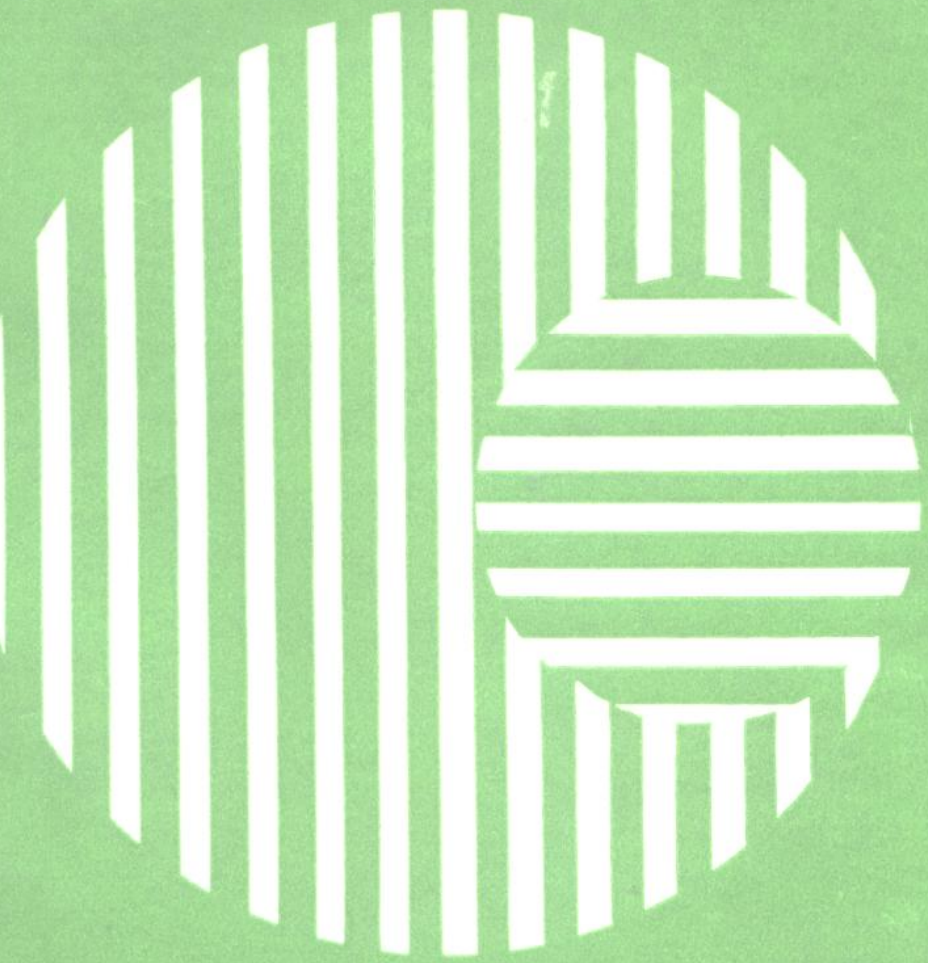
标准C语言程序设计



孟庆昌 刘振英 编著

学苑出版社

标准C语言程序设计

孟庆昌　刘振英　　编著

宇航出版社

（京）新登字 181 号

内 容 简 介

本书根据 ISO/IEC 公布的 C 程序设计语言的文本，结合教学、软件开发和水平考试等方面的需求，对标准 C 语言及其程序设计方法和应用进行了系统、全面而又实用的介绍。全书共分十七章，主要内容包括：程序设计概述；C 语言的常量和变量；基本数据类型及其转换；各种运算符的功用、优先级和结合性；表达式、语句与控制流；变量存储类；数组；函数；指针；结构、联合和枚举；预处理、输入/输出和库函数使用；水平考试辅导等。在附录中给出 C 语言的语法汇总。每章的最后都附有习题，供读者练习。

全书以通俗的语言由浅入深地介绍标准 C 语言的语法规则，以有代表性的实例突出重点地说明 C 语言的具体应用。本书可供高等院校计算机专业及其相关专业的师生用作教学参考书，也可供从事软件开发和应用的计算机工程技术人员作参考资料，还可作为全国计算机软件人员水平考试的辅导材料及有关人员的自学用书。

