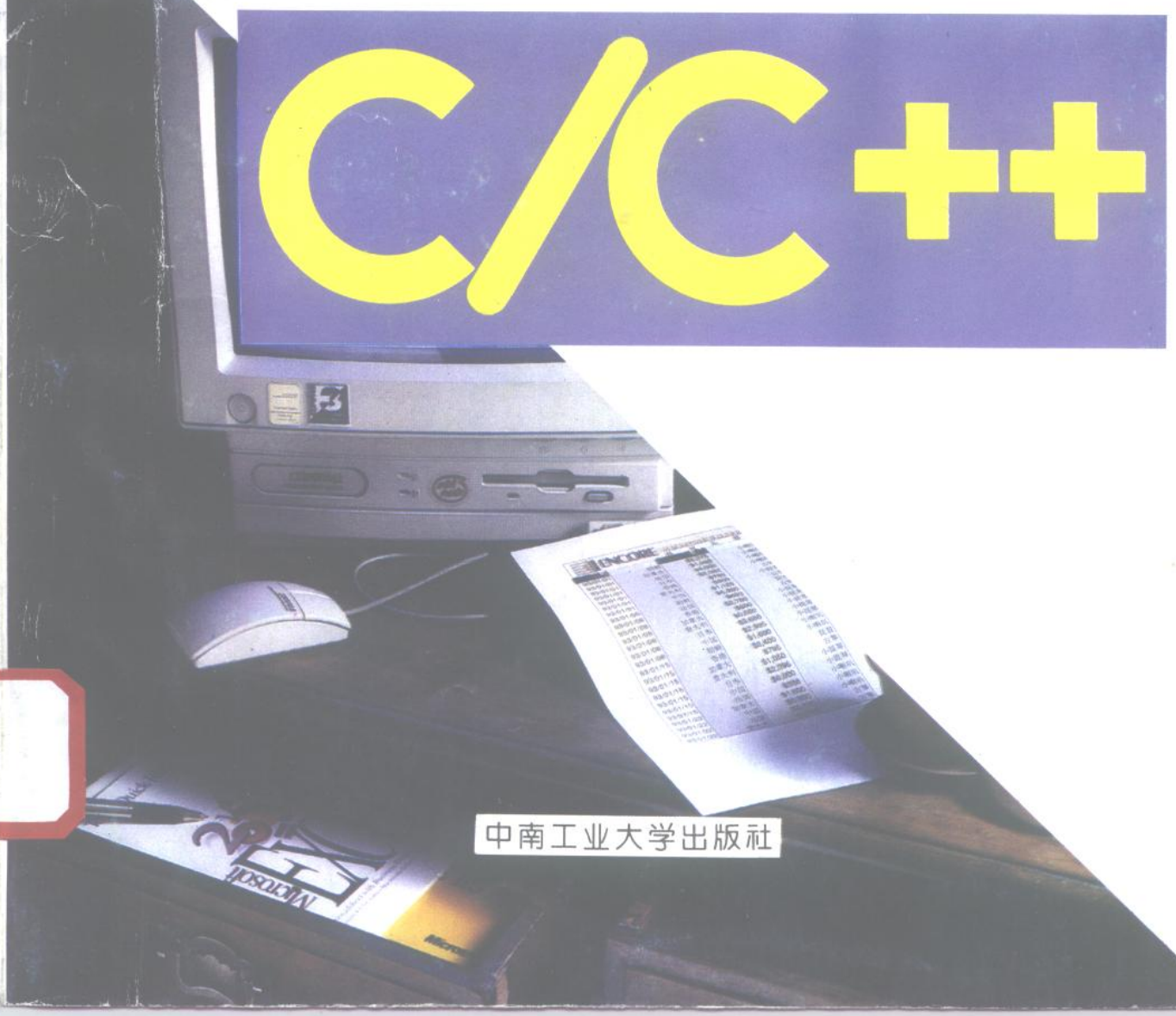
●计算机基础教育系列教材●

C/C++语言实用教程

毛先成　邓顺华
戴塔根　柳建新　叶柏龙　编著



中南工业大学出版社

C/C++ 语言实用教程
中南工业大学出版社

C/C++语言实用教程

毛先成　邓顺华
戴塔根　柳建新　叶柏龙　编著

中南工业大学出版社
·1996·

C/C++语言实用教程

毛先成　邓顺华
叶柏龙　编著
戴塔根　柳建新

责任编辑：肖梓高

国防工业大学出版社出版发行
[illegible]230研究所印刷厂印装
新华书店总店北京发行所经销

开本：787×1092　1/16　印张：13　字数：324千字
1996年6月第1版　　1996年6月第1次印刷
印数：0001－4000

ISBN 7-81020-880-2/TP·069
定价：15.00元

前　言

随着电子计算机的发展，C语言在计算机软件开发中的作用日益重要。C语言同时具有高级语言和低级语言的各种优点，表现出语言功能丰富、表达能力强、编程灵活、目标代码效率高、可移植性强等特点，因此，它不仅适合于编写系统软件，也适合于编写各种应用软件，它是广大程序员尤其是优秀程序员最广为使用的编程语言。

同时，随着软件开发技术的不断进步，面向对象方法已发展成为现代软件工程最优秀的方法，它较传统的结构化方法更为优越和先进，它已形成了一整套的软件开发方法学，包括面向对象分析、面向对象设计、面向对象的编程语言和面向对象的软件系统。面向对象的方法要求软件开发人员采用面向对象的编程语言。C++语言就是一种面向对象的编程语言，它具有面向对象编程的各种先进特性，同时，它又继承了C语言的通用性、灵活性、可移植性等所有特点。实际上，C++语言是C语言的超集，是C语言的扩展或增强版本。因此，C++语言综合了C语言和面向对象编程语言的各种优点，它比C语言有更强的性能，可编写出更大、性能更强的程序。C++语言正在日益成为程序员的通用编程语言。

