组、字符串、结构体、文件的顺序由浅入深地阐述了 C 语言的语法和句法规则。

3. 实例丰富多样。本书给出了大量的解决实际问题的程序设计实例，详细阐述了程序设计方法。大部分实例都给出了 N-S 结构化流程图表示的算法及程序代码。

4. 习题覆盖全面。本书习题包括选择题、填空题、编程题 3 种题型，覆盖范围广。选择题着重语法和句法规则的使用，填空题着重程序阅读和典型经典实例，编程题着重实际应用。

本书由赵山林、高媛主编。其中，第 2、5、8、11、13 章由赵山林编写，第 7、9、10、12 章及附录由高媛编写，第 1 章由乔志会编写，第 3、4 章由李军莉编写，第 6 章由樊永生编写。

编　者

2012 年 8 月

目 录

第 1 章 绪论 ····· 1
1.1 计算机基础知识 ····· 1
1.1.1 冯·诺依曼体系结构的计算机及工作原理 ····· 1
1.1.2 数制及数制间的转换 ····· 3
1.1.3 数据在内存中的存储 ····· 7
1.1.4 软件系统 ····· 12
1.2 程序设计基础 ····· 14
1.2.1 利用计算机解决实际问题的具体步骤 ····· 15
1.2.2 算法及其表示 ····· 16
1.2.3 结构化程序设计 ····· 21
1.3 C 语言简介 ····· 22
1.3.1 C 语言的发展 ····· 22
1.3.2 C 语言的符号集 ····· 24
1.3.3 C 程序结构 ····· 25
习题 ····· 27

第 2 章 数据及运算 ····· 29
2.1 数据类型 ····· 29
2.2 C 程序中数据的表示方法 ····· 30
2.2.1 常量 ····· 30
2.2.2 变量 ····· 34
2.3 运算符及其表达式 ····· 37
2.3.1 C 运算符概述 ····· 37
2.3.2 算术运算符及算术表达式 ····· 37
2.3.3 赋值运算符和赋值表达式 ····· 39
2.3.4 自增自减运算符及含自增自减运算的表达式 ····· 40
2.3.5 逗号运算符和逗号表达式 ····· 41
2.3.6 表达式小结 ····· 42
2.4 类型转换 ····· 42
2.4.1 类型的自动转换 ····· 42
2.4.2 类型的强制转换 ····· 47
习题 ····· 48

第 3 章 顺序结构程序设计 ····· 50
3.1 C 语句概述 ····· 50
3.2 常用库函数使用 ····· 51
3.2.1 库函数的使用 ····· 52
3.2.2 输出函数 ····· 53
3.2.3 输入函数 ····· 59
3.3 顺序结构程序设计举例 ····· 64
习题 ····· 66

第 4 章 选择结构程序设计 ····· 69
4.1 逻辑量 ····· 69
4.1.1 任意确定的值 ····· 69
4.1.2 关系表达式 ····· 69
4.1.3 逻辑表达式 ····· 70
4.2 if 语句 ····· 72
4.2.1 if 语句的格式 ····· 72
4.2.2 条件表达式 ····· 75
4.2.3 if 语句的嵌套 ····· 77
4.3 switch 语句 ····· 81
4.4 选择结构程序设计举例 ····· 85
习题 ····· 90

第 5 章 循环结构程序设计 ····· 94
5.1 循环结构的概念 ····· 94
5.2 实现循环的语句 ····· 95
5.2.1 while 语句 ····· 95
5.2.2 do～while 语句 ····· 99
5.2.3 for 语句 ····· 102
5.3 break 和 continue 语句 ····· 105
5.3.1 break 语句 ····· 105
5.3.2 continue 语句 ····· 107
5.4 循环的嵌套 ····· 108
5.5 循环结构程序设计举例 ····· 111
习题 ····· 116

第 6 章 函数 ····· 122
6.1 概述 ····· 122

6.1.1 函数概述……122
6.1.2 C 程序结构……124
6.1.3 函数分类……125
6.2 函数的定义……125
6.2.1 函数定义的一般形式……125
6.2.2 关于函数定义的几点说明……128
6.3 函数的调用……130
6.3.1 函数声明……130
6.3.2 函数调用……132
6.3.3 函数调用的执行机制……133
6.4 函数的嵌套调用和递归调用……134
6.4.1 函数的嵌套调用……134
6.4.2 函数的递归调用……135
6.5 变量的作用域与生存期……139
6.5.1 变量的属性……139
6.5.2 局部变量……140
6.5.3 全局变量……142
6.6 函数举例……147
习题……151