标准 C 语言程序设计

编著：孟庆昌
刘振英

责任编辑：廖寿琪　高丹平

*

宇航出版社出版发行

北京和平里滨河路 1 号　邮政编码 100013

各地新华书店经销

北京隆昌印刷厂印刷

*

开本：787×1092　1/16 印张：18.75 字数：470 千字

1993 年 4 月第 1 版第 1 次印刷　印数：1～6000 册

ISBN 7-80034-553-X/TP・033　定价：13.00 元

编 者 按

（代 序）

C语言是国际上广为流行的一种计算机高级程序设计语言，它几乎运行在所有类型的计算机上。我国自80年代中期利用C语言进行软件开发以来，由于众多专家、学者和工程技术人员的共同努力，已在许多领域中取得了可喜的应用成果，并培养了一大批人材。这对加速发展我国计算机事业、加强中外计算机技术与应用成果交流、尽快赶超世界先进科学技术水平，建设具有中国特色的四个现代化，无疑地是宝贵的财富。

宇航版《C语言及其应用》一书，四年来畅销不衰、先后印刷四次，该书已被不少高等院校选作教材；在近年来的计算机应用软件人员水平考试中，又作为主要参考书之一，成为广大读者的“好友”。

随着计算机事业的飞跃发展，1990年国际标准化组织ISO/IEC公布了国际标准C程序设计语言的文本；1991年中国也提出了国家标准程序设计语言C的征求意见稿。不论是国际的或国内的各种版本的C语言，要想在中国推广使用，都必须贯彻国际标准和国家标准，才能在各类计算机中充分发挥出C语言的功能，保证开发软件的质量，增强软件的可移植性。

《标准C语言程序设计》就是遵循C语言国际标准和国家标准，为提高全国计算机软件人员的水平及其解决实际问题的能力而组织编写的。现在已经录制了教学用录像片，供有关院校、部队、机关、科研单位、企事业部门及参加软件水平及资格考试的人员等使用。该书的作者之一孟庆昌同志从80年代初期就从事UNIX操作系统和C语言的分析、研究和教学工作，曾参加过“UNIX系统分析与改造”、“可移植操作系统国家标准”的制订等科研项目。先后作为主要编写人编著有《C语言程序设计》、《C语言及其应用》、《C语言初级教程》、《操作系统教程——UNIX系统V实例分析》等图书，后者还在台湾以繁体字版本发行。

《标准C语言程序设计》一书的特点是：文字通顺、内容通俗易懂、全书由浅入深、层次清晰、语法规则有条不紊，应用举例联系实际，其程序已在计算机上全部通过，可直接在开发中应用；特别是书中结合历年的软件水平考试实例，既有程序说明和程序，又有答案与分析，起到了画龙点睛的作用。这本书是《C语言初级教程》的读者继续深造的参考书，也是辅导软件人员参加水平考试时应当配备的教科书，又可作为即将出国从事软件开发的工程技术人员的随身工具书。

向读者推荐这本书，还因为它会使读者分清标准C与经典C的区别，掌握标准C语言的结构、语法规则、编程技巧等，相信能起到抛砖引玉的作用。愿致力于计算机软件开发的炎黄子孙们奋发图强，刻意追求，以自己的聪明才智和拼搏精神结出硕果，在21世纪的高科技领域中昂首于世界之林。

前　言

C语言是举世公认的优秀程序设计语言之一，近几年来它得到迅速推广和普及，其应用范围遍及软件开发的各个领域。每当评述C语言的巨大成功时，人们无不对它的简洁、高效、可移植性好等众多优点啧啧称赞。

回顾C语言的发展历史，它最初是伴随UNIX操作系统而诞生的，经过内部使用，对外推广，其功能进一步完善，成为通用的程序设计语言。在其发展过程中，出现了很多不同的版本。这些版本虽保持C语言的基本特色，但彼此间的差异妨碍了C语言的应用，也有损于C语言的形象。因而，对C语言的标准化工作受到世界各国有关专家的重视，1985年美国提出非正式标准草案（即ANSI标准草案），1989年ISO/IEC提出国际标准草案，1990年ISO/IEC的正式标准被公布于世（我们称之为标准C）。从此，C语言就有了统一的国际化的标准，这标志C语言进入新的时代。

1981年C语言进入中国。原北京大学二分校（现北京信息工程学院）率先把C语言作为高校学生的程序设计语言课程。在1984年召开的全国第一次C语言认识研讨会上，我们和有关同志曾就C语言的形式化描述等问题提出过一些建议并进行了讨论。随着UNIX系统在中国的推广和各种微机上C语言版本的使用，C语言以宏伟的气势迅速占领了软件工程、系统开发、事务处理及编程教学等领域的阵地，现在C语言已成为中国最主要的编程语言之一。

