第 2 章　C 语言基础

2.1　C 程序的基本概念

C 语言有着严格的格式和语法规则，用户需要按照这些要求来编写程序。本章将讲解 C 程序的组成、语句、注释等基本概念。

2.1.1　C 程序的基本结构

C 程序由语句、函数、包含文件等部分组成。本节将以一个简单的 C 程序实例来讲解 C 程序的基本结构问题。这个程序实现两个整数的大小判断功能。

（1）单击“主菜单”|“系统工具”|“终端”命令，打开一个终端。在终端中输入“vim”命令，打开 VIM。

（2）输入程序。在 VIM 中按“i”键，进入到插入模式，然后在 VIM 中输入下面的代码。需要注意的是，/* */符号中的内容是注释，程序中不必输入这些注释。

```
#include <stdio.h>          /*包含文件。*/
int max(int i , int j)
{
                            /*注释：这是一个自定义函数，实现两个整数的大小比较功能。 */
if(i>=j)
  return(i);                /*返回较大的数。*/
else
  return(j);
}
int main()                  /*主函数。*/
{
int i =3;                   /*定义三个变量，并且赋值。*/
int j =5;
int t;
t=max(3,5);                 /*将较大的数赋值给 t。*/
printf("the big number is %d\n",t);     /*输出文本和结果。*/
}
```

（3）保存文件。输入这些代码以后，按“Esc”键返回到普通模式，输入命令“:w 2.1.c”，保存这个文件。然后输入命令“:q”退出 VIM。

（4）在终端中输入下面的命令，编译这个程序。

```
gcc -o 2.1.out 2.1.c
```

（5）在终端中输入下面的命令，对这个程序添加可执行权限。

```
chmod +x 2.1.out
```

（6）输入下面的命令，运行这个程序。

```
./2.1.out
```

（7）程序的运行结果如下所示，显示较大值是 5。

```
the big number is 5
```

由上面的例子可知，C 程序的源代码有下面的特点。

（1）程序一般用小写字母书写。

（2）大多数语句结尾必须要分号作为终止符，表示一个语句结束。同一个语句需要写在一行上。

（3）每个程序必须有一个主函数，主函数用 main()声明，并且只能有一个主函数。在 Linux 系统中，main 主函数应该是 int 类型。

（4）每个程序中的自定义函数和主函数，需要用一对大括号括起来。

（5）程序需要使用#include <>语句来包含系统文件，这些系统文件完成系统函数的定义。

（6）一个较完整的程序大致包括下面这些内容。

- 包含文件：一组#include <*.h>语句。
- 用户自定义函数：用户已编写的完成特定功能的模块。
- 主函数：程序自动执行的程序体。
- 变量定义：定义变量以存储程序中的数据。
- 数据运算：程序通过运算完成各种逻辑功能。这些运算由各种语句和函数实现的。
- 注释：注释写在/* */符号之间，不是程序的必需部分，但是可以增强程序的可读性。

需要注意的是，C 程序是严格区分大小写的，程序中同一个字母的和大写小写字母代表不同的内容。可以将多个语句写在一行上，但是每个语句必须有一个分号结束。例如本节代码中的 max 函数，也可以写成下面的形式。

```
int max(int i , int j){  if(i>=j)  return(i);  else  return(j);}
```

这段程序也是可以正常执行的，但是不便于程序的阅读与修改。在编程中需要使用正确的格式，每个语句写在一行上，并且使用正确的注释与缩进，使代码的层次清晰明了。

2.1.2　C 程序的一般格式

通过上一节的例子可知程序的主要部分是主函数和自定义函数。C 程序的一般结构如下所示。

```
程序的包含文件
子函数类型声明
全局变量定义
main 主函数()
{
   局部变量定义
   <程序体>
}
```

```
自定义函数 1()
{
   局部变量定义
   <程序体>
}
自定义函数 2()
{
   局部变量定义
   <程序体>
}
……
自定义函数 N()
{
   局部变量定义
   <程序体>
}
```

主函数与自定义函数的位置是随意的。自定义函数可以写在主函数的前面或后面，如果放在后面，需要在程序的前面声明这个函数。

2.1.3 C 程序中的注释

所谓注释，就是在编写程序时对代码的补充说明，不影响程序。任何一个程序员，都不可能完全记忆所写过的代码，注释的作用是对代码的阅读和理解进行提示。程序在编译时，会自动去除注释的内容。

