曾垂昌 付恩平
李忠国 马少泊 编著

计算机教程

C语言实用教程

SHIYONGJIAOCHENG

科学出版社

C语言实用教程

曾垂昌　付恩平
李忠国　马少泊　编著

科学出版社
2000

内 容 简 介

C 语言是一种功能很强、应用广泛的系统程序设计语言。本书全面系统地介绍了 C 语言及其程序设计方法和应用，共十二章，内容包括 C 语言的数据类型、各种语句、各种结构类型数据以及程序设计的基本方法、技巧和实际应用。本书突出 C 语言的特点，并通过大量的程序实例详细讲述了 C 语言的编程方法。

本书深入浅出，通俗易懂，内容充实，实用性强，是学习和掌握 C 语言及其应用的良师益友。本书是针对大学本科学生编写的，但也适合于其他有关人员阅读。

图书在版编目(CIP)数据

C 语言实用教程/曾垂昌等编著. -北京：科学出版社，2000
ISBN 7-03-007673-7

Ⅰ.C… Ⅱ.曾… Ⅲ.C 语言-程序设计-教材 Ⅳ.TP 312

中国版本图书馆 CIP 数据核字（1999）第 65138 号

科学出版社 出版
北京东黄城根北街 16 号
邮政编码：100717
北京科普印刷厂 印刷
新华书店北京发行所发行 各地新华书店经售
*
2000 年 3 月第 一 版 开本：787×1092 1/16
2000 年 3 月第一次印刷 印张：24
印数：1—5 100 字数：563 000

定价：32.00 元

（如有印装质量问题，我社负责调换(环伟)）

前　言

C 语言是当前国内外非常流行的一种程序设计语言。它功能强、效率高、简洁灵活、可移植性好，既具有高级语言的优点，又具有低级语言的许多特点，因此，在软件工程领域里越来越受到人们的普遍重视，其应用十分广泛。

C 语言是程序员特别是系统程序员必须要掌握的一种语言，现在，学习使用 C 语言的人越来越多。但由于 C 语言内容丰富，概念、规则复杂，编程灵活，使许多初学者感到吃力。本书根据作者多年教学经验和用 C 语言编制系统软件的体会，针对 C 语言的特点，从应用出发，详细解说了 C 语言的语法和程序设计，仔细分析编程思想，以减轻读者学习上的困难。本书通俗易懂，适合课堂教学和自学。本书主要针对大学本科学生和软件开发人员而编写，也可以作为二级考试教材。

本书突出了以下几个特点：

1. 本书对有关程序的讲解分两种思路进行，一种是先给出完整的程序，然后对程序进行详细解说；另一种是先提出问题，然后分析问题，得出求解问题的算法和思路并且分别编写出各个独立的程序功能块，最后将这些功能块合成为一个完整的程序，以此启示读者学习编制程序的方法。
2. 本书采用作者独创的按优先级分层说明的方法，利用图文并举的方式对 C 语言中的重点和难点进行了详细的分析和说明，使读者易于融会贯通，举一反三。
3. 本书从 C 语言的本质出发，突出 C 语言的特点，详细解说了 C 语言对存储地址的操作，并结合具体应用，介绍各种实例。同时，对 C 语言的精华部分——指针、数组和结构作了深刻的阐述。
4. 本书对于递归调用这个难点，进行了深入的讨论，从递推原则和归纳方法出发，逐步分析，最后总结出递归程序编制的基本思路和方法，并以实例说明其实现技巧。
5. 本书以 ANSI C 标准为主线，基于 Turbo C 3.0 进行讲解，所有例题都上机调试通过。

为了帮助广大读者学习，我们还正在制作与该教材配套的多媒体学习光盘，该光盘以适合课堂教学为宗旨，模拟课堂教学的方式，采用多种生动形象的教学方法和手段，进行多媒体教学，争取提高教学效果和教学效率，同时，也给自学者创造良好条件。

