# C语言程序设计及其应用

龚杰民　金益民　编

子科技大学出版社

# C语言程序设计及其应用

龚杰民 金益民 编

西安电子科技大学出版社

1988

## 内 容 简 介

C语言是现代最流行的通用程序设计语言之一，它简洁、紧凑、灵活、实用、高效、可移植性好，并且还考虑了硬件对程序的影响。UNIX操作系统本身及其支持的许多应用软件都是用C语言书写的。C语言在我国也正在迅速推广。

本书介绍了C语言的历史、特点及新版本的变更规定，全面、系统地阐述C语言的所有成份，不仅包括C的数据类型、语句、存贮类别，还较详细地叙述预编译程序、C与宿主机上运行环境的接口。本书共有150多个程序实例。各章节的实例注重说清概念，并介绍程序设计的方法与技巧；最后一章为综合应用实例，包括测试C语言版本与UNIX操作系统特性的基准程序及其它应用程序。每章末附有习题，书后有六个附录。

本书可供高等院校计算机、自动化、无线电、经济管理等专业作为教材，也可供计算机工作者、需要使用与研究C语言的科研人员和工程技术人员参考。

**C语言程序设计及其应用**

龚杰民 金益民 编

西安电子科技大学出版社出版发行
（原西北电讯工程学院出版社出版）
西北电讯工程学院印刷厂印刷
新华书店经销

开本 787×1092 1/16 印张16 6/16 字数 395 千字
1986年2月第1版 1988年6月第2次印刷 印数 5001—9000

ISBN 7-5606-0065-4/TP·0021
统一书号：15322·41
定价：2.75元

# 前　言

C语言自七十年代问世以来，由于它具有简洁、代码紧凑、灵活、高效、实用和可移植性好等特点，已成为当今最流行的程序设计语言之一。它的设计者最初用它来书写UNIX操作系统及其支持的应用软件，现在它的应用已不限于UNIX操作系统，国外生产的各种计算机系统(从大型机到小型机、微型机)几乎都配有C编译程序。它是通用的程序设计语言，特别适合于书写操作系统、编译程序、数据库管理系统等系统程序，也适合于书写网络软件和应用软件。这几年随着UNIX操作系统引入国内，C语言的重要性愈加明显，C语言的应用、研究及标准化等工作正在大力开展，以UNIX为实际背景的操作系统教材已陆续出版。在这样的形势下，迫切要求推广普及C语言。但是目前国内的C语言教材寥寥无几，本书正是在这种情况下编写的。

1980年以来，我们先后承担了与C语言有关的教学、科研工作，分别对美国的大学生，我国的大学生、研究生、教师、科技工作者讲授了C语言或系统程序设计方面的专题。1984年初，我们编写了《C程序选编》，并得到了院内、外同志们的鼓励，希望我们早日编写出一本教材。于是我们参阅了一些新的文献资料，设计并整理了各种程序实例150多个，结合几年来的教学体会，写成了这本教材。

本教材的对象是学过一门以上程序设计语言的大学生、研究生与科技工作者。§1.1首先介绍了C语言的历史与特点，这一节既是C语言概述的开始，也是对C语言的小结，读者在学习各章时或学完本书后再读一下本节，可以加深体会。从§1.2到第八章，详细地、由浅入深地阐述了C语言的程序结构与语言的所有成份，并着力讨论了初学者较陌生的指针类型、存贮类别及预处理程序等内容。在讨论方法上，我们从程序设计的角度提出问题，举出引例，然后逐步展开，系统阐明概念与语法规则，指出难点与易混淆之处；并通过较多的实例来说清概念与示范如何应用。学习程序设计语言，与学习自然语言有相似之处，不能只学语法，还必须大量实践，因而读一些恰当的程序实例是必不可少的环节。为此，我们不仅在各章、节都给出中、小型的例题，在第十章又集中给出了综合应用的实例，包括测试C语言与UNIX操作系统特性的基准程序及各种应用程序。很多程序是我们自己设计、编写并调试通过的，大部分程序还印出了输入数据与运行结果。考虑到读者英语程度的差异，有的注释用中文，有些还在程序清单边上加旁注，以便于阅读理解。

