第 1 章　C 语言概述

要点：计算机是机器。计算机没有智慧，其“智慧”就是程序员的智慧。因为计算机的工作过程是程序员通过编写程序使用指令控制其实现的。

1.1　C 语言简介

要点：C 语言在算法描述方面是无与伦比的。

1. C 语言的发展过程

C 语言是在 20 世纪 70 年代初问世的。1978 年，美国电话电报公司(AT&T)贝尔实验室正式发布了 C 语言。同时，B. W. Kernighan 和 D. M. Ritchit 合著了著名的 *The C Programming Language* 一书。1983 年，美国国家标准协会(American national standards institute)在此基础上制定了一个 C 语言标准，通常称为 ANSIC，其是一个面向过程的程序设计语言。在 C 语言的基础上，贝尔实验室的 Bjarne Stroustrup 同年推出了 C++ 语言，C++ 语言进一步扩充和完善了 C 语言，成为一种面向对象的程序设计语言。因此，掌握了 C 语言，再进一步学习 C++，就能以一种熟悉的语法来学习面向对象的语言，从而达到事半功倍的效果。

2. C 语言的特点

(1) 简洁、紧凑，使用方便，易于理解和记忆。

(2) 运算符丰富。C 语言的运算符共有 34 种。C 语言把括号、赋值、逗号等都作为运算符进行处理，从而使 C 语言的运算类型极为丰富，可以实现其他高级语言难以实现的运算。

(3) 数据结构类型丰富。C 语言中的数据类型从基本数值类型到数组和指针再到自定义数据类型，为各种复杂算法程序的实现提供了必要的数据存储和操作保障。

(4) 具有 9 个结构化的控制语句。结构化控制语句使程序设计规范，易读性强，可移植性强，使面向过程的程序设计更加简单。

(5) 语句组合灵活，程序设计自由度大。数量有限的控制语句结合丰富的运算符而构成的表达式和表达式语句，可以设计出各种复杂算法程序和过程控制程序。

(6) C 语言具有强大的底层硬件操作能力。C 语言允许直接读写物理地址和使用地址访问对应单元，同时 C 语言具有位(bit)操作指令，能实现汇编语言的大部分功能，可以直接对硬件进行操作。因此 C 语言是目前嵌入式系统开发中极为重要的语言工具。

(7) 生成目标代码质量高，程序执行效率高。C 语言编译器效率与其他高级语言相比，

最接近汇编语言或机器语言。无论是代码存储效益还是处理器执行效率,C 语言都是最优秀的,所以特别适合那些程序存储空间受限和处理器运算处理能力不高的嵌入式系统开发。

1.2 C 语言程序设计基本语法规则

要点:“麻雀”虽小但五脏俱全。

与学习其他高级语言程序设计一样,在全面学习 C 语言程序设计主要内容之前,通过简单的几个例程来介绍 C 语言程序设计的基本结构和基本语法规则。对简单程序进行剖析可以为后续相关知识的学习打下基础,使理论知识学习与实践操作能够结合进行。

1. 结合实例介绍基本语法

【例 1-1】 输出“Hello Word!”文字信息的基本程序。

```
#include<stdio.h>
void main(void)
{
    printf("Hello Word!\n");
}
```

(1) C 语言是结构化程序设计语言,也是函数式程序设计语言。所有执行性指令语句必须写在函数内部。只有部分声明性语句、定义性语句和预处理指令语句才可以写在函数外部。

(2) 每个程序的所有程序代码可以写在一个文件中,也可以写在多个文件中,由项目管理器将其组织和管理起来。这些文件中可以根据需要书写(严格来讲称为定义)多个函数,但在这个工程中(即这个程序的所有代码)必须有且只有一个主函数,即名字为 main 的函数。注意,main 全部是小写字母。

(3) 函数由函数首部和函数体(函数定义主体)构成,如例 1-1 中 main 字样所在的一行是位于 main 函数定义整体代码的第一行,即首行位置,所以称为函数首部。函数首部之下是由一对“{”“}”界定起来的函数体。函数体是实现本函数功能的所有语句。程序从起括号“{”后面的第一条语句开始执行,默认按照从上到下、从左到右的顺序执行其中的语句,如果其中有结构化指令,则按照结构化指令规则执行,一直执行到 return 语句或回括号“}”位置结束。函数的具体定义规则在第 9 章中进行讲解。