参加本书编写的人员还有陶扬、李真子、袁安心、张志祥、杨鹏、杨华昕、李一华、刘海涛、白雪飞等。

在本书的编写和出版过程中，还得到了张景生、贾可荣、周亦明、蒋东星、李一凡、彭建明、郭海涛、刘欣等同志的关心和帮助，他们对本书的录入、校对、编排和画图等工作都付出了艰巨的劳动。特别是张景生同志对本书的结构和内容提出了宝贵的意见。作者在此感谢所有为本书的出版给予过支持的人们。

编　者

1999 年 6 月

目　录

第一章　C语言概述 …… (1)
1.1　C语言的来源和特性 …… (1)
1.1.1　C语言的来源 …… (1)
1.1.2　C语言的特性 …… (2)
1.2　C语言简单程序介绍 …… (2)
1.2.1　main（）函数 …… (2)
1.2.2　常量、变量和关键字 …… (4)
1.3　数据类型 …… (9)
1.3.1　整型数据 …… (10)
1.3.2　字符型数据 …… (12)
1.3.3　浮点（float）类型 …… (14)
习题 …… (15)
第二章　数据的输入输出 …… (18)
2.1　格式化输出库函数 printf（） …… (18)
2.1.1　“控制字符串”中的 Type 字段 …… (19)
2.1.2　关于［类型大小］（［size］）字段 …… (22)
2.1.3　关于［寻址模式］（［Address-mode］）字段 …… (23)
2.1.4　关于［标记］［域宽］［.精度］（［flag］［width］［.precision］）字段 …… (24)
2.2　格式化输入函数 scanf（） …… (28)
2.3　scanf（）函数的附加功能 …… (36)
2.4　printf（）和 scanf（）的返回值 …… (38)
2.5　输入输出宏指令 getchar（）和 putchar（） …… (40)
2.6　getch（），getche（）以及 putch（） …… (44)
习题 …… (45)
第三章　C语言的运算符和表达式 …… (50)
3.1　C语言的运算符 …… (50)
3.1.1　算术运算符和算术表达式 …… (50)
3.1.2　赋值运算符和赋值表达式 …… (53)
3.1.3　关系运算符及关系表达式 …… (56)
3.1.4　逻辑运算符与逻辑表达式 …… (57)
3.1.5　自增1自减1运算 …… (59)
3.1.6　sizeof 运算符 …… (60)
3.1.7　多元赋值运算符 …… (61)
3.1.8　移位运算 …… (62)

3.1.9 位逻辑运算 …… (63)
3.1.10 条件运算符 …… (65)
3.1.11 逗号运算符 …… (65)
3.1.12 强制类型转换符 …… (66)
3.2 优先级和结合性 …… (66)
3.3 算术运算中的类型转换 …… (68)
习题 …… (69)
第四章 流程控制语句 …… (72)
4.1 if 语句 …… (72)
4.1.1 简单 if 语句 …… (72)
4.1.2 嵌套 if 语句 …… (75)
4.2 switch... case 语句 …… (79)
4.3 程序举例 …… (83)
4.4 循环语句 …… (86)
4.4.1 while 语句 …… (86)
4.4.2 do... while 语句 …… (89)
4.4.3 for 语句 …… (90)
4.4.4 循环语句的嵌套 …… (96)
4.5 break 语句和 continue 语句 …… (99)
习题 …… (102)
第五章 函数 …… (111)
5.1 函数的定义和说明 …… (111)
5.1.1 函数的定义 (Function Definetions) …… (111)
5.1.2 函数说明 (Function Declarations) …… (114)
5.2 函数调用及函数参数 …… (116)
5.2.1 无参调用 …… (116)
5.2.2 有参调用 …… (117)
5.2.3 函数的返回值 …… (119)
5.3 变量存储类别 …… (120)
5.3.1 自动变量 (auto) …… (121)
5.3.2 静态变量 (static) …… (123)
5.3.3 外部变量 (extern) …… (125)
5.3.4 寄存器变量 (register) …… (127)
5.4 函数的嵌套调用与递归调用 …… (128)
5.4.1 函数的嵌套调用 …… (128)
5.4.2 函数的递归调用 …… (129)
习题 …… (134)
第六章 编译预处理 …… (141)
6.1 预处理命令 …… (141)