在编写程序时，需要正确地书写注释，养成良好的注释习惯。例如下面代码中使用的注释方法就是常用的注释方法。其中，程序右侧的注释可以按“Tab”键使注释对齐。

```
/*  作者：小明。
    内容：变量的输入与输出。
    时间：2007.11.18
    程序的最前面注释出代码的信息。*/
#include <stdio.h>                          /*包含文件。*/
int main()                                  /*主函数。*/
{
int i ;                                     /*定义一个变量。*/
scanf("%d",&i);                             /*输入一个变量。*/
printf("%d",i)                              /*输出变量。*/
}
```

2.2 数据类型

数据类型指的是一类数据的集合，是对数据的抽象描述。数据类型的不同决定了所占存储空间的大小不同。每个变量在使用之前必须定义其数据类型。C 程序有整型（int）、浮点型（float）、字符型（char）、指针型（*）、无值型（void）这些常用数据类型。还有结构体（struct）和联合体（union）两种自定义数据类型。本章将讲解前三种基本的数据类型。

2.2.1　整型（int）

整型可以简单理解为整数。与整数不同的是，一个整型变量有一定的字节长度和存储数据范围。整型变量是有正负的，在定义整型变量时，需要注意变量的正负问题。如表 2.1 所示是不同长度与正负的整型变量。

表 2.1　整型变量

类　型	说　明
signed short int	有符号短整型数。简写为 short 或 int，字长为 2 字节，数的范围是-32768~32767
signed long int	有符号长整型数。简写为 long，字长为 4 字节，数的范围是-2147483648~2147483647
unsigned short int	无符号短整型数。简写为 unsigned int，字长为 2 字节，数的范围是 0~65535
unsigned long int	无符号长整型数。简写为 unsigned long，字长为 4 字节，数的范围是 0~4294967295

例如下面的代码中定义和使用了整型变量。

```
#include <stdio.h>
int main()
{
int i ;                          /*定义一个 signed short int 变量 i。*/
long j;                          /*定义一个 signed long int 变量 i。*/
unsigned short k;                /*定义一个 unsigned short int 变量 i。*/
i=123;                           /*i 赋值为 123。*/
j=1234L;                         /*j 赋值为 1234。*/
k=0x5e;                          /*k 赋值为十六进制的 5e。*/
printf("%d\n",i);
printf("%ld\n",j);
printf("%d\n",k);
}
```

用下面的命令编译这段代码。

```
gcc 2.2.c
```

然后对编译的程序添加可执行权限。

```
chmod +x a.out
```

输入下面的命令运行这个程序。

```
./a.out
```

程序的运行结果如下所示。

```
123
1234
94
```

在程序中整型变量有以下三种表示方法。

- 普通十进制数，在程序中除了 0 以外不以 0 开始。如 0，123，-123 等。
- 八进制数，程序中以 0 开始，如 013，-013 等。
- 十六进制数，程序中以 0x 或 0X 开始，如 0x1E、0XEF 等。

另外，可在整型常数后添加一个“L”或“l”字母表示该数为长整型数，如代码中的 j=1234L，表示 j 为长整型数。

2.2.2 浮点型（float）

浮点型数指的是数值的小数点在存储空间中可以移动，可以用来表示不同精度与大小的数值。小数、科学计数法的值都是用浮点型变量来存储的。根据所占的存储空间与表示范围不同，浮点型数可以分为下面的两种。

- float 单精度浮点数，字长为 4 个字节共 32 位二进制数。可表示数的范围是 3.4x10-38E~3.4x10+38E。
- double 双精度浮点数，字长为 8 个字节共 64 位二进制数，可表示数的范围是 1.7x10-308E~1.7x10+308E。

```
#include <stdio.h>
int main()
{
float m;                              //定义一个 float 变量。
double n ;                            //定义一个 double 变量。
m=1.234;                              //对 m 赋值。
n=.1234;
printf("%f\n",m);                     //输出变量。
printf("%f\n",n);
}
```

输入下面的命令，编译这个程序。

```
gcc 2.3.c
```

输入下面的命令，对编译的程序添加可执行权限。

```
chmod +x a.out
```

输入下面的命令，运行这个程序。

```
./a.out
```

程序的运行结果如下所示。

```
1.234000
0.123400
```

与整数变量相比，浮点型变量在下面这些方面是不同的。

- 浮点常数只有十进制一种进制。
- 绝对值小于 1 的浮点数，其小数点前面的零可以省略。如 0.123 可写为.123。
- 用默认格式输出浮点数时，都是保留 6 位小数，没有小数的后面加零。

2.2.3 字符型(char)

