C语言简明教程
与
TURBO C程序开发系统

姜文清 编

西安电子科技大学出版社

C 语言简明教程
与 TURBO C 程序开发系统

姜文清 编

西安电子科技大学出版社
1993

（陕）新登字010号

内容简介

本书在使读者了解C语言全貌的基础上，从C的特点和难点的角度介绍C语言，并且通过各种类型颇有价值的程序实例介绍C的编程方法和技巧。

全书共分十章。前六章是关于C语言的简明教程，作者根据多年教学和开发软件的经验，对C中常用但又难以掌握的结构，像指针、参数传递、返回值以及递归函数等，通过实例和图示，进行了详细的讲解。各章后并附有精选的习题。后四章以Turbo C为例，介绍C的大量库函数，并且具体地说明了屏幕窗口与图形程序的设计方法，最后讨论了Turbo C的存储模式，给编大程序奠定了基础。全书篇幅不长，既可作为教材使用，又提供了工程需要的参考资料。

本书的作者早在80年代初曾参加首次移植UNIX，并且首次用C写出了《数据结构与算法》。

本书既适合作为各类大学“C语言程序设计”课程的教科书，又适合具有一定的计算机基础知识、希望快速掌握C语言的读者参考。

C语言简明教程
与TURBO C程序开发系统
娄文清 编
责任编辑 夏大平

西安电子科技大学出版社出版发行
陕西省大荔县印刷厂印刷
新华书店经销
开本 787×1092 1/16 印张 12 2/16 字数 286千字
1993年6月第1版 1993年6月第1次印刷 印数 1—8 000

ISBN 7-5606-0243-6/TP·0087 定价：7.00元

前　言

C 语言诞生于70年代初。1971年，D. M. Ritchie 写出第一个 C 编译程序，1972年投入使用。1973年，K. Thompson 和 D. M. Ritchie 用 C 写出 UNIX。1978年由 AT&T Bell 实验室正式发表 C 语言，并且由 B. W. Kernighan 和 D. M. Ritchie 合著的“THE C PROGRAMMING LANGUAGE”问世。在此基础上，美国国家标准学会制定了一个 C 语言标准，通常称为 ANSI C。ANSI C 与 Kernighan 的 C 完全兼容，同时引入一些 Pascal 的表示形式，做了一些扩充。

C 问世初期，用 C 写出了 UNIX 的核心程序及其庞大的外层软件，包括系统命令、实用程序、库以及各种应用软件。于是，C 作为主要的程序开发工具，在 UNIX 及其各种移植版本上运行。

近年来，随着 IBM PC 系列微型计算机的推广，C 被移植到 DOS 上运行。在移植过程中，一般参照 ANSI 标准，并且建立了良好的程序开发环境，很受微机用户的欢迎。

目前，全世界普遍认为 C 是用途最广泛的“软件工程语言”。

我国自1982年开始开发使用 C 语言，用来设计、移植操作系统和编译程序。近几年来普遍用来开发人工智能、模式识别、实时控制、通讯网络，以及各种绘图和管理软件。

笔者自1982年开始参加我国首次移植 UNIX。完成之后，首次用 C 写出了关于“数据结构”的成套程序，并选择了其中的基本部分经整理于1988年由上海交通大学出版社出版（见参考文献［1］）。

本书的初稿写于1986年，作为哈尔滨工业大学的油印教材在校内使用。由于当时的对象是计算机系即将毕业的本科学生，所以编写的指导思想是使已具备计算机基本知识的读者尽快地掌握 C 语言。这样，既使读者了解到 C 的全貌，又讲清了 C 的特点和难点，收到了事半功倍的效果。

随着 Microsoft C 和 Turbo C 的相继引入，我国 C 的用户越来越多，而且基础水平也越来越高。由于软件系统不断更新，教学内容也不断改进。在教学过程中不仅要使学员了解 C 的全貌，讲清其特点和难点，而且还要结合软件工程，提供编程方法和技巧。基于这种思想，笔者对讲稿进行了充实、精选和提高。该书以《C 编程技巧》名，于1992年12月在哈尔滨工业大学内部再次印出。现将其易名修订正式出版，作为教学总结，献给对 C 感兴趣的广大读者。

