第 1 章　C 语言概述

计算机系统是由硬件和软件两部分组成的。硬件是指计算机中有尺寸和质量的物理部件，软件主要是指控制计算机硬件工作的程序。要运行计算机硬件来实现各种功能、完成各种任务，就必须让它执行相应的程序指令，而这些程序指令是依靠计算机语言编制的。

计算机语言是用来编写计算机软件程序的，就像自然语言编写文章一样，计算机语言也包括文字（称为字符）、词汇（称为标识符）和语法规则。计算机语言分为低级语言和高级语言，低级语言面向计算机硬件，用来编写程序时十分麻烦；高级语言面向问题，其词汇和语法规则接近于自然语言和数学表达式，程序的编写和移植（移植指的是程序经过简单修改后就能应用在不同类型的计算机上）都比较方便。由于问题的多样性，高级语言出现了很多种类，用于解决不同类型的应用问题。

在多种高级语言当中，C 语言有其独到之处。它作为一种高级程序设计语言，具备方便性、灵活性和通用性等特点，适合编写应用软件程序；同时，它还向程序员提供了直接操作计算机硬件的功能，具备低级语言的特点，适合于各种类型的系统软件的开发。因此，C 语言是深受软件工作者欢迎的程序设计语言。

本章首先简单介绍 C 语言的发展与特点，然后从程序设计的角度介绍有关的基本概念，最后通过几个简单的 C 程序实例描述 C 语言程序的基本结构。

1.1　C 语言的发展与特点

1.1.1　C 语言的发展

C 语言是由美国贝尔实验室的 Dennis Ritchie 设计，在 BCPL（Basic Combined Programming Language）语言和 B 语言的基础上，于 1972 年在一台使用 UNIX 操作系统的 DEC PDP-11 计算机上实现的，它取 BCPL 的第二个字母 C 而得名。

BCPL 语言是由英国剑桥大学的 Martin Richards 于 1967 年提出的一种系统程序语言。1970 年，美国贝尔实验室的 Ken Thompson 以 BCPL 语言为基础，做了进一步简化，设计出了很简单的而且很接近硬件的 B 语言，并且用 B 语言在 PDP-7 计算机上写出了第一个 UNIX 操作系统，1971 年又用 B 语言在 PDP-11/12 计算机上写出了 UNIX 操作系统。

C 语言既保持了 BCPL 语言和 B 语言的优点（精炼、接近硬件），又克服了它们的缺点（过于简单、功能有限、数据无类型等）。最初的 C 语言只是为描述和实现 UNIX 操作系统而设计的一种系统工作语言。C 语言与 UNIX 操作系统密切相关，UNIX 操作系统、编译程序和应用程序大多由 C 语言写成。1973 年，Ken Thompson 和 Dennis Ritchie 两人合作，将原来由他们于 1969 年用汇编语言写的 UNIX 操作系统的 90%以上用 C 语言改写，形成了 UNIX 第 5 版。

最初的 C 语言主要在美国贝尔实验室内部使用。直到 1975 年 UNIX 第 6 版公布后，C 语言的突出优点才引起人们的普遍关注。1977 年出现了不依赖于机器的 C 语言编译文本《可移植 C 语言编译程序》，使得用 C 语言开发的软件在移植到其他机器时所做的工作大大简化，

这也推动了 UNIX 操作系统迅速地在各种机器上实现。随着 UNIX 操作系统日益广泛的使用，C 语言也迅速得到推广。在 C 语言与 UNIX 操作系统的发展过程中，它们相辅相成，相互促进。1978 年以后，C 语言已先后移植到大、中、小、微型机上，已独立于 UNIX 操作系统和 PDP 计算机。

1978 年，Brian Kernighan 和 Dennis Ritchie 两人以 UNIX 第 7 版中的 C 语言编译程序为基础，合著了影响深远的名著《C 程序设计语言》（*The C Programming Language*），这本书中介绍的 C 语言此后一直是事实上的 C 语言标准版本。

1983 年初夏，美国国家标准化协会（ANSI）专门成立一个委员会为 C 语言制定了 ANSI 标准，这个标准称为 ANSI C，相比原来的 C，它有了很大的发展。1988 年，Brian Kernighan 和 Dennis Ritchie 按照 ANSI C 标准修改了他们的经典著作《C 程序设计语言》。Turbo C、Visual C++等完全是按照 ANSI 的 C 语言标准实施的，是一种快速、高效的编译程序，并提供了集成开发环境。