C语言也在不断地完善和演化。1978年美国Bell实验室的Dennis Retchie和Steve Johnson对C语言的定义作了扩充和修改。除了增加枚举类型外，主要对结构与联合类型放宽了限制。目前已经实现的编译程序大多数已作了相应的修改。但目前我们所见到的国内C语言教材或参考书尚未提到这些变动，本书对这些变动作了介绍(见§2.2与§7.3)。

C语言与宿主系统的关系十分密切。第九章较详细地阐述C语言与宿主系统的接口。目录标有“*”的小节初学者可以不看。对于标准库函数的调用形式，C编译程序的各种任选开关，编译命令等使用方法都作了扼要介绍，有些在附录中给出。

由于C语言的标准化、形式化工作尚在进行之中，为慎重起见，本书不另给形式描述，在附录六中引用了文献[1]的附录A：语法参考手册中的几节。

本书的第七、九、十章(除去§9.2)，§1.1和附录中的说明由金益民同志编写，其余各章由龚杰民同志编写。

西安交通大学计算机系主任、全国计算机与信息处理标准化技术委员会程序设计语言分技术委员会C语言工作组副组长施鸿宝同志在百忙中仔细审阅了本书，并提出了许多宝贵意见，我们在此表示衷心的感谢。本书的编写还得到我院许多同志的热情鼓励与支持，我院203室机房、101室机房和图书馆机房为我们提供了上机条件，何振邦、刘坚同志为本书提出了有益的建议，我们在此一并致谢。由于学术水平和经验的限制，错误和缺点在所难免，敬请各位专家、广大读者批评指正。

编　者

于西北电讯工程学院

# 目　录

前言
第一章　C 语言概述和简单的 C 程序
§1.1　C 语言的历史与特点……1
§1.2　C 程序的书写格式、编译和运行……4
§1.3　变量及变量的类型说明……6
§1.4　符号常数……8
§1.5　常量……9
§1.6　输入/输出初步……10
习题……16
第二章　数据、运算符和表达式
§2.1　基本数据类型……17
§2.2　枚举类型……18
§2.3　变量的初值设置……20
§2.4　数组……20
§2.5　算术运算符……23
§2.6　关系和逻辑运算符……24
§2.7　类型转换……25
§2.8　加 1 和减 1 运算符……27
§2.9　字位逻辑运算符……28
§2.10　赋值运算符……29
§2.11　计算的优先级和顺序……30
§2.12　表达式……31
习题……33
第三章　语句和流程的控制
§3.1　表达式语句和空语句……35
§3.2　复合语句(分程序)……35
§3.3　if 语句(如果语句)……38
§3.4　循环……42
§3.5　break 语句(中止语句)……51
§3.6　switch 语句(开关语句)……52
§3.7　goto 语句和带标号的语句……56
§3.8　continue 语句(继续语句)……57
§3.9　return 语句(返回语句)……58
习题……58

**第四章 函数**

§4.1 函数的定义与调用……61
§4.2 函数的类型……66
§4.3 关于函数参数的讨论……70
§4.4 递归……72
习题……78

**第五章 指针和数组**

§5.1 指针的概念……79
§5.2 指针参数……81
§5.3 指针和数组……84
§5.4 指针与多维数组……90
§5.5 命令行参数……92
§5.6 指向函数的指针……99
习题……101

**第六章 各种存贮类的区别**

§6.1 自动变量……102
§6.2 外部变量……105
§6.3 静态变量……115
§6.4 寄存器变量……118
§6.5 类型定义……120
习题……121

**第七章 结构和联合**

§7.1 结构(struct)……123
§7.2 联合(union)……143
*§7.3 有关结构与联合的规定的变动……145
习题……148

**第八章 C语言预处理程序**

§8.1 宏替换……149
§8.2 包含文件……152
§8.3 条件编译……153
§8.4 行控制……154
习题……154

**第九章 C语言与宿主系统的接口**

§9.1 与C语言有关的UNIX部分概述……156
§9.2 输入输出函数……159
§9.3 低级输入输出……173
*§9.4 进程……177
*§9.5 信号(signal)与中断……182
习题……186