全书共分十章。前六章是 C 语言简明教程，后四章介绍 Turbo C 开发系统，其中大部分在各种 C 开发系统里都是兼容的，可作为软件工程的参考资料。

在简明教程里，开头两章介绍 C 的数据、表达式、控制流语句和常用的标准 I/O 函数；并且给出了关于字符处理、数值计算和处理菜单命令的例子，同时引入了逐步求精的程序设计方法（例2.2.2），以便给初学者奠定编程基础。第三章结构化数据，重点介绍指针及其与数组的关系，这是 C 的一个难点，也体现 C 的特点。通过实现程序，结合图示，详细说明各种表示形式的意义。在介绍指针数组时引入了一个反转法交换数据位置的例子，颇为有趣。在介绍结构、联合以及动态数据结构的过程中，一直以人事登记表为例，给从事计算机管理的读者提供实用的参考。第四章函数，重点介绍 C 参数传递的基本规则。不少用户调试

C程序遇到困难，往往因为套用其它语言的规则，而没能掌握C传值这一基本规则。在这一章里，给出3个不同的分类程序，特别是§4.6的程序例十分有趣，请读者仔细研读。第五章专门讨论递归，这是学习程序设计过程中感到棘手的问题。这里先通过像阶乘、数的分解这样简单的例子，讲清递归程序的执行过程和基本的设计方法，然后结合递归下降法处理算术表达式，介绍根据“语法”设计递归程序的方法，并给出一个根据表达式建立二元树的例子。此外，这一章还通过一个实例讨论了递归程序的变换。第六章介绍C读写文件的基本方法。要想把C作为得心应手的软件开发工具，一定要掌握好这前六章的内容。前六章的各章后并附有精选的习题。

为了提高C程序设计的效率，发挥软件系统的功能，后四章以Turbo C为背景，介绍C的大量库函数。第七章以高度浓缩的形式，按照.h文件的顺序介绍库函数，供读者作为资料参考。为满足开发交互式应用软件和绘图程序的需要，第八章介绍屏幕窗口程序设计的所需的数据和函数，并通过两个不同类型的实例，介绍窗口程序的设计方法。事实上，这是两个程序“架子”，供读者在实用中补充和修改。第九章是图形程序设计的资料，这里给出应用基本图形函数绘图的方法。第十章简要介绍Turbo C程序开发系统，介绍其组成、主菜单、热键命令和编辑操作。为了解除读者对Mode、far、near和huge的困惑，这一章重点讨论Turbo C的存储模式。这四章的内容在不同的实现系统中大部分是兼容的；有一些函数尽管在名称和形式上有所差异，但从概念和处理方法上基本都是相同的。

在整理过程中，笔者力图工程实用、提高效率、压缩篇幅、降低成本。所以，本书虽然篇幅不长，但内容十分丰富。

本书可供大专院校师生和从事软件工程的技术人员参考，也可作为C语言课程的教材。

由于编者水平有限，书中难免存在缺点错误，希望广大读者批评指正。

编者　1993年2月

目　录

第一章 程序基础

§1.1 简单程序例

例 1.1.1 从键盘上打入一行字符，使其按相反顺序输出，如图 1.1 所示。

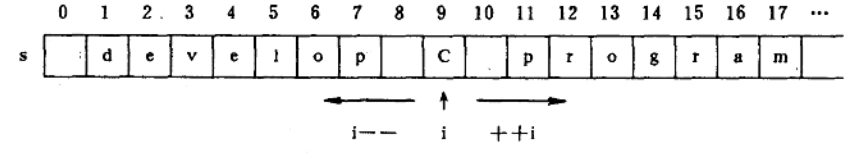

图 1.1 字符的输入和输出顺序

程序如下列所示，为了说明方便，在程序中每一行的前边加了一个序号：

```
(1) #include <stdio.h>
(2) main()                    /* This is the first example */
(3) {
(4)     char s[80], c;
(5)     int i = 0;
(6)     while ((c = getchar()) != '\n')
(7)         s[++i] = c;
(8)     while (i != 0)
(9)         putchar(s[i--]);
(10)}
```