目前，C 语言已风靡全球，成为应用最广泛的计算机高级语言之一。C 语言不仅可用来开发系统软件，也可用来开发应用软件。C 语言既具有一般高级语言的特性（可读性、可移植性好，结构化程序设计），又具有低级语言（如汇编语言）的特性（可直接对硬件进行操作，如对位、字节和内存的操作），是一种集高级语言和低级语言优点于一身的语言。因此，C 语言常被称为计算机“中级语言”。

1.1.2 C 语言的特点

C 语言是世界上自 20 世纪 70 年代以来最有影响的计算机语言之一，也是最常用的语言之一。C 语言能得到越来越广泛的使用，要归功于它有不同于（或优于）其他语言的特点，特别是优良的“可移植性”。它不需要做大的修改就能在每台支持 C 语言的机器上运行。概括起来，C 语言的主要特点如下：

1）结构简洁、紧凑。C 语言一共只有 32 个关键字（即保留字），压缩了一切不必要的成分，因此 C 语言编写的源程序非常简练，使得 C 语言被公认为一种比较强有力的语言。C 语言的运算符和表达式的表示方法也力求简练，易于学习，便于理解和使用。

2）数据类型丰富，控制结构完善。C 语言数据类型丰富，它不仅有不同长度的整型、字符型和单双精度实型（即浮点型）等基本数据类型，还有指针类型以及数组、结构体和联合体等构造数据类型，能用这些类型来实现各种复杂的数据结构（如链表、树、栈等）的运算。尤其是指针类型，使用起来更为灵活、多样。

C 语言提供了 9 种控制语句来实现 3 种结构（顺序、分支和循环结构）的程序设计，以支持良好的程序结构。同时用函数作为程序模块以实现程序的模块化，每个 C 程序由若干函数组成，用户不仅可以调用丰富的库函数，而且可以调用自己编写的自定义函数。因此，C 语言是结构化的理想语言，符合现代编程风格要求。

3）运算符丰富，表达能力强。C 语言的运算符非常丰富，共有 34 种运算符（详见第 3 章）。它不仅具有算术运算符、关系运算符、逻辑运算符等，还把括号、赋值、逗号、变量值增（减）1、强制类型转换等都作为运算符处理，同时还将字符、逻辑值等数值化。C 语言的运算符不仅具有优先级的概念，而且具有结合性的概念。因此，C 语言的运算类型极其丰富，表达式类型多样化。灵活使用各种运算符和表达式，不仅可以使程序简洁，而且可以实现在其他语言中难以实现的运算。

4）语法限制不太严格，程序设计自由度大。C 程序的书写形式自由，主要用小写字母，一个语句可以写在几行，一行也可以写几个语句。

一般的高级语言语法检查比较严格，能检查出几乎所有的语法错误，而 C 语言允许编写者有较大的自由度，因此放宽了对语法的检查。程序员应仔细检查程序，保证其正确，而不要过分依赖 C 编译程序去查错。“限制”与“灵活”是一对矛盾，严格限制就会失去灵活性，而强调灵活必然会放松限制。对于一个不熟练的人员，特别是初学者，编写一个正确的 C 程序可能要比编写一个其他高级语言程序难一些。

5）同时具有高级语言和低级语言的特点。C 语言允许直接访问物理地址，能进行位操作，可以直接对硬件进行操作，能实现汇编语言的大部分功能。因此，C 语言既具有高级语言的功能，又具有低级语言的许多功能；既可用来编写系统软件，又可用来编写应用程序。

6）目标代码质量高，可移植性好。C 语言程序不仅编译的目标代码质量高，程序执行效率高，而且可移植性好。可移植性是指从一种硬件环境中不加修改或稍加修改地移到另一种环境中运行的性能。虽然 C 适合用于许多计算机机型，但独立于具体的计算机系统。灵活地运用预处理程序可以提高程序的可移植性。

1.2　C 语言的字符集与标识符

1. 字符集

满足 C 语言词法要求的字符包括所有大小写的英文字母、阿拉伯数字及部分特殊符号，其中的特殊符号如图 1.1 所示。

```
+    -    *    /    %    _    =    <    >    &
~    (    )    [    ]    .    {    }    :    ?
;    ″    !    #    ′    ^    空格
```