**第十章　C 语言程序综合应用举例**

§10.1　一些基准程序 ……187
§10.2　数值计算应用举例 ……202
§10.3　动态结构的应用举例 ……208
§10.4　系统程序设计举例 ……213
§10.5　两个游戏程序 ……221
附录一　C 语言关键字表 ……236
附录二　UNIX C 编译程序的主要调用开关说明 ……236
附录三　UNIX 联结装配程序 ld 的部分调用开关说明 ……236
附录四　UNIX 系统中部分.h 文件的内容 ……237
附录五　C 语言标准输入输出库说明 ……243
附录六　C 语言语法摘要 ……249
参考文献 ……252

# 第一章　C语言概述和简单的C程序

## §1.1　C语言的历史与特点

### 一、C语言的简要历史

六十年代，随着计算机科学的形成与发展，高级程序设计语言的研究得到了蓬勃的发展。但是，当时得到广泛应用的高级程序设计语言中缺乏用于书写操作系统和编译程序等系统程序的工具，系统程序设计仍然主要依赖于汇编语言。根据这种情况，人们开始探索能用于系统程序设计的有足够表达能力并且高效率的高级程序设计语言。作为这种探索工作的结果之一，剑桥大学的 Martin Richards 在1969年设计并实现了BCPL (Basic Combined Programming Language)语言。(该语言后来被移植到了多种计算机上，并且至今仍得到相当广泛的应用。

同年稍后至次年 Bell 实验室的 Ken Thompson设计了一种类似于 BCPL的语言，称为B，并用 B语言书写了在PDP-7机上实现的第一个实验性的UNIX操作系统。接着，在1972至1973年间，Bell实验室的Denis Ritchie在B语言的基础上，重新设计了一种语言，首先在PDP-11 机上实现，并用它重写了 UNIX 操作系统。由于它是 BCPL 和 B 语言的后继，因而取名为 C。但是，C 语言与它的前驱不同，BCPL和 B 是无类型的语言，而C语言却能支持许多种数据类型，因而更能反映当代大部分计算机的结构，参考文献[12]中给出了它们主要的异同处。

为了使C语言具有高度的可移植性，在 PDP-11C 编译程序的基础上，Bell实验室的小组作了进一步的努力。首先在 1974 年，麻省理工学院的Alan Snyder 根据与Bell实验室的合同，作为他的硕士论文基础，书写并实现了一个可移植的 C编译程序。但由于该编译程序速度太慢、过于复杂，并且存在一些严重的错误，未能进入实用阶段。

在 Alan Synder工作的基础上，Bell的 S.C.Johnson在1977年实现了一个成功的可移植编译程序，简称 PCC，其核心思想是把编译程序中与机器有关和与机器无关的部分分离。结果是编译程序的约75% 是与机器无关的，只有约 25% 与机器有关，在移植时需作改动。后期实现的大部分C编译程序，如UNIX第七版，UNIX 系统Ⅲ和系统 V 支持的 C编译程序，都采用 PCC 编译程序。

在短短的几年内，C 语言得到了不断的修改、完善和扩充。如初始化从不用等号改为用等号；赋值运算符把运算符从等号右边移至左边以避免混淆；增加了结构的直接赋值，枚举类型等。在 PDP-11的C版本中为了支持抽象数据类型，还增加了类属(Class)说明等。

随着 UNIX 操作系统的逐步推广，也由于C语言本身的特点，大多数计算机从大型机到微型机现在都配有C 编译程序。即使非 UNIX 操作系统的机器，也实现了C的编译程序。例如CP/M80支持的C语言就有 Aztec C, BDS C, Q/C, Eco-C, Supersoft C, Whitesmith C等多种；CP/M-86和 MS-DOS也支持多种 C语言。在 UNIX 环境下还有 Lint等工具帮助检测程序设计的错误，可以大大减轻程序员调试程序的负担。

