21世纪大学计算机基础规划教材

C语言程序设计

王煜　王苗　吴玉霞　徐建民　编著

- 本书文字严谨、流畅，例题典型，文档规范。
- 在本书编写过程中，遵循了知识讲授和能力训练并重的原则；遵循了软件工程的原则。
- 本书可作为高等院校学生学习C语言的教材，也可作为程序设计人员的参考书。

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

21世纪大学计算机基础规划教材

C语言程序设计

王煜　王苗　吴玉霞　徐建民　编著

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

内 容 简 介

本书是高等学校计算机科学与技术及相关专业C程序设计课程教材，内容包括程序设计基础、简单的数据类型和运算符及表达式、输入输出及顺序结构程序设计、选择结构程序设计、循环结构程序设计、编译预处理、函数、指针、数组、结构体、共用体和枚举类型、位运算、文件。

本书文字严谨、流畅，例题丰富，文档规范，注重程序设计技能训练，可作为高等院校学生学习C语言的教材，也可作为程序设计人员的参考书。

图书在版编目（CIP）数据

C语言程序设计/王煜等编著. —北京：中国铁道出版社，2005.1

（21世纪大学计算机基础规划教材）

ISBN 7-113-06361-6

Ⅰ.C… Ⅱ.王… 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(2005)第005648号

书　　名：C语言程序设计

作　　者：王　煜　王　苗　吴玉霞　徐建民

出版发行：中国铁道出版社（100054，北京市宣武区右安门西街8号）

策划编辑：严晓舟　姜淑静

责任编辑：苏　茜　秦绪好　张雅静

封面制作：白　雪

印　　刷：河北省遵化市胶印厂

开　　本：787×1092　1/16　印张：15.25　字数：365千

版　　本：2005年2月第1版　2005年2月第1次印刷

印　　数：1～5000册

书　　号：ISBN7-113-06361-6/TP·1417

定　　价：22.00元

前　言

高级程序设计语言是计算机科学与技术及其相关专业重要的必修课之一，也是数据结构、操作系统等课程的选修课。C 语言于 20 世纪 70 年代由贝尔实验室在 B 语言的基础上提出，并成功地用来编写了 UNIX 操作系统。由于 C 语言的强大功能和各方面的优点，其提出后迅速传播，成为计算机科学与技术及相关专业首选的高级程序设计语言之一。

本书根据高等院校计算机科学与技术专业 C 程序设计课程教学大纲，在作者十几年讲授 C 程序设计的基础上编写而成。考虑到 C 语言程序设计既是数据结构、操作系统等课程的选修课，同时也是计算机科学与技术专业学生必备的技能之一，因此在编写本书过程中，作者遵循了知识讲授和能力训练并重的原则：在讲清基本知识的基础上，注意了例题的选择，适当增加了例题和习题的数量和类型。遵循了软件工程的原则：根据软件模块高内聚的原则注意将相近的内容尽可能安排在一起；根据模块大小适中的原则注意每一章、每一节乃至每一段落都大小适当；根据清晰第一的原则注意了程序文档的规范。

本书内容包括 12 章，分别为程序设计基础、简单的数据类型和运算符及表达式、输入输出及顺序结构程序设计、选择结构程序设计、循环结构程序设计、编译预处理、函数、指针、数组、结构体、共用体和枚举类型、位运算和文件。其中 7、8、9、10 章由王煜编写，2、3、4、5 章由王苗编写，1、6、11、12 章由徐建民和吴玉霞共同编写。最后由徐建民完成了第一遍统稿，在修改的基础上由王煜进行了第二遍统稿，最后的定稿由王煜和徐建民共同完成。

河北大学的张晓丽、刘振鹏二位教授在百忙中审阅了书稿，并提出了不少中肯的意见，在此对他们的帮助表示衷心感谢。杜瑞忠老师对本书的编写给了很多帮助，研究生刘进波、邵艳华、柴变芳和赵爽同学帮助收集了部分资料和录入了部分文字，在此一并表示感谢。

编写出一本优秀的教材是一件非常不容易的工作，涉及到诸多的因素。由于编者水平、精力所限，再加上时间紧迫，书中肯定会有不足甚至谬误，恳请读者批评指正。

编　者

2004 年 12 月

目 录

第 1 章　程序设计基础

计算机的发明引起了社会生产力的迅猛发展，过去由人工来进行的大量繁琐的工作，现在可以通过计算机迅速而便捷地来实现。目前，计算机应用领域日益广泛，除了可以进行复杂的科学计算，还能对文字、图像、声音等进行识别和处理。

