High Level Language Programming

高级语言程序设计

（C语言描述）

陆黎明　朱媛媛　蒋　培　编著

科学出版社

高级语言程序设计
（C语言描述）

陆黎明　朱媛媛　蒋　培　编著

科学出版社
北京

内 容 简 介

本书以目前流行的 C 语言为例，全面阐述了高级语言程序设计的基本概念、基本方法和基本技术。主要内容包括 C 程序设计基础，数据类型、运算符和表达式，结构化程序设计，数组、指针，函数，结构体类型、文件等。

本书强调程序设计方法的教学，通过大量具有趣味性和实用性的例题来说明 C 语言中语法的应用，以及程序设计的概念、方法和技巧，并对例题做了详细的分析，富有启发性；将初学者较难掌握的指针数据类型提前到数组这一章节，使学生有较多的时间来理解和掌握它的应用；所配的练习题有针对性，贴近生活，能够激发学生学习的兴趣和积极性；结构合理，重点突出，难点分散，图文并茂，格式规范，有利于学生学习 C 语言和培养良好的程序设计风格及习惯。

本书可作为各类高等学校本科、高职高专、成人教育的教材，也可作为计算机等级考试（二级 C）的参考书和自学教材。

图书在版编目(CIP)数据

高级语言程序设计：C 语言描述/陆黎明，朱媛媛，蒋培编著. —北京：科学出版社，2013.1

ISBN 978-7-03-036505-7

Ⅰ.①高… Ⅱ.①陆… ②朱… ③蒋… Ⅲ.①C 语言-程序设计 Ⅳ.①TP312

中国版本图书馆 CIP 数据核字(2013) 第 012662 号

责任编辑：贾瑞娜 张丽花 / 责任校对：鲁 素

责任印制：阎 磊 / 封面设计：陈 敬

科学出版社 出版

北京东黄城根北街 16 号

邮政编码：100717

http://www.sciencep.com

北京市文林印务有限公司印刷

科学出版社发行 各地新华书店经销

*

2013 年 1 月第 一 版 开本：720×1000 B5

2013 年 1 月第一次印刷 印张：14 3/4

字数：335 000

定价：32.00 元

(如有印装质量问题，我社负责调换)

前　言

高级语言程序设计是高等学校计算机专业重要的基础课程，也是理工科各相关专业的基础课程。学习本课程，不仅要求学生掌握一门高级程序设计语言的知识，更重要的是要让学生掌握程序设计的思想和方法，培养学生程序设计语言的应用能力和问题的求解能力。一本好的教材则是完成这一目标的第一步。但多年的教学实践暴露出一个问题，相当数量的学生仅仅掌握了一些高级语言的语法知识，普遍不清楚这些语法知识的作用和使用场合，也没有掌握基本的程序设计方法和技巧，导致问题的求解能力和编程能力普遍较弱。本教材就是要改变目前“重语言语法轻应用设计”的现状，树立“将高级程序设计语言作为程序设计的工具，变语言语法知识的传授为对问题的求解能力和语言的应用能力的培养，突出程序设计方法的教学”这一教学理念，为培养软件设计人员打下良好的基础。

本书具有以下几个特点：

（1）在内容的选择上，考虑到C语言使用的广泛性，它既能编写应用程序又能编写系统软件，所以选择C语言来阐述高级语言程序设计的基本概念、基本方法和基本技术。但C语言功能丰富、使用灵活，也给程序设计的初学者带来了困难。本教材不刻意追求C语言语法知识的大而全，对不是主要的语法知识（如主函数参数、位运算、共用体、枚举类型、带参数的宏定义和条件编译等）不作介绍，对较少使用又较难理解的语法知识（如指向数组的指针、指针数组、二级指针等）只作简要介绍。鉴于目前相当多的专业把C语言程序设计安排在第一学期，学生缺乏程序设计必要的基础知识，本书在第1章中对于数在计算机内的表示形式以及程序设计和算法等概念作了简要的介绍。

（2）考虑到指针类型及其应用是C语言中的难点，也是后续课程（如数据结构）的重点，本教材不像有的教材那样，最后集中讲解指针类型及相关内容，而是对这一难点进行分解，将指针类型提前到数组这一章，使学生能尽早理解指针概念和掌握它的基本用法。而在后续章节中对指针的运用进行更详细的描述，如指针法访问数组元素、字符指针与字符数组、数组形参、返回指针值的函数、指向函数的指针等，目的是使学生能有比较长的时间来学习指针，由简到难掌握指针的应用。同时，这样使教材体系很好地反映了知识点的内在联系，增加了指针应用的机会，降低了指针学习的难度。另外，递归函数执行过程的理解是C语言中的又一个难点，本教材通过独特的图示展示了递归函数执行时形参的变化过程，以帮助学生清楚地理解递归函数的执行过程。

