第1章

C语言概述

为使读者对C语言有一个概括的了解，在详细介绍C语言程序设计之前，本章简单地介绍C语言的产生过程及特点、C语言程序的结构与书写格式以及C程序的开发过程。

1.1 C语言的产生过程及特点

20世纪60年代，随着计算机科学的迅速发展，高级程序设计语言得到了广泛地应用，然而，还没有一种可以用于书写操作系统和编译程序等系统程序的高级语言，人们不得不用汇编语言（或机器语言）来书写，但汇编语言存在着不可移植、可读性差、研制软件效率不如高级语言等缺点，给编程带来很多不便。为此，人们对能用于系统程序设计的高级语言的开发就变得势在必行了，于是，在20世纪70年代初产生了一种能够用来研制各种系统程序的高级语言——C语言。

1.1.1 C语言的产生过程

C语言的出现是与UNIX操作系统紧密联系在一起的，C语言本身也有一个产生过程，表1—1给出了C语言的产生过程。

表1—1 C语言的发展历史

语言名	设 计 者	年份
CPL	C. Strachey 等	1968
BCPL	M. Richards	1969
B	K. Thompson	1970
C	D. M. Ritchie	1972

C语言起源于1968年发表的CPL(Combined Programming Language）语言。它的许多重要思想来源于Martin Richards在1969年研制的BCPL(Basic Combined Programming Language)语言,以及以BCPL语言为基础的而由Ken Thompson在1970年研制成的B语言。K. Thompson用B语言写了第一个UNIX操作系统，用在PDP—7计算机（现已被淘汰）上。D. M. Ritchie1972年在B语言的基础上研制出C语言，并用C语言写了第一个在PDP—11计算机上实现的UNIX操作系统。UNIX操作系统的巨大成功也伴随着C语言的巨大成功。

目前，从微型到大型计算机都配有C编译程序。不仅在装配UNIX操作系统的机器上，而

且在非 UNIX 操作系统的机器上也配有多种 C 的编译程序。由于 C 语言本身具有许多特点，现在它已经成为在微、小、大、巨型计算机上，从系统程序设计到工程应用程序都能使用的一种高级程序设计语言。

1.1.2 C语言的特点

C 语言的特点可以从多方面来阐述，这里仅从使用者的角度加以讨论，其主要特点如下：

1. 表达能力强且灵活。C 语言是处于汇编语言和高级语言之间的一种记述性程序设计语言。C 语言既有面向硬件和系统，像汇编语言那样可以直接访问硬件的功能，又有高级语言面向用户、容易记忆、便于阅读和书写的优点。

2. 程序结构清晰且紧凑。因为 C 语言程序通常由若干个函数组成，所以它是一种模块化程序设计语言。因此，它十分利于把整体程序分割成若干相对的功能模块。并且，它为程序模块间的相互调用以及数据传递提供了便利，这种模块化结构的程序不但清晰而且紧凑。

3. 书写简单、易学。例如,C 语言用 {和 }来代替 Pascal 语言中的 begin 和 end 作为复合语句标号，它的运算符也尽量缩写等。

4. 目标程序的质量高。C 语言提供了一个较大的运算符集合，并且其中大多数运算符与一般机器指令相一致，可直接翻译成机器代码，因此，用它编写程序生成的代码质量高。实践证明，其它高级语言相对汇编语言的代码效率要低得多，而 C语言的代码效率只比汇编语言低 10%—20%。但 C 语言在描述问题时编程迅速、可读性好、表达能力强等优点是汇编语言无法相比的。

5. 可移植性好。C语言的语句中，没有依存于硬件的输入／输出语句，程序的输入／输出功能是通过调用输入／输出函数实现的。而这些函数是由系统提供的独立于 C 语言的程序模块库,因此,C 语言虽然具有直接访问硬件的功能，但 C 语言程序本身并不依存于机器硬件系统，从而便于在硬件结构不同的机种间实现程序的移植。