(4) main 函数中使用“printf("Hello Word!\n");”语句完成向标准输出设备(显示器)输出“Hello Word!”字样的程序任务。该语句由 printf 函数调用和“;”构成。C 语言中除部分结构控制语句外,其他语句是需要使用“;”结尾的,所以在 C 语言中的“;”称为语句结束标志。

(5) 在函数体中可以使用(也称调用)开发环境系统中已有的系统函数或之前写好的功能函数,但一般需要使用 #include 预编译指令将其对应的头文件包含到本程序中。#include语句要写在函数外部,一般写在文件最前面(最上面)的行上,如本例中的 #include<stdio.h>就是在 main 函数中调用的 printf 函数的对应头文件,注意,其是编译预处理指令(在第 10 章中会详细讲解),所以不能在后面加“;”。

【例 1-2】 简单程序。

```
#include<math.h>
#include<stdio.h>
void main()
{
    int a=0;
    double x,s;
    printf("input number:\n");
    scanf("%lf",&x);
    x=a+2+x;
    s=sin(x);
    printf("sine of %lf is %lf\n",x,s);
}
```

(1) 在 main 函数之前的两行称为预处理命令(详见第 10 章)。预处理命令还有其他几种,这里的 include 称为文件包含命令,其意义是把尖括号＜＞或引号""内指定的文件包含到本程序中,成为本程序的一部分。

被包含的文件通常是由系统提供的或者在本工程所在文件夹中的且扩展名为.h 的文件,因此也称为头文件。C 语言编辑环境中的头文件中包括程序设计时需要用到的标准库函数的函数原型,因此,凡是在程序中调用一个库函数,都必须包含该函数原型所在的头文件。

在本例中使用了三个库函数,即输入函数 scanf、正弦函数 sin 和输出函数 printf。sin 函数是数学函数,其头文件为 math.h 文件,因此在程序的主函数前用 include 命令包含了 math.h。scanf 和 printf 是标准输入/输出函数,其头文件为 stdio.h,在主函数前也用 include 命令包含了 stdio.h 文件。

(2) C 语言规定对 scanf 和 printf 这两个函数可以省去对其头文件的包含命令,所以在本例中也可以删去第二行的包含命令＃include＜stdio.h＞。同样,在例 1-1 中使用了 printf 函数,也可省略包含命令。

(3) main 函数的函数体从结构上分为两部分,是按位置前后来划分的,每个区域中只能书写符合其要求的代码,不能混排(C++ 语言中不受此规则限制)。

前面区域是声明区,只能书写声明或定义语句,如“int a＝0;”和“double x,s;”这两个都是变量定义语句,具体语法在后面章节中进行讲解。只有当所有需要的定义或声明性语句都写完了,才能写运算或处理性可执行语句,如本例中的“printf("input number:\n");”到“printf("sine of %lf is %lf\n",x,s);”的所有语句。

(4) 程序的功能。程序中先定义一个整数型(int 类型)变量 a,初值是数字 0,再定义两个双精度(double 类型)变量 x 和 s,但没有初值,在后面的程序中要先给变量赋值后才能参与运算或被其他函数使用。有了变量,后面就是使用这些变量通过计算、存储和处理等操作程序实现相应功能。本程序完成的功能是使用“printf("input number:\n");”语句输出提示信息“input number:”后,从键盘输入 1 个数给 x 变量,再将 a 变量的值加上数字 2 再加上 x 变量的值,回写给 x 变量,之后使用 sin 函数求 x 的正弦值,再将得到(函数返回)的正弦值通过赋值运算符写入 s 变量中,然后使用“printf ("sine of %lf is %lf\n",x,s);”调用 printf 函数,在标准输出设备上输出以下结果:

```
sine of 7.000000 is 0.656987
```

2. C 语言运算或处理性可执行语句

C 语言的运算或处理性可执行语句又分单纯的函数调用语句、表达式语句、含有函数调用的表达式语句、结构控制语句和复合语句等。这里只涉及前三种，其他语句在后续章节中进行详细讲解。