数字、汉字、图像和声音的表象千差万别，但对于计算机而言，它们都被称为数据或信息。计算机的基本功能就是对数据进行处理。

程序设计是用来沟通算法与计算机的桥梁；程序是编程者写的、计算机能够理解并执行的一些命令的集合，是解决问题的具体步骤在计算机中的实现。计算机程序是指为让计算机完成特定的任务而设计的指令序列。

C 语言是国际流行的高级程序设计语言。它既可以作为系统描述语言用来编写系统软件，也可用来编写应用软件。

1-1　计算机内的数据表示

在计算机科学中，数据是指所有能被计算机所识别、存储和处理的符号的总称。数据在计算机内部是以二进制形式表示的。

1-1-1　数制及其转换

1. 常用计数方法

进位计数制是常用的计数方法。进位计数制采用有限个数码来表示数据，数据中各个数字所处的位置决定它的权值，每个数字所表示的数值就等于该数字本身乘以它的位置所代表的权值。

日常生活中，最常用的是十进制计数法，例如：

$(1234.56)_{10}=1\times10^3+2\times10^2+3\times10^1+4\times10^0+5\times10^{-1}+6\times10^{-2}$

其各位的权值分别为：10^3、10^2、10^1、10^0、10^{-1}、10^{-2}。

十进制计数法是“逢十进一”的。对于一个十进制数，个位数的权值是 $1(10^0)$，那么个位数所能表示的最大数值为 $9\times10^0=9$，超过 9 就要向十位进位；十位数的权值是 $10(10^1)$，那么十位数所能表示的最大数值为 $9\times10^1=90$，依此类推。

在计算机内部，数据用二进制计数法表示。计算机内用电子器件的两种不同状态来表示数字信息，如用高电平表示 1，用低电平表示 0，因此对于计算机的物理实现而言，用二进制表示数据更方便。例如：

$(1001101.101)_2=1\times2^6+0\times2^5+0\times2^4+1\times2^3+1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$

其各位的权值分别为：2^6、2^5、2^4、2^3、2^2、2^1、2^0、2^{-1}、2^{-2}、2^{-3}。

二进制计数法是“逢二进一”的。二进制数的每一位或者为 0，或者为 1；超过 1 就要向高位进位。

采用二进制计数法表示数据的一个缺点是所需位数很多，容易出错，因此为了便于阅读和书写，在程序中常采用八进制数与十六进制数来代替二进制数。例如：

$(621)_8=6\times8^2+2\times8^1+1\times8^0$

其各位的权值分别为：8^2、8^1、8^0。

八进制计数法是“逢八进一”的。八进制计数法的数码为：0、1、2、3、4、5、6、7。这八个数码相当于十进制数的 0 到 7，接下来下一位数字就向高位进位，即$(10)_8$，相当于十进制数 8，那么$(11)_8$相当于十进制数 9，……

又如：$(8A1F)_{16}=8\times16^3+10\times16^2+1\times16^1+15\times16^0$

其各位的权值分别为：16^3、16^2、16^1、16^0。

十六进制计数法是“逢十六进一”的。十六进制计数法的数码为：0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F。十六进制中有 16 个数码，但从 10 开始到 15 就没有阿拉伯数字可用了，因此规定用字母 A(a)、B(b)、C(c)、D(d)、E(e)、F(f)来表示 10、11、12、13、14、15。

2. 数制转换

（1）任意进制转换为十进制

由 r 进制数转换为十进制数可按照如下公式进行多项式展开求和即可：

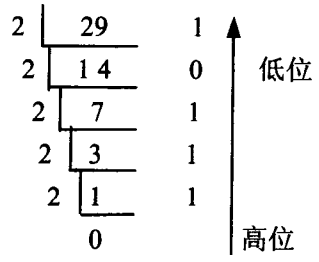
$K_nK_{n-1}\ldots K_1K_0.K_{-1}K_{-2}\ldots K_{-m}$

$=K_n\times r^n+K_{n-1}\times r^{n-1}+\ldots K_1\times r^1+K_0\times r^0+K_{-1}\times r^{-1}+K_{-2}\times r^{-2}+\ldots K_{-m}\times r^{-m}$