图 1.1　C 语言字符集中的特殊符号

2. 标识符

C 语言中的标识符主要用来表示常量、变量、函数和类型等的名字，它们是只起标识作用的一类符号，包括关键字、预定义标识符和自定义标识符三类。

关键字是指具有特定含义的字符串，这些字符串不允许用户把它们当作变量名使用。C 语言的关键字共有 32 个，它们都用小写英文字母表示，如图 1.2 所示。后续各章中会逐步介绍它们的含义。

```
auto      break     case      char      const     continue  default   do
double    else      enum      extern    float     for       goto      if
int       long      register  return    short     signed    sizeof    static
struct    switch    typedef   union     unsigned  void      volatile  while
```

图 1.2　C 语言的 32 个关键字

除上述的关键字外，还有一类具有特殊含义的标识符，它们被用作库函数名和预编译命令，这类标识符在 C 语言中称为预定义标识符。一般来说，不要把这些标识符再定义为自定义标识符使用，如 `printf`、`include` 等属于预定义标识符。

自定义标识符是程序员根据自己的需要定义的一类标识符，用于标识变量、符号常量、

自定义函数、构造类型名和文件名等。这类标识符只能由英文字母、数字和下画线构成，但开头字符只能是字母或下画线。下画线常用来把几个词语隔开，以增强代码的可读性，如变量 num_of_people。

C 语言区分大小写，关键字和预定义标识符全部使用小写字母，自定义标识符也区分大小写，如 NUMBER、Number 和 number 被视为三个不同的标识符。

一个自定义标识符可由一个或多个字符组成，但其长度是有限制的，C 语言只能识别前 31 个字符，因此如果两个标识符的前 31 个字符相同，那么它们被视为同一个标识符。

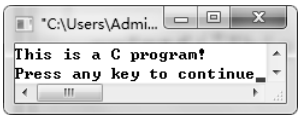
1.3 程序举例

下面介绍几个简单的 C 程序，以便分析 C 程序的构成及其特性。

【例 1.1】输出一行信息“This is a C program!”。

```
#include<stdio.h>       /*将标准输入/输出函数的头文件包含到本文件中*/
void main()             /*主函数头*/
{                       /*函数体开始 */
  printf("This is a C program!\n");/*输出一串字符，\n 为换行符，不显示*/
}                       /*函数体结束 */
```

运行结果：



说明：运行结果最后的“Press any key to continue”是运行窗口暂停的显示提示，表示按任意键继续。后面的例题都是这样，因此不再说明。

main 是主函数名，每个 C 程序都必须有一个 main 函数；函数名后要有圆括号，圆括号内可以有函数的参数，本例是一个无参函数，所以是空括号。函数体是由花括号“{}”括起来的部分，函数体内有若干语句。C 语言的每个语句都以分号“;”结束。

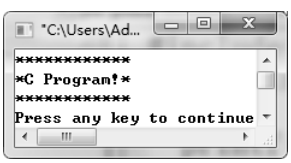
printf 是输出函数，它属于 C 语言的库函数，库函数名 printf 是系统定义的预定义标识符。

第一行的“#include<stdio.h>”内容与输出函数 printf 有关联，几乎每个 C 程序都有这一行内容。

“/*.....*/”是注释部分，它表示从“/*”开始到“*/”之间的内容为注释，注释只是给程序做的标记，目的是便于读懂程序，对编译和运行不起作用。注释可以加在程序中的任何位置。

注意：程序调试运行方法参见附录 A。

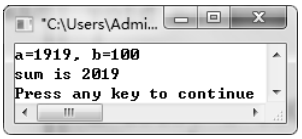
思考练习：输出如下结果应该怎样改写上面的程序？最好用两种改法。



【例 1.2】求两个整数之和。

```
#include<stdio.h>       /*将标准输入/输出函数的头文件包含到本文件中*/
void main()             //主函数头
{
    int a,b,sum;        /*定义 a、b、sum 为整型变量*/
    a=1919;b=100;       /*给变量 a、b 赋值*/
    sum=a+b;            /*给 sum 赋值*/
    printf("a=%d,b=%d\n",a,b);        /*输出变量 a、b 的值*/
    printf("sum is %d\n",sum);        /*输出变量 sum 的值*/
}
```