6. C语言是一种结构化程序设计语言，它提供了一整套循环、条件判断和转移语句，实现了对程序逻辑流程的有效控制，有利于结构化程序设计。

7. C语言提供了丰富的数据类型。它不仅具有字符型和几种尺寸的整型数以及单、双精度的浮点数等基本数据类型，而且允许程序员自己设计更为复杂的数据类型，如数组、结构、联合等来适应特殊的程序需求。

8. C 语言允许程序员定义各种类型的变量指针和函数指针。指针是与机器内存地址相关的说明项，因此指针是让程序员以相同于机器码的形式存取内存的数据。正确地使用指针可提高程序的效率。C 语言还支持指针运算，允许程序员直接访问和操纵内存地址。

9. C语言的预处理是一种正文处理，是编译之前对正文文件（源程序文件）的再安排。其中，用得最多的是定义程序的变量、代替函数调用的宏（可较快运行）和基于某种特定条件的编译指令。

C语言是一种很灵活的语言，它允许程序员做出各种决定。为了保持这种特性，C语言很少在类型转换等方面加以强制性的限制，这通常是很有益的。但是，C 语言类型检验太弱，转换比较随便，存在着不安全因素，程序员在使用 C 语言时必须注意到这一点。

由于 C语言具有上述众多特点，近年来迅速地得到广泛普及和应用。C 语言被称为‘高级汇编语言”。特别是在微处理机和微型计算机的软件开发，以及各种软件工具的开发中，使用 C 语言的趋势日益增强，最近呈现出 C 语言有可能取代汇编语言的发展倾向。

1.2 IBM—PC 微型机所用的 C 语言

近年来，适用于各种不同操作系统（UNIX、MS—DOS、CP/M86 等）和不同机种（8bit～32bit）的 C 语言编译系统相继出现，在当前国内主流微机 IBM—PC 上流行在 MS—DOS 下的 C 语言编译程序有：Computer Innovation 公司的 C86（或者优化 C86），Lattice 公司的 LC、Microsoft 公司推出的 MS C（5.0 或 6.0 版）以及 Quick C、Turbo 公司推出的 Turbo C 等。

适用于不同操作系统和不同机种的 C 语言编译程序有几十种之多，不同版本的 C 语言的语句功能基本上一致，可以圆满地解决 C 语言程序在不同机种、不同系统间的移植问题。但是，不同版本之间也存在着某些差异，它主要体现在标准函数库中收纳的函数的种类、格式和功能上稍有差别。本书以当前最新的 1987 年美国国家标准 C 语言（87 ANSI 标准 C）为基础，同时兼顾其它不同版本中通用性、一致性的内容予以叙述。读者在使用 C 语言编制实用程序时，最好首先参考您所使用的计算机上配备的 C 编译系统的有关资料。

1.3 C 语言程序的结构及书写格式

1.3.1 C 语言程序的结构

C 语言程序是由一个或几个函数所组成的。请看下面一个简单的 C 语言程序示例。

例 1.1 求 X、Y、Z 三个整型数之和。

程序清单：

```
/ * File name:EX1-1.C * /
/ * The sum of x, y and z * /
main( )
{
  int x,y,z,sum;
  scanf("%d%d%d",&x,&y,&z);
  sum=x+y+z;
  printf("sum=%d\n",sum);
}
```

执行此程序时，首先从键盘上以十进制形式键入三个整型数，然后求三数之和，并将和以十进制形式在屏幕上显示出来。

在 C 语言中，以 /＊开头，并以 ＊／结束的字符行为程序的注释部分，注释可以出现在程序的任何部分，它可以帮助阅读和理解程序。

C 语言程序的基本结构为：

```
main( )
{
      语句
```

}

其中,main( ) 是一个函数，而且是一个特殊的函数，每个程序必须有一个且只有一个名为 main 的函数。程序在 main () 函数（主函数）的开始处开始执行。 main 名后面的圆括号 () 是必须的,这是 C 语言函数的标志。圆括号对 () 中为参数表。 main () 函数可以有参数表，也可以没有参数表。这里为没有参数表的情况，但圆括号对 () 必须有，不能省去。一般情况下，main( ) 函数没有参数表，有参数表的情况请参阅本书第 9 章的命令行参数部分。

花括号 { } 内是函数的函数体。花括号对 { } 将构成函数的语句序列括起来，C 语言中的 { } 与 Pascal 语言中的“ begin”和“end ”类似。无论函数体中是一组语句，还是“空”（一个语句也没有）,{ } 都是必须有的，也就是说，对应于一个函数至少要包含一对 { } 符号。C 语言中的语句大致分为两类：一类为说明语句，用来描述数据，决定内存的分配；另一类为执行语句，用来对数据进行操作，决定内存的内容。