一、程序结构

C 语言的程序为块结构，每个块称为一个“函数”(fuction)。一个完整的 C 语言程序至少有一个函数，称为主函数，名为 main。C 的各函数之间没有顺序要求。本例中的程序只有一个主函数，其中调用了库函数 getchar 和 putchar。

二、函数结构

C 的每个函数都有一个函数名，用户定义的函数可以用任意的标识符表示函数名。函数名后边必须紧跟括弧；当函数没有参数时，只有一对空括弧。关于函数的参数以及返回值的类型，将在第四章详细讨论。

在上述的程序里，第 (1) 行是个蕴含控制行，它不属于 C 语言本身，是 C 预处理程序处理的行，意思是把文件 stdio.h 中的内容插在这里。第 (2) 行表示一个函数开始，main 是函数名。一对/* ... */里包含的内容是注释，为程序加一个注解。C 的注释可以出现在程序中空格或回车键能出现的任何位置。第 (3) 和第 (10) 行的一对花括号括起来的是函数体，亦称分程序。分程序开始花括号后面是关于本函数的变量说明。第 (4) 行说明一个能存 80 个字符的数组变量 s 和能存一个字符的简单变量 c。第 (5) 行说明一个变量 i，并且令

其初值为 0。从第（6）行开始是程序要执行的语句。第（6）行用来控制读字符的循环，其中的循环条件

```
(c = getchar()) != '\n'
```

表示“读一个字符赋给变量 c，c 不是换行符”。第（7）行在第（6）行的条件为真时执行：先使数组 s 的下标 i 增 1，再把 c 存入 s 中 i 所指的单元；然后重复判第（6）行的条件。第（8）和第（9）行构成输出字符的循环，即当 i 不等 0 时打印 s 中的一个字符，然后使其下标 i 减 1；当 i 等 0 时程序终止。

三、C 程序的开发和执行

开发 C 程序时，首先要通过系统开发环境的编辑程序输入源程序；完成之后，通过编译程序进行语法检查，在没有任何语法错误的情况下，编译的结果生成目标程序；再经过连接，得到可执行的目标程序。如果编译程序指出语法错误，需要再用编辑程序进行修改。

执行可执行目标程序时，如果程序需要输入数据，比如，例 1.1 的程序中含有从键盘上读字符的函数 getchar，则要为程序输入适当的数据。上例中的程序执行时需要从键盘上打入一行字符，比如：

```
develop C program
```

于是，程序将这一行字符读入计算机，然后按照与输入相反的顺序输出，如下列所示：

```
margorp C poleved
```

四、流行的 C 语言开发环境

C 最初的开发及运行环境是 UNIX 操作系统。在 UNIX 操作系统中提供了 C 编译、连接需要的各种说明、库函数的代码等支持软件，为开发程序提供了极大方便。由 UNIX 移置而产生的各种类 UNIX 操作系统，譬如 XENIX、ZEUS 等，也都提供了和 UNIX 基本相同的关于 C 语言程序的开发环境。

近年来，C 已被移置在 DOS 操作系统上运行。目前使用比较广泛的 C 开发系统有：

C86，C88：Computer Innovation，Inc 公司

C4.0，5.0，6.0：Microsoft 公司

Turbo C 1.0，2.0：Borland 公司

这些系统都具有下列特点：第一，模拟了 C 在 UNIX 上运行的环境，所以和 UNIX 上的 C 兼容；第二，提供给用户的系统是一个完整的开发环境，使用方便。其中，Turbo C 里的开发命令 TC 提供一系列菜单命令，并以屏幕窗口的方式工作，使用极为方便，而且由于系统中的菜单命令“成文自懂”，加上简明的“help”，使用户不必专门学习，而在使用过程中就能很快掌握它的用法。

§1.2　标识符和关键字

一、标识符

标识符用来表示变量、类型和函数等名称。以字母或下划线开头，后边可以跟若干个字母、下划线或者数字。例如，

```
s       S       i       l       x10     x01
count   stdout  exit    _exit   EXIT
```