本书以ISO标准为基础，参考众多专家、读者、学生的反馈意见，较全面、系统、由浅入深地介绍了标准C语言及其程序设计方法和应用范例。全书共分十七章和三个附录，每章末尾给出一些习题，可供读者练习，以利对所学内容的巩固和贯通。

对于标准C语言与以往经典C语言有差异之处，书中都明确地予以说明。对标准C语言的语法和程序结构等，力求用通俗语言进行解释，对各部分的要点做了归纳，并举出大量浅显而又有代表性的实例，特别注意到对程序算法的说明。近二三年来，参加全国计算机软件人员水平考试的人数逐年递增，需要有一本按照国际标准、系统介绍C语言，并且有益于水平考试的书，这是促成编写本书的一个动力。为此在第十七章中通过对一些试题的分析，介绍解题时应注意的几个问题。

由于我们对C语言标准的理解尚很肤浅，加之水平有限，所以书中肯定存在不足以至某些错误，恳请广大读者批评指正。

由于标准C语言的个别功能还未在机器上得以实现（起码在我们所用的3B2和SUN机器上），对这些新特点就不便通过可运行的具体示例来加以说明。我们对此也感到遗憾。希望不久能见到完全支持标准C语言的编译。

在本书的编写过程中，得到宇航出版社廖寿琪同志的大力支持和帮助，他对本书的内容和结构提出了许多宝贵的建议。原高档微机协会秘书长徐国平先生把函授班上大量学员的反馈信息及时告诉编者，从而使本书注意克服有关问题。我们还得到过许多同志的关心和鼓励。在此向上述所有同志表示衷心感谢，并期望得到更多朋友的指导。

编　者

1992年6月

目　录

第一章 概 述

自从1946年第一台电子计算机问世以来，至今才短短的46年历史，电子计算机科学和技术已取得惊人发展，电子计算机工业已成为强大的独立工业部门。

要使一个完整的计算机系统运转，离不开硬件和软件。硬件是组成计算机的物质设备，如CPU、内存、外设等；软件是完成特定功能的程序的集合，它由汇编语言或高级语言编写。如果把计算机系统比拟成一个“人”，那么硬件是“人”的躯体，而软件是“人”的灵魂、思想。

本章简要介绍程序设计、高级语言等基本概念，回顾C语言的发展历史及其特点，并通过示例讲解C程序的一般知识及上机实习过程。

1.1 程序设计

唱歌要有乐谱。乐谱中规定了什么地方是高音，什么地方是低音，什么地方连唱，什么地方停顿等等。计算机的动作是执行程序的过程，程序就相当于“乐谱”。而程序设计就是编制“乐谱”的过程。简单地说，程序设计就是根据所提出的任务，用某种程序设计语言编制一个能正确完成该任务的计算机程序。例如，用FORTRAN语言编写一个求解线性方程组的程序，用C语言编写求解八皇后摆法问题的程序等。

如何进行程序设计呢？一般说来，包括以下步骤：

(1) 问题定义——把所要解决的问题、所涉及的输入数据以及希望得出的结果等，用日常语言尽可能清晰、完整、准确地表达出来，经过抽象，建立相应的数学模型；

(2) 算法设计——确定解决问题的方法，并把任务分解成计算机能够执行的几个特定操作。如对排序问题，是采用选择排序、分段变换排序还是快速排序等。

(3) 流程图设计——用形象的、适于编写程序的方法表达算法。可用自然语言描述，也可用流程图符号表示，或者将二者结合起来。

(4) 程序编制——用选定的语言，按流程图提供的步骤写出程序。

(5) 程序调试、测试及资料编制——对编完的程序进行编译、运行，查出其出错位置，并予以纠正。对有实用价值的程序，还要测试其正确性及效率等，并编写程序的使用和维护说明书，供别人参考。

要成为一名好的程序员，除了要熟练掌握一些语言（高级语言或汇编语言）外，还必须多读多练。多读是吸取他人的经验和长处，多练是转化成自己的实际才干。

1.2 高级语言

程序设计是伴随着电子计算机而产生和发展起来的。它源于电子计算机的出现，又推动其普及应用，促进其更新换代。

在计算机发展初期，仅使用机器可直接识别的二进制码，它们是由“0”、“1”构成的序列，称为“机器语言”。机器语言过于繁杂，使用困难，进一步发展便产生了汇编语言。汇编语言是一种便于记忆的“符号语言”，但它属于低级语言，就是说，汇编语言依赖于具体的机器。

在汇编语言的基础上，出现了各种高级语言，如ALGOL、BASIC、FORTRAN、COBOL、PASCAL和C等。高级语言与具体机器的指令系统没有直接关系，以更接近于自然语言（如英语）或数学公式的形式来编写程序。高级语言的出现大大推动了计算机的应用和普及。