运行结果：



本程序的作用是求两个整数之和。先把两个整数放在变量 a 和 b 中，再将和值存放在变量 sum 中。

main 是主函数名，每个 C 程序都必须有一个 main 函数；函数名 main 后的空括号表示本例是一个无参函数。函数体是由花括号“{}”括起来的部分，函数体内首先是变量的定义（也称变量的声明）部分，然后是执行语句部分。C 语言的每个变量定义语句和每条执行语句都以分号“;”结束。

第 4 行定义变量 a、b 和 sum 是整型（int）变量（此变量只存放整数）。第 5 行和第 6 行均为赋值语句。第 5 行包含两条赋值语句，分别将变量 a 和 b 赋值为 1918 和 100，这里的“=”是赋值符号，意思是把右面的值赋给左面已经定义的变量；第 6 行将表达式 a+b 的值赋给变量 sum。

第 7 行和第 8 行均是输出函数调用语句，printf 是 C 语言中的输出函数（详见第 4 章）。双引号括起来的是格式控制字符串，它可以包括“格式说明”和普通字符，其中的每个“格式说明”都由“%”后接格式字符构成，字符“d”是以十进制形式输出整型数据的格式字符；格式控制字符串中的普通字符按原样输出，其中“\n”是转义字符，表示换行（即换行符）。printf 函数中括号内的最右边给出输出表达式，这里给出的是变量 a、b 或 sum，输出表达式的个数应该与格式控制字符串中的“格式说明”的个数一致。

对于一行尾部的注释，在 VC++ 2010 环境中也可以使用“//...”的形式，它表示本行中从“//”开始到行尾的内容为注释。本程序的第 2 行就是采用这种方式注释的。

第 1 行是编译预处理命令中的“文件包含”命令（详见第 9 章），它的作用是将标准输入/输出函数的头文件 stdio.h 包含到本程序文件中。stdio.h 头文件中包含了与标准输入/输出操作有关的函数的原型声明等信息，因此，凡是需要使用标准输入/输出函数（如 printf、scanf 等函数）的 C 程序，都应该在文件的开头加上这条文件包含命令，以便编译系统在编译时能够从头文件中获得有关的信息。

思考练习：输出 3 门课的平均分（如 a=60;b=85;c=70;），如何改写上面的程序？最好写出多种改法，看看运行结果有什么问题。为什么与预测的结果不完全相同？

说明：C 语言中的加、减、乘、除运算符分别为+、-、*、/，先算乘除，后算加减。

【例 1.3】从键盘输入两个整数，求它们之中最大的数。

```
#include<stdio.h>
int max(int x,int y)        /*定义 max 函数，返回值为整型，x、y 为形参变量，x、y 为整型*/
{
    int z;                  /*定义 max 函数中用到的变量 z 的类型*/
    if(x>y)  z=x; else z=y; /*分支语句*/
    return (z);             /*返回变量 z 的值，通过 max 带回到调用处*/
}
  void main()
  {
    int  m, a, b;
    printf("Please input 2 integer numbers: \n");
    scanf("%d, %d",&a,&b);       /*从键盘输入两个整数分别给 a、b*/
    m=max(a,b);                  /*调用 max 函数，将 a 和 b 中较大的值赋给变量 m*/
    printf("Max number is %d\n",m);       /*输出变量 m 的值*/
}
```

运行结果：

```
Please input 3 integer numbers:
5, 3↙
Max number is 5
```

说明：在本书中，↙代表回车键。

本程序由两个函数构成：主函数（即 main 函数）及自定义函数 max。函数 max 的作用是：将形参变量 x、y 中较大者的值赋给变量 z，并通过 return 语句将变量 z 的值返回给主调函数。形参变量的值是从主调函数中的函数调用语句的实参中传送过来的。main 函数的作用是：从键盘输入两个整数分别给变量 a、b，通过调用函数 max 求两个数中最大一个的值。调用函数 max 求出两个数中较大的一个。

main 函数在调用 max 函数时，将实参 a、b 的值分别传送给 max 函数中的形参变量 x、y；通过执行 max 函数得到一个返回值（即 max 函数中变量 z 的值），把这个值（它是变量 a 和 b 中较大一个的值）赋给变量 m。调用函数 max 后，变量 m 已是变量 a、b 中最大的值。