例 1.1 程序中函数 main () 内有四个语句：

int x, y, z, sum ；是数据类型说明语句。这里说明 x, y, z, sum 四个变量均为整型，分别占据内存两个字节的空间。

其余三个语句均为可执行语句。其中：

scanf( ) 为 C 语言的格式化输入函数（系统提供的标准输入函数）。

sum=x+y+z ；为赋值语句。

printf( ) 为 C 语言的格式化输出标准函数。

C 语言的每个语句（除以后要介绍的个别语句外）结束时，必须以分号“ ；”结尾。

1.3.2 C 语言程序的书写格式

1. 用 C 语言书写程序时较为自由，既可以一行写多个语句，也可以一个语句分几行来写。它不像 BASIC, FORTRAN 程序有严格的书写格式。

2. C 语言要求关键字都使用小写字母。而且 C 编译程序区分大小写字母，即 C 编译程序认为 A 和 a 为两个不同的标识符（名字）。这一点 C 语言与其它语言不同，其它语言（如 BASIC, FORTRAN, Pascal 等）编译程序自动将小写字母转为大写字母，它们将 A 和 a 识为同一个标识符。因此，为了避免出错，一般使用 C 语言书写程序时均用小写英文字母。

3. 前面已介绍说 C 语言程序没有严格的书写格式，与 Pascal 语言一样，采用自由格式，每个语句以“ ；”结尾。但是，为了避免程序书写的层次混乱不清，为了便于阅读和理解程序，一般都采用有一定格式的习惯书写方法。因为 C 语言是结构化程序设计语言，为了结构层次分明，书写程序时不同结构层次的语句，从不同的起始位置开始，同一结构层次中的语句，缩进同样个数的字符位置。同一结构层次的花括号对 { } 亦缩进同样个数的字符位置。

4. 一般一个语句占一行，为了增加可读性，适当地加一些注释行或空行。

请看下面 C 语言程序书写格式示例。

例 1.2 统计输入文本中的行和字符个数的程序。

```
#include  <stdio.h>
main( )
{
    int c,nl,nc;
    nl=nc=0;
```

```
    while((c=getchar( ))! =EOF)
    {
      nc++;
      if(c==='\n')
            nl++;
    }
    printf("line=%d%t character=%d\n",nl,nw,nc);
}
```

1.4 C 语言程序的开发过程

绝大多数 C语言程序为编译执行，因此，C 语言程序的基本开发过程如图 1—1所示。

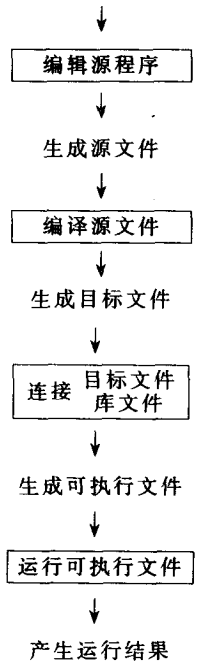

图1—1 C语言程序的开发过程

1.4.1 编辑源程序

大部分 C 编译系统带有独立的编辑程序，可用于输入或修改源程序。也可使用操作系统提供的编辑程序，如 UNIX 或 CP/M 下的 ED，MS—DOS 或 PC—DOS 下的 EDLIN 等，还可使用功能更强的专用编辑软件，如 WordStar 等。源程序经编辑程序由键盘输入后，形成源程序文件以文本文件的形式存储在外存储器（软盘或硬盘）中。源程序文件的名字由用户选定，但扩展名均为 C(即 C 语言程序源文件均带有后缀".C")。

例如：

filename.C

f1.C

均为 C 语言源程序文件名。

1.4.2 编译源文件

在程序运行之前，必须用系统提供的编译程序对源程序文件进行编译。编译程序要进行语法检查，若没有发现错误，那么编译后将产生目标文件。目标文件是由“目标代码”组成的，目标代码通常是指可在机器上直接运行的二进制码（机器码）。在 MS—DOS 下目标文件带有.OBJ 后缀（文件扩展名）。若编译程序发现有错误，则输出错误信息，此时，程序员应对程序进行再编辑，改正程序错误后，再进行编译，直到编译正确为止。