(1) 单纯的函数调用语句。调用已有函数(别人定义好的，一般要使用 # include 将相应的头文件包含在调用代码所在的文件中)实现已知的功能，被调用的函数没有返回值(数学意义上的结果值)或者有返回值但不需要使用，语句以分号结束。如“printf("input number:\n");”语句和“scanf("%lf",&x);”分别是向标准输出设备(对于嵌入式系统要看具体定义，一般是串行通信接口)输出信息“input number:”和从标准输入设备输入一个高精度实数给变量 x，这两个函数使用的规则和特点在后面章节中进行讲解。

(2) 表达式语句。由运算符和运算对象组合构成的一个语句，语句运行完会得到一个表达式的值，同时可以将其赋值给一个变量，语句以分号结束。如“x=a+2+x;”语句是完成 a 变量的值加上数字 2 再加上 x 变量的值，得到的结果再写入 x 变量中，覆盖掉本指令执行前 x 中的值。

(3) 含有函数调用的表达式语句。也是表达式语句中的一种，只不过是函数调用的返回值作为表达式的一个运算对象，再参与表达式的运算过程，语句以分号结束。如本例中“s=sin(x);”语句，其调用 sin 函数得到返回值(数学意义上的结果值，具体程序实现方法在第 9 章中进行讲解)，通过赋值运算符写入变量 s 中。

3. 输入/输出函数简介

在前两个例子中用到了输入/输出函数 scanf 和 printf，以后会详细介绍。这里先简单介绍一下它们的格式。

scanf 和 printf 这两个函数分别称为格式输入函数和格式输出函数。其意义是按指定的格式输入/输出数据值。完成计算中存储的二进制数与输入/输出设备上输入/输出的文字信息之间的转换。这两个函数调用语句中的括号里由格式控制字符串和参数表两部分组成，中间使用“,”间隔。

(1) “格式控制字符串”是一个由多种功能结构的格式字符串组合起来的一个组合字符串，所以也可以叫作“格式控制组合字符串”。其必须用双引号括起来，其中包括输入/输出格式控制子串和非格式控制符。每个格式控制子串控制输入/输出信息或数据量的位置和格式，一般由%开始，再由多个格式控制符组成，具体内容在第 4 章中进行详细讲解。

(2) 在 printf 函数中，如果在格式控制字符串内出现非格式控制字符，在标准输出设备(如显示屏)上原样输出该字符；在 scanf 函数中，如果在格式控制字符串内出现非格式控制字符，则要在标准输入设备(如键盘)上的当前位置原样输入该字符。

(3) 参数表中给出了输入或输出的数值或可以计算出具体数值的表达式。当有多个输入或输出的数量时，必须按顺序书写并用逗号间隔。参数表的顺序与格式控制字符串中各个格式控制字符子串一一对应。格式控制字符子串的位置就是以该格式控制字符子串控制的格式输入/输出对应数量值的对应位置。例如：

```
printf ("sine of %lf is %lf\n",x,s);
```

例 1-2 中该 printf 函数调用语句中的两个%lf 为格式字符子串，表示按双精度浮点数处理。

它在格式字符串中两次出现，对应需要输出的 x 和 s 两个变量值的输出格式和位置。其余字符为非格式字符，则照原样输出在显示器上，对应运行输出结果 sine of 7.000000 is 0.656987 。

【例 1-3】 程序的基本结构。

```
int max(int a,int b);              //函数说明
void main()                        //主函数
{
    int x,y,z;                     //变量说明
    //int max(int a,int b);        //函数说明
    printf("input two numbers:\n");
    scanf("%d%d",&x,&y);           //输入，赋值给 x 和 y 变量
    z=max(x,y);                    //调用 max 函数
    printf("maxmum=%d",z);         //输出
}
int max(int a,int b)               //定义 max 函数
{
    if(a>b)
        return a;                  //把结果返回主调函数
    else
        return b;                  //把结果返回主调函数
}
```

(1) 本程序的功能是由用户输入两个整数，程序执行后输出其中较大的数。

(2) 一个程序由多个文件中的多个函数及声明部分组成。多个函数可以由 main 函数＋多个系统函数的调用构成，也可以由一个 main 函数＋多个程序员定义的函数及调用构成，还可以由一个 main 函数＋多个程序员定义的函数＋调用多个系统函数构成。