（2）十进制转换为任意进制

可以采用除基取余法将十进制整数转换为 r 进制整数：将十进制整数除以 r，得到商和余数，余数对应为 r 进制数低位的值；继续让商再除以 r，得到商和余数，……重复此操作，直至商为 0，如此得到的一系列的余数就是相应 r 进制数的各位数字，先得到的是低位，后得到的是高位。

例如，将$(29)_{10}$转换为二进制整数：



因此，$(29)_{10}=(11101)_2$

可采用乘基取整法将十进制小数转换为 r 进制小数：将十进制小数乘以 r，去掉乘积的整数部分，再将余下的纯小数乘以 r，重复此操作，直至乘积等于 0 或达到所需的精度为止，如此得到的一系列整数就是 r 进制小数的各位数字，先得到的是高位，后得到的是低位。

例如，将$(0.625)_{10}$转换为二进制小数：

0.625×2=1.25　　1　高位
0.25×2=0.5　　0
0.5×2=1　　1　低位

因此，$(0.625)_{10}=(0.101)_2$

由于整数和小数的转换方法截然不同，将十进制数转换为 r 进制数时，整数部分和小数部分要分开来进行转换。r 进制小数能精确地转换为十进制小数，但十进制小数通常不能精确地转换为 r 进制小数。

（3）二进制与八进制、十六进制之间的转换

对于一个二进制数，只要依次（整数部分由低位到高位，小数部分由高位到低位）将其每 3 位或 4 位分成一组，就可以直接转换为八进制数或十六进制数。

例如，二进制数$(1000101100010011.10111)_2$

若按 3 位一组划分，就可以直接写出其对应的八进制数：

1　000　101　100　010　011．101　11
1　0　5　4　2　3．5　6

即$(1000101100010011.10111)_2=(105723.56)_8$

若按 4 位一组划分，就可以直接写出其对应的十六进制数：

1000　1011　0001　0011．1011　1000
8　B　1　3．B　8

即$(1000101100010011.10111)_2=(8B13.B8)_{16}$

反之，将八进制数中的每一位用 3 位二进制数表示，将十六进制数中的每一位用 4 位二进制数表示，就能转换为对应的二进制数。

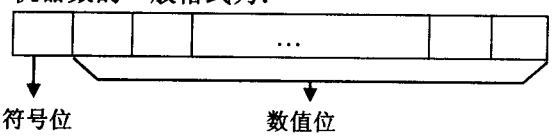
1-1-2　原码、反码及补码

在计算机中，“位”是数据的最小单位。计算机中的存储量就是以字节为单位来计算的，一个字节是 8 位二进制位。

在计算机内，不仅数值是用二进制数表示的，符号也是用二进制数表示的，一般规定：用 0 表示正号“＋”，用 1 表示负号“-”；符号位放在数值位之前。

一个数连同其符号在机器中的二进制表示形式称为机器数，它所代表的数值称为机器数的真值。

机器数的一般格式为：



为简便起见，下面讨论中用一个字节来表示带符号的数据。

在计算机中，带符号的数的表示方法有三种：原码、反码和补码。

1. 原码

原码表示法是符号位用 0 表示正数，用 1 表示负数，数值位表示数值本身。例如：

$[+27]_{原}=00011011$　　$[-27]_{原}=10011011$

在原码表示法中，0 的表示方法不惟一：

$[+0]_{原}=00000000$　　$[-0]_{原}=10000000$

2. 反码

反码表示法中正数与负数的表示方法不同，正数的反码与原码同形，如：

$[+27]_{反}=00011011$

负数的反码为：符号位仍为 1，数值位是对原码取反，如：

$[-27]_{反}=11100100$

在反码表示法中，0 的表示也不惟一：

$[+0]_{反}=00000000$　　$[-0]_{反}=11111111$

3. 补码

采用原码表示的数据进行加减运算很不方便，当两个加数的符号相同时，可以数值位相加，结果符号不变；当两个加数符号不同时，加法运算实际上要转换为减法进行；为了进行减法，应先判断两数的绝对值，让绝对值大的数减去绝对值小的数，运算结果的符号与绝对值大的数的符号相同。计算机实现上述过程比较复杂。