1.4.3 连接目标文件及库文件

源程序经编译程序编译后产生的目标文件虽然是由机器指令代码组成，但是它还是不能直接在机器上运行，它是一个浮动的程序模块，也就是说，它是可重定位的。这意味着其中机器码指令的内存地址并未绝对地确定，只有偏移量是确定的。因此，程序需要重定位在确定的绝对地址上，这由连接程序完成。并且，所有的 C 语言编译程序都是和标准函数库文件一起提供的，保存在函数库文件中的函数也都是可重定位的。当用户 C 语言程序调用了一个标准库函数（或者别人编写的函数）时，编译程序“记忆”它的名字，随后，连接程序（LINK）将用户编写的程序产生的目标代码同标准函数库文件中找到的目标码结合起来，这个过程称为“连接”。连接时，对程序和函数进行重定位，内存偏移量被用来产生实际地址（绝对地址）。经过连接重定位后产生的文件称为可执行文件，可执行文件以 .EXE 作为后缀，它可以在机器上直接运行。

1.4.4 运行程序

经过编译和连接后产生的可执行文件（.EXE 文件），只需在操作系统下打入可执行文件名，程序即可启动运行。

综上所述，一般运行 C 语言程序都必须事先经过编译、连接后才能运行，它不像运行解释型 BASIC 程序那样直接启动解释程序即可边解释边执行程序。不过不同的 C 编译系统对于编译和连接的具体操作也不尽相同，有的步骤明显（例如，MS C），有的步骤不明显，由系统自动完成编译、连接过程（例如，Turbo C 和 MS C6.0 的集成环境下）。具体的上机操作步骤请参看具体编译系统的有关资料（参阅附录 A 有关 Turbo C 的介绍）。

C 语言允许进行分割编译，即将大的程序分成若干块，装入若干文件，每一个文件可单独编译，当所有的文件编译完成后，再进行连接。连接程序将分别编译后产生的所有目标文件和库函数中的函数进行连接，形成一个完整的可执行文件。

习 题 1

1. 试将 C 语言程序结构与你熟悉的其它语言（例如：BASIC、FORTRAN、Pascal，COBOL 等）的程序结构进行比较。

2. 请指出下列 C 语言程序中的错误，并改正之。

```
/ * File name:error.C * /
```

```
/ * The program is error program!
main( );
{
    INT a,b,c,sum;
    a:=1;b:=2;C:=3;
    SUM=A+B+C;
    printf("SUM=%d\n",sum)
}
```