都是不同的标识符。C 区分大写和小写字母。

二、关键字

有些标识符在程序中具有特殊的意义，把这样的标识符称为关键字。

C 用作定义说明数据类型的关键字有：

auto	char	double	extern	float
int	long	register	short	static
struct	typedef	union	unsigned	

用作控制流语句的关键字有：

break	case	continue	default	do
else	for	goto	if	return
switch	while			

还有一个用作一元运算符的关键字：sizeof。

C 的关键字全用小写字母，不能用大写字母。此外，C 的关键字是保留字，就是说不能用作标识符使用。

§1.3　基本数据类型

1.3.1　常量

C 有 4 种基本类型的常量，分别如下：

一、整常数

整常数由数字串组成，有下列 3 种：

十进制数：非 0 开头，如：

1234　　10　　9

八进制数：0 开头，如：

01234　　C10 = 8　　08 = 010

09 = 011　　07 = 7

十六进制数：0x 开头，如：

0x10 = 16　　0x11 = 17　　0x20 = 32

0x9 = 9　　0xa = 10　　0xb = 11

0xc = 12　　0xd = 13　　0xf = 15

二、浮点常数

带小数点、带指数部分的数，用双精度表示。如：

3.14159265　　0.314e+1　　0.314e1　　314e−2

三、字符常数

括在单引号里的单个字符可以看成整数，其值为 ASCII 代码值。

'A'　'a'　'0'　'1'　'\n'

'A' = 65 = 0101 = 0x41 = (0100 0001)

'B' = 66 = 0102 = 0x42 = (0100 0010)

'a' = 97 = 0141 = 0x61 = (0110 0001)
'b' = 98 = 0142 = 0x62 = (0110 0010)
'0' = 48 = 060 = 0x30
'1' = 49 = 061 = 0x31
'2' = 50 = 062 = 0x32

转义字符：在某些字符之前 加反斜线 "\" 表示特殊意义或控制功能，有：

'\0' = 0，空字符，作为字符串的结束标志
'\t'　　水平制表，Tab 键的功能
'\n'　　回车换行，Enter 键的功能
'\"'　　一个双引号
'\\'　　一个反斜线
'\b'　　退一格，back space
'\f'　　走纸，form feed

四、字符串

括在双引号里的字符，其末尾以'\0'标志结束。如：

"This is a C program"
"The result is: x = %d\n"

1.3.2 变量及其类型

一、变量

在程序中用标识符表示变量。一个变量代表存储数据的一块存储空间，如图 1.2 所示，通过变量的名称访问其中的数据。

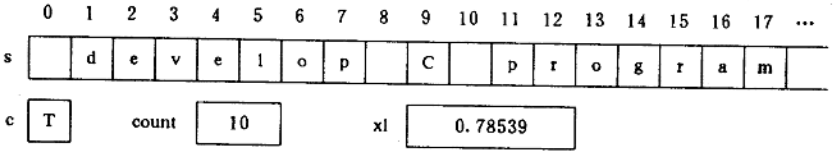

图 1.2　变量及其所表示的数据

二、变量的类型

变量的类型规定该变量所表示的数据需用存储空间的长度和数据的格式。所以，程序中使用的变量必须说明其类型，以便系统分配它所需要的存储空间。

说明变量类型的语句称为变量说明语句，其一般形式如下：

```
类型名　变量名表;
```

其中，变量名表可以包含若干个变量，彼此之间用逗号分开。例如：

```
char c1, c2;        /* 说明 c1, c2 是字符型的变量 */
```

```
int i, count;          /* 说明 i, count 是整型的变量 */
```

C 的变量有 7 种基本类型，其类型名关键字和在字长为 16 位的计算机上占用的二进制位（bit）数如下列所示：

char	字符型	8 bits	
int	整型	16 bits	(−32768～32767)
short	短整型	16 bits	
long	长整型	32 bits	
unsigned	无符号数	16 bits	(0～65536)
float	浮点数	32 bits	
double	双精度数	64 bits	