6.2 宏定义 …… (141)
6.3 文件包含 …… (147)
6.4 条件编译 …… (149)
习题 …… (152)
第七章 数组与字符串 …… (156)
7.1 一维数组的说明及赋值（Array Declaration And Initiation） …… (156)
7.1.1 一维数组的说明 …… (156)
7.1.2 数组的初值设置 …… (157)
7.1.3 程序举例 …… (159)
7.2 多维数组 …… (164)
7.2.1 二维数组的说明及其初始化 …… (165)
7.2.2 应用举例 …… (168)
7.3 字符串与数组 …… (172)
7.3.1 字符数组的说明和初始化 …… (172)
7.3.2 字符数组的引用 …… (174)
7.3.3 字符串库函数 …… (175)
7.3.4 二维字符串数组 …… (183)
习题 …… (184)
第八章 指针 …… (188)
8.1 指针的概念 …… (188)
8.1.1 什么是指针 …… (188)
8.1.2 指针变量的说明 …… (189)
8.1.3 给指针变量赋值 …… (191)
8.2 指针的运算 …… (195)
8.2.1 取地址运算符 & 和间接取运算符 * …… (195)
8.2.2 指针的算术运算 …… (198)
8.2.3 指针运算举例 …… (199)
8.3 指针作参数（call by address） …… (201)
8.3.1 两种传值方式的差别 …… (201)
8.3.2 在什么情况下指针作为函数参数 …… (202)
8.4 指向指针的指针（二维指针和三维指针） …… (207)
8.5 指向数组的指针（A Pointer And To An Array） …… (210)
8.5.1 程序举例 …… (210)
8.5.2 指针与二维数组的关系 …… (211)
8.5.3 访问二维数组元素 a [i] [j] 的几种方法 …… (213)
8.6 指向字符串的指针 …… (217)
8.6.1 一个字符串常量代表一个地址 …… (217)
8.6.2 指向字符的指针数组 …… (220)
8.7 指向函数的指针 …… (222)

8.7.1 指向函数的指针变量的说明及赋值 …… (223)
8.7.2 用指向函数的指针变量调用函数 …… (224)
8.7.3 函数指针变量作为函数参数 …… (225)
8.8 命令行参数 …… (227)
8.9 用 typedef 定义新的类型名 …… (228)
习题 …… (229)
第九章 结构、联合与枚举 …… (236)
9.1 结构说明 …… (236)
9.1.1 有关结构概念的一个实例 …… (236)
9.1.2 结构类型定义的一般形式 …… (237)
9.1.3 结构型变量的定义 …… (238)
9.1.4 给结构变量分配存储空间 …… (240)
9.2 结构初始化 …… (241)
9.3 对结构成员的引用 …… (242)
9.4 结构数组 …… (247)
9.4.1 结构数组的定义 …… (247)
9.4.2 结构数组的初始化和引用 …… (249)
9.5 结构型指针 …… (253)
9.5.1 结构型指针变量的定义 …… (253)
9.5.2 用结构型指针变量引用结构的各成员 …… (254)
9.5.3 结构变量运算小结 …… (258)
9.6 结构型变量作函数参数 …… (259)
9.6.1 将实参的数值传送给形参 (call by Value) …… (259)
9.6.2 将实参的地址值传送给形参 (call by Address) …… (259)
9.7 动态数据结构 …… (264)
9.7.1 线性链表的一般概念 …… (264)
9.7.2 线性链表的建立和输出 …… (267)
9.7.3 线性链表的插入操作 …… (270)
9.7.4 线性链表的删除操作 …… (272)
9.8 联合 …… (273)
9.8.1 联合类型及该类型的变量定义 …… (274)
9.8.2 联合型变量所占存储空间 …… (274)
9.8.3 对联合变量及其成员的操作 …… (275)
9.9 位域 …… (280)
9.10 枚举类型 …… (281)
9.10.1 枚举类型及其变量定义的一般格式 …… (281)
9.10.2 枚举类型的运算 …… (282)
习题 …… (283)
第十章 文件 …… (288)