第 7 章 指针……156
7.1 指针的基本概念……156
7.1.1 变量的直接访问和间接访问……156
7.1.2 指针与指针变量……157
7.2 指向变量的指针变量……158
7.2.1 指针变量的定义……158
7.2.2 指向关系的建立……159
7.2.3 间接访问……160
7.3 指针变量做函数形参……163
7.4 指向函数的指针和返回指针值的函数……166
7.4.1 指向函数的指针……166
7.4.2 指向函数的指针变量……167
7.4.3 指向函数的指针变量做函数参数……169
7.4.4 返回指针值的函数……171
7.5 多级指针……172
7.5.1 二级指针……172
7.5.2 多级指针……173
习题……173

第 8 章 数组……176
8.1 数组的基本概念……176
8.2 一维数组……177
8.2.1 一维数组的定义……177
8.2.2 一维数组元素的引用……178
8.2.3 一维数组元素的初始化……180
8.2.4 一维数组程序设计举例……180
8.3 指向数组元素的指针变量……184
8.3.1 指向数组元素的指针变量……184
8.3.2 指针变量做函数形参访问主调函数中的数组元素……190
8.4 二维数组……195
8.4.1 二维数组的定义……195
8.4.2 二维数组元素的引用……196
8.4.3 二维数组元素的初始化……197
8.4.4 二维数组程序设计举例……198
8.5 二维数组和指针……200
8.5.1 二维数组的指针……200
8.5.2 指向二维数组元素的指针变量……201
8.5.3 指向一维数组的指针变量……202
8.5.4 指向一维数组的指针变量做函数形参……203
8.6 指针数组……204
8.6.1 指针数组的定义……205
8.6.2 指针数组的引用与初始化……205
8.6.3 利用指针数组处理二维数组……205
8.6.4 一维指针数组和二级指针的关系……206
习题……206

第 9 章 字符串……210
9.1 字符数组和字符串……210
9.1.1 一维字符数组表示字符串……210
9.1.2 二维字符数组表示字符串……214
9.2 指向字符的指针变量和字符串……216
9.2.1 指向字符的指针变量表示字符串……216
9.2.2 使用指针数组表示多个字符串……224
9.3 命令行参数……227
习题……228

第 10 章 结构体、联合体与枚举……232
10.1 结构体类型及结构体变量……232
10.1.1 结构体类型的定义……232

10.1.2　结构体变量……233
10.1.3　结构体嵌套……237
10.1.4　结构体变量做函数形参……239
10.2　结构体数组……240
10.2.1　结构体数组的定义……241
10.2.2　结构体数组的引用……241
10.2.3　结构体数组的初始化……242
10.3　结构体指针……243
10.3.1　指向结构体变量的指针变量……243
10.3.2　指向结构体数组元素的指针……245
10.3.3　结构体指针变量做函数形参……247
10.4　链表……248
10.4.1　单向链表的概念……248
10.4.2　内存的动态分配与释放函数……249
10.4.3　单向链表算法……251
10.5　联合体……260
10.5.1　联合体类型的定义……260
10.5.2　联合体变量的定义……261
10.5.3　联合体变量的引用……262
10.6　枚举……266
10.6.1　枚举类型的定义……266
10.6.2　枚举变量的定义和引用……266
10.7　类型别名……268
习题……269

第 11 章　位和位段……273
11.1　位运算……273
11.1.1　位运算符……273
11.1.2　位运算举例……278
11.2　位段……280
11.2.1　位段结构体类型的定义……280
11.2.2　位段结构体类型变量的定义……281
11.2.3　位段结构体变量成员的引用……283
习题……284

第 12 章　文件……287
12.1　文件概述……287
12.1.1　文件的概念……287
12.1.2　缓冲文件系统……288
12.2　文件的使用……288
12.2.1　文件类型指针……289
12.2.2　文件的打开……289
12.2.3　文件的关闭……291
12.2.4　文件的读写……291
12.2.5　文件的定位……297
12.2.6　文件的检测……299
12.3　文件程序设计举例……300
习题……303