三、类型定义

在程序中，用户可以通过已有的变量类型自己定义新的类型。定义时用关键字 typedef，规定一个标识符，表示一个类型名。然后可以用这个类型名说明变量。例如：

```
typedef unsigned index;    /* 定义 index 为 unsigned      */
index count;               /* 说明变量 count 为 index 类型 */
```

四、类型转换

在程序中，为了实现类型匹配，需要使某种类型的数据转换成另一种类型的，在这种情况下需要进行类型转换。类型转换时，只须在要转换的数据前加新的类型名，并将其用括弧括起来。例如：

```
double x; int a;          /* 说明两个不同类型的变量 */
x = 2.73;                 /* 令 x 的值为 2.73 */
a = (int) x + 5;          /* 先使 x 变成整数再加 5 赋给 a */
```

§1.4　运算符和表达式

C 有多种运算符，可以构成各种各样的表达式。

1.4.1　算术运算

算术运算符连接操作数组成算术表达式，实现算术运算。算术运算符有如下 5 种：

+（加）　　−（减）　　*（乘）　　/（除）　　%（模除）

例如，下列所示为 C 的算术表达式：

```
a * b − c / d
(a + b) / c / d
(a + b) / (c * d)
a % b
```

C 的算术运算须遵循下列运算规则：

1. 运算的优先顺序为先做 *、/、%，后做 +、−。
2. 两个相同类型的操作数，运算结果的类型也相同。例如：

```
3 * 4          结果为整型
```

3.14 * 4.5　　结果为双精度型

3. 两个不同类型的操作数，运算结果的类型和其中精度较高的类型相同。例如：

3.14 * 4　　结果为双精度型

4. 整数乘、除运算从左往右进行。两个整数相除，结果仍然为整数。例如：

3 * 4 / 5　　结果为 2

3 / 5 * 4　　结果为 0（和 (3 / 5) * 4 相同）

5. 模除运算也称求余数运算，即两个整数相除，结果为除得的余数。例如：

15 % 12　结果为 3　　26 % 12　　结果为 2

9 % 12　结果为 9　　12 % 12　　结果为 0

−15 % 12 结果为 −3　　15 % (−12)　结果为 3

余数的符号和被除数的符号相同。

1.4.2 关系运算

关系运算符连接两个操作数组成关系表达式，比较两个操作数的大小或相等关系。关系运算符有如下 6 种：

C 的关系运算符	数学中的关系符
>	>
>=	≥
<	<
<=	≤
==	=
!=	≠

C 的关系运算须遵循下列运算规则：

1. 关系运算的结果：若关系成立，则产生整数 1，表示逻辑真；否则，产生整数 0，表示逻辑假。例如：

13 >= 12　结果为 1　　13 < 12　结果为 0

13 == 12　结果为 0　　13 != 12 结果为 1

2. 关系运算符的操作数可以是算术表达式。例如：

a + b < c + d　　a − b >= 0

even % 2 == 0　　odd % 2 != 0

3. 关系运算符的优先顺序低于所有算术运算符的优先顺序。运算时，先求算术表达式的值，然后进行关系运算。在关系运算符中，“==”和“!=”的优先级比其它 4 个运算符的优先级低。

1.4.3 逻辑运算

逻辑运算按操作数的整体值即按字进行。运算时只考虑操作数的值是否为 0；0 表示逻辑假，不为 0 表示逻辑真。运算的结果若为真，则产生整数 1，否则产生 0。

C 的逻辑运算符及其运算规则如下：

&&（逻辑与）　　||（逻辑或）　　!（逻辑非）

1. 逻辑与，算符为“&&”，仅当两个操作数的值都为真时，其结果为真，否则为假。例如，下列左边所示是表示运算条件的数学不等式，右边是相应的 C 的逻辑表达式：

$1 \geqslant x > 0$	x <= 1 && x > 0
x ≠ 0 并且 y ≠ 0	x != 0 && y != 0
x ≠ 0 并且 y ≠ 0	x && y