由于C++语言是C语言的超集，是C语言的扩展或增强版本，故二者在逻辑上存在着本质联系。本书以C语言和C++语言的内在联系为主线，由浅入深地介绍C语言和C++语言的编程方法和技巧，便于读者循序渐进地掌握C语言和C++语言。这既有利于C语言初学者顺利地掌握好C语言和C++语言，也有利于C语言程序员自然平滑地向C++语言程序员过渡。

本书按照"保证基础、精选内容、由浅入深、以利教学"的原则编写，是作者多年来从事C语言和C++语言教学经验的总结，可作为大中专计算机语言课程的教材，也宜供C/C++语言初学者及C语言程序员使用。

全书共分15章，分别介绍了C/C++语言起源及C/C++语言程序的结构特点、C语言的编程方法和技巧、C++语言对C语言的扩展、C++语言的编程方法和技巧。各章节过渡自然，实例丰富完整。书中全部源程序均在Borland C/C++环境下调试通过。

本书由中南工业大学毛先成副教授、邓顺华副教授、戴塔根教授、柳建新讲师、叶柏龙副教授共同编写。

由于编者水平有限，加之时间仓促，难免出现错误或不妥之处，热忱欢迎广大读者及专家提出指导、批评，以使本书不断改进、完善。

编者

1996年3月

目　　录

1 C/C++语言概述

C语言自从问世以来,已由最初的作为操作系统的描述语言,迅速发展成为用途广泛的通用程序设计语言。它不仅可用来编写系统软件,也适用于编写应用软件。由于它的灵活性和编程限制少,C语言已成为广大程序员尤其优秀程序员最广为使用的编程语言。

C++语言是ANSI标准C语言的扩展或增强版本,即C++语言是C语言的超集。由于C++语言继承了C语言的通用性、灵活性等所有特点,并增加到扩充了面向对象编程等先进特性,因此,C++语言比C语言有更强的性能,可编写出更大更复杂的程序。C++语言正在日益成为程序员的编程语言。