3. 请参阅附录 A,在你的计算机上用 Turbo C 编译系统将例 1.1 的程序进行编辑、编译、连接和运行。

4. 用 Turbo C 编译系统将例 1.2 的程序进行编辑、编译、连接和运行。

第2章

常量与变量

数据处理是数字计算机的基本功能。在C语言中，数据处理的基本对象是常量和变量。C语言提供了多处数据类型的常量和变量。

2.1 标识符命名

在C语言中，标识符是对常量、变量、函数、标号和其它各种用户定义的对象的命名。

2.1.1 标识符的构成规则

标识符的构成必须遵循下列语法规则：

1. 以字母（大小写均可）或下划线符中任一字符为起端（某些C编译系统可能不允许下划线作为标识符的起始字符）。

2. 在第一个字符之后，可跟随任意的字母、数字或下划线符组成的字符序列，该字符序列可以是空串。例如，下面的字符序列均为合法的C语言标识符。

i,inword_ file,file1,F_name

而下面的字符序列则均为不合法的C语言标识符：

2i,α_#file,a/b

2.1.2 注意事项

构造标识符时除了必须按照以上规则外，还应注意以下事项：

1. 在C语言中，大小写字母有不同的含义，例如AB、Ab及ab为三个不同的标识符。

2. ANSI标准规定，标识符可以是任意长度，但是外部名必须至少能由前6个字符唯一地区分。这里外部名指的是连接过程中所涉及的标识符，其中包括分割编译时文件间共享的函数名和全局变量名。这是因为对某些仅识别前6个字符的编译程序而言，下列的外部名

f_name1 f_name2 f_name3

将被当作同一个标识符处理。

3. ANSI标准还规定，内部名必须至少能由前31个字符唯一地区分。这里内部名指的是仅在定义该标识符的文件中使用的标识符。在进行C语言程序设计时，读者应查阅实际使用的C编译程序的用户手册，以确认本系统实际识别标识符的前多少个字符。

4. 另外，在构造标识符时，不能和C语言的关键字相同，也不能和C语言库函数同名。

2.2 基本数据类型

2.2.1 C语言的数据类型

C语言提供了丰富的数据类型，C语言的数据类型有基本数据类型、聚合类型（aggregate-types）和指针类型。

基本数据类型包括：字符型(char)、整型(int)、浮点型(float)、双精度浮点型(double)和无值(void)。

聚合类型包括：数组、结构、联合、位域(bit filed)、枚举和用户定义类型。

C语言的数据类型综合如下：

- 数据类型
  - 基本类型
    - 字符型(char)
    - 整型(int)
    - 浮点型(float)
    - 双精度型(double)
    - 无值(void)
  - 聚合类型
    - 数组
    - 结构
    - 联合
    - 位域
    - 枚举
    - 用户定义类型
  - 指针类型

在这里仅介绍基本类型数据，对于复杂类型数据（聚合类型和指针类型）将在以后的章节里陆续介绍。

2.2.2 基本类型数据的宽度及范围

尽管五种基本类型数据的宽度（位数）和所表示数的范围随处理器的类型和C语言编译程序的实现而异，但一般来说，整数的宽度与处理器的字长相等，一个字符占用一个字节，而浮点数值的确切格式则根据实现而定。对于多数微型机，包括基于8088和8086的机型，五种数据的字长和范围如表2—1所示。

表2—1 基本类型数据的宽度及范围

类型	宽度	范围
char(字符型)	8	0～255
int(整型)	16	−32768～32767
float(浮点型)	32	绝对值在10^{-38}～10^{38}(约7位有效数字)
double(双精度型)	64	绝对值在10^{-308}～10^{308}(约15位有效数字)
void(无值型)	0	无值

void 类型是 ANSI 标准中增加的类型，有的 C 语言可能不支持这种类型，但多数 C 语言编译程序支持这种类型。它的用法将在以后的章节中介绍。

2.2.3 基本类型修饰符

除 void 类型外，基本类型的前面均可以带有各种类型修饰符，修饰符用来改变基本类型的意义，以便更准确地适应各种情况的需求。

基本类型修饰符有以下四种：

signed（有符号）

unsigned（无符号）

long（长型）

short（短型）

以上四种类型修饰符适用于字符和整数两种基本类型，而 long 还可用于 double（由于 long float 与 double 为同种类型，因此 ANSI 标准删除了多余的 long float）。

因为整数的缺省定义是有符号数，所以 signed int、signed short int、signed long int 中的修饰符 signed 是多余的，一般不写，但仍允许使用。

某些编译系统允许将 unsigned 用于浮点型，如 unsigned double。但是这一用法降低了程序的可移植性，故建议一般不要采用。

表 2—2 给出了按照 ANSI 标准而组合的各种类型、宽度（所占位数）和表示数的范围。此表中的宽度和范围适用于字长为 16 位的系统中，在字长大于 16 位的系统中，short int 与 signed char 可能不等。

表 2—2 ANSI 标准中的数据类型

类　型	宽　度	范　围
char（字符型）	8	ASCII 字符
unsigned char（无符号字符型）	8	0～255
signed char（有符号字符型）	8	－128～127
int（整型）	16	－32768～32767
unsigned int（无符号整型）	16	0～65535
signed int（有符号整型）	16	同 int
short int（短整型）	8	－128～127
unsigned short int（无符号短整型）	8	0～255
signed short int（有符号短整型）	8	同 short int
long int（长整型）	32	－2147483648～2147483647
signed long int（有符号长整型）	32	同 long int
unsigned long int（无符号长整型）	32	0～4294967295
float（浮点型）	32	绝对值在 10^{-38}～10^{38} 约 7 位有效数字
double（双精度型）	64	绝对值在 10^{-308}～10^{308} 约 15 位有效数字
long double（长双精度型）	128	约 24 位有效数字