2. 逻辑或，算符为“||”，两个操作数的值只要有一个为真，其结果即为真，否则为假。例如，下列左边所示是表示运算条件的数学不等式，右边是相应的 C 的逻辑表达式：

x > 1 或者 x < −1	x > 1 || x < −1

3. 逻辑非，算符为“!”，一元运算，若操作数的值为真，其结果为假，否则为真。

以上 3 个逻辑运算符的优先顺序按

$$! \rightarrow \&\& \rightarrow ||$$

的顺序递减排列。

除逻辑非以外，二元逻辑运算的优先级都比关系运算的优先级低。这样，当表达式中同时出现上述 3 种二元运算符时，按下列顺序进行运算：

算术运算 → 关系运算 → 逻辑运算

1.4.4　位式运算

位式运算的操作数是整数，按照整型数中的二进制位（bit）进行运算，运算的结果还是整数。位式运算有如下 6 种运算符：

&（与）　|（或）　^（异或）　~（反）　<<（左移）　>>（右移）

1. 位式与，算符为“&”。对 a & b，仅当 a、b 对应位均为 1，结果该位为 1，否则该位为 0。例如：

$$3 \& 5 = 0011 \& 0101 = 0001 = 1$$

其中，0011 和 0101 分别为 3 和 5 的二进制数。

2. 位式或，算符为“|”。对 a | b，只要 a、b 的对应位有一个为 1，结果该位为 1，否则该位为 0。例如：

$$3 | 5 = 0011 | 0101 = 0111 = 7$$

3. 异或，亦称按位加，算符为“^”。对 a ^ b，若其对应位相同，则结果该位为 0，否则该位为 1。例如：

$$3 \wedge 5 = 0011 \wedge 0101 = 0110 = 6$$

4. 取反，算符为“~”。对应操作数每位上的 0，结果为 1；对应每位上的 1，结果为 0。例如：

$$\sim 5 = \sim 0000\ 0101 = 1111\ 1010$$

5. 左移位，算符为“<<”。对 x << n，把 x 的每一位向左移 n 位，右边空出的位填 0。例如：

$$5 << 2 = 0000\ 0101 << 2 = 0001\ 0100 = 20$$

6. 右移位，算符为“>>”。对 x >> n，把 x 的每一位向右移 n 位；对有符号数来说，左边空出的位填符号位上的值，称算术移位；对无符号数来说，左边空出的位填 0，称逻辑移位。在整数的二进制补码表示中，用最高位表示数的符号，0 表示正，1 表示负。例如，

假定用八位二进制补码表示整数，则

$$5 >> 2 = 0000\ 0101 >> 2 = 0000\ 0001 = 1$$

$$-5 >> 2 = 1111\ 1011 >> 2 = 1111\ 1110 = -2$$

对变量来说，用关键字 int、short 或 long 说明的是有符号数，用 unsigned 说明的是无符号数。

1.4.5　一元变换

通过下列运算符分别可以取变量的地址、指针所指单元的内容和测某种类型的数据所占存储空间的长度：

&（取地址）　　*（取内容）　　sizeof（测长度）

1. 取地址，算符为“&”。&V 取变量 V 的地址，亦称 V 的左值或指针。

2. 取内容，算符为“*”。*P 取指针 P 所指单元的内容，亦称 P 的右值。

3. 测试长度，算符为“sizeof”。sizeof E 测试表达式 E 的结果值占用存储空间的长度，用字节数表示。sizeof（T）测试类型为 T 的一项数据占用存储空间的长度。

1.4.6　增 1 和减 1 运算

增 1 和减 1 运算使一个整型变量的值在访问之前增 1 或减 1，或者在访问之后增 1 或减 1。有下列运算符：

++（增 1）　　--（减 1）

设 V 是一个整型变量，增 1 和减 1 运算的规则如下：

1. 先增运算，++V，先使 V 的值增 1，然后访问 V。

2. 后增运算，V++，先访问 V，然后使 V 的值增 1。

3. 先减运算，--V，先使 V 的值减 1，然后访问 V。

4. 后减运算，V--，先访问 V，然后使 V 的值减 1。

例 1.4.1　从键盘上打入一行字符，使其按相反顺序输出。程序如下：

```
#include <stdio.h>
main()                          /* This is the first example */
{
    char s[80], c;
    int i = 0;
    while ((c = getchar()) != '\n')
        s[++i] = c;
    while (i != 0)
        putchar(s[i--]);
}
```