因此，在计算机中采用补码表示法来表示数据。

采用补码表示数据，能够简化设计与运算，可以将减法运算转化为加法运算。

下面以时钟为例说明补码的原理，假设当前时刻是 5 点，若问两个小时前是几点，那么可以将时针向前拨两小时，即 5-2=3，得到 3 点；还可以将时针向后拨 10 小时，即 5＋10=15=12＋3，从时钟上看，还是 3 点。

对于这种运算，5 减去 2，与 5 加上 10 能够达到同样的效果，二者殊途同归。这是因为时钟的刻度是以 12 为周期的，超过 12 就再从头开始计数。称 2 和 10 是互补的，即减去 2 就相当于加上 10。补码的运算原理与此类似。

在计算机中，以定长的存储单元来表示数据，所能表示数据的范围是有限的，超过它的表示范围后，高位就会溢出。

正数的补码与原码、反码同形，如：

$[+18]_{原}=[+18]_{反}=[+18]_{补}=00010010$

负数的补码为：符号位为 1，数值位等于原码的数值位取反，再加 1，或者说：

$[x]_{补}=[x]_{反}+1$；

如：$[-18]_{原}=10010010$

$[-18]_{反}=11101101$

$[-18]_{补}=11101110$

采用补码表示数据，使得正、负的关系转换成了一种纯数值的关系。补码形式的数据进行运算时，符号位和数值位一样参与运算，不必考虑它特别的含义（正、负）。

例如，计算-36＋58 的值。

$[-36]_{补}=11011100$　　$[+58]_{补}=00111010$

```
    11011100
  + 00111010
  ------------
溢出 ←[1] 00010110
```

显然，$(11011100)_2+(00111010)_2=(00010110)_2=(22)_{10}$，从而验证了：“-36+58=22”。

可见，采用补码表示的数据进行运算的规则是很简单的。而且，在补码表示法中，0 的表示是惟一的，假设用一个字节存放数据，规定：

$[0]_{补}=00000000$，　$[-128]_{补}=10000000$

1-2　算法及其表示

1-2-1　算法

为解决一个问题而采取的方法和步骤，称为算法。

解决任何问题都是需要一定的次序的。例如乐队演奏乐曲、教师按教学计划授课、工程师设计施工方案等等都必须按照一定的次序进行。

算法就是被精确定义的一组规则，规定先做什么，再做什么，以及判断某种情况下做哪种操作。

例如，下面是用自然语言表示的对三个数进行从小到大排序的算法。

（1）输入三个数 x、y、z；

（2）将 x 与 y 比较，若 x>y，交换 x 与 y 的值；

（3）将 x 与 z 比较，若 x>z，交换 x 与 z 的值；

（4）将 y 与 z 比较，若 y>z，交换 y 与 z 的值；

（5）输出此时的三个数 x、y、z。

1-2-2　算法的特性和目标

解决不同的问题，需要不同的策略，因而相应的算法也千变万化、繁简不同。但一个算法必须具有以下五个特性：

（1）确定性

算法中的每一步都必须有确切的含义，不允许存在二义性；对于相同的输入数据，必须有相同的输出结果。

例如：“将个子高的同学名单打印输出”，在这一描述中“个子高”是很不明确的说法，身高 170cm 以上是“个子高”，还是 180cm 以上是“个子高”？

（2）可行性

算法中的每一步操作都能通过可以实现的基本运算执行有限次来完成，并最终得到确定的结果。

例如：当 B=0 时，A/B 是不能有效执行的。

（3）有穷性

一个算法必须总是在执行有限个操作步骤和可以接受的时间内完成其执行过程。也就是

说，对于一个算法，要求其在时间和空间上均是有穷的。

例如：一个采集气象数据并加以处理进行天气预报的算法，如果为了得到第二天的天气情况要运算一个星期才能得出结果，显然就超出了可以接受的时间，起不到预报天气的作用。

（4）输入

一个算法有零个或多个输入数据。有些算法需要提供输入数据，有些算法则不需要。

例如：计算 1～100 的累计和，无须输入数据，而对若干个整数进行排序，却需要输入待排序的数据。

（5）输出

一个算法应该有 1 个或多个输出数据。执行算法的目的是为了求解，而“解”就是输出，因此没有输出的算法是毫无意义的。算法的输出数据往往与其输入数据存在某些特定的关系。