字符型变量在计算机中以 ASCII 码方式表示，长度为一个字节。各种字母、符号都可以是一个字符型的变量。数字 0~9 也可以存储为一个字符型变量。当数字存储为字符型变量时，

保存的是这个数字的 ASCII 码的值，与 int 型保存数值是不相同的。

> **注意**：ASCII 码是计算机内部的字符编码集。所有的信息都是以 ASCII 码的形式保存在计算机中的。一个字符变量对应 ASCII 码表中的一个字符。

在程序中，字符可直接用单引号引起来。如‘A’、‘b’、‘9’、‘@’都是一个字符型变量，也可以用这一字符的 ASCII 码值来表示一个字符，例如 87 表示大写字母‘W’。

除了屏幕上可以直接显示的字符以外，还有一些字符是用来进行格式控制，并不能直接显示在屏幕上。例如\n 表示一个换行符，ASCII 码为八进制 010，而不是一个字符\加一个字符 n。C 程序中常用的转义字符如表 2.2 所示。

表 2.2　转义字符

字　　符	名　　称	ASCII 码
\a	响铃（BEL）	007
\b	退格（BS）	008
\f	换页（FF）	012
\n	换行（LF）	010
\r	回车（CR）	013
\t	水平制表（HT）	009
\v	垂直制表（VT）	011
\\	反斜杠	092
\?	问号字符	063
\'	单引号字符	039
\"	双引号字符	034
\0	空字符（NULL）	000

在程序中字符变量需要用 char 来定义。例如下面的代码，是字符变量的定义与使用实例。

```
#include <stdio.h>
int main()
{
char m;
char n;
char t;
m='a';
n='!';
t='\n';
printf("%c\n",m);                    /*输出 a 再输出一个换行。*/
printf("%c\n",n);                    /*输出!再输出一个换行。*/
printf("%c\n",t);                    /*输出的 t 为一个换行.*/
}
```

输入下面的命令，编译这个程序。

```
gcc 2.4.c
```

输入下面的命令，对编译的程序添加可执行权限。

```
chmod +x a.out
```

输入下面的命令，运行这个程序。结果如下所示。

```
a
!
```

2.2.4 变量名

变量名只能由字母、数字、下画线组成，且头一个字符只能是字母或下画线，变量名之间不能有空格或其他字符。这样的变量才是合法的。例如下面这些变量名是合法变量名。

```
ab   _a   a2  AAA   A_B   __AB
```

下面这些变量名，不符合变量名的要求，是不合法的。

```
3a                  /*不能以数字开头。*/
a.b                 /*变量名之间不能有特殊符号。*/
$ab                 /*不能以特殊符号开头。*/
a b                 /*变量名中不能有空格。*/
```

不合法的变量名会引起程序的错误。除了这些变量名以外，系统关键字也不能作为变量名。例如下面这些词是系统关键字，作为变量名时会引发错误。

```
int   printf   float   exit
```

2.2.5 字符 NULL

NULL 是一个特殊的字符，在 C 程序中有着重要的功能。NULL 在 ASCII 码表中是 0，在程序中表示空字符，或者没有这个数值。在定义变量时，如果要对变量赋值为空值，可将这个变量赋值为 NULL，如下面代码所示。

```
int a=NULL;
char b=NULL;
float c=NULL;
```

> **注意**：ASCII 码是计算机字符的内部编码，每一个字符对应 ASCII 码表中的一个编号。详细介绍见本书 8.5.3 节中的内容。

2.3 变量的赋值与输出

变量的赋值指的是对一个变量指定一个值，这个值可以用于程序的运算。变量的输出指的是将变量的值显示在屏幕上，一般情况下，程序执行完毕以后需要将结果输出。

2.3.1 变量的赋值

赋值符号是等号，含义是将等号右边的值或表达式结果传递给等号左边的变量。例如下面的代码就是定义变量和赋值的例子。

```
#include <stdio.h>
int main()
```

```
{
int i =0,j=1,k=123;                    /*定义三个整型变量，定义变量同时赋值。*/
int s;
char a,b,c;                            /*定义三个字符型变量。*/
s=j;                                   /*将变量 j 的值赋值给 s。*/
a='a';                                 /*将字符 a 赋值给变量 a。*/
b=a;                                   /*把变量 a 的值赋值给 b。*/
c=b;                                   /*把变量 b 的值赋值给 c。*/
}
```

2.3.2　printf 函数输出变量

在前面的代码中，都是使用 printf()函数进行变量的输出。printf()函数称为格式输出函数，功能是按用户指定的格式，把指定的数据显示到屏幕上。

Printf()函数是一个标准库函数，它的函数原型在头文件“stdio.h”中，该函数的使用方法如下所示。

```
printf(“格式控制字符串”,输出变量列表)
```