2.3 常量

常量是在程序执行过程中其数值不发生变化的量。常量在程序中不必进行任何说明就可以直接使用。C 语言中的常量有三种：数值常量、字符常量和字符串常量。此外，C 语言中还经常使用两种表现形式不同的常量：转义字符常量和符号常量。

2.3.1 数值常量

C 语言中的数值常量有两类：整数常量和浮点数常量。

1. 整数常量

整数常量（整常数）通常用十进制表示（不能以 0 开头），也可以用八进制和十六进制表示。表示整型数据时，不能带小数点，有小数点的数为浮点数。

用八进制表示时，开头要写上 0(零），其后再写上要表示的八进制数。八进制数各位用数码 0～7 之一表示。

用十六进制表示时，开头要写上 0x，十六进制数的各位除了可以用数码 0～9 外，还可以用英文字母 a～f 表示。a～f 对应十进制的 10～15。

例如,数值 1234 有下列三种不同的表示形式：

十进制表示法：1234

八进制表示法：02322

十六进制表示法 0x4d2

而数 21、021、0x21 却是三个数值不同的数。

21 的数值为十进制 21。

021 的数值为十进制 17。

0x21 的数值为十进制 33。

整数可以是正数，也可以是负数，因此整数前面可以带有正（＋)负(－)号,正整数时一般不写前面的＋号。

整数的取值范围因 CPU 的不同和编译系统的不同而不同。取值由 CPU 的字长所决定，即为 int 类型的取值范围。对于 IBM—PC 机来说，取值范围为：

－32768～＋32767

对于超过这个范围的整数，可以使用长整数。长整数的表示方法是在整数后缀上一个字母 L 或 l。如：

123456L,12334567l,01234567l,0x1234al 都是长整型数。

长整数的取值范围为 long int 类型的取值范围。在 16 位机上为：

－2147483648～2147483647

2. 浮点数常量

浮点数常量（浮点常数）是一个十进制数，它表示一个带符号的实数。C 语言中的浮点常数的构成规则如下：

[整数][．无符号整数] [E|e[－|＋] 无符号整数]

从浮点数的构成规则可以看出：浮点常数的值包括整数部分、小数部分和指数部分。其中

的无符号整数是一位或多位十进制数字(0～9 数字)。E(或 e)是指数符号。

整数部分和小数部分靠小数点连接。可以省略小数点前的整数部分，也可以省略小数点后的小数部分，但二者不可同时省略。小数点在没有小数部分时亦可省略。

3. 指数部分由 E(或 e)开头，后面跟着一个整常数，这个整数值可正可负。指数部分为正时可省略“＋”号不写。在浮点常数中任何地方不能出现空白字符。

4. 在不加说明的情况下，浮点常数为正值，如要表示负值，则需在常数值前面加一负号。

例如：

C语言浮点常数	表示十进制数
16.85	16.85
168.5E－1	16.85
168.5E2	16850
268e－2	2.68
－0.3e2	－3
－4.5e－2	－0.045
.35e2	35
.75	0.75
－.75	－0.75
.0075e3	7.5
－.0075e3	－7.5
－.0075e－2	－0.000075

2.3.2 字符常量

字符常量是括在单引号内的单个字符。例如：

'X'.'2'.'j'.'a'.'?'.'*'.'g'

都是 C 语言字符常量。其中的单引号只是作为定界符使用，并不表示字符常量本身。在两个单引号中包围的字符不能是单引号（'）和反斜线(\),即'''和'\'是错误的表达形式。关于这两个字符作为字符常量的表达形式，将在 2.3.3 中给予说明。

在 C 语言中，字符常量具有数值，其值为该字符在机器字符集中的代码值。因此，可以说字符常量实际上是一个字节的正整数。例如，在 ASCII 字符集（见附录 C)中：

'A'的数值为 65

'a'的数值为 97

'f'的数值为 102

'0'的数值为 48

'2'的数值为 50

'?'的数值为 63

'F'的数值为 70

字符常量可以像其它数一样，参与数值运算，进行字符间的比较。例如在程序中，经常进行的大写字母与小写字母之间的转换，就可以通过字符常量之间的数值运算来实现。比如，把大写字母 F 转换成小写字母 f,仅需将 F 的代码值加上 a 与 A 代码值之差，即：

'F'＋('a'－'A')＝70＋(97－65)＝102＝'f'.