第 13 章　编译预处理……304
13.1　宏定义……304
13.1.1　不带参数的宏定义……304
13.1.2　带参数的宏定义……307
13.2　文件包含……309
13.3　条件编译……311
习题……313

附录 1　常用字符的 ASCII 码表……315

附录 2　C 的运算符及优先级和结合性……316

附录 3　Visual C++ 6.0 集成开发环境……317

附录 3.1　在 Visual C++ 6.0 环境下编写 C 语言程序……318
附录 3.2　Visual C++ 6.0 常用命令……322
附录 3.3　常见错误提示信息及修改建议……324

附录 4　常用标准库函数……328

第 1 章 绪论

1.1 计算机基础知识

计算机发展到今天已经形成一个完整的、复杂的体系，其涉及门类广，内涵层次深，已经成为一门综合性学科，与其相关的研究也已深入到了各个基础学科和高尖端学科中。而对计算机基础知识的了解是掌握计算机在各学科中应用的基础。本节将介绍计算机的组成及工作原理、数制、数据在内存中的存储以及计算机语言等与 C 程序设计的学习密切相关的基础知识。

1.1.1 冯·诺依曼体系结构的计算机及工作原理

1945 年，冯·诺依曼在领导设计 EDVAC 计算机的过程中，提出了现代计算机的若干设计思想，被后人称为冯·诺依曼思想，这是计算机发展史上的里程碑。自 1945 年至今计算机的设计大多采用的是冯·诺依曼体系结构，冯·诺依曼也因此被人们称为“计算机之父”。

1. 冯·诺依曼体系结构的基本思想

冯·诺依曼思想包括以下两个核心内容。

（1）二进制

二进制是指所有数据必须以二进制形式表示才能被计算机存储、传输和处理。数据是所有能输入到计算机并能被计算机处理的物理符号的总称，包括数值、文字、程序、声音、图像和图形等。

（2）存储程序控制

① 存储程序。程序是利用计算机解决实际问题的具体操作步骤的有序指令集合。存储程序的含义是指将事先编制的程序和与程序相关的原始数据存入存储器。

② 程序控制。当存储程序后，程序就获得了对计算机的控制权。计算机按照程序中的指令，控制计算机各部件协同工作，完成程序规定的任务。

2. 冯·诺依曼体系结构的计算机

按照冯·诺依曼思想设计的冯·诺依曼体系结构的计算机由运算器、存储器、控制器、输入设备和输出设备五大部件构成。

（1）运算器

运算器是计算机中处理数据的核心部件，主要由执行算术运算和逻辑运算的算术逻辑单元（ALU）、存放操作数和中间结果的寄存器组以及连接各部件的数据通路组成，用以完成各种算术运算和逻辑运算。

（2）控制器

控制器是计算机中控制管理的核心部件。主要由程序计数器、指令寄存器、指令译码器、时序控制电路和微操作控制电路等组成。在系统运行过程中，控制器不断地取出指令、分析指令，并向计算机各个部件发出操作控制信号，指挥各个部件高速协调工作。

由于运算器和控制器在逻辑关系和电路结构上联系十分紧密，尤其在大规模集成电路制作工艺出现后，这两大部件往往被集成在同一芯片上，被称为中央处理器，简称 CPU。

（3）存储器

存储器是用来存放程序和数据的，起着存储、缓冲、传递数据和指令的作用。存储器分为主存储器和辅助存储器。

① 主存储器。主存储器又称为内存储器，简称主存或内存，用来存放计算机上正在运行的程序和数据，可直接与 CPU 交换数据。

计算机内存是由电子元器件构成的。从使用功能上内存又可分为随机存储器（Random Access Memory，RAM，又称为读写存储器）和只读存储器（Read Only Memory，ROM）。只读存储器用来存放监控程序、系统引导程序等专用程序。在生产制作只读存储器时，将相关的程序指令固化在存储器中，并永久保存下来，不会因断电而丢失。在正常工作环境下，只能读取其中的指令，而不能修改或写入。随机存储器用来存放正在运行的程序及所需要的数据，具有存取速度快、集成度高、电路简单等优点，但断电后信息将丢失。随机存储器可分为动态（DRAM）和静态（SRAM）两大类。DRAM 的特点是集成度高，主要用于大容量内存储器；SRAM 的特点是存取速度快，主要用于高速缓冲存储器，CPU 中的寄存器就是 SRAM。