10.1 文件概述 …… (288)
10.1.1 DOS 环境下的文件类型 …… (288)
10.1.2 缓冲文件系统（高级 I/O）和非缓冲文件系统（低级 I/O） …… (289)
10.2 缓冲文件系统 …… (291)
10.2.1 文件（FILE）类型指针 …… (291)
10.2.2 文件的打开和关闭 …… (291)
10.2.3 字符 I/O …… (294)
10.2.4 文件的结束判断及出错检测（feof 和 ferror 函数） …… (298)
10.2.5 字符串 I/O …… (301)
10.2.6 格式 I/O …… (306)
10.2.7 数据块 I/O 函数 …… (308)
10.2.8 文件的定位与随机读写 …… (310)
10.3 非缓冲文件系统（低级 I/O） …… (315)
习题 …… (318)
第十一章 Turbo C 的文本屏幕处理 …… (323)
11.1 基本概念 …… (323)
11.1.1 两种屏幕操作模式 …… (323)
11.1.2 窗口的概念 …… (323)
11.2 文本屏幕处理函数 …… (323)
11.2.1 文本输入输出及管理函数 …… (324)
11.2.2 窗口和模式控制函数 …… (328)
11.2.3 属性控制函数 …… (329)
11.2.4 状态查询函数 …… (332)
第十二章 图形处理 …… (334)
12.1 图形操作的基本概念 …… (334)
12.1.1 基本概念 …… (334)
12.1.2 进入图形系统 …… (334)
12.2 基本图形处理函数 …… (337)
12.2.1 图形系统控制函数 …… (337)
12.2.2 绘图及填充函数 …… (340)
12.2.3 屏幕和视口的管理函数 …… (358)
12.2.4 图形方式下的文本输出函数 …… (363)
12.2.5 颜色控制函数 …… (367)
12.2.6 错误处理函数 …… (372)
12.2.7 状态查询函数 …… (373)
附录 常用字符与 ASCⅡ码对照表 …… (374)

第一章　C 语言概述

C 语言是一种通用的计算机语言，它不仅能在 UNIX 系统下运行，还能在许多操作系统(如 PC-DOS, CP/M, VMS 等)支持下，在各种 16 位和 32 位计算机上运行。C 语言是介于低级语言(如汇编语言)与高级语言(如 COBOL 语言)之间的一种语言，有人称之为中级语言，它适合于编写系统程序，故又称为系统程序设计语言；同时它也完全适合编写不同领域中的应用程序，往往充当一些程序的主语言(如某些数据库语言的主语言)。历经多年，人们借助于 C 语言已经开发了各种各样的系统程序和应用程序，如图形、通信、工程、数据库、CAD/CAM 以及字处理方面的软件。本书将逐一地详细解说 C 语言的有关基本语法，介绍某些库函数，为读者掌握和应用 C 语言打下良好的基础。

本章介绍的主要内容：

- C 语言的来源和特点
- C 语言程序的构成
- 构成 C 程序的各要素

1.1　C 语言的来源和特性

1.1.1　C 语言的来源

追根求源，C 语言的开发大致经过如下几个阶段(如图 1.1 所示)。

1967 年，Martin Richards 开发了 BCPL(Basic Compound Programming Language)语言。C 语言的许多重要观点和思想源于 BCPL 语言。到现在为止，C 语言仍间接受到 BCPL 语言的影响(通过 B 语言)。

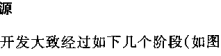


图 1.1　C 语言的来源

1970 年，Ken Thompson 开发了 B 语言。B 语言是基于 BCPL 设计的一种语言，是在 DEC PDP-7 这样一台陈旧的机器上，为描述 UNIX 系统开发出来的语言。它比较简单，缺乏丰富的数据类型，以字编址，所以 B 语言并没有流行。