用汇编语言或高级语言编写的程序不能直接在机器上运行，必须经过“转换”，变成相应的机器语言后才可执行。把汇编语言转换成机器语言的程序，称作“汇编程序”；把高级语言转换成机器语言的程序，称作“编译程序”。不同的高级语言对应于不同的编译程序。

1.3 C语言发展历史及特点

C语言是1972年在贝尔实验室开发出来的，它由D·里奇一个人设计和编写。设计C语言的最初目的就是为了更好地描述UNIX操作系统（由K·汤普逊和D·里奇设计）。因此，C与UNIX操作系统紧密地联系在一起。正如美国“Electronics”杂志向其开发者颁发1982年度最高成就奖时所评价的：UNIX系统与C语言的紧密组合对从微型机到大型机的广泛应用产生了重要影响。当然，如今C语言已独立于UNIX系统，可在各种硬件环境上运行。

C语言属于Algol语系，其派生关系如下：

Algol 60

由一个国际委员会设计，1960

↓

CPL

(Combined Programming Language)

剑桥大学和伦敦大学，1963

↓

BCPL

(Basic Combined Programming Language)

M·里查德，剑桥，1967

↓

B

K·汤普逊，贝尔实验室，1970

↓

C

D·里奇，贝尔实验室，1972

多年来，UNIX 第 5 版操作系统上配备的 C 语言一直被作为 C 语言的公认标准。但随着微型机的普及，出现了一大批 C 语言系统，它们之间具有很高的兼容性。但由于没有统一标准，必然有些差异。为了改变这种局面，ANSI（American National Standards Institute）于 1983 年夏初设立了一个委员会以制定 C 语言标准。随着制订以 UNIX 为基础的操作系统的标准——POSIX（Portable Operating System Interface for computer environments）这一工作的开展，ISO 于 1990 年通过了 C 程序设计语言的标准，它是以 ANSI 标准为基础，吸收了国际上的意见后制定的。

C 语言能取得今天的显赫地位，是与它具有众多的优点分不开的。主要有：

（1）简洁　C 是一种表达式语言。它有功能很强的运算符，如增 1 运算符（++）、减 1 运算符（--）、取地址运算符（&）、间接运算符（*）等，用这些运算符可构成书写简洁而功能很强的表达式，从而可提高软件生产效率。另外，C 语言未提供输入/输出机制，而由其运行的宿主环境（操作系统，如 UNIX、DOS、VMS 等）提供，这样就使得 C 编译程序小巧、紧凑。

（2）高效　C 语言通常被称为中级计算机语言。这并不含贬意，相反，把高级语言的成分和汇编语言的功能结合起来，表示 C 语言，因而能充分反映机器硬件的功能，如允许对位、字节和地址这些计算机功能中的基本成分进行操作，所以代码效率极高，几乎接近于汇编语言代码。这是目前其他高级语言尚未达到的。在应用中，C 语言可以取代汇编语言，这也是 C 语言被程序员广泛使用的一个原因。

（3）可移植　C 语言程序非常容易移植，可将为某种计算机写的软件改编到另一种机器上去，从而使编写的程序独立于具体的计算机体系结构，实现“共用”目标。

（4）结构化　C 语言的主要结构成分是函数，它是完成程序功能的基本构件。这样，一个程序的诸多任务就可被分别定义和编写，各由一个单独的函数完成，从而使程序模块化。结构化语言总是现代的，它比非结构性语言（如 BASIC、FORTRAN）更易于程序设计，写出的程序结构清晰，便于维护。

C 语言还有其它一些优点，读者可在学习及实践中体会。当然 C 语言也和其他语言一样，存在不足之处，如某些运算符优先顺序与习惯不完全一致；类型转换比较随便，不是强类型语言等。尽管如此，相比之下，C 语言仍不愧为优秀的程序设计语言之一。

1.4　C 程序示例

下面用几个具体的 C 程序作示例，简要介绍 C 程序的结构和各个成分的构成及作用，使读者对 C 语言编程有一个感性认识。

〔例 1-1〕 计算三个整数的和。

```
main(  )
{
    int  a,b,c,sum;
    a=1;
    b=10;
    c=100;
    sum=a+b+c;
    printf("sum=%d\n",sum);
}
```

现在对这个程序进行解释：

第 1 行 main( )是函数首部，它告诉系统：这个函数的名称是 main。怎样知道 main 是表示函数名呢？这是由于在 main 后面紧跟着一对圆括号，向 C 编译程序表明这是函数。main 是 C 语言中标识主函数的专用名，表示该 C 程序从这里开始执行。