② 辅助存储器。辅助存储器又称为外存储器，简称辅存或外存，能长久保存数据。外存的特点是存储容量大、成本低，但存取速度相对较慢。外存储器中的程序和数据不能直接被运算器、控制器处理，必须先调入内存储器才能处理。目前广泛使用的外存储器主要有硬盘、光盘和 U 盘等。

（4）输入/输出设备

输入/输出设备又称为外部设备，简称 I/O 设备，是与计算机主机进行信息交换，实现人机交互的硬件设备。

输入设备用于输入人们要求计算机处理的数据、字符、文字、图形、图像和声音等信息以及处理这些信息所必需的程序，并把它们转换成计算机能接受的形式。常用的输入设备有键盘、鼠标、摄像头、扫描仪和麦克风等。

输出设备用于将计算机处理的结果以人们可识别的形式输出。常用的输出设备有显示器、打印机、音响设备和绘图仪等。

3. 冯·诺依曼计算机工作原理

按照冯·诺依曼思想，计算机的工作过程可以简单概括为存储程序和程序控制。利用输入设备将程序和原始数据输入到计算机内存；计算机在控制器的控制下，从内存逐条取出程序中的每一条指令交给运算器去执行，并将运算结果送回存储器；利用输出设备将程序执行结果输出。冯·诺依曼体系结构计算机的工作原理如图 1.1 所示。计算机自动执行时，有数据流和控制流两种流动

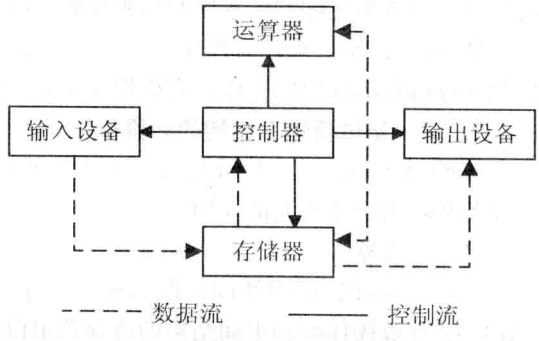


图 1.1　计算机工作原理图

的信息。数据流是指程序、原始数据、中间结果和最终结果等数据信息在五大组成部分之间的传输；控制流是由控制器进行指令分析后向各部件发出的控制命令，指挥各部件协调工作。控制器是整个计算机的核心，无论是数据的输入/输出、存储器的读/写，还是数据的运算处理等，都必须在控制器的控制下有序地进行。

计算机的工作过程就是执行指令的过程。指令是能被计算机识别并执行的二进制代码，规定了计算机能完成的某一种操作，一条指令通常由操作码和操作数组成，如指令“A＋B”的操作码就是加法操作，操作数就是 A 和 B，加法操作和操作数都以二进制形式表示。下面通过执行指令“A＋B”简单描述一下计算机的工作原理。

① 取指令。从内存中取出指令“A＋B”，送入控制器中的指令寄存器。指令寄存器是用于存放待执行指令的高速存储器。

② 分析指令。控制器对指令寄存器中的指令进行分析，将指令的操作码“+”转换成相应的控制信号并确定操作数 A 和 B 在内存中的存储位置。

③ 执行指令。从内存中读取数据 A 和 B 送入运算器中的数据寄存器，由运算器完成加法，并将结果存入运算器中的数据寄存器。

计算机的工作就是连续的取指令、分析指令、执行指令的过程。

1.1.2　数制及数制间的转换

1. 数制

（1）数制的定义

按进位的原则进行计数的方法，称为进位计数制，简称数制。在日常生活中常采用十进制进行计数，例如，9＋1＝10，1 元＝10 角等。除此之外还有许多非十进制的计数方法。例如，12 个月为 1 年，用的是十二进制计数法。7 天为 1 周，用的是七进制计数法。在进位计数制中，如果用 r 个基本符号（例如：0、1、2，…、$r-1$）表示数值，则称其为 r 进制。如十进制中基本符号个数 $r=10$，其基本符号为 0、1、2、3、4、5、6、7、8、9。如取 $r=2$，则为二进制，其基本符号为 0、1。计算机中所处理的数据都是用二进制来表示的。