第 11 行的输出函数调用语句用来输出一行提示信息，双引号内的字符串原样输出。第 12 行是输入函数调用语句，scanf 是 C 语言中的输入函数（详见第 4 章）。双引号括起来的是格式控制字符串，含义同前例。scanf 函数中括号内的最右边给出输入变量存储单元指针（请读者先将指针的概念理解为存储单元的地址，有关指针的概念详见第 10 章），这里给出的是变量 x 的存储单元地址，“&”是求地址运算符。输入变量存储单元地址的个数应该与格式控制字符串中“格式说明”的个数一致。如果格式控制字符串中有普通字符，那么在输入时这些普通字符也得按原样输入。

思考练习：如果输入三个数求最大值，那么怎样改写上面的程序？

通过以上几个例子，我们可以看到：

1）C 程序是由函数构成的。一个 C 程序必须包含且只能包含一个 main 函数，不能有一个以上的 main 函数；除 main 函数外，可以包含若干其他函数。被调用的函数既可以是库函

数（即系统提供的标准函数），又可以是用户根据自己的需要编写的函数（即自定义函数）。函数是 C 程序的基本组成单位。

C 语言中的函数相当于其他语言中的子程序、过程。C 语言的这种特点使其容易实现程序的模块化。

2）一个 C 函数由函数头和函数体两部分构成。详见第 8 章。

① 函数头部分。包括函数属性（如内部函数和外部函数）、函数类型（即函数返回值的类型）、函数名、形参变量及其类型声明列表。

一个函数名后面必须跟一对圆括号，函数参数可以没有，但这对圆括号不能没有。

例如，例 1.3 中 max 函数的头部分如图 1.3 所示。

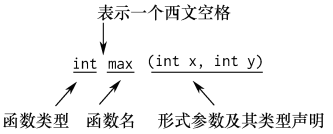


图 1.3　max 函数头部分的说明

② 函数体。即函数头部分以下花括号{}内部的部分。如果一个函数内有多个花括号，那么最外层的一对{}为函数体的范围，内层的{}为复合语句。

函数体一般包括：

a. 变量定义部分。变量定义部分应放在函数体的执行语句之前，一个函数可以没有变量定义部分。

b. 执行语句部分。由若干语句组成，一个函数也可以没有执行语句部分。如空函数

```
void dump()
{
}
```

既没有变量定义部分，又没有执行语句部分，它什么也不干，但这是合法的。

3）一个 C 程序中各个函数的定义是独立的、平行的。无论各个函数之间的位置如何安排，一个 C 程序总是从 main 函数开始执行；如果不出现意外的话，那么它总是从 main 函数结束整个程序的运行。

4）C 程序书写格式自由，一行内可以写几个语句，一个语句也可以写在几行上。C 程序的语句没有行号，只要求每个变量声明和执行语句均以分号“;”作为结束标记，分号是 C 语句的必要组成部分。

5）C 语言本身没有输入/输出语句。输入/输出操作是由 scanf 和 printf 等库函数来完成的。C 语言对输入/输出实行“函数化”。

6）可以用/*……*/对 C 程序中的任何部分进行注释。为增加程序的可读性，源程序都应加上必要的注释。

1.4　本章小结

本章简单介绍了 C 语言的发展历史、功能和特点。计算机最终是靠执行程序完成任务的，而程序是用计算机语言编写的，C 语言是众多高级语言中流行的一种。

本章通过几个案例，简单分析了 C 程序的基本构成，即 C 程序是由函数构成的，函数

是由语句构成的，语句是由基本词汇构成的，基本词汇是由字母、数字、符号构成的。

尽管目前我们还没有编写较大的程序，但是已经可以领会到算法的重要性。在编写程序之前，必须深思熟虑，形成解题的思路即算法，利用某种方式将其描述出来，并以此为依据编写程序。

建议读者阅读书后的附录 A，以便了解 C 语言的运行和调试环境，为后续学好 C 语言打下良好的基础。

习题

1.1 指出下列 C 语言标识符哪些是不合法的。合法的标识符中哪些是关键字？哪些是预定义标识符？哪些是自定义标识符？

```
Total,_debug,Large&Tall,printf,Counter1,begin_,for,Case,A3_B3,void,_123,IF
```

1.2 C 语言程序可以由哪些不同的部分组成？函数由哪几部分组成？

1.3 C 语言中的关键字共有多少个？是否可以用大写字母表示？

1.4 编写一个 C 程序，输入 a,b,c 三个值，输出其中最小的值。

1.5 C 语言有哪些主要用途？