第 2 行只有一个开花括号｛，它等同于 PASCAL 语言中的 BEGIN，而第 9 行的闭花括号｝等同于 END。这一对花括号往往被称为语句括号，它们把一组数据说明和执行语句括在一起，构成这个函数的函数体。在编写程序时要注意，在任何情况下，开花括号与闭花括号都必须成对出现，不允许某一个出现的次数多于另外一个。

第 3 行至第 8 行构成这个函数的函数体。第 3 行

```
int  a, b, c, sum;
```

是数据说明语句，表示 a，b，c 和 sum 是定义过的四个整数类型（由 int 标志）的变量。说明语句的末尾是以分号结束。

第 4 行至第 7 行是四个赋值语句。在 C 语言中，赋值符是“=”，先计算赋值符右边表达式的值，然后赋给其左边的变量。于是左边变量的值就是右边表达式计算的结果。如第 7 行，先把 a（值为 1）、b（值为 10）、c（值为 100）三个量的值相加，得到 111，然后赋给 sum，这样 sum 的值就是 111。

注意在每一个语句的后面都带有一个分号（;），这是 C 语言的规定。因为在 C 语言中，分号是语句终止符，是语句的一个组成部分，而不是一般意义上的分隔符。所以，一个语句末尾的分号是不可省略的。

第 8 行 printf(…);是一个打印语句。在 printf 后面紧跟有一对圆括号，表示它是一个函数。这里是对 printf 函数进行调用，由它输出运算的结果（即 sum 的值）。其实，printf 是 C 语言标准程序库中的一个函数，用来进行格式输出。此例中，圆括号内有两个实参，其间以逗号分开。第一个实参是用一对双引号括起来的字符串，表示有关输出格式的控制信息。这里“%d”是输出转换的标志，表示把第二个实参（即 sum）中的内容按十进制整数形式输出。%d 前的字符串按原样输出。所以该程序运行后输出结果是：sum=111。

在这个字符串中最后的两个字符，即反斜线\和字母 n 合起来作为一个换行字符。当遇到换行字符时，光标或打印头就移到下一行的开头。因此，在换行字符后的任何字符，在终端或显示屏幕上都出现在下一行中。

这里提到程序的第几行，只是为了说明方便，实际上 C 程序是没有行号的。就是说，上面的程序不能写成带行号的形式：

```
1 main(  )
2{
3     int a,b,c,sum;
        ⋮
9}
```

〔例 1-2〕 求给定三个电阻的并联和串联的电阻值。

大家知道，电阻 r1、r2 和 r3 的串联电阻值 rs 等于各电阻值之和，即：rs=r1+r2+r3；而它们的并联电阻值 rp 的倒数等于各电阻值倒数之和，即：$\frac{1}{rp}=\frac{1}{r1}+\frac{1}{r2}+\frac{1}{r3}$。

可以先设一个变量，如 rr，表示 rp 的倒数。利用上式求出 rr 之后，再求出 rp。

下面是相应的程序。

```
/* Calculating the values of series and parallel of resistances of r1,r2 and r3 */
main(  )
{
      float r1,r2,r3;  /* resistances */
      float rs;/* series value */
      float rp;/* parallel value */
      float rr; /* reciprocal of rp */
      printf("Input:R1=? R2=? R3=? \n");
      scanf("%f %f %f",&r1,&r2,&r3);
      rs=r1+r2+r3;
      rr=1/r1+1/r2+1/r3;
      rp=1/rr;
      printf("The series value is %f;",rs);
      printf("The parallel value is %f·\n",rp);
}
```

现在对这个程序的结构进行解释。为了说明方便，仍采用“第几行”这种方式。

第 1 行是以/*开始，以*/结束，中间有一字符串，这是注释行。注释行用于对整个程序、或某些特定段的功能、或者某些重要数据的作用进行解释或提示，从而大大提高程序的可读性。同样，在第 4 行、第 5 行、第 6 行和第 7 行中也用了注释行。从这五个注释行出现的位置可以看出，它可以出现在整个程序的前面，也可插在某个程序段的前面或后面，甚至还可插在某行代码的前后。就是说，注释行可出现在程序的任何位置上。因为在进行编译时，注释行不生成目标码。即，C 编译程序在处理时把它看作是一个空白，并不作任何解释。