1.1 C/C++语言起源

C语言的最初设计和实现是由Dennis Ritchie在DEC PDP-11上的UNIX系统环境下完成的。C语言的前身是BCPL语言。BCPL语言是由Martin Richards设计的,经Ken Thornpson简化为B语言。Ken Thompson用B语言编写了UNIX操作系统和相应实用程序。尽管B语言具有精炼和接近硬件等优点,但同时具有缺少数据类型、过于简单等缺点。1972~1973年,贝尔实验室的Dennis Ritchie,在继承B语言优点、克服B语言缺点的基础上,设计开发出了C语言,并在1973年和Ken Thompson合作将UNIX操作系统重写了一遍。

多少年来,C语言的标准版一直是UNIX操作系统支持的版本。Brian Kerhighan和Dennis Ritchie于1978年出版的《The C Programming Language》一书是C语言的经典描述,称为C语言的标准。随着计算机的发展及普及,出现各种C语言版本,各版本之间存在着差异。为克服C语言各版本间的不兼容性,美国国家标准化协会(ANSI)根据各版本对C语言的发展与扩充,制定出C语言新的标准,称为ANSI C语言。现在ANSI C语言标准已被各种C/C++语言编译系统所采用。目前微机上常用的C/C++语言编译系统主要有MSC/C++、Quick C、Turbo C、Borland C++等。

C++语言是C语言的扩展和超集。C++语言对C语言的扩展是Bjarne Stroustrup于1980年在贝尔实验室实现的。当时这种扩展后的新语言被称为"带类(Class)的C语言",1983年才被正式称为C++语言。

为解决编写大型程序时的复杂性问题,Bjarne Stroustrup借用了Simula67语言的面向对象特性,将面向对象编程(又称为OOP)等内容扩充到C语言中。

Bjarne Stroustrup在设计和实现C++语言时,既保留了C语言的有效性、灵活性,便于移植等全部精华和特点,又添加了面向对象编程的支持,因此,C++语言是C语言的灵活有效性和Simula67的面向对象编程的综合,故而具有强大的编程功能,编写出的程序具有构造清晰、易扩充维护、不失有效性等优良特性。C++适合于各种系统、应用软件的程序设计,尤其适宜编写复杂的大型软件。

1.2 C语言程序结构及特点

下面给出一个简单的C语言程序示例，以便分析其结构和特点。

例1-1 从键盘输入三个数，将它们求和，并在屏幕上显示求和结果。

```
/* 求三个数之和的C语言源程序    */
#include <stdio.h>                          /* 标准输入输出头文件   */
int sum(int x, int y, int z)                /* 求和函数说明   */
{
  int z;          /* 定义局部变量              */
  z=x+y+z;        /* 求和并赋值给 z            */
  return z        /* 返回求和结果              */
}
void main(void)  /* 主函数说明                 */
{
  int s, a, b, c                            /* 定义局部变量   */
  scanf("%d, %d, %d", &a, &b, &c);          /* 键盘输入三个数   */
  s=sum( a, b, c);                          /* 求三个数之和   */
   printf("sum=%d", s);                     /* 输出和   */
}
```

从该例可看出，C语言程序在结构上具有如下特点：

(1)C语言程序是由函数组成的。该程序中，包括main()和sum()二个函数。一个完整的C语言程序至少、且只能有一个函数main()。该函数称为主函数，是程序执行的起点，且函数名总为main()。在C语言程序中，函数相当于其它语言中的子程序和函数。一方面，用户可按上例的方式编写自己的函数(sum和main)，还可使用C语言的函数库所提供的函数(如scanf和printf)。函数库是由C语言编译系统提供的。

(2)一个函数由函数说明和函数体组成。

①函数说明。包括函数类型、函数名和参数说明。如上例中的sum函数说明为int sum(int x, int y, int z)，其中int sum指明函数类型为整型、函数名为sum，int x等为参数说明(int指明参数的类型为整型)。函数说明中可以缺失函数类型，这时隐含地表示函数的类型为整型；参数说明也可以缺失，这时表示该函数为无参数。值得注意的是，void main(void)中的第一个void表示函数类型为空类型，第二个void表示无参数(同缺失参数说明)。