除了UNIX环境以外，C也已被用于其它几个环境中。用C书写的程序范围很广，包括UNIX操作系统，C编译程序本身，以及基本上全部UNIX的应用软件和大量的游戏程序。除了系统软件外，C已被成功地用于主要的数值计算，文字处理和数据库领域中。尽管它是一个高级语言，由于它强有力的表达能力和效率，在许多环境中它已经或可以完全取代汇编语言。

目前C语言还存在一些缺点和不足之处，还没有一个一致公认的标准。C的标准化问题现已引起国际标准化组织许多成员国的重视。我国也已组织了专门的小组研究C的形式化描述和规范。随着越来越多的人学习、掌握、应用、关心和研究C语言，它一定会更趋完善，在计算机科学和应用领域中发挥更大的作用。

**二、C语言的主要特点**

C语言是在七十年代初期设计的。在很多方面继承和发扬了在六十年代出现的许多高级语言的成功经验与特色。如数据类型，多种存贮类别、自由格式，一定程度的模块化结构，参数的传值方式，结构化的控制，分别编译等。但由于它是作为一种系统程序设计的工具设计的，也就决定了它具有许多特殊性。与几乎同时出现的PASCAL语言相比，由于它们的设计目标不同，差异较大。熟悉PASCAL的程序员在初次接触C语言时往往会感到C程序难以调试。这是因为PASCAL为保证程序的正确性而对语言规定了许多限制，在程序编译时通过检查是否违反了这些限制来帮助程序员检测程序设计的错误。而C语言为了使语言有较大范围的可应用性，尽可能少作限制。语言鼓励使用表达式的副作用，使用指针，交替地使用数组和指针等。这种以灵活性为主要设计目标的结果，使得C语言程序较为难以验证。

C语言的主要特点是相对简单，表示法简洁，大量使用指针运算，高度灵活，容易移植，错误检测能力较差。

C语言的编译程序一般规模为7000～8000行源程序。语言的基本成份如下：

1. 数据类型

C的基本数据类型有char(通常为8位字节),int, short, long(长度不同的整数),float和double(浮点数)。

此外，从概念上讲，可由这些基本类型导出无穷的其它类型：如果T是一种类型，则可以有指向T类型变量的指针，数组元素为T类型的数组,包含T的结构和联合(相应于PASCAL的记录与变体记录)，还有函数指针和枚举类型。

C不具备字符串数据类型。字符串以字符数组表示，通常以空字符作终止符，并由库函数操作。

指针运算是C的整体组成部分。如果P是指向T的指针类型，且当前指向类型为T的数组中某一元素，则P+1指向数组中下一元素。即指针的运算以该指针所指对象的大小衡量，程序员不需(也不应该)关心其实际大小。

2. 运算符和表达式

除了常见的加、减、乘、除等算术运算符和大于、等于、小于等关系运算符及与、或、非等逻辑运算符外,C还有多种其它运算符。在常用的高级语言中，除了APL外，没有比C的运算符更丰富的了。C允许访问指定的物理地址，并可对一个字中的某一或某几位进行操作，包括与、或、非，异或，左移、右移等。位操作的存在，是C可以取代汇编语言的一个重要原因。在C中，任何一种二元算术运算符和大多数二元位操作运算符都有对应的“赋值

运算符”。例如“+”对应的赋值运算符为“+=”，语句：

V=V+表达式；

可以更紧凑地写作：

V+=表达式；

一元运算符++和--分别对它们的操作数加1和减1，语句：

++thing；与语句

thing=thing+1；等价。

C表达式的类型可以通过加在表达式之前的说明符强制性地转换为另一类型。例如语句：

x=sqrt((double) integer expression)；中，说明符double将整型表达式转换为sqrt函数所要求的双精度浮点实型。

3. 控制流

C的控制流基本上是传统的形式，有条件语句、循环结构(for语句、while语句、do-while语句)、开关语句、从开关语句或循环体立即出口的break语句、使下次循环开始的continue语句等。C语句还有标号和GOTO语句。