在程序中插入一些注释行是一种良好的习惯，提倡尽量多用。在使用注释行时，也要注意：①/*和*/要成对出现，并且在/与*以及*与/之间都不能插入空格；②注释行不能嵌套，就是说，不能在一个大的注释行里面包含有若干小的注释行，如下面的形式是不允许的：

```
/*……/*……*/……*/
```

③注释行不要插到一个字符常量（如‘A’、‘\n’）或一个字符串常量（如“abc”、“ok!”）的中间。

第 4 行至第 7 行是数据说明语句，即对程序中所使用的变量的类型进行说明，在这里分

别说明 r1，r2，r3，rs，rp 和 rr 是浮点型（float）变量。各行后面的注释指出：r1，r2 和 r3 表示各电阻值，rs 表示串联电阻值，rp 表示并联电阻值，rr 表示 rp 的倒数，即并联电阻值的倒数。这样就增加了程序的可读性。

第 8 行是函数调用语句——调用标准库函数 printf（详见第三章或第十六章），打印出提示信息：Input：R1＝？　R2＝？R3＝？　这样在运行程序时，一旦看到上述提示行，就知道应该打入给定三个电阻的各自的数值了。

第 9 行也是函数调用语句——调用标准库函数 scanf（详见第十六章），它接收用户从键盘打入的浮点数，分别赋予变量 r1，r2 和 r3，从而给定了这三个变量的初值。例如，当在键盘上打入下面输入行时：29.5　10.44　26.0，则 r1，r2 和 r3 将分别取值为 29.5，10.44 和 26.0。

第 10 行、第 11 行和第 12 行是赋值语句。第 10 行语句执行之后，理论上 rs 的值应是 65.94。

随后两行分别对函数 printf 进行调用，分别输出给定三个电阻的串联值 rs 和它们的并联值 rp。其印出结果是：

The series value is 65.940002；The parallel value is 5.947238。

为什么两次调用 printf，而结果却输出在一行上呢？这是由于第一个 printf 的格式控制实参中没有换行符（\n），于是第二个 printf 的输出信息就接在第一个 printf 的输出之后打印（显示）出来。如果需要各单独占用一行，那么第一个 printf 的调用形式应改为：

printf("The series value is %f\n",rs)；

为什么理论上 rs 的值是 65.94，而输出结果却是 65.940002 呢？这是由于使用格式转换符%f，只保证在小数点后面保持 6 位有效数字。但浮点变量并不一定有足够的二进制位把给定的小数值恰好存放，这就造成了最后精度的误差，从而出现理论计算值与实际输出结果不同。在进行浮点运算时类似现象经常出现，这是由于表示数值时的精度误差造成的。

另外，细心的读者会发现，这个程序是不"安全"的：如果输入的电阻值为负数或者为 0 时，怎么办？（为负数时，不仅违背通常的电阻的物理含义，还会造成 rs 的值为 0 的恶果！）这可通过对输入数据进行判别来解决：如果出现这种情况，就给出告警信息，并要求用户重新打入数据。如果读者有兴趣对这个程序进行改进（这样做很好！），不妨先画出算法流程图，等学完第四章和第五章的知识之后，再动手编写程序。

〔例 1-3〕　把数 1996 转换成罗马数字表示。

```
/*
 * * Program to Print 1996 in Roman numerals.
 */
main(  )
{
    int  a=1996;

    a=romanize(a,1000,'m');
    a=romanize(a,500,'d');
    a=romanize(a,100,'c');
    a=romanize(a,50,'l');
    a=romanize(a,10,'x');
```

```
    a=romanize(a,5,'v');
    romanize(a,1,'i');
    putchar('\n');          /* end with a newline */
}
/*
 * *Print the character c as many times as there are
 * *j's in the number i,then return i minus the j's.
 */
romanize(i,j,c)
char c;
int i,j;
{
    while(i>=j)
      {
        putchar(c);
        i=i-j;
      }
    return(i);
}
```

这个程序分成两个函数:main( )和 romanize( )。前者的工作主要是安排减法运算次序及程序的起始和终止。而后者完成实际的减法运算，并利用一个库函数 putchar( )将罗马数字打印出来。

C 语言中根据功能的需要，一个程序可以分成若干个文件（在 UNIX 操作系统中，这些文件名通常带有后缀·c，表示是用 C 语言书写的源程序）。每一个文件中可以包含若干个函数。不管整个 C 程序由多少个函数组成，其中必须有且只能有一个名为 main 的函数。以上三例的程序都比较简单，各自存于一个文件中（例如分别为 sum. c,cal. c,rom. c）。前两个程序都只有一个函数，因此函数名一定是 main；例1-3的程序中有两个函数，其中只能有一个 main 函数。