②函数体。一对大括号{}括起来的内容即为函数体。当有多对大括号时，函数体括号是指最外层的一对大括号。函数体中一般由变量定义(数据定义)部分和执行部分组成。该二部分分别由定义语句和执行语句组成。函数体括号中可以无任何内容，这时称为空函数。当调用执行空函数时，空函数不执行任何语句即返回调用处。

(3)C语言程序中的语句(变量定义及声明语句和执行语句)总是以分号(;)结束。

(4)C语言中没有输入输出语句。C语言程序中的输入输出功能是通过调用库函数(如scanf和printf)来实现的。scanf和printf等用于输入输出的库函数的使用方法见后述章节。

(5)C语言程序的书写格式极为自由。可在一行内写一个或多个语句(甚至整个函数),也可以一个语句写在多行上。由此可见换行仅起分隔的作用。但需注意的是,编程者应尽量按一定的规范格式书写程序,以便于程序易读。

(6)C语言程序中,常有编译预处理指令(又称编译预处理语句),它总是#开头,是一个语句(指令)占一行,后不能跟分号或其它语句。它告诉编译系统在编译源程序之前做什么。例如,上例中的#include <stdio.h>指示编译系统将文件stdio.h嵌入于本程序中,然后开始编译。实际上,文件stdio.h中包含有scanf和printf函数的说明。一般,在C语言程序中使用函数(包括库函数和编程者编写的函数)时,应先对被使用的函数进行说明。

(7)C语言程序是区分大小写的。一般,C语言程序主要用小写字母书写,而大写多用来表示宏定义、常量或用以增强程序的可读性。

(8)C语言程序通过/*和*/配对使用进行注释,/*和*/之间的任何内容均为注释内容。

1.3 C++语言程序结构及特点

C++语言是C语言的超集,除继承了C语言的结构和特点外,它还具有面向对象等许多新的特征。

下面给出一个简单的C++语言程序,以便读者对C++语言程序的结构和特点有一个初步的认识。

例1-2 栈的操作。

```
#include "iostream.h"
#define SIZE 100

//定义栈类型
class stack
{
  int stck[SIZE];
  int tos;
public:
  void init(void);
  void push(int i);
  int pop(void);
};

//栈的初始化
void stack::init(void)
{
  tos=0;
{
```

```
//进栈操作
void stack::push(int i)
{
   if(tos==SIZE)
   {
    cout<<"The stack is full!";
    return;
    }
   stck[tos]=i;
   tos++;
}

//出栈操作
int stack::pop(void)
{
    if(tos==0)
    {
    cout<<"The stack is underflow!";
     return 0;
    }
     tos--;
     return stck[tos];
}

void main(void)
{
   stack stack1, stack2;

     stack1.init();
     stack2.init();

     stack1.push(1);
     stack2.push(2);
     stack1.push(3);
     stack2.push(4);

     cout<<stack1.pop()<<" ";
     cout<<stack1.pop()<<" ";
     cout<<stack2.pop()<<" ";
     cout<<stack2.pop()<<"\n";
```

在该程序中，定义了一个对象类型 stack，stack 由数据成员(变量 stck, tos)和方法成员(函数 init, push, pop)两部分组成，这样，对象类型 stack 就将全部信息(数据或变量、方法或函数)都封装在其自身之中，这是C++语言程序的最基本的特征。一旦定义了对象类型 stack，就可以用对象类型 stack 定义对象变量(如该程序中的 stack1 和 stack2)，并在对象变量(stack1，stack2)之上进行各种操作(初始化、进栈、出栈等)。

C++语言对C语言的扩展及其面向对象特征的详细介绍请见第 11 章。

2 数据类型、运算符和表达式

操作符对变量和常量进行操作构成表达式。表达式是C语言的基础，读者在深入学习C语言之前，必须理解本章提出的重要概念。C语言与其它计算机语言(如BASIC)不同，那些语言的变量、操作符及表达式都很简单，C语言的变量、操作符及表达式功能很强而且十分重要。读者在学习时必须掌握本章阐述的一些十分重要的原理。本章和以后各章将以Turbo C为背景，给出一些详尽的实例进行阐述，这些实例一般都可在MS-C/C++，Visual C++，Borland C++环境中运行。