无论哪种进制，它们都包含两个要素：基数和位权。

① 基数。每一种数制都有固定的符号集，不同数制中允许选用的数码个数称为该数制的基数。如十进制，其数码有 10 个：0、1、2、3、4、5、6、7、8、9；二进制，其数码有两个：0 和 1；八进制，其数码有 8 个：0、1、2、3、4、5、6、7；十六进制，其数码有 16 个：0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F。

② 位权。处于不同位置的数码所代表的数值不同，其值等于该数码乘以位权。位权是以基数为底、数码所在位置的序号为指数的整数次幂。n 个数码符号组成的整数从低到高每一位的位置序号依次是 0、1、2…$n-1$。

例如：

十进制整数 1234 中，第 0 位是 4，位权为 10^0，所代表的值是 4×10^0；第 1 位是 3，位权为 10^1，所代表的值是 3×10^1；第 2 位是 2，位权为 10^2，所代表的值是 2×10^2；第 3 位是 1，位权为 10^3，所代表的值是 1×10^3。

二进制整数 1101 中，第 0 位是 1，位权为 2^0，所代表的值是 1×2^0；第 1 位是 0，位权为 2^1，所代表的值是 0×2^1；第 2 位是 1，位权为 2^2，所代表的值是 1×2^2；第 3 位是 1，位权为 2^3，所代表的值是 1×2^3。

（2）常用数制及其表示方法

现实世界中的数据常用十进制表示，计算机内部数据的表示使用的是二进制。但二进制无论是书写、阅读还是记忆均很不方便，所以经常采用八进制或十六进制简化对二进制的描述。常用的数制如表 1-1 所示。

表 1-1　　常用的数制

		二 进 制	八 进 制	十 进 制	十 六 进 制
进位原则		逢二进一	逢八进一	逢十进一	逢十六进一
基数		2	8	10	16
数码		0,1	0,1…7	0,1…9	0,1…9,A…F
表示方法	后缀	B（Binary）	O（Octal）	D（Decimal）	H（Hexadecimal）
	下标	$(N)_2$	$(N)_8$	$(N)_{10}$	$(N)_{16}$

为了区分不同进制的数据，书写时可以使用表 1-1 中的表示方法。可以通过后缀形式来表示不同数制，如 1234D 表示十进制数 1234、1101B 表示二进制数 1101、1470O 表示八进制数 1470、1A0FH 表示十六进制数 1A0F。但是八进制后缀 O 和数字 0 容易混淆，故常用 Q 来表示八进制数，如 1470Q 表示八进制数 1470。还可以使用右下标形式表示不同数制，如$(1234)_{10}$、$(1101)_2$、$(1470)_8$、$(1A0F)_{16}$分别表示十进制、二进制、八进制和十六进制数。

十进制是日常生活中常用的数制，书写时可以省略其后缀 D 或者右下标，如 1234 默认为十进制。

同一数据值，使用不同进制表示时不完全一样。如十进制 2 用二进制表示就是 10。表 1-2 中列出了常用的几种数制之间的关系。

表 1-2　　常用数制之间的关系

二 进 制 数	十 进 制 数	八 进 制 数	十六进制数
0	0	0	0
1	1	1	1
10	2	2	2
11	3	3	3
100	4	4	4
101	5	5	5
110	6	6	6
111	7	7	7
1000	8	10	8
1001	9	11	9
1010	10	12	A
1011	11	13	B
1100	12	14	C
1101	13	15	D
1110	14	16	E
1111	15	17	F
10000	16	20	10

2. 数制转换

（1）任意进制转换为十进制

任意进制表示的数都可以写成按其位权展开的多项式之和，称为位权展开式。任意一个 r 进

制整数 N 可表示为：

$$N = a_{n-1} \times r^{n-1} + \ldots + a_i \times r^i + \ldots + a_1 \times r^1 + a_0 \times r^0 = \sum_{i=0}^{n-1} a_i \times r^i$$

其中，a_i 是第 i 位的数码；r 是基数，表示不同的进制；r^i 是位权；n 为整数 N 的位数。

当把任意进制数转换为十进制数时，只需将任意进制数的位权展开式按照十进制的运算法则运算即可得到转换之后的十进制数。

例如：

$1234 = 1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0 = 1234$

$1101B = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13$

$1470Q = 1 \times 8^3 + 4 \times 8^2 + 7 \times 8^1 + 0 \times 8^0 = 824$

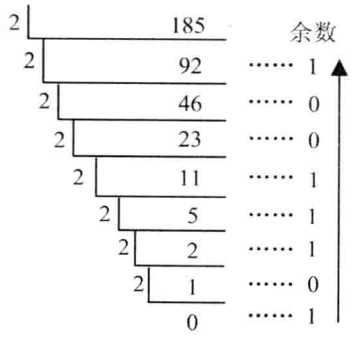
$1A0FH = 1 \times 16^3 + 10 \times 16^2 + 0 \times 16^1 + 15 \times 16^0 = 6671$