4. 程序结构

一个C程序由存于一个或多个文件中的可分别编译的一系列变量和函数说明构成。C的程序结构是分程序结构，但函数定义不能嵌套。

在最高一级的对象——函数或外部变量，可被说明为全局的，或仅在给出说明的源文件范围内有效；一个函数的内部变量可以是自动类的(进入该函数时出现，从该函数出口时消失)，也可以是静态类的(即使从该函数出口，变量值仍保存)。C中还有register说明，告诉编译程序该变量在程序中用得很频繁。通常，编译程序将把这类变量存贮于硬件寄存器中以获得快速存取。但register说明只是一种建议，对编译程序无约束力。

C中的函数均可递归调用。参数的传递是传值方式，但在必要时也可用传递指针的方法来传地址。函数的参数和返回值可以是任何基本类型、指针、结构、联合或枚举类型。数组作参数是通过传递指向该数组的第一个元素的指针来实现的。

C还具有一个预处理器，提供了源程序的并入，条件编译，和符号名及能在一行内定义的函数的宏处理功能。

5. 运行环境

与其它常见的高级语言不同，C语言中不包括任何输入输出语句，它也不提供存贮管理或字符串处理功能。类似地，尽管C最初是为系统程序设计应用设计的，但它只提供了单控制流结构，没有多道程序设计，并行操作，同步或共行程序。上述这些高级功能，均必须由另外的功能块提供。对大多数C程序来说，运行环境是由一个标准的输入输出库提供的。将语言本体与运行环境分开，使C语言成为一个相对简单的语言，便于学习掌握。

由于C语言比其它高级语言更能反映计算机的硬件结构，具有某些通常只有汇编语言才有的功能，因此，C语言在某种意义上说是一种较“低级”的语言。有人把它称为是现代计算机的“可移植汇编语言”[27]。“可移植”指同一程序可在许多不同的机器上运行；“汇编”指C程序通常比较紧凑，运行速度快，并且程序对机内数据有完全的控制。

C语言不是一个强类型的语言。它对数据类型缺乏一致性检测，运行时对数组下标越界

也不作报告，为保证程序的正确性，程序员的负担较重。另外，过于紧凑的格式和函数的不允许嵌套，对规模较大的程序，在一定程度上影响了可读性。这些缺点，前者由于lint工具的存在得到了一定程序的弥补，后者借助于结构程序设计的方法和C语言分别编译的能力来补偿。

总之，从C语言的特点不难看出C语言是一种通用的程序设计语言，特别适合于系统程序设计，具有很强的生命力，在我国也将得到愈来愈广泛的使用。下面我们全面讲述C语言的各个语法成份及程序设计方法，并给出大量的程序实例。

## §1.2 C程序的书写格式、编译和运行

C语言的源程序是什么样的呢？让我们看几个简单的C程序。

**例1-1** 打印字符串的C程序

我们要求终端屏幕上出现如下两行字符串：

```
welcome to UNIX!
****************
```

用C语言来写源程序就是

```
main( )
{  printf("welcome to UNIX!\n");
   printf ("****************\n");
}
```

在UNIX环境下，用系统的编辑程序的各种命令建立这一文件，并存到磁盘上。源文件名以.c为结尾，例如本例用wel.c为源文件名。然后打命令

```
cc  wel.c
```

进行编译。如果源程序没有错误，编译结束就产生一个可执行的文件，文件名是a.out。打命令

```
a.out
```

就使它运行，在屏幕上印出所期望的那两行字符串。

现在让我们来看看源程序的结构和书写形式。

任何C程序都由一个主函数和若干个(或零个)其它函数组成。主函数的名字一律用main，其它函数的名字由你自己命名(须符合标识符的有关规定，见下文)。函数名后面紧跟着的圆括弧用来放参数。参数传递是函数之间数据联系的一种方式，这与FORTRAN、ALGOL、PASCAL语言有相似之处。例1-1的main( )表示主函数为无参函数，但圆括号不能省略，因为它们是函数的标志。