2.1 标识符

标识符是变量、函数、标号和用户定义的名字的总称。C语言的标识符可以由一个或多个字符组成，第一个字符必须是字母或下划线，其它部分可以是由字母、数字或下划线组成的字符串。如count、var1、sum_length均是合法标识符，但1count、hi! there、sum..length却是非法标识符。

在标识符的定义中，读者必须注意以下几方面的问题：

(1)在C语言中，字母的大小写是有区别的，所以sum，Sum和SUM是三个完全不同的标识符。

(2)标识符不能和C语言的关键字相同，也不能和已定义的函数名和库函数名相同。

(3)标识符的有效长度在1～32个字符之间。

(4)建议标识符的命名采用表意命名法或匈牙利命名法，使标识符易于理解。

2.2 数据类型

1.**基本数据类型** 在C语言中，有五种基本数据类型：字符型、整型、实型、双精度实型和无值型。以上数据类型的字节长度和值域如表2-1所示。

表2-1 C语言基本数据类型的长度和值域

类 型	字节长度	值 域
char	1	-128 to 127
int	2	-32768 to 32767
float	4	3.4E-38 to 3.4E+38
double	8	1.7E-308 to 1.7E+308
void	0	无值

字符型(char)变量用于保存8位的ASCII字符(如'A'，'B'，'1')，也可存储8位的二进制数。整型(int)变量用于存储整型量。实型(float)和双精度实型(double)变量用于存储带小数的、非常大或非常小的数，实型与双精度实型之间的差别是所表示的最大数和最小数不同，

double 比 float 表示实数的范围大得多。

无值型(void)有两个用途。第一个用途是明确地表示一个函数不返回任何值;第二个用途是产生同一类型的指针。这两个用途将在以后章节中讨论。

C 语言还支持若干种复合数据类型,包括结构、联合、位域、枚举和用户定义类型。它们都由基本数据类型组成。这些复合类型将在第七章进行讨论。

2. **类型修饰符** 类型修饰符指在基本数据类型前添加一定的修饰前缀,即修饰限定词。除了无值类型外,基本数据类型可以带有各种修饰前缀。修饰符用于明确基本数据类型的含义,以准确地适应不同情况下的变量定义要求。类型修饰符的种类有: signed(有符号), unsigned(无符号), long(长)和 short(短)。

修饰符 signed, unsigned, long 和 short 可用于字符型和整型的修饰。此外, long 还可用于双精度实型量。表 2-2 列出了 ANSI 标准中全部允许的数据类型及它们的长度和取值范围。

表 2-2 C 语言基本数据类型及其修饰符的所有组合

类 型	二进制位长	值 域
char	8	-128 to 127
unsigned char	8	0 to 255
signed char	8	-128 to 127
int	16	-32768 to 32767
unsigned int	16	0 to 65535
signed int	16	-32768 to 32767
short int	16	-32768 to 32767
unsigned short int	16	0 to 65535
signed short int	16	-32768 to 32767
long int	32	-2147483648 to 2147483647
signed long int	32	-2147483648 to 2147483647
unsigned long int	32	0 to 4294967295
float	32	3.4E-38 to 3.4E+38
double	64	1.7E-308 to 1.7E+308
long double	64	1.7E-308 to 1.7E+308

整数前面的 signed 可以省略,因为整数定义本身规定就是有符号的。

有符号(signed)和无符号(unsigned)的整型量的区别在于它们的最高位的定义不同。如果定义的是有符号的整型(signed int), C 语言编译程序所产生的代码就设定整型数的最高位为符号位。如果最高位为 0,则该整数为正;如果最高位为 1,则该整数为负。例如,