这是例 1.1.1 的程序。关于赋值语句

```
s[++i] = c;
```

执行时，先使 i 的值增 1，然后对 i 所指的单元进行赋值。所以，这个语句相当于如下的两个语句：

```
i = i + 1;
```

```
s[i] = c;
```

而对语句

```
putchar (s[i--]);
```

来说，先输出s[i]的值，然后使i减1，所以，这个语句相当于如下的两个语句：

```
putchar (s[i]);
i = i - 1;
```

1.4.7 条件运算

条件运算根据判定条件决定一个表达式结果的值。例如，

若x≥y，则A为x，否则，A为y

可以写成

```
A = (x >= y) ? x : y
```

条件表达式用两个运算符，三个操作数，其一般形式为

```
E1 ? E2 : E3
```

其中E1、E2、E3均为表达式。如果E1为真，即其结果值不为0，则该条件表达式的结果为E2的值，否则为E3的值。

例如，条件表达式

```
(++i % 8) ? ' ' : '\n'
```

表示：先使i的值增1，将i的结果值除以8。若其余数不为0，则该表达式的结果值是一个空格符；否则，即除以8的余数为0，则该表达式的结果值是一个换行符。

1.4.8 赋值运算

在大多数程序设计语言里通常称为赋值语句的结构形式，在C语言里称为赋值表达式。基本赋值表达式的一般形式如下：

变量 = 表达式

其中，“=”表示赋值，称为赋值号，意思是先求表达式的值，然后将其按照变量的类型赋给变量，即存入变量所对应的存储空间。

由于基本赋值表达式的右侧还是表达式，所以，对于不同的变量V1、V2、… Vn：

```
V1 = V2 = … = Vn = 表达式
```

还是一个赋值表达式，称为多重赋值。执行时，把表达式的值按照Vn、... 、V2、V1的顺序依次赋给每个变量。

除了基本赋值表达式之外，还可以在赋值号前面加一个二元算术运算符或二元位式运算符，构成二元运算赋值表达式。假定oper表示这样的一个二元运算符，E是一个表达式，二元运算赋值表达式的形式如下：

```
V oper= E
```

与这个赋值表达式等价的形式为：

```
V = V oper E
```

就是说，对变量 V 和表达式 E，经 oper 运算，结果仍存入变量 V。例如：

```
i += 3;          /* 变量 i 增加 3 结果仍存入 i    */
i <<= 3;         /* 变量 i 左移 3 位结果仍存入 i  */
```

例 1.4.2　编程序，测试系统中无符号数的长度，即其所用的二进制位数。

算法：

(1) 设置无符号数 detector，令其初值为 0；

(2) 对 detector 取反，则其为全 1；

(3) 将 detector 左移一位并计数，重复进行，直到 detector 变成 0；

(4) 输出计数值即为无符号数的二进制位数。

程序如下：

```
main ()
{
    unsigned detector = 0, count = 0;
    detector = ~detector;              /* 令 detector 为全 1 */
    while (detector != 0) {
        detector <<= 1;
        count ++;
    }
    printf ("An unsigned number has %d bits. \n", count);
}
```

执行该程序的输出结果如下：

```
An unsigned number has 16 bits.
```

1.4.9　运算符的优先顺序

C 各种运算符求值的优先顺序如表 1.1 所示，按照自上而下优先级递减，同一行里的算符优先级相同。当优先级相同的算符连续出现时，由其结合性确定求值的顺序。

表 1.1　运算符的优先级

No	运　算　符	结合性
1	()　[]　->　.	左→右
2	一元的 !　~　++　--　-　(类型)　*　&　sizeof	右→左
3	*　/　%	左→右
4	+　-	左→右
5	<<　>>	左→右
6	<　<=　>　>=	左→右
7	==　!=	左→右
8	&	左→右