（2）十进制转换为任意进制

十进制整数转换为任意进制数采用“除基逆序取余法”。即用十进制整数除以任意进制的基数，如果商不为 0，则把商看做被除数，继续除以基数，直至商为 0。将每次相除所得的余数逆序排列即可得到转换后的进制数。

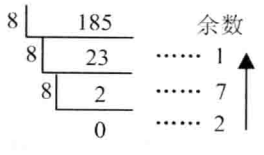
① 十进制整数转换为二进制。

例如：185 = 10111001B



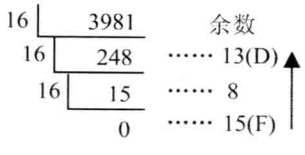
② 十进制整数转换为八进制。

例如：185 = 271Q



③ 十进制整数转换为十六进制。

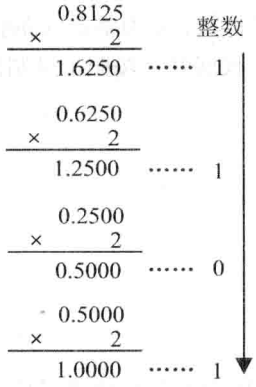
例如：3981 = F8DH



十进制小数转换为任意进制数采用“乘基正序取整法”。即用十进制小数乘以任意进制的基数，

如果乘积的小数部分不为 0，则继续用基数乘以乘积的小数部分，直至乘积的小数部分为 0。将每次乘积的整数部分正序排列即可得到转换后的进制数。

例如：0.8125 = 0.1101B



注意，“乘基正序取整法”可能存在乘积的小数部分永远不为 0 的情况，这时只需按照要求的精度得到部分小数位即可。

（3）二进制与八进制之间的转换

① 二进制转换为八进制。

由表 1-2 可知，三位二进制数 000～111 正好可以完全表示八进制数的数码 0～7，所以二进制数转换为八进制数可采用“三合一”的方法进行转换。具体规则是从低位到高位，每三位二进制组成一组，不足三位时，在高位用 0 补齐，把每一组二进制数用一位等值的八进制数表示，即可得到转换后的八进制数。

例如：1011010101110B = 13256Q

001 011 010 101 110 B
1 3 2 5 6 Q

② 八进制转换为二进制。

八进制数转换为二进制数可采用“一拆三”的方法进行转换。具体规则是把每一位八进制数写成等值的三位二进制数即可得到转换后的二进制。

例如：14270Q = 1100010111000B

1 4 2 7 0 Q
001 100 010 111 000 B

（4）二进制与十六进制之间的转换

① 二进制转换为十六进制。

由表 1-2 可知，四位二进制数 0000～1111 正好可以完全表示十六进制数的数码 0～9、A～F，所以二进制数转换为十六进制数可采用“四合一”的方法进行转换。具体规则是从低位到高位，每四位组成一组，不足四位时，在高位用 0 补齐，把每一组二进制数用一位等值的十六进制数表示，即可得到转换后的十六进制数。

例如：1011010101110B = 16AEH

0001 0110 1010 1110 B
1 6 A E H

② 十六进制转换为二进制。

十六进制数转换为二进制数可采用“一拆四”的方法进行转换。具体规则是把一位十六进制数写成等值的四位二进制数即可得到转换后的十六进制数。

例如：B4F7H = 1011010011110111B

B　4　F　7　H
1011 0100 1111 0111　B

1.1.3　数据在内存中的存储

1. 内存

内存是由一系列电子线路单元组成，每个电子线路单元有 2 种稳定的状态，分别表示二进制的 0 或 1，称之为位（bit），位是组成内存的最小单位。如图 1.2（a）所示。

（1）内存中的数据单位

① 字节（Byte）。一个字节由 8 个连续的二进制位组成，每个二进制位有一个编号，其编号顺序按位自右至左为 0～7，也可称为第 0 位、第 1 位…第 7 位，如图 1.2（b）所示。字节是数据存储的最小单位。