（3）本书不仅着眼于C语言语法知识的讲解，更注重说明该语法在程序设计中的应用意义，不对那些没有实际意义的语法现象进行罗列，如（++a）+（++a）这样的表达式。本教材通过恰当的举例，着重说明了自增自减运算符、逻辑运算符的短路求值规则、指针变量作数组名、返回指针值的函数、指向函数的指针、静态局部变量、外部函数等语法知识的应用。另外，一般的教材都将typedef放到最后的章节才介绍，许多学生学完了也不知道它的作用。本教材把它提前到数组这一章讲解，通过使用typedef来介绍二维数组的本质是一维数组、指向数组的指针、指针数组、指向指针的指针、指向函数的指针等语法知识，不但概念清晰，而且也有利于学生的理解。

（4）本书特别注重程序设计方法和技巧的教学，除了介绍最基本的累加、求最大最小等程序设计方法外，还介绍了常用的顺推法、倒推法、迭代法、穷举法、回溯法等程序设计方法，排序和查找等算法，以及巧用下标、状态变量使用等程序设计技巧。本教材列举了大量具有趣味性和实用性的程序设计例题（书中将仅仅说明语言语法的举例称为示例，以示区别），在举例时不是先给出程序再去解释程序的内容，而是先分析人脑解决此问题的过程，从中找出解决问题的方法，再编写程序，并对已有的方法提出改进的可能，从而启发学生的思维，旨在培养学生勤于思考的习惯。本教材所选配的练习题有针对性，贴近生活，能够激发学生学习的兴趣和积极性。

本书例题全部在Dev-C++ 4.9.9.0集成开发环境下调试通过，当然也可在Visual C++ 6.0等集成开发环境下调试编译这些程序，读者可根据自己的具体情况选用。

本书是上海市教委重点课程建设项目“高级语言程序设计（C语言描述）”的建设成果，参加本书大纲讨论和部分编写工作的还有：徐晓钟、严忠林、王笑梅、王爱华、郑晓妹等。本书在编写过程中得到了上海师范大学副校长高建华教授，上海市高等学校计算机等级考试专家、华东理工大学顾春华教授的悉心指导。初稿完成后，江苏大学的赵跃华教授、上海大学的陈章进教授进行了认真的审阅，提出了许多宝贵意见。另外，科学出版社的编辑进行了认真负责的编辑工作，保证了本书的顺利出版。在此一并向他们表示衷心的感谢。

本书结构合理，重点突出，难点分散，图文并茂，格式规范，有利于学生的学习和培养良好的程序设计风格和习惯。本书适合作为各类高等学校本科、高职高专、成人教育的教材，也可作为计算机等级考试（二级C）的参考书和自学教材。

由于编者水平有限，虽然力求精准，但疏漏与不足之处仍在所难免，敬请专家和读者指正。

编　者

2013年1月于上海

目　录

第 1 章　程序设计基础

1.1　数在计算机内的表示形式

数是计算机程序处理的基本对象，了解数在计算机内的表示形式是学习程序设计的前提。

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，十进制计数系统是在 8 世纪由阿拉伯数学家发明的。以下两个概念在讨论进位计数制时很重要。

（1）基数。基数就是进位计数制中允许使用的不同基本符号的个数。例如，十进制共有 10 个基本符号（0、1、2、3、4、5、6、7、8、9），其基数是 10；二进制共有 2 个基本符号（0、1），其基数是 2。

（2）权值。权值是进位计数制中的一种因子，权值的概念可用以下实例来说明。十进制数 $3656.83=3\times10^3+6\times10^2+5\times10^1+6\times10^0+8\times10^{-1}+3\times10^{-2}$，在这个数中，有些相同的数字由于所处位置不同，因而所代表的数值大小也不同，各位数字所代表的数值大小由权值来决定。在这个十进制数中，从左到右各位数字的权值分别为 10^3、10^2、10^1、10^0、10^{-1}、10^{-2}，它们都是 10（这个数的基数）的整数次幂。十进制数的特点是“逢 10 进 1”。