二进制的 127 为: 00000000 01111111

二进制的 -127 为: 10000000 01111111

但是,为了运算的方便,目前大部分计算机(包括以 80X86 为 CPU 的计算机)都使用二进制的补码来表示负整数。这样就会使得 -127 的表示方法有所不同,然而符号位的使用是一样的。负整数的二进制补码是将其对应的正整数的各位取反,而后加 1 求得。换言之,若已知最高位为 1,则知其为负整数,其所表示的负数的值由该负数按二进制位取反,而后加 1 求得。例如, -127 的二进制补码形式为 11111111 10000001。

有符号整数对于很多运算都是很重要的,但是它所表示的最大数的绝对值只是无符号数

的一半，这一点应引起读者注意，以防计算溢出。例如，32767 表示为：

01111111 11111111

如果其最高位设置为 1，则该数就会由最大的正整数变为最大的负整数，即 -1；如果在该数上再加 1，则变为 10000000 00000000 即为 -32768。然而，如果将该数定义为无符号整数，当将 32767 的最高位置 1 时，它就变成了 65535。

如果读者对刚才讨论的整数二进制表示弄不清楚，可暂时放下，在学习了后面章节的位运算后，反过来再看，相信读者一定会明白的。

3. **访问修饰符** C 语言有两个用于控制访问和修改变量的方式的修饰符，它们分别是常量(const)和易变量(volatile)。

const 所修饰的变量在程序运行的过程中，始终保持不变。例如，

```
const int a;
```

将产生整型变量 a，其值不能被程序所修改，任何对 a 的赋值企图会被编译程序发现，并报告错误，但它可以在其它类型的表达式中使用。读者可能会问，a 不能被赋值，其值从何而来呢？编译程序允许常量在说明时赋初值，如，

```
const int a = 100;
```

另一种赋初值的方法是通过某些硬件的方法来进行，读者在深入学习 C 语言后，才会弄清楚。修饰符 volatile 用于提醒编译程序，该变量的值可以通过程序中，未明确定义的方法来改变。例如，一个全程变量的地址可以传给操作系统的时钟例行程序，从而该变量可用于存储系统的实时时钟值。在这种情况下，变量的内容在程序中没有明确的赋值语句对它赋值时，其内容也会发生变化。这一点是很重要的，因为在假定表达式内变量内容不变的前提下，C 语言编译器可能会自动优化某些表达式。此外，在编译过程中，有的优化处理将改变表达式的求值顺序。修饰符 volatile 可防止上述情况的发生。

const 和 volatile 可同时使用。例如，假如 0x30 是一个随外部条件而变化的口地址值，那么就恰好需要用下述说明来避免偶然因素所产生的副作用的影响。

```
const volatile unsigned char * port = 0x30;
```

2.3 常量

1. **常量类型** 在 C 语言中，常量是指不可由程序修改的固定值。常量可为任何基本的数据类型。每个常量的表示方法取决于它的类型。如表 2-3 所示。

表 2-3 常量类型及其取值举例

数据类型	常量举例
char	'a', '', '9'
int	1, 1200, -100
long int	35000, -120000
short int	10, -11, 80
unsigned int	1000, 863, 50000
float	12.45, 3.2E-3
double	123.45, 1.4E45, -0.98

2.**十进制和八进制常量** 有时,编程使用以8或16为基数的数字系统比以10为基数的数字系统更方便些。以8为基数的数字系统称为八进制,它使用数字0至7,八进制10等于十进制数8。以16为基数的数字系统称为十六进制,它使用0至9加上字母A至F,字母A至F分别表示10,11,12,13,14,15。例如,十六进制10等于十进制16。因为这两个数字系统使用频繁,所以,C语言允许定义八进制和十六进制常量。八进制常量以"o"开头,后跟0至7的数字。十六进制常量以0x开头,后跟0至9和A至F。下面给出一些例子:

```
hex = 0xFF; /* 十进制数 255 */
hex = 0xafcd; /* A至F字母的大小写是等效的 */
```