(3) 本程序由两个自定义函数及系统函数调用构成。函数之间是并列关系。main 函数和 max 函数是用户函数，需要编程者定义；scanf 函数和 printf 函数是系统函数，是已经存在的函数，调用前只要有对应的、必要的头文件包含预处理指令就可以，包含指令写在文件开头的声明部分。

(4) 程序执行从主函数开始，主函数中可以调用其他函数，如 max 函数。max 函数的功能是比较两个数，然后把较大的数返回给主函数。

(5) max 函数是用户自定义函数。如果定义在调用点["z＝max(x,y);"语句]之后或在别的文件中，需要在调用点之前的声明部分(可以是调用点所在函数的声明部分，也可以是本文件的声明部分)使用声明语句(函数首部加分号)进行声明。在此例中也可以放在 main 函数体的"{"后的声明部分，这种方式现在使用得较少，所以使用//改成注释文字。

可见，在程序的说明部分中，不仅可以有变量说明，还可以有函数说明。关于函数的详细内容将在第 9 章中进行介绍。

(6) 在程序的每行后用//可以开始写注释文字，该注释文字不被编译软件编译。//开始的注释文字只能写一行，不能换行，如果要写多行注释，必须在每行的注释文字前加//。

另外有的时候需要写很多行注释，或者把大面积的代码注释掉，不让编译软件对其进行编译，这时可以使用/＊和＊/将所有需要注释的文字括起来。

注意：/＊和＊/要成对使用，并且/＊在前、＊/在后。

(7) 本例程序的执行过程是按 main 函数体中的语句和其调用的函数体中的语句的书写顺序和结构控制语句执行的：首先在屏幕上显示提示串，请用户输入两个数，按 Enter 键后由 scanf 函数语句接收这两个数并送入变量 x 和 y 中，然后调用 max 函数，并把 x 和 y 的值传送给 max 函数的参数 a 和 b。在 max 函数中比较 a 和 b 的大小，返回大者并赋值给主函数的变量 z，最后使用 printf 函数在显示器上输出 z 的值。

4. C 语言程序的基本语法规则总结

(1) 一个 C 语言源程序可以由一个或多个源文件组成。

(2) 每个源文件可由一个或多个函数组成。

(3) 每个函数的定义体部分由说明性或定义性语句以及运算或处理性语句组成。

(4) 一个源程序无论由多少个文件组成，都有且只有一个 main 函数，即主函数。

(5) 源程序中可以有预处理命令(include 命令仅为其中的一种)，预处理命令通常应放在源程序文件的最前面。

(6) 每一个说明性或定义性语句以及每一个运算或处理性语句(结构化程序设计语句除外)都必须以分号结尾。

注意：预处理命令不是 C 语言的执行指令，结尾不必须加分号。函数首部和花括号“{”之后不能加分号。

(7) 标识符与关键字之间必须至少加一个间隔符，一般为空格。如果已有语法规则中要求的明显间隔符，也可不再加空格符来间隔。

(8) 原则上一个说明性或定义性语句或一个运算或处理性语句占一行，也可以一行写多条这样的执行性语句。同一行的多条语句，语句执行顺序是从左向右依次执行。复合的结构化语句一般写成多行，必要时使用{ }限定复合内容。

1.3 C 语言的字符集及词汇

要点 1：语言是由文字传承的，因此语言是由文字组成的。掌握了对应的文字组成元素和用途，就掌握了语言。

要点 2：同样的文字有不同的用法，其代表不同含义。

1. 字符集

字符是组成语言最基本的元素。C 语言程序设计使用的是键盘上的英文半角，可输入字符集。只有在字符常量、字符串常量和注释中可以使用汉字或其他 ASCII 码表中的图形符号。具体字符集如下。

(1) 字母：a～z，共 26 个；A～Z，共 26 个。C 语言是区分大小写的，关键字必须使用小写字母，其他自定义名称中字母的大小写不同时，系统认为是不同名称。

(2) 数字：0～9，共 10 个。

(3) 空白符：空格符、制表符、换行符等统称为空白符。空白符只在字符常量和字符串常量中起作用。在其他地方出现时，只起间隔作用，称作标准间隔符。因此在程序中使用空白符个数多少，对程序的编译不发生影响，但在程序中适当的地方使用空白符将增加程序的清晰性和可读性。