解决某一特定问题往往可以选择多种算法。设计一个好的算法对于正确、高效地解决问题有着重要的意义。一个好的算法应达到以下目标：

- 正确性：算法应该能够满足预先规定的功能和性能要求。
- 可读性：算法不仅仅是让计算机来执行的，更要让人来阅读，可读性好的算法有助于调试程序、发现和修改错误，使得日后对软件功能的扩展、维护易于实现。一个算法应当思路清晰、层次分明、简单明了。
- 健壮性：指算法能够对非法的输入做出合理的处理，而不是产生莫名其妙的结果。
- 高效率与低存储空间需求：指解决特定问题的算法的执行时间应尽量短，算法执行过程中需要的存储空间应尽量小。

1-2-3 算法的表示

算法可以使用各种不同的方法来描述。常见的算法表示方法有：自然语言、伪码、传统流程图、N-S 结构图等。

1. 用自然语言表示算法

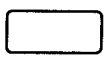
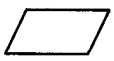
自然语言就是人们日常使用的语言，可以是中文、英文等。如前面对 3 个数进行从小到大排序的算法，就是用自然语言描述的。

用自然语言表示的算法简单、通俗易懂，但文字冗长，表达上不易准确，易有二义性。所以，一般不用自然语言描述算法。

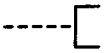
2. 用传统流程图表示算法

传统流程图是用规定的一组图形符号、流程线和文字说明来表示各种操作的算法表示方法。传统流程图常用的符号如表 1-1 所示。

表 1-1 传统流程图常用符号

符 号	符号名称	含 义
	起止框	表示算法的开始和结束
	输入/输出框	表示输入/输出操作

续上表

符 号	符号名称	含　义
	处理框	表示对框内的内容进行处理
	判断框	表示对框内的条件进行判断
	流程线	表示流程的方向
	连接点	表示两个具有同一标记的“连接点”应连接成一个点
	注释框	表示对流程图中某些框的操作做必要的补充说明

用传统流程图描述对 3 个数进行从小到大排序的算法如图 1-1 所示。

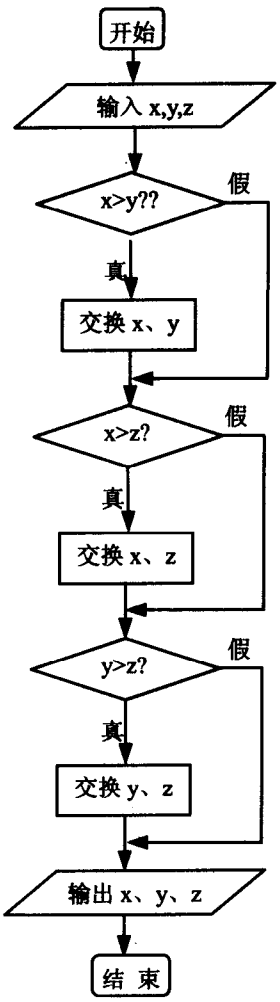


图 1-1　用传统流程图表示对 3 个数进行排序

用传统流程图表示算法直观形象，算法的逻辑流程一目了然，便于理解；但画起来比较麻烦，且由于允许使用流程线，使用者可以随心所欲，使流程可以任意转移，从而造成阅读和修改上的困难。

为克服上述弊病，提出了结构化的程序设计方法。在结构化的程序设计方法中，流程图只包括三种基本程序结构：

（1）顺序结构

在顺序结构中，要求顺序地执行按先后次序排列的每一个基本的处理单位。顺序结构的流程图如图 1-2 所示，该图表示先执行处理过程 A，然后再顺序执行处理过程 B。

（2）选择结构

在选择结构中，要根据判断条件的成立与否，选择执行不同的处理过程。选择结构的流程图如图 1-3 所示，当判断条件成立时，执行处理过程 A，否则执行处理过程 B。

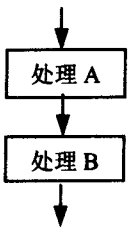


图 1-2　顺序结构

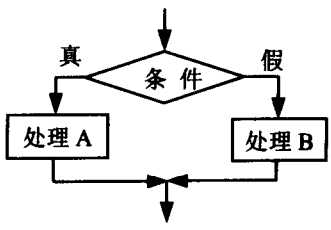


图 1-3　选择结构

（3）循环结构

循环结构是根据一定的条件，对某些语句重复执行的结构。被重复执行的部分称为循环体。