在例1-3中，main( )函数体的开头一行：

```
int a=1996;
```

实际是一种变量说明，并给它赋了初值。它说明 a 是整型量（int），为了省写就把初值直接写在其后。当然可以改写成：

```
int a;
a=1996;
```

效果完全一样。注意，在 C 中变量都要先说明后引用，说明要给出其类型和存储类。

紧接下面的7个语句，调用了7次函数 romanize，其中前6次把返回值赋给了 a，最后一次未赋值给 a。这是因为前6次都需要把返回值用作下一次函数调用的实参值。现在解释一下 romanize 的工作：

```
romanize(i,j,c)
```

表示它有3个参数 i，j，c。因为此时其具体值是多少尚不可知，所以称之为形式参数，简称形参。

```
    char c;
    int i, j;
```

说明形参 c 为字符型（char）量，i 和 j 是整型（int）量。其作用是告诉编译程序有关引用量的存储空间大小。有时这两行也称为参数区分。

上述三行总称为函数首部，针对此例，它包括函数名 romanize，参数表（i，j，c）和参数说明 char c；int i，j；

其实每一个函数的第一个开花括号（{）前面都称为函数首部，如函数 main（ ）中，main（ ）即是首部，而函数 romanize 的首部占三行。

romanize( )里的函数体部分是一些可执行的代码。这里包含两个语句：while 和 return。而 while 语句中又包括了两个语句，为此要用花括号括起，构成一个复合语句。只要条件 i>=j（i 大于等于 j）成立，这个复合语句中的两个语句就不断执行。while 语句称为循环语句，是 C 语言中的一类重要的控制语句，与它有关的还有 for 和 do-while 语句。return（i）；是一个返回语句，它向调用者返回一个值，并把控制也返回调用者。例中调用者为函数 main。

现具体说明这两个函数的调用关系：在 main 中，第一次是以 a=1996，1000，'m' 来调用 romanize，这里的1996，1000，'m' 分别对应于 romanize 中的形参 i，j，c，称之为实在参数，或简称为实参。当控制转到 romanize 之后，由于（i>=j）（因为1996>=1000）条件成立，所以执行 while 循环体中的两个语句：由 putchar（c）；印出字符 m。经过第二句运算后得到 i 为996，再转回去时，（i>=j）不成立，所以退出 while 循环，从而执行 return 语句，把 i（=996）返回给了调用者。当控制回到 main（ ）时，第一个语句

```
    a=romanize(i,1000,'m');
```

a 得到返回值996。之后又第二次，第三次，直到第六次调用函数 romanize，此时 a 得到的返回值是1。最后一次调用函数 romanize，只是把控制返回，而调用者不需要返回值。

在 C 语言程序中，函数出现位置的先后是无关紧要的，因为它总是从函数 main 开始执行。另外，一个名字之后紧跟的圆括号（ ）是函数的标志。

用 C 语言进行程序设计时，鼓励人们把一个大问题划分成若干小问题，为每一个小问题编制一个函数，然后把这些小“部件”（函数）安装起来就构成一个完整的运行良好的大“机器”（程序）。原则上 C 语言是无格式语言，即在格式上没有任何特殊的要求，一个程序的代码可以一个字符接一个字符地排成“长蛇阵”。但是为了便于阅读和修改，读者在编写时注意缩进和上下对齐，使程序的层次一目了然，如上例所示。

1.5 C 程序的编辑、编译和运行

把手写的 C 程序输入到文件中去，一般需要使用宿主系统（如 UNIX、PC-DOS、VMS 等）所提供的编辑程序（如行编辑 ed，屏幕编辑 vi 等）。编辑好的程序应放在某个（些）以.c 结尾的文件（称作源文件）中，比如〔例1-1〕放在文件 sum.c 中；〔例1-2〕放在文件 cal.c 中。

对编辑好的源文件就可以进行编译。在 UNIX 环境下，如果欲编译文件 sum. c 中的程序，就打入如下命令：

```
    cc sum.c
```

如果是编译文件 cal.c 中的程序，则打入命令：

cc cal.c

可以看到，一般的编译命令的格式是：

cc　文件名

打入这个命令后，系统把 C 编译程序调入内存，对要编译的程序进行词法分析、语法分析、代码生成等多种处理。如果没有错，就按默认原则，把最后生成的可执行目标代码放在临时共用文件 a.out 中。为了运行这个可执行目标代码并获得所需结果，只要打入命令：