2.3.3 转义字符常量

某些非图形字符和用于功能定义的特殊字符如：单引号 ′ ,双引号 ″ 和反斜线＼等，也可以表示成字符常量，其方法是用转义序列来表示字符。转义序列字符用反斜线＼后面跟有一个小写字母或一个三位的八进制或十六进制数表示。将转义序列字符用单引号括起来，就成为字符常量。这样表示的字符常量称为转义字符常量。表 2—3 列出了若干常用的转义字符。

表 2—3 常用的转义字符

转义序列	含　义
\b	退格
\f	换页（走纸）
\n	回车换行
\r	回车
\t	横向跳格（即 TAB）
\″	双引号
\′	单引号
\0	空值
\\	反斜线
\v	竖向跳格
\a	响铃报警
\ddd	八进制常量
\xdd	十六进制常量

其中，转义字符\ddd(d 为八进制数字 0～7 中之一）为八进制代码值和转义字符＼xdd(d 为十六进制数字 0～f 中之一）为十六进制代码值，它将字符的代码值转换为对应的字符，用它可以表示任一个字符。例如：

\101 或\x041 表示字符 A

\010 或＼x008 表示控制字符\b

\134 或\x05c 表示反斜线＼

这种表示方法的特例是＼0(后面无数字），它表示字符 NULL。

转义字符常量是由单引号将转义序列括起来表示的，例如：

′\n′　回车换行字符常量

′\0′　空操作字符常量

′\′′　单引号字符常量

′\\′　反斜线字符常量

′\″′　双引号字符常量

均为转义字符常量。

2.3.4 字符串常量

在 C语言中，字符串常量是用双引号括起来的字符序列表示的。例如：

″This is a string literal″

就是一个字符串常量，其中双引号仅作为定界符使用，并不是字符串中的字符。字符串常量中

的字符序列可以是字符集中的任何字符，包括转义字符。当为双引号"和反斜线\字符时，必须用转义字符表示。字符序列中字符的个数可以是零个或多个。

C语言的字符串常量在内存中存储时，编译系统自动在其尾部追加一个NULL字符。前面已介绍过NULL字符在ASCII码中，其代码值为0。在程序中NULL字符常用转义字符\0表示。每个字符占用一个字节，NULL字符亦占用一个字节，因此，长度为n个字符的字符串常量，在内存中占用n+1个字符空间。例如，字符串"string"有6个字符，将它存储在内存时，却占用7个字节空间，如下所示。

s	t	r	o	n	g	\0

字符串常量在内存中存储时，实际上存储的是各字符的代码值。例如上面的字符串常量在内存中的存储值为：

115	116	114	105	110	103	0

需要注意的是，字符常量与只有一个字符的字符串在存储形式上的不同。例如：字符常量'B'与字符串常量"B"其存储形式是不同的。

'B'的存储形式为 66 ，占用一个字节。

"B"的存储形式为 66 00 ，占用两个字节。

另外，在C语言中，字符串""表示空串，空串在内存中占用一个字节，其值为NULL的代码值0，存储形式为 00 。

2.3.5 符号常量

在C语言中，常量可以用符号代替，代替常量用的符号称为符号常量。为了便于与一般变量区分，符号常量一般采用大写英文字母。符号常量在使用之前必须预先定义，其定义的一般格式为：

#define 符号常量 常量

例如：

```
#define NULL 0
#define EOF -1
#define COUNT 100
#define PI 3.1415926
```

这里定义了四个符号常量：NULL、EOF、COUNT和PI，它们代替的常量分别是0、-1、100和3.1415926。

每个符号常量定义式只能定义一个符号常量，并且占据一个书写行，定义时必须以#号打头，而且每个定义式后面不能加分号；。实际上，它们不是C语言语句，而是发布给编译系统的预处理命令。关于预处理命令本书将在第9章中详细介绍。

符号常量定义一般写在程序的最前面，符号常量一经定义，它们就可以在程序中代替常量使用，且在程序中不允许修改它的值。

例2.1 计算半径为1,2…,100时的圆的面积和周长

```
# define   PI   3.1415926
# define   COUNT   100
main( )
{
    int i;
    float s,l;
    for(i=1;i<=COUNT;i++)
    {
        s=PI*i*i;
        l=2*PI*i;
        printf("r=%d\ts=%f\tl=%f\n",i,s,l);
    }
}
```