1972 年，Dennis Ritchie 在贝尔实验室将 B 语言修改成 NB 语言，也就是后来的 C 语言。可见，C 语言是为描述 UNIX 系统而开发经修改形成的一种语言，它和 UNIX 是一对亲密的"伙伴"。长期以来，它们互相支持，互相依赖。UNIX 系统及在它之上运行的程序(如数据库管理系统和网络通信程序)都是用 C 语言编写的；而 C 语言编写的程序又可以调

用 UNIX 的强大功能,使其自身更加完善。

1983 年,美国国家标准化协会召开会议,讨论 C 语言的新标准,这个标准被称为 ANSI 标准或 ANSI C。1988 年完成了 ANSI C 标准的制定,现在流行的各种 C 编译支持该标准的绝大部分特性。本书以 Turbo C 3.0(属 ANSI C 标准)为基准进行编写。

1.1.2 C 语言的特性

C 语言有其明显的特性,其突出的特性是具有指针变量,能对存储地址进行操作。可以说指针是 C 语言的生命,对存储地址进行操作是 C 语言的精华。C 语言与其他高级语言的区别也正体现在这里。有了指针变量并能对存储地址进行操作,就可以取变量的地址,也可以通过指针访问变量,易于编写系统程序。此外,C 语言还具有如下特性:

(1)有丰富的库函数。C 语言不能对组合对象如字符串、集合、数组和表进行操作,它没有定义对这些整体对象进行操作的语句,没有定义动态存储分配和释放语句,也没有像 READ 和 WRITE 这样的输入输出语句。所有这些功能都是明显地以调用库函数的方式实现的。C 编译提供了大量的库函数,包括系统调用、算术运算,以及对屏幕、对文件的操作等等。程序员应用这些函数,可以大大简化其开发软件的工作量,提高开发效率。不仅如此,程序员还可以设计自己的程序库,以加强他们对应用程序的开发。

(2)C 语言提供了预处理器。程序可以利用宏指令以提高程序的可读性、可移植性和可靠性。

(3)C 语言提供的数据类型除了字符型、整型和浮点型这些基本类型而外,还提供了组合类型(a hierarchy of derived data type),如数组、结构和联合等。用户可以用这些类型定义复杂的数据结构以适应各种需求。

(4)C 语言具有低级语言的功能,能对位进行运算;同时也具有高级语言的语句,具有强有力的结构化控制流的特点,便于写结构化程序。

(5)C 语言程序由文件组成,每个文件由函数组成,函数间可以嵌套调用和递归调用。

借助于寄托在某种操作系统(最理想的是 UNIX)上的 C 语言,可以开发各式各样的系统程序和应用程序,如图形、通信、工程、数据库、CAD/CAM 以及字处理方面的软件。

1.2 C 语言简单程序介绍

1.2.1 main( )函数

什么叫程序?程序是一组按某种规则组合在一起的指令的集合,它能使计算机按照这些指令产生相应的操作。程序一般是用某种语言编写的。设计不同的语言具有不同的目的。BASIC 语言适用于初学者,FORTRAN 语言适用于公式计算,PASCAL 语言适用于教学,而 C 语言则是功能强大的专业化语言。许多系统软件都是由 C 语言编制的,UNIX 系统几乎全由 C 语言编制。

C 语言程序结构是怎样的呢?它由函数组成,其中必有一个而且只有一个主函数(main( )函数)。

例 1.1 main( )函数举例。

```
void main(void)
```

```
{
printf("Hello!");
}
```

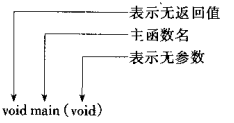
对以上函数的说明：

①main 是个标识符，它与后面的()(即 main( ))结合，表明 main 这个标识符是一个函数名。()内的 void 代表该函数没有参数，main( )之前的 void 代表该函数没有返回值，传统的 C 语言不必写 void，ANSI C 要求写上 void。

②void main(void)后有一对花括号{ }，花括号内由 C 语言的语句组成，成为函数体，左"{"表示块的开始，右"}"表示块的结束。块内可再有块。

③块内的每个语句以分号";"结束，这里的"printf("Hello");"就是个语句，它调用 C 语言的内部库函数 printf( )(别忘了它后面的";")，其功能是原样显示"Hello!"。程序的输出为：Hello!