因此，任意进制的数都可以表示为它的各位数字与权值乘积之和。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 $p=d_{m-1}d_{m-2}\cdots d_1d_0d_{-1}\cdots d_{-n}$，它可表示为

$$p=d_{m-1}\times r^{m-1}+d_{m-2}\times r^{m-2}+\cdots+d_1\times r^1+d_0\times r^0+d_{-1}\times r^{-1}+\cdots+d_{-n}\times r^{-n} \tag{1.1}$$

习惯上将上述 r 进制数的表示式中幂的指数写成十进制数的形式。如果对式（1.1）（它是一个多项式）中的幂运算和乘法运算都按照十进制的法则进行，所得到的结果就是该 r 进制数 p 的十进制数值。因此，将 r 进制数转换成十进制数是非常方便的。r 进制数的特点是“逢 r 进 1”。

人们在日常生活中经常使用十进制数，但在表示时间时，用十二进制（或二十四进制）表示小时，用六十进制表示分和秒。在计算机内部，通常采用二进制数，它只有 2

个基本符号，即0和1，特点是“逢2进1”。但二进制数的缺点是不易记忆和书写，所以人们又提出了八进制数和十六进制数。八进制数有8个基本符号，即0、1、…、7，它们是从十进制数中借来的。十六进制数有16个基本符号，除了从十进制数中借来的0、1、…、9外，还加上6个英文字母A、B、C、D、E、F。

为了避免各种进制数在使用时产生混淆，在给出一个数时，同时应指明它的进制，通常用下标10、2、8、16（或字母D、B、O、H）分别表示十进制、二进制、八进制、十六进制。例如，$(586)_{10}$、$(11011)_2$、$(356)_8$、$(4AF)_{16}$或$(586)_D$、$(11011)_B$、$(356)_O$、$(4AF)_H$等。4种进制的对应关系如表1.1所示。

表1.1　4种进制数的对应关系

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数的原因主要有两个：①表示容易，二进制数中的0和1可以很方便地用晶体管的两个稳定状态（导通和截止）来表示；②运算简单，二进制数的加法和乘法各只有3条运算规则，即0+0=0、0+1=1、1+1=10和0×0=0、0×1=0、1×1=1。而十进制数有10个基本符号，要用10种状态才能表示，实现困难。另外，十进制数的运算规则也比较复杂，这样也增加了实现的困难程度。

这里要特别说明的是，计算机内部采用二进制数，普通应用采用十进制数，这样就需要在二进制和十进制之间进行相互转换，这种转换相对来说不是很方便。为了便于转换，同时方便记忆和书写，所以提出了八进制数和十六进制数。八进制数和十六进制数

是给专业人员使用的，不是给普通用户使用的。

1.1.2　数制转换

1. 十进制数与 r 进制数之间的相互转换

1）r 进制数转换为十进制数

从式（1.1）可知，r 进制数（r 可以为二、八或十六）转换为十进制数的转换规则是：各位数字与相应权值的乘积之和，即为转换结果。例如：

$(10110.1)_2=2^4+2^2+2^1+2^{-1}=(22.5)_{10}$

$(456.45)_8=4\times8^2+5\times8^1+6\times8^0+4\times8^{-1}+5\times8^{-2}=(302.578125)_{10}$

$(2AF.48)_{16}=2\times16^2+A\times16^1+F\times16^0+4\times16^{-1}+8\times16^{-2}$

$=2\times16^2+10\times16+15\times1+4\times16^{-1}+8\times16^{-2}=(687.28125)_{10}$

2）十进制数转换为 r 进制数

（1）十进制整数转换为 r 进制整数的转换规则是：“除基数取余”，即十进制数反复地除以基数，并记下每次得到的余数，直到商是 0 为止，将所得余数按从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：