② 字（Word）。连续的两个字节称为字，如图 1.2（c）所示。连续的两个字称为双字，如图 1.2（d）所示。连续的两个双字称为 4 字，如图 1.2（e）所示。

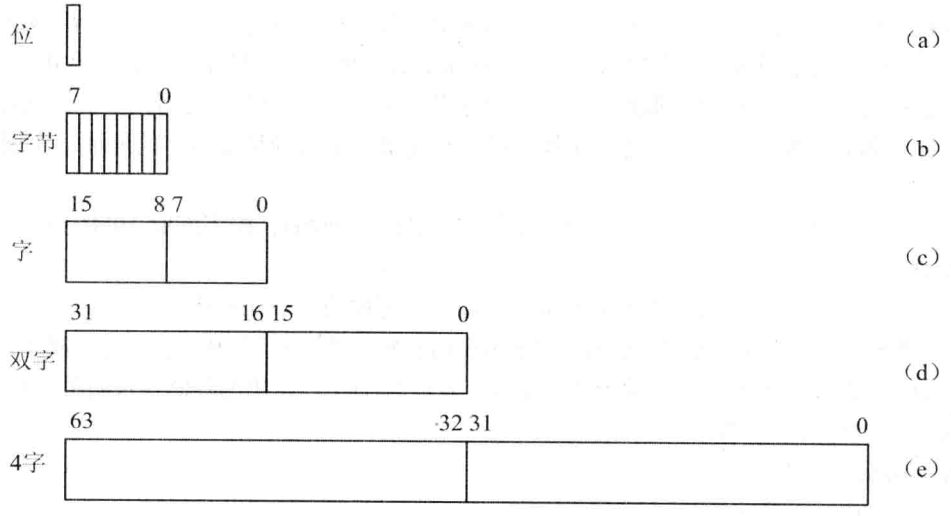


图 1.2　数据的存储形式示意图

③ 字长。中央处理器在单位时间内能一次处理的二进制数的位数叫字长。所以能处理字长为 8 位数据的 CPU 通常就叫 8 位 CPU，32 位 CPU 可在同一时间内处理字长为 32 位的二进制数据。字长是计算机性能的一个重要指标，字长越大计算机处理数据的速度就越快。早期计算机字长一般为 8 位和 16 位，目前大多数为 32 位或 64 位。

（2）内存的组织形式

计算机对数据的存储并不是杂乱无章的，有一套完整的存储体系结构，以保证数据的完整性和安全性，这一整套体系就是内存的组织形式。

① 地址。内存管理时，为了正确地存取信息，对每个字节都赋予了唯一的编号，该编号称为地址。地址从 0 开始编号，按照加 1 递增的顺序编址。在计算机中用二进制表示地址，为了书写

方便常用十六进制表示。存储器单元编址如图 1.3 所示。

② 存储单元的内容。在存储器中，存放在存储器单元中的数据称为该单元的内容，存储单元的地址与内容的关系如图 1.4 所示。图中，地址 2006H 单元的内容为 FBH；2007H 单元的内容为 10H。

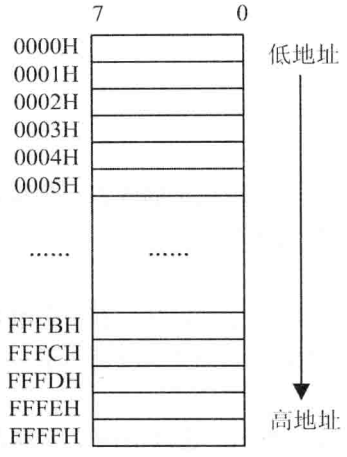


图 1.3 存储器单元编址示意图

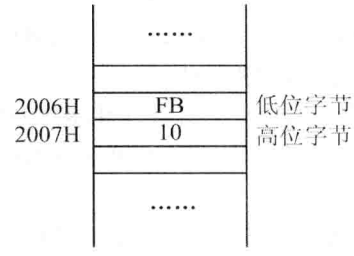


图 1.4 存储单元内容示意图

根据地址可以存取字节、字、双字、连续的多个字节。按字节操作时，只要指出该字节的操作地址，就可以实现字节操作。字是由 2 个字节组成，其存储顺序是低字节内容（第 0～7 位为低位字节，也称为低 8 位）存储在低地址单元，高字节内容（第 8～15 位为高位字节，也称为高 8 位）存储在相邻的高地址单元。在进行字操作时，规定低字节的地址是该字的地址，实现字的存取操作。