(4) 键盘可输入的半角标点和特殊字符。

2. 词汇

在 C 语言中使用的词汇分为 6 类，即关键字、标识符、运算符、分隔符、常量和注释符。

1) 关键字

关键字是由 C 语言规定的具有特定意义的字符串，通常也称为保留字。标准 C 语言一共有 32 个关键字，分别是 auto、break、case、char、const、continue、default、do、double、else、enum、extern、float、for、goto、if、int、long、register、return、short、signed、static、sizof、struct、switch、typedef、union、unsigned、void、volatile 和 while。

后期不同版本的 C 语言对其关键字和语法进行了扩充。C 语言的关键字分为以下几类。

(1) 类型说明符。用于定义和说明变量、函数或其他数据结构的类型，如前面例题中用到的 int 和 double 等。

(2) 语句功能符。用于表示一个语句的功能部分，如例 1-3 中用到的 if 和 else 就是条件语句的两个语句功能符，即条件语句关键字。

(3) 预处理命令字。用于表示一个预处理命令，如前面各例中用到的 # include。

2) 标识符

在程序中使用的变量名、函数名、标号等自定义名称统称为标识符。C 语言规定，标识符只能是由 A～Z、a～z、0～9 和下画线“_”组成的字符序列，并且其第一个字符必须是字母或下画线，标识符不能与 C 语言关键字相同。

例如，以下标识符是合法的：

```
a,x,x3,BOOK_1,sum5,_abc,_asd_d,_12h
```

例如，以下标识符是非法的：

```
3s              (以数字开头)
s * T           (出现非法字符 * )
-3x             (以减号开头。如果是下画线“_”就是正确的)
bowy-1          (出现非法字符"-",如果是下画线“_”就是正确的)
```

在使用标识符时还必须注意以下几点。

(1) 标准 C 语言不限制标识符的长度，但对于各种不同版本的 C 语言编译系统，其限制长度是不同的。例如，在某版本 C 语言中规定标识符前 8 位有效，当两个标识符前 8 位相同时，则被认为是同一个标识符。

(2) 在标识符中，大小写是有区别的。例如，BOOK 和 book 是两个不同的标识符。

(3) 标识符虽然可由程序员随意定义，但标识符是用于标识某个量的符号。因此，命名应尽量有相应的意义，以便于阅读理解，做到“顾名思义”，而且尽量不重复。虽然有些标识符是可以重名的，但要掌握重名标识符的性质和使用规则才行。

3) 运算符

C 语言中含有相当丰富的运算符，如算术运算符＋、－、＊、/等。运算符与常量、变量和函数调用返回值一起组成表达式，表示各种运算功能。运算符由一个或多个字符组成，在后续章节中进行详细讲解。

4) 分隔符

在 C 语言中采用的分隔符有逗号、分号、冒号和空格等。逗号主要用在类型说明和函

数参数表中，用于分隔各个变量。空格多用于语句各组成部分的关键字和标识符之间，作间隔符，其他的在一些特定语句里使用。在关键字、标识符之间必须要有一个以上的空格符作间隔，否则将会出现语法错误。

例如，把“int a;”写成“inta;”，C 编译器会把 inta 当成一个标识符来处理，其结果必然出错。

5）常量

C 语言中使用的常量可分为数字常量、字符常量、字符串常量、符号常量和转义字符等多种，在后面章节中将专门给予介绍。

6）注释符

C 语言的注释符是//、/ * …… * /。在与//同一行之后的任何文字和/ * …… * /之间的任何文字符号都为注释。程序编译时，不对注释做任何处理。注释可出现在程序中的任何位置。注释用来向用户提示或解释程序的意义。在调试程序中对暂不使用的语句也可用注释符注释掉，使程序编译过程跳过该行或不对该段代码做编译。

1.4 习　　题

本章的习题内容请扫描二维码观看。

第 1 章课后习题

第 2 章　算法及算法描述

要点：程序＝数据结构＋算法。

关于程序的定义在程序设计语言领域有很多不同的说法，但大家都比较认可的是 Nikiklaus Wirth 提出的公式：程序＝数据结构＋算法。

数据结构(data structure)是数据组织形式，包括原始数据的存储、处理过程中的数据存储和结果数据的存储。原始数据为程序提供必要的数据来源；中间过程数据存储为程序处理完成程序功能提供保障；结果数据存储为程序功能实现后必要的人机交互和各个程序间的信息交互提供载体。