花括号{和}是函数体的界限，也可用作语句括号，它们和ALGOL、PASCAL中的begin、end的作用类似。函数体内通常有说明部分和语句部分。例1-1没有任何变量，因而不需要说明部分，仅由两个语句构成。每个语句须以分号结尾。书写方式为自由格式，一行可以写多个语句，一个语句也可分成几行写，但一个标识符不能分成两行。

例1-1的两个语句都是按格式输出。printf是标准输入/输出库提供的“按格式输出函数”的名字，这里调用该函数两次。函数名后面必须有括号，其中放参数，也称“参数表”。

双引号中的字符序列叫做字符串或字符串常数。其中要特别注意反向斜线\，\n代表换行字符，应看成只代表一个字符，它使光标移动到下一行的左端，如果例 1-1 中的\n都去除，则输出就不分行，且运行结束时光标也不移到下一行左端。

在C语言中为表示一些非图形字符或看不见的字符，经常用反斜线为首的转义序列，如：

| | |
|---|---|
| \n | 换行 |
| \t | 横表 |
| \b | 退格 |
| \r | 回车 |
| \f | 换页 |
| \\ | 反斜线本身 |
| \′ | 单引号 |
| \ddd | 位模式(d为8进制数字)，至多三个数字 |
| \″ | 双引号 |

**例 1-2**

```
main( )/*It is main function */
{  printf("Welc ome to UNIX! \n");
   stars( );
}
stars( )/*The function is to print a lot of stars */
{  printf("**********\n"); }
```

该程序的运行结果与例 1-1 相同。但它们打印一行星号是由函数 stars( )来实现的。主函数中出现调用 stars( )，就转到该函数的定义段；stars( )的函数体内只有一个语句——调用 printf，当执行完这语句，又返回到主函数，程序结束。C程序中各函数的书写次序可以是任意的，main( )也可以放在 stars( )函数体结束后，而且各函数可以分布在几个文件中，并可分别编译。如果例 1-2的两个函数分别在文件 welm.c和 wels.c中，用下述命令编译

```
cc  welm.c wels.c
```

产生的浮动目标代码将在文件 welm.o 和 wels.o 中，并将它们装配在可执行文件 a.out中。如果你要修改其中一个文件，例如把 welm.c 修改，而 wels.c不修改，只要用命令

```
cc  welm.c wels.o c
```

重新编译 welm.c，并将结果与原来的 wels.o装配在一起。大型程序通常都分几个文件设计，并充分利用已有的模块。例如上述 wels.o 可以与其它需要调用它的函数组装在一起。

/* 和 */是一对注释的开始符号与结束符号。程序中可以有任意个注释，并可出现在允许空格出现的任何地方，因为一个注释等价于一个空格。注释用于帮助读者阅读和理解程序，在编译时被忽略掉，即并不产生目标代码。人们经常在程序的开头写有说明整个程序功能的注释，在每个函数的开头写有该函数功能的注释。在语句行的末尾或其它地方进行他需要的任意注释。

在UNIX 环境下用命令

cc 文件名1 文件名2 … 文件名n

产生的可执行文件名必为a.out。显然，最新形成的可执行文件会把a.out中原来的内容冲掉。如果要把可执行文件保留在磁盘上而不被下次形成的a.out冲掉，就应该使用带任选项-o的编译命令。例如我们要把例1-1的可执行代码保留在名为wel的文件中，则打命令

cc -o wel wel.c

这时-o后跟着的字符串wel就是可执行代码的文件名，因而运行时应打命令

wel

cc命令还可以带其它任选项(也称为"调用开关")。见附录二或使用手册。

## §1.3 变量及变量的类型说明

**例1-3** 三个整数相加的程序

```
main( )/*The sum of three integers */
{  int a, b, c, sum; /* declaration */
   a = 1; b = 2; c = 3;
   sum:=a+b+c;
   printf("sum is %d\n", sum);
}
```