例如，按字节读取图 1.4 中所示的 FBH 时，其地址为 2006H；按字读取 10FBH 时，其地址也为 2006H。

由此可见，同一个地址即可视为字节单元地址，又可视为字单元地址。

③ 存储器的容量。存储器字节数的总和表示存储器的容量。在计算机中，为方便起见，存储器容量以 $2^{10}B = 1024B$ 为基本单位，称为 1K 字节，表示为 1KB。常用的存储容量的单位有：KB、MB、GB、TB，其大小分别为：

$1KB = 1024B = 2^{10}B$

$1MB = 1024KB = 2^{20}B$

$1GB = 1024MB = 2^{30}B$

$1TB = 1024GB = 2^{40}B$

2. 数据在内存中的存储

数据在内存中的存储形式称为机器码，机器码所表示的实际值称为真值。

（1）有符号整数的存储

下面以在内存中占两个字节的整数为例来介绍有符号整数在内存中的存储。当存储有符号数时，2 个字节的最高位为符号位（1 表示负数，0 表示非负数），其余位是数据位，如图 1.5 所示。

计算机中有符号整数的存储是以补码形式存储的。一个整数有以下 3 种编码。

① 原码。原码是符号位数码化了的二进制。十进制整数数码化为原码的方法是首先把十进制

整数转换成二进制，然后将数据位在高位用 0 补足 15 位，最高位添上符号位。

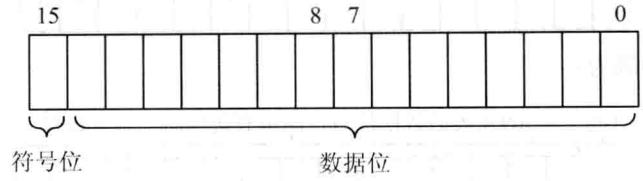
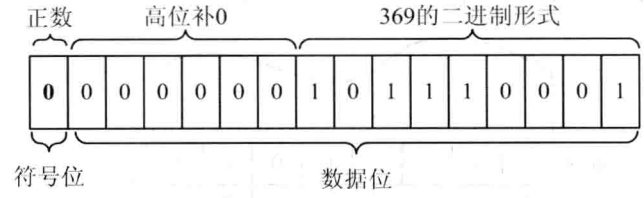


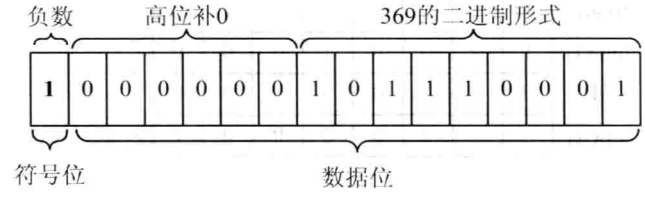
图 1.5　有符号整数存储形式

例如：

十进制数 369 的原码为：



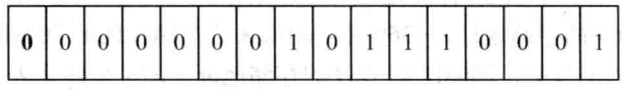
十进制数−369 的原码为：



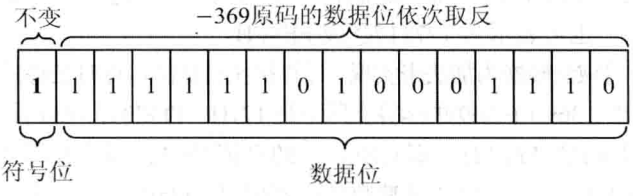
② 反码。对正数而言，原码即为反码；对负数而言，反码是将原码中除符号位以外的其余位依次取反，即将 0 变成 1，将 1 变成 0。

例如：

十进制数 369 的反码为（同原码）：



十进制数−369 的反码为：



③ 补码。对正数而言，原码即为补码；对负数而言，补码是在反码的基础上加 1。在求补码过程中符号位不发生变化，当数据位的最高位有进位时，舍弃进位。

例如：

十进制 369 的补码为（同原码）：