主函数(即 main( ))的结构如下：



main( )函数是 C 语言的面有函数，没有 main( )函数不成为 C 程序。每个 C 程序都从 main( )函数开始执行，面不管 main()函数在程序中什么位置出现。

C 程序书写格式是自由的，例如：

```
void main(void)
{
  printf(.....);
}
```

或

```
void main(void){printf(.....);}
```

都是合法的，但读者一定要按照第一种格式书写，这样的书写易读、易维护。计算机软件的开发，往往要许多人用多年的时间才能完成，所以为了自己也为了别人能易于读懂程序常在合适的地方加上注解。

例 1.2

```
/* 0102.c */
# include <stdio.h>
# include <conio.h>
void main( void )
{  int  i = 1;     /* 初始化变量 i 并赋值 */
   int  j,k;      /* 说明变量 j,k */
```

预处理宏指令及定义函数不加分号";"

说明部分

```
    j=2;              /*给变量 j 赋值*/
    k=i+j;            /*求 i 与 j 之和并赋值给 k*/
    printf("i=%d,j=%d,k=%d\n",i,j,k);
    getch();
}
```
执行部分

程序解说：

① 说明部分：定义变量 i,j,k 并给 i 赋值，变量必须先说明(或叫定义)后使用。

② 执行部分：求 k (k=i+j)。

printf(.....)； 显示执行结果。

getch( )；使用户可直接看到执行结果，执行后按任意键则回到编辑界面。

③ 预处理宏指令：这里 stdio.h 和 conio.h 是两个头文件，函数 printf( )包含在 stdio.h 中，函数 getch( ) 包含在 conio.h 中，# include < ..... > 不是 C 语言语句，是预处理指令，它以 # 开头，最后不加分号，它将 stdio.h 和 conio.h 装入内存，其目的是使 printf( )和 getch( )可以被主函数(或其他用户编写的函数)调用，通常这些预处理宏指令放在程序最开始。

④ 不论是说明语句还是执行语句都以分号";"结束。/* ..... */内的部分为注解，只供用户阅读，系统对之"视而不见"，不予理会，但对阅读程序十分有用。

可以进一步给出 main()函数的结构如下：

```
预处理宏命令      /*必须以#开头*/
void main(void)   /*函数*/
    {
      说明语句;    /*注释*/
      执行语句;    /*注释*/
    }
```

一般地，一个典型的 C 语言程序由多个源文件组成，每个源文件由头文件说明和函数组成，函数中有一个也只有一个主函数。如图 1.2 所示。

1.2.2 常量、变量和关键字

常量、变量、关键字等是 C 语言的重要成分。下面分别讨论：

1. 常量

在程序运行过程中(从开始到结束)，其数值不能由程序改变的量叫常量。可分为：

整型常量 不带小数点的数，如：2,4,5；在整型常量中除有十进制数外还有八进制数与十六进制数。例如：323(数字为 0 到 9)表示十进制数，可正可负；0323(使用数字 0 到 7，最前面的一个字符为 0)表示八进制数，只能是无符号的整数(相当于正数)；0x323(数字开头为 0x)表示十六进制数，只能是无符号数(相当于正数)，使用字符 0 到 9 与 a 到 f。另外还有一种称之为长型的常量，例如：12L(或 12l)、014L(014l)、0Xel(0xel)它们分别表示十进制、八进制和十六进制长型数，在计算机存储器中占用 4 个字节。