与PASCAL等语言类似，C语言程序中的每个变量必须在使用前进行类型说明或定义(此外还有存贮类的说明，见第六章)，使编译程序便于对变量分配存贮单元。变量的基本类型有四种：

int　　代表整型；
char　　代表字符型；
float　　代表单精度浮点型；
double　　代表双精度浮点型。

上述表示类型的四个名字都属于保留字，也就是说它们具有专门的意义和用途，用户不能用它们来表示其它变量或函数的名字(即自定义的标识符)，见附录一。

此外，还有三个可用于int型的限定词：short、long和unsigned，例如：

long int a; 表示长整型变量a，
short int b; 表示短整数变量b，
unsigned int c; 表示无符号整数。

在short、long、unsigned后面的int可以省略。一般说，long类型的变量占的存贮空间比int型长，因而允许的最大值较大，short型变量占的存贮空间比int型短，允许的最大值也就小些。例如某些机器上整型数占2个字节，short型占1个字节，long型占4个字节，详见§2.1。类型说明的形式为

类型名　　变量名；

例如某程序有以下说明：

```
int    a;
int    x;
```

```
float   y;
char    c;
```

对于相同类型的变量，可以用如下形式：

类型名　变量1，变量2，　…，变量n；

以上说明因而可以等价地表示为

```
int     a, x;
float   y;
char    c;
```

例1-3中4个变量具有同一类型，因而采用了这种较紧凑的表示法。

本例中出现了赋值语句，请注意赋值号为=(与FORTRAN相同)，而不同于PASCAL中的:=。

标识变量、常量、函数、程序、数据类型、存贮类名字的字符序列都称为标识符，保留字是说明固定含义的标识符、标识符构成须符合以下规则：

标识符以英文字母为首，后面可跟任意个(也可为零个)字符，这些字符可从英文字母、下划线—、数字中任选。大小写字母作为不同的字母。C语言中除了表示常量的标识符用大写外，其它标识符(包括保留字)一般都用小写字母。在大部分C语言的具体实现中，标识符的前面8个字符是有意义的，也就是说多于8个字符的那些字符不能分辩，例如变量名1 2345678与a12345679被认为是同一变量。函数名与外部变量名要用于汇编程序和装入程序，它们的字符个数受到的限制较多：

| | |
|---|---|
| DEC PDP-11 | 七个字符，大小写加以区别； |
| Honeywell 6000 | 六个字符，大小写不加区别； |
| IBM 360/370 | 七个字符，大小写不加区别； |
| Interdata 8/32 | 八个字符，大小写加以区别。 |

C语言的关键字有28个，见附录一。

为了使程序易读易理解，变量名尽量选取能表达含义的英文词或汉语拼音词，当词较长时可选其中一节或缩写。

例1-3的函数调用printf具有两个参数。以双引号为界的是第一个参数，也称控制字符串，逗号后的sum是第二个参数。控制字符串中每出现一个%符号就表示其它参数(第二、第三、…参数)将用%后指示的格式打印，例中%d就表示sum的值将用十进制数的格式打印。因此例1-3运行结果为

```
sum is 6
```

**例1-4**　浮点数运算实例

某商店将每斤价为1.80元的硬糖35斤与单价为2.10元的软糖15斤混杂成什锦糖果出售。编一程序计算什锦糖的价格与这批糖果的总价并打印出来。

我们以元为单位，则单价与总价都须用浮点数类型，源程序如下：

```
main( )
{  float total, price;
   total = 1.80 * 35.0 + 2.10 * 15.0;
   price = total/(35.0 + 15.0);
```

```
    printf("total:$%-6.2f\n", total);
    printf("price:$%4.2f\n", price);
}
```

例中的变量都是浮点型，因而相应的常数也须用浮点型来表示。1.80、35.0等常数就是浮点型常数。对于+、-、*、/、一元减运算符-，当运算对象为整型时，运算结果也为整型，而两个运算对象之一或全部为浮点型时，则运算结果为浮点型。例如5/8得到的商为整数0，而5.0/8.0得到的商为0.625000。本例需要进行浮点运算，因而常数都用浮点型。事实上这里用35和15来代替35.0和15.0，并不影响计算效果。第二章将系统地叙述类型转换规则。