a.out

如果要保留可执行目标代码，则在编译时要用任选项－o，它把可执行代码放在紧随－o 之后的文件中。比如，

cc　－o　sum　sum.c

或　cc　－o　cal　cal.c

那么执行时只要打入

sum

或　cal

就行。

如果一个程序由若干文件组成，为了对它们统一编译，就把所有这些文件名都列在 cc 命令之后，如：

cc　p1.c　p2.c …pn.c

其中 p1.c，p2.c，… pn.c 就是一个程序的几个源文件名。如果不想立即把所有源文件都进行编译，而想一个个或一部分一部分地编译，那么须采用部分编译手段。在 UNIX 环境下是在用 cc 命令时带任选项－c，否则就会发出警告。比如〔例1-3〕，如果把函数 main(　)放在 p1.c 中，而函数 romanize(　)放在 p2.c 中，那么打入命令

cc p2.c

就会发生警告：

Undefined:

-main

它表明在 p2.c 中没有主函数 main；但是打入下列命令就不会产生警告信息：

cc －c　p2.c

因为这里用了部分编译功能，根据默认规则，其编译好的目标码将放在 p2.o 中。以后重新打入

cc　p1.c　p2.o

仍是正确的。

图1-1是 C 语言程序编辑、编译和运行的全过程的框图。

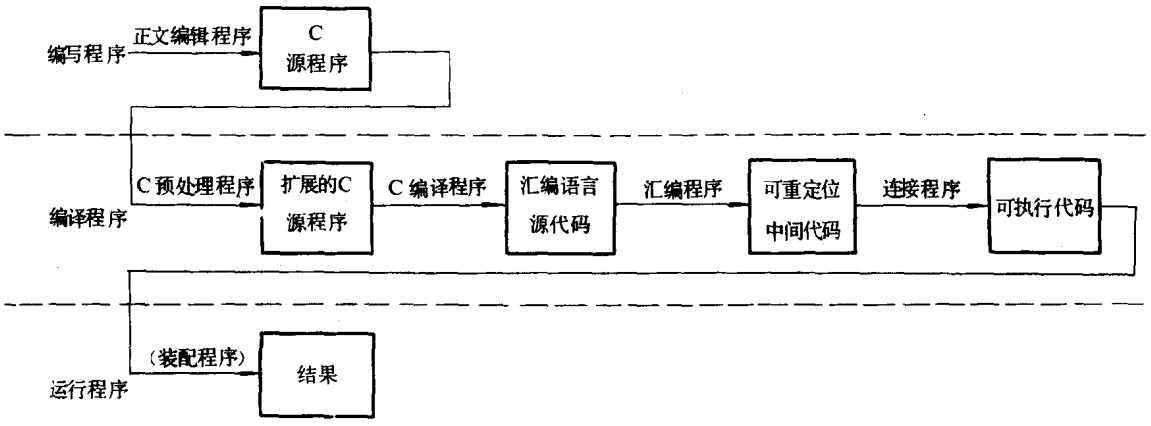


图1-1 C程序的编写、编译和运行过程

1.6 字符集及词法约定

1.6.1 字符集

任何一种高级语言都需要有自己的字符集，C语言也不例外。利用C语言编写源程序时用到的字符集中有数字、字母、图形符号、转义序列和三联字符序列等。

(1) 数字：通常的十进制的10个数字：0，1，……9。

(2) 字母：英文字母大、小写各26个：A、B、…Z和a、b、…z；UNIX系统喜欢用小写字母，而且在内部处理时也以小写为主，所以大写字母和小写字母在UNIX系统中并不等同。

(3) 图形符号：29个图形（可见）字符：

! " # % & ' ( ) * + , - . / : ; < = > ? [\ ] ^ _ { | } ~

它们主要用于表示各种运算符。

(4) 转义序列：某些不可见（不可打印）或难以打印的字符需要在源程序中出现，就采用转义序列的方式来表示。比如在前面程序中用到的换行符\n，看上去是两个字符，因为输入时要打两个键：字符\和字母n。实际上\n合起来表示一个换行符。这里字符\被称为转义符，表示尾随其后的那个字符（如n）失去原有含义，而具有另外特定的意义。表1-1列出了C语言源程序中常用到的一些转义序列及其含义。

表1-1 转义序列及其含义

字符名	表示形式	含义	备注
换行	\n	把打印（显示）位置移到下一行的起始位置	
水平制表	\t	把打印位置移到当前行的下一个制表点（通常是右移8个字符的间隔）	与实现有关
垂直制表	\v	把打印位置移到下一行制表点起始位置	与实现有关
退格	\b	把打印位置在当前行上向后退一个字符位置	与实现有关