浮点型常量 带小数点的数或用指数形式表示的数，如：3.14,3.0,13E02(= 1.23 *

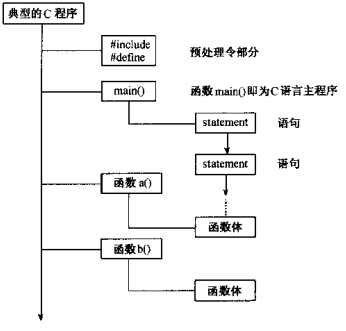


图 1.2　典型的 C 语言程序结构

10^2),3. 和 1230E－01(＝1230＊10^{-1})。

字符常量　用单引号括起来,单引号内只能是一个字符这样的一个量叫字符常量,如:'A'、'a'、'1'。

注意:　'1'和 1 是不相等的,'1'是字符,其 ASCII 编码为 49,而 1 是整数。

C 语言中有一种用"\"表示的字符,称为转义字符或叫 ESC(Escape Seqnence)字符,如:"\n"表示换行符,"\b"表示退格符,"\\"表示反斜杠,"\'",表示单引号。见表 1.1。

表 1.1 中的"\ddd"是三位八进制数 ddd 所对应的 ASCII 码表中的字符(最多 3 位八进制数,也可以用 2 个或 1 个八进制数表示相应字符)。例如:\40＝空格字符,\61＝1,\101＝A。

表 1.1　转义字符

字符形式	功能	字符形式	功能
\n	换行	\f	走纸换行
\t	横向跳格	\\	反斜杠字符
\v	竖向跳格	\'	单引号字符
\b	退格	\ddd	1 到 3 位八进制数所代表的字符
\r	回车	\xhh	1 到 2 位十六进制数所代表的字符

注意:如果 ddd 中有一位超过 7,则这位和这位以后的字符原样输出,"\"不起作用。

例 1.3　\ddd 格式举例。

```
void main(void)
```

```
{
char c1,c2,c3;
c1='\040';c2='\061';c3='\101';
printf("%c,%c,%c",c1,c2,c3);
}
```

输出：,1,A

例 1.4 \ ddd 中有一位超过 7 的例子。

```
void main(void)
{
printf("%c",'\8');
}
```

输出:8

因为 8 不是八进制数，“\”不起作用。

对于字符'\ xdd'中 x 标记后面的 dd 为十六进制数,它等于十六进制数 dd 所对应的 ASCII 码字符。例如：\ x20 = ' '; \ x31 = '1'; \ x41 = 'A'。同样,dd 有一个为非十六进制数字时,则它本身及它之后原样输出。

例 1.5 \ xdd 格式举例。

```
void main(void)
{
printf("%c, %c, %c\n",'\x20','\x31','\x41');
printf("%c, %c, %c\n",'\%','\\','\'');
}
```

输出： ,1,A

%,\, '

ESC 字符 \ n, \ b 等的功能见表 1.1,其中包含了若干控制字符。

字符串常量 用双引号括起来的连续字符序列称为字符串常量。例如:"This is a sample.";"3219876";"a"。C 语言中字符串以空字符“\ 0"(ASCII 码为 0)结束,因此"a"实际上占有两个存储单元[a|\0],因此它为字符串常量。""也作为字符串常量,表示双引号之间没有任何字符。即为空字符。空字符的 ASCII 码为 0,它占有一个字节空间,可以表示成 \ 0 或 0 或 NULL,NULL 在头文件 stdio. h 中定义。

符号常量 在程序中往往先定义一些符号常量,然后再引用这些符号常量。

例如,用如下的宏指令定义符号常量：

```
#define   PRICE      30
#define   MAXNUM     100
```

定义后,如果程序中用到 PRICE 时,系统会自动用 30 代替之,同样,程序中若出现 MAXNUM,自动用 100 代替之。

例 1.6 符号常量举例。

```
#define   PRICE     30
#define   MAXNUM    100
void main (void)
```

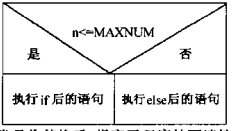
```
{
int n;
float total;
printf("Input integer n: \n");
scanf("%d",&n);
if(n < = MAXNUM)
{
 total = n * PRICE;
 printf("%d %f \n",n,total);
 }
else
  printf("输入的数量%d.超过规定范围%d \n",n,MAXNUM);
}
```