算法(algorithm)是对数据的处理过程或操作过程的描述。可以简单理解为对完成任务需求(即程序功能)的各个指令或语句或功能代码的排列顺序的描述。算法设计首先根据程序功能要求设计数据结构以用于存储数据，再根据数据结构设计相应的处理过程来完成任务，处理过程中同时考虑在必要的地方进行人机交互或程序间交互。

程序设计语言和软件编辑环境是完成程序设计和实现任务需求的工具，同一个功能程序可以使用不同的程序设计工具实现。每种工具各有优点，程序员或开发人员根据需要选择不同的工具使用。

C 语言由于其特性优秀、功能强大和能够更接近底层硬件进行功能程序设计，衍生出各种不同的编译软件和集成编辑环境。不同编译软件和集成编辑环境面向不同处理器系列的程序设计，不仅可以用于计算机领域的桌面系统程序设计，而且在嵌入式系统程序设计中应用得更为广泛。目前是物联网与人工智能快速发展的时代，嵌入式处理器相关程序设计需求得以爆发式发展，所以对于电子信息类学生和科技工作者来说，掌握 C 语言程序设计方法，特别是与嵌入式系统程序设计相关的重点内容尤为重要。

2.1　算法举例及描述

要点：算法是程序的灵魂，程序功能由算法决定，程序代码是功能实现的具体表现。同一个算法可以使用不同语言和不同代码实现。

1. 算法举例

【例 2-1】 求 1×2×3×4×5。

(1) 原始算法如下。

S1：先求 1×2，得到结果 2。

S2：将步骤 1 得到的乘积 2 再乘以 3，得到结果 6。

S3：将 6 乘以 4，得到结果 24。

S4：将 24 再乘以 5，得到结果 120。

这样的算法虽然正确，但太烦琐。

(2) 改进的算法如下。

S1：使 t=1。

S2：使 i=2。

S3：使 t×i，乘积仍然放在变量 t 中，可表示为 t×i→t。

S4：使 i 的值+1，即 i+1→i。

S5：如果 i≤5，返回 S3；否则输出结果，算法结束。

如果计算 100!，只需将 S5 的“如果 i≤5”改成“如果 i≤100”即可；如果求 1×3×5×7×9×11，只需对算法做以下很小的改动。

S1：1→t。

S2：3→i。

S3：t×i→t。

S4：i+2→i。

S5：如果 i≤11，返回 S3；否则结束。

该算法不仅正确，而且简洁高效，因为计算机是高速运算的自动机器，实现循环轻而易举，特别是对于那些迭代次数比较多的计算机处理任务尤为突出。

【例 2-2】 有 50 个学生，要求将他们之中成绩在 80 分以上者打印出来。

如果 n 表示学生学号，n_i 表示第 i 个学生的学号，g 表示学生成绩，g_i 表示第 i 个学生的成绩，则算法可表示如下。

S1：1→i。

S2：如果 $g_i \geqslant 80$，则打印 n_i 和 g_i；否则不打印。

S3：i+1→i。

S4：如果 i≤50，返回 S2；否则结束。

【例 2-3】 判定 2000—2500 年中的每一年是否是闰年，将结果输出。

(1) 闰年的条件如下。

如果年份是 4 的倍数，且不是 100 的倍数，为普通闰年；如果年份是 100 的倍数，且是 400 的倍数，为世纪闰年。归结起来就是：四年一闰；百年不闰，四百年再闰。具体描述如下。

① 能被 4 整除，但不能被 100 整除的年份；

② 既能被 100 整除，又能被 400 整除的年份。

(2) 设 y 为被检测的年份，则该算法的一种表示形式如下。

S1：2000→y。

S2：如果 y 不能被 4 整除，则输出 y“不是闰年”，然后转到 S5。

S3：如果 y 能被 4 整除，不能被 100 整除，则输出 y“是闰年”，然后转到 S4。

S4：如果 y 能被 4 整除，且能被 100 和 400 整除，则输出 y“是闰年”；否则输出 y“不是闰年”，然后转到 S5。

S5：y+1→y。

S6：当 y≤2500 时，返回 S2 继续执行；否则结束。