例1-4中，控制字符串中有格式转换描述%-6.2f和%4.2f。f代表浮点数，%4.2f表示这个浮点数至少用4个字符宽度的空间印出，小数点后为两位数字。如果该数的宽度少于4，就在左边填空格符，保证宽度为4，这叫做"右侧对齐"。%-6.2f的总宽度至少为6，小数点后面两位，负号表示"左侧对齐"，就是说当数的宽度小于6，在右侧填空格符。当数的实际宽度超过格式描述指定的域宽时，按实际宽度印出。其它类型输出同样可指定域宽。详见§1.6。

## §1.4 符号常数

我们把例1-4中出现的几个常数用大写字母组成的标识符代替，程序格式就成了如下形式：

**例1-5**

```
#define HWEIGHT 35.0
#define SWEIGHT 15.0
#define HPRICE  1.80
#define SPRICE  2.10
main( )
{  float total, price;
   total=(HPRICE*HWEIGHT+SPRICE*SWEIGHT);
   price=total/(HWEIGHT+SWEIGHT);
   printf("total:$%-6.2f\n", total);
   printf("price:$%4.2f\n", price);
}
```

运行结果

```
total:$94.50
price:$1.89
```

程序中大写字母组成的符号名HWEIGHT，SWEIGHT等称为符号常数，通常在程序开始处用#define来构造符号常数，使它定义为特定的字符串。编译程序将对程序中以后出现的这些名字都替换成相应的字符串(例如HWEIGHT用串35.0替换……)。但是双引号内的字符串不被替换，若在例1-5中增加语句行printf("HWEIGHT")，那末这个HWEIGHT不会被替换，执行时仍印出HWEIGHT。

采用符号常数使程序易读，因为定义的名字往往取有含义的词，如本例中 HWEIGHT 可看出“重量”的意思，前缀 H 与“硬”(hard)有关。同时程序修改方便，若要把硬糖重量更改为 25，只要在程序开始处修改一次

```
#define HWEIGHT 25.0
```

而不必在程序中多处寻找，也不致将同一数值的其它变量给误改了。

第八章讨论C的预处理程序，那时读者将会对符号常数有更深的理解。

## §1.5 常　量

上节例子中已出现了十进制整数型常量(如例 1-3 中的 1,2,3)和十进制浮点型常量(如例 1-4 中的 1.80，35.0 等)。常量(或称“常数”)不止这些形式，而有下列多种。

### 一、数值常量

1. 十进制常数

如 71　-3.1　.3　3.9e-7　4　3.1E2 等均为合法的；而 -.E　E3　1E1.5　3E 都是非法的。这里要注意：1)浮点数的小数点两旁至少一边要有数字；2)E 或 e 的左边必须有数字，E或e的后面必须有整数；3)每个浮点数是双精度的，因此记号 e 既表示 float又表示 double。超出int型数值范围的整数可用长整型常数表示，以L结尾的形式，如37100L.

2. 八进制常数

整型常数前有一个前导零就隐含地表明它是八进制的，例如十进制数31可写成八进制数037。在八进制常数后也可以带 L表示长常数。如 0123456L。

3. 十六进制常数

整型常数前有前导的 0x 或 0X 表示十六进制常数。例如 0x1f或0X1F都是合法的，其值就是十进制数31。

### 二、字符常数

字符常数(或字符常量)是用单引号括起来的单一字符，例如'p'，'q'等。它在机器中占一个字节。一个字符常数的值是该字符在机器的字符集中的编码数字值。对于采用 ASCII 字符集的机器，字符 '0' 的值为48，而在采用 EBCDIC字符集的机器中却为240，写成'0'就使得程序同该特定值无关。

字符常数可以象整数一样参与数值运算，这是由于它的值本来就是一个整数的缘故。我们以后讲类型转换时还会说到：运算时char型自动转换成 int 型。

§1.1 节介绍了用转义序列表示非图形字符，实际上这也是字符常数，如 '\n'(换行)，'\0' (具有值的零字符，注意不同于 '0')。凡是能存放在一个字节中的量都可以作为字符量对待。例如：

```
char quest, newline, flags, formfeed;
guest = '!';
newline = '\n';
flags = 077;    等价于flags = 63; 或flags = '?';
```