在这个程序的 main()函数中出现的 PRICE 和 MAXNUM 分别被 30 和 100 代替,然后再进行运算。PRICE 和 MAXNUM 称为符号常量。

程序的功能为在最大数量规定范围内购买某物,求出总的价格。若购买的数量超出规定范围,给出提示,不能购买。读者在此遇到的新语句是:

```
 if(n < = MAXNUM) / *
 {
  total = n * PRICE;
  printf("%d %f \n",n,total);
 }
 else
   printf("输出的数量%d.超过规定范围%d \n",n,MAXNUM);
```

其语意为:



用符号常量作替换后,提高了程序的可读性,PRICE 显然代表单价,而 MAXNUM 代表最大数量,读起来一目了然。除此之外,程序还易于维护和修改,比方说 PRICE 由 30 上升到 40,则只要将第一个 # define PRICE 30 宏指令中的 30 改为 40 即可,不用修改程序的其他部分。

符号常量可以大写,也可以是小写。但习惯上用大写字母代替常量,而用小写字母代表变量。

2. 变量

程序运行过程中可以改变的量称之为变量。

变量用标识符表示,标识符是以英语字母或下划线开头的,由英语字母、下划线和数

字组成的序列。例如:ax1,w3a,my _ name,_ Mine 等是合法的标识符,而 π,My-name,35x 等则是非法的。

不同的 C 语言版本有不同的标识符长度。标准 C 只识别标识符的头 8 个字符。例如 A123 _ 456 与 A123 _ 456 _ b789,它们的前 8 个字符一样,标准 C 将认为它们是同一标识符。现在有的 C 语言版本规定标识符最长可达 32 个字符,还有的甚至更多。

程序中引用变量时必须先说明(或定义),例如下列程序段:

```
int x , y ;
  ……
z = x + y ;
  ……
```

虽然说明了变量 x,y,但没有说明 z,系统将会提示出错信息,程序员只有在加上说明语句"int z;"后,程序才可能正常运行。

每个变量与一个存储单元相对应。程序在变量的说明点给其分配存储空间。例如:

```
int x , y ;
```

则给变量 x,y 分别分配一个二字节空间,如图 1.3 所示。

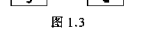


图 1.3

程序员可以对变量 x,y 进行运算和赋值。例如:

```
x = 3 ; y = 4 ;
```

它们分别是将 3 和 4 存入变量 x 和 y 代表的单元中,每个单元与一个地址相对应,设 x 对应于地址 0180,y 对应于地址 0280,则 3 和 4 将分别存入 0180 单元和 0280 单元中,见图 1.3。

可见变量具有双重特性,一是它的地址,二是它的内容。前例的变量 x 和 y 都具有两个值:一个是它们的地址值,分别为 0180 和 0280;另一个是它们的内容,分别为 3 和 4。

下列赋值语句:

```
x = x + 1 ; y = y ;
```

分别代表:将 x 的内容 3 加上 1 存入 x 单元中(地址 0180),将 y 的内容 4 存入 y 单元中(地址 0280)。尽管"y = y ;"这样的语句在程序中是合法的,但却是无用的,因为它对 y 单元没有进行任何有用的操作。另一方面,它也说明了赋值号"="左边与右边的变量的含义是不同的,右边代表内容,称为右值,左边代表单元号(地址),称为左值。虽然是同一变量 y,但其含义是完全不一样的。变量显然可以作为左值,因为变量与一个单元相对应,C 语言规定,凡是能与存储单元相对应的表达式都可以作为左值(我们将在第 8 章作进一步的讨论)。

一般情况下,取变量名时,最好能"见名思义"。例如:用 weight 代表重量;用 price 代表价格。

应注意的是,书写变量时,C 语言是区分大小写的,例如 WEIGHT 和 weight 是两个不同的变量。我们可以利用这一点,有时借助大小写混合书写,使变量的含义更明确。如