3.**字符串常量** C语言除了支持前面定义过的常量外,还支持另一种类型的常量,即字符串。

字符串是以双引号引起来的一组字符。例如,"this is a example"就是一个字符串常量。读者必须注意的是,不能把字符串和字符混淆起来,字符是由单引号引起来的。例如'a'为一个字符,而"a"是一个只包含一个字母的字符串,这两个常量在存储结构上是有严格区别的:'a'占用一个字节,"a"占用两个字节。

4.**转义字符常量** 单引号引起来的字符常量通常用作打印字符,但有些字符(如:回车,走纸,响铃)不能从键盘输入。因此,在书写源程序时,为了明确地表示这些不可直接键入的字符,C语言提供了专门的转义字符常量,这些转义字符常量常称为控制字符。如表2-4所示。

表2-4 常用转义字符常量

标识码	含 义
\b	退格
\n	换行
\r	回车
\f	换页
\t	水平制表符
\v	纵向制表
\"	双引号
\'	单引号
\0	空字符(整数值为0)
\\	反斜杠
\a	响铃
\o	八进制常量
\x	十六进制常量

转义字符的用法和其它字符的用法完全相同.如,

```
ch = '';
printf("Error \n");
```

5.**符号常量** 在C语言中,常量可以用符号来代替。代替常量用的符号称为符号常量。为了便于与一般的变量相区分,符号常量一般全部使用大写英文字母表示。符号常量在使用之前必须预先定义,其定义的一般格式是:

```
#define 符号常量 常量
```

例如, #define NULL 0

```
#define EOF -1
#define PI 3.1415926
```

其中 NULL、EOF、PI 都是符号常量，它们代替的符号常量分别是 0、-1、3.1415926。

每个符号常量定义式只能定义一个符号常量，而且在整个程序中只能定义一次，一经定义不得改变，每个定义占一个书写行。必须注意的是，它们必须以 # 开头，而且后面不能加分号。实际上，它不是 C 语言的程序语句，而是传递给编译系统的预处理指令。关于预处理指令将在后面章节讨论。

符号常量在程序中用于代替具有一定实际意义的常量。如前面定义的 EOF，它的意义是"文件尾"，即程序中出现的这个 -1 不仅是个数，而且具有确定的意义。因此，使用符号常量增强了程序的可读性和易理解性。

此外，程序中某些常量，在调试、扩充或移植时，需要修改其值，这种常量通常也定义为符号常量。当它们需要修改时，只需改变其定义即可。在一个符号常量多处使用的大型程序中，充分体现了这种优越性。同时也避免了一个常量在多处修改中，因失误造成的不一致而导致的错误。

2.4 变量

1. **变量说明** 变量是用来存储数据的，在 C 语言中也是如此。在向变量存储数据时，必须事先说明。其说明的一般形式是：

变量类型 变量名列表；

其中"变量类型"必须是一个有效的 C 语言数据类型，它可以是 C 语言的基本数据类型、带有类型修饰符的数据类型或用户自定义数据类型。"变量名列表"可由一个或多个标识符名加上逗号分隔符组成。例如，下面给出一些变量说明：

```
int i;
int k, l, m;
long int len;
unsigned long int ul;
double height;
```

在 C 语言中，与其它一些计算机语言不同的是，变量名与其类型无关，即任意标识符可说明为任意类型的变量，但建议读者在为变量取名时，不要和 C 语言的保留字同名，以免引起编译混乱。

2. **变量说明的位置** 变量在何处说明对程序其它部分如何使用这个变量影响较大。以变量说明的位置为基础，决定变量如何使用的规则叫做该语言的"作用域规则"。关于这些规则及细则的完整讨论要到读者对 C 语言有更多了解时才能进行，这里只讨论基本的东西。

在 C 语言程序中，可在三个位置说明变量，第一是在所有函数（包括 main()函数）之外，这种变量叫全程变量，程序的任何部分都可使用这种变量；第二是在函数内，这种变量叫做局部变量，仅仅本函数内的语句可以使用这种变量，从本质上讲，局部变量仅为本函数内的代码所认识，本函数外的代码"看不见"它；第三是函数的形式参数说明部分，这些参数除了接收传递