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

可得 $(83)_{10}=(1010011)_2$

（2）十进制小数转换为 r 进制小数的转换规则是：“乘基数取整”，即十进制小数乘以基数，将乘积的整数部分取出来，小数部分再乘以基数，重复上述过程，直到乘积的小数部分为 0 或满足转换精度要求为止。将每次取得的整数按从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：

```
  0.8125
×      2
1  1.625
×      2
1  1.25
×      2
0  0.5
×      2
1  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。但要特别注意的是，有些十进制小数不能完全精确地转换为对应的二进制小数，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。例如：十进制小数0.1对应的二进制小数是一个无限循环小数，即

$(0.1)_{10}=(0.0\underline{0011}\,\underline{0011}\,\underline{0011}\,\underline{0011}\cdots)_2\approx(0.0001100110011010)_2$（取16位小数）

（3）一个十进制数既有整数部分，又有小数部分，将其转换为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连接起来即为转换结果。例如：将$(124.625)_{10}$转换为二进制数，因为$(124)_{10}=(1111100)_2$，$(0.625)_{10}=(0.101)_2$，所以$(124.625)_{10}=(1111100.101)_2$。

2. 二进制数与八进制数、十六进制数之间的相互转换

由于二进制数与八进制数、十六进制数之间的特殊关系（8和16都是2的整数次幂，即$8=2^3$，$16=2^4$），所以使得二进制数与八进制数、十六进制数之间的相互转换非常简单。

1）二进制数与八进制数之间的相互转换

二进制数转换为八进制数的转换规则是："三位并一位"，即以小数点为基准，整数部分从右到左每三位一组，最左一组不足三位时前面补0；小数部分从左到右每三位一组，最右一组不足三位时后面补0，然后，每组用等值的八进制数字代替即可。例如：

$(10010001.0011)_2=(\underline{10}\,\underline{010}\,\underline{001}.\underline{001}\,1)_2=(\underline{010}\,\underline{010}\,\underline{001}.\underline{001}\,\underline{100})_2=(221.14)_8$

八进制数转换为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即可。注意，0也要写成三位二进制数000。例如：

$(506.36)_8=(\underline{101}\,\underline{000}\,\underline{110}.\underline{011}\,\underline{110})_2=(101000110.01111)_2$

2）二进制数与十六进制数之间的相互转换

只要把二进制数与八进制数之间的相互转换时用到的"三位并一位"和"一位拆三位"改为"四位并一位"和"一位拆四位"，即可实现二进制数与十六进制数之间的相互转换。例如：

$(10101001.0011)_2=(\underline{1010}\,\underline{1001}.\underline{0011})_2=(A9.3)_{16}$

$(506.36)_{16}=(\underline{0101}\,\underline{0000}\,\underline{0110}.\underline{0011}\,\underline{0110})_2=(10100000110.0011011)_2$

1.1.3　码制

在计算机内部，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。计算机内部最常用的信息单位是字节（byte，1byte=8bit），常见的PC机内存储器一个存储单元的大小（即存储容量的基本单位）是1字节。由于PC机的内存储器有许许多多的存储单元，用字节来描述存储容量的大小显得比较烦琐，其他常用的单位还有千字节（KB）、兆字节（MB）和吉字节

（GB），1KB=2^{10}bytes=1024bytes，1MB=2^{10}KB=1024KB，1GB=2^{10}MB=1024MB。目前PC 机内存储器的存储容量普遍达到了 4GB。

1. 机器数

在计算机内部表示数，要考虑数的长度、符号和小数点的表示等问题。把数本身（指数值部分）以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。机器数有以下几个特点：

（1）有固定的位数。在计算机内部，所能存储的二进制数的长度（最大位数）是固定的，通常的位数有 8 位、16 位、32 位、64 位（即 1 字节、2 字节、4 字节、8 字节）等。

（2）数的符号数字化。一个数有符号（即正号+或负号–），而计算机的内存储器是由二进制位构成的，不能直接表示正负号，必须要进行变换，即数字化。通常用“0”表示正，用“1”表示负，并将相应的位称为符号位。

（3）依靠格式上的约定表示小数点的位置。

机器数的表示方法最常用的是原码、补码、反码等。下面讨论二进制整数的机器数形式。

2. 原码

一个数的原码是：最高位（最左边一位）是符号位（“0”表示正号，“1”表示负号），其余各位给出数的绝对值的机器数表示方法。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 位，则