符号常量在程序中用于代替具有一定实际意义的常量。例如前面定义的 EOF,它的意义是“文件尾标”,NULL 的意义是字符串尾标,PI 代替圆周率的值等，因此，符号常量增强了程序的可读性。

另外，程序中某些常量，在调试、扩充或移植时要求修改其值。例如，上例中的 COUNT ,在程序调试时可暂时定义为 5，程序调试正确后，再计算大量的数据，改为 100等。这种常量通常也定义为符号常量。当它们需要修改时，只需修改其定义即可。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性而导致的错误。

2.4 变量

变量是在程序中其值发生变化的量。“说明”用于在程序中建立变量、函数与类型的名字及属性的关系。在 C 语言中，所有变量在使用之前必须加以说明。

2.4.1 变量 的说明

变量说明语句的一般形式为：

[存储类型] 数据类型 变量名表；

其中数据类型名为 2.2节中介绍的所有 C 语言有效的数据类型。变量名表可以由一个或由逗号分隔符分隔的多个标识符（变量名）构成。其存储类型本书稍后再作介绍。存储类型可以缺省。例如：

```
int i,j,k;
short int si;
unsigned int ui;
Long int li;
float f1,f2,s;
```

```
double d1,d2;
```

均为变量说明语句。

变量说明语句可以位于函数内部，也可以位于所有函数的外部（即在程序首部）。例如：

```
int a,b;
main( )
{
    int i,j;
    char c;
     ⋮
}
```

在函数内部说明的变量，称之为内部变量。在所有函数外部说明的变量，称之为外部变量。上例中，a 和 b 为外部变量，i、j 和 c 为内部变量。

在变量说明时，除了指定它的数据类型以外，还可以指定它的存储类型。C 语言提供了以下 4 种存储类型：

auto（堆栈型或称自动型）

static（静态型）

register（寄存器型）

extern（外部参照型）

例如：

```
auto int i,j;
static char c1,c2 ;
register int l,m ;
extern int x,y ;
```

在变量说明时，可以不指定存储类型（即存储类型可以缺省），缺省时内部变量的存储类型为 auto 型。

关于存储类型的具体意义，本书将在第 9 章中再详细介绍。

2.4.2 变量的初始化

在 C 语言中，变量说明时，可以通过在变量名之后放置一等号和常量给该变量赋初值，这称之为变量的初始化。例如：

```
char C='B' ;
int i=100,j=200;
static int k=300 ;
float x=4.5,y=7.65;
```

不同存储类型的变量，其初始化的意义也不同，关于这一点本书将在第 9 章中再做详细介绍。

习 题 2

1. 判断下列 C 语言标识符，指出哪些是合法的，哪些是不合法的。合法者在其后面的括号内打√号，不合法者打×号，并注明不合法的理由。

(1) Ab(　　)
(2) ab(　　)
(3) 3C(　　)
(4) C3P(　　)
(5) π(　　)
(6) x^2(　　)
(7) DPD—11(　　)
(8) —P5(　　)
(9) PDP _ 11(　　)
(10) x3(　　)

2. 判断下列浮点常量的合法性。合法者在括号内打√号，并写出它所表示的十进制数值。不合法者在括号内打×号，并注明其不合法的理由。

(1) 36.78e+4 (　　)
(2) 36.78e-4 (　　)
(3) 36.78 (　　)
(4) .3678 (　　)
(5) .e+4 (　　)
(6) .e-4 (　　)
(7) 36.78e4 (　　)
(8) 36.78e4.5 (　　)

3. 判断下列字符常量或字符串常量的合法性，合法者打√，否则打×。

(1) 'abc' (　　)
(2) ',' (　　)
(3) "," (　　)
(4) "ab,?" (　　)
(5) "ab','" (　　)
(6) '"' (　　)
(7) "ab\7" (　　)
(8) "ab\"" (　　)

4. 判断下列变量说明语句的正确性，正确的打√，表达不正确者打×，并请将其更正。

(1) char:C1,C2,C3; (　　)
(2) C1,C2,C3:char; (　　)
(3) char C1,C2,C3 (　　)
(4) int,i j k; (　　)
(5) int ijk; (　　)