$[+83]_{原}$=01010011　　$[-83]_{原}$=11010011

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

原码的优点是简单直观，缺点是加减运算较复杂。例如，两个原码表示的机器数做加法运算：

10001101+00001011=?　　即–13+11=?

表面上看是做加法，事实上由于两数异号要做减法，还要根据数的绝对值决定到底是哪个数减哪个数，最后还要决定减法结果的符号。

3. 补码

引进补码的目的主要是为了把减法运算化为加法运算，从而降低计算机 CPU 内部运算器的复杂度，降低制造成本。减法运算如何能够化为加法运算呢？答案是：任何有模的计量器，均可化减法运算为加法运算。“模”是指一个计量系统的计数范围。例如，有一块手表显示 5 点 15 分，但是它快了 5 分钟，理应做减法（即分针倒退 5 小格），使得它显示正确的时间 5 点 10 分。事实上，做加法（即分针前进 55 小格）也可达到同样的

目的。因为做加法 15+55 时，结果 70 超出了手表分针的表示范围，分针停在 10 分上，而 60 被自动舍去，这里的 60 就是模。

由于机器数的位数是固定的，也是有限的，即计算机内的运算都是有模运算。模在计算机中是表示不出来的，若运算结果超出能表示的数值范围，就会自动舍去溢出量（模）。由此，引进了补码的概念。假设机器数的位数是 n，模为 2^n，数 x 的补码定义为

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

例如，假设机器数的位数是 8 位，则

$[+83]_{补}$=01010011 $[-83]_{补}=2^n+x=2^8+(-1010011)$=100000000−1010011=10101101

$[+127]_{补}$=01111111 $[-127]_{补}=2^n+x=2^8+(-1111111)$=100000000−1111111=10000001

$[+0]_{补}=[-0]_{补}$=00000000 $[-128]_{补}=2^n+x=2^8+(-10000000)$=100000000−10000000= 10000000

表 1.2　补码的位数与可表示的数的范围对应关系

补码的位数	可表示的数的范围
8	−128～+127
16	−32768～+32767
32	−2147483648～+2147483647

原码不能表示−128，这是因为 0 的原码表示不唯一，有“正零”和“负零”之分，而 0 的补码表示是唯一的。所以，10000000 在原码中表示“负零”，而在补码中可表示−128。补码的位数与可表示的数的范围对应关系如表 1.2 所示。

同一个数的原码与补码的相互转换规则如下。

（1）原码转换为补码：正数的补码即为原码，负数的补码在它原码的基础上符号位不变，其余各位取反后末位加 1。原码 10…0 是个例外，不适合本转换规则。

（2）补码转换为原码：正数的原码即为补码，负数的原码在它补码的基础上符号位不变，其余各位取反后末位加 1。补码 10…0 是个例外，不适合本转换规则。

数用补码表示后，就能将减法运算化为加法运算。假设机器数的位数是 8，则

$$[83-127]_{补}=[+83]_{补}+[-127]_{补}=01010011+10000001=11010100=[-44]_{补}$$

根据补码转原码规则，得$[-44]_{原}$=10101100，运算结果完全正确。再如：

$$[83-2]_{补}=[+83]_{补}+[-2]_{补}=01010011+11111110=\underline{1}01010001=[+81]_{补}$$

因为模为 2^8，上式中的有下划线的 1 会自动舍去，正数的补码就是原码，得$[+81]_{原}$=01010001，运算结果完全正确。

4. *反码*

假设机器数的位数是 n，数 x 的反码定义为

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

反码很少直接用于计算，而是作为计算补码的过渡手段。

5. *移码*

假设机器数的位数是 n，数 x 的移码定义为：

$$[x]_{移}=2^{n-1}-1+x,\quad -2^{n-1}\leqslant x<2^{n-1}$$

无论 x 为正数还是负数，其移码均加 2^{n-1}–1。移码在计算机中主要用于表示浮点数中的阶码。

1.1.4　定点数和浮点数

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。定点整数的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，$(83)_{10}=(1010011)_2$，若定点数长度为 8 位（最高位为符号位），则该数的机内表示为 01010011。

（2）定点小数。定点小数的小数点位置固定在符号位与最高数值位之间，用来表示纯小数。例如，$(-0.8125)_{10}=(-0.1101)_2$，若定点数长度仍为 8 位，则该数的机内表示为 11101000。

定点数的运算规则比较简单，但不适合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数表示，则小数点的位置是不固定的，可以浮动。为了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 2.718，它可以等价地表示为多种指数形式，如 2.718×10^{0}、0.2718×10^{1}、0.02718×10^{2}、27.18×10^{-1} 或 271.8×10^{-2} 等。不难看出，小数点的位置可以向左或向右浮动（表示数缩小或扩大若干倍），只要在小数点位置浮动的同时增减指数的值，即可保证数的值不变。任何一个二进制数 N 都可以表示为下面的指数形式：

$$N=2^{e}\times M$$