输出变量列表中给出了各个输出变量。格式控制字符串用于指定输出格式，由格式字符串和非格式字符串两种组成。

- 格式字符串是以%开头的字符串，在%后面跟有各种格式字符。不同格式字符串的作用是说明输出数据的类型、形式、长度、小数位数等。如“%d”表示按十进制整型输出，“%ld”表示按十进制长整型输出。这些格式字符串与输出变量列表中的变量一一对应。
- 格式字符串以外的内容是非格式字符串，在输出时按照原有的字符直接输出。

格式字符串的内容和意义如下所示。

- %c：输出单个字符，参数为该字符的 ASCII 码。
- %d：以十进制形式输出带符号整数（正数不输出符号）。
- %e 或%E：以指数形式输出单、双精度实数，默认保留 6 位小数。
- %f：以小数形式输出单或双精度实数，默认保留 6 位小数。
- %g 或%G：以%f 或%e 中较短的输出宽度输出单、双精度实数。如果指数小于-4 或大于等于默认精度，则使用%e 或%E 格式输出。否则用%f 格式输出，省略末尾多余的 0。
- %i：以十进制形式输出带符号整数，同%d。
- %o：以八进制形式输出无符号整数（不输出前缀 0）。
- %s：输出字符串，参数为 char 指针，显示字符串中所有的字符。
- %u：以十进制形式输出无符号整数。
- %x 或%X：以十六进制形式输出无符号整数，%x 表示输出小写，%X 表示输出大写。

例如下面的代码是 printf()函数格式化输出的实例。

```
#include <stdio.h>
int main()
{
  int i=117;                                   /*定义整型变量 i=117。*/
  char a='m';                                  /*定义字符定量 a。*/
```

```
    float m=123.1234567 ;                    /*定义一个浮点型变量。*/
    printf("%d\n",i);                        /*以整型输出 i。*/
    printf("%c\n",i);                        /*以字符型输出 i。*/
    printf("%o\n",i);                        /*以八进制输出 i。*/
    printf("%x\n",i);                        /*以十六进制小写输出 i。*/
    printf("%X\n",i);                        /*以十六进制大写输出 i。*/
    printf("%c\n",a);                        /*以字符型输出 a。*/
    printf("%d\n",a);                        /*以整型输出 a。*/
    printf("%o\n",a);                        /*以八进制输出 a。*/
    printf("%x\n",a);                        /*以十六进制小写输出 a。*/
    printf("%X\n",a);                        /*以十六进制大写输出 a。*/
    printf("%f\n",m);                        /*以浮点型输出 f。*/
    printf("%e\n",m);                        /*以科学计数法输出 m。*/
}
```

输入下面的命令，编译这一个文件。

```
gcc 2.5.c
```

输入下面的命令，对编译的文件添加可执行权限。

```
chmod +x a.out
```

输入下面的命令运行这一个程序。

```
./a.out
```

程序的运行结果如下所示。

```
117
u
165
75
75
m
109
155
6d
6D
123.123459
1.23123e+02
```

从程序的运行结果可知，字符是以 ASCII 码的形式保存的，相对应一相整型的。当整型数 117 以整型输出时是 117，而以字符型输出时是 u。

2.3.3 scanf 函数从键盘读入变量

scanf()函数是键盘格式化输入函数，可以从键盘输入读取信息，这个函数的使用方法如下所示。

```
scanf("<格式化字符串>", <地址表>);
```

这里的格式化字符串与上一节中的格式化字符串含义是相同的。当有多个要读取的变量

时，可以用空格或其他符号隔开，在键盘输入时也相应地输入这些符号。例如下面的代码就是使用 scanf()函数从键盘读取输入，再输出变量的例子。

```
#include <stdio.h>
int main()
{
   int i ,j ;                                    /*定义程序中的变量。*/
   int m;
   printf("please enter a char:\n");              /*提示信息。*/
   scanf("%c",&m);                               /*输入一个字母。*/
   printf("the char is %c.\n",m);                 /*输出一个字母。*/
   printf("please enter a number:\n");
   scanf("%d",&i);                               /*输入一个整数。*/
   printf("please enter another number:\n");
   scanf("%d",&j);
   printf("%d + %d = %d \n",i ,j ,i+j);          /*输出两个整数和这两个整数的和。*/
}
```

输入下面的命令，编译这一个文件。

```
gcc 2.9.c
```

输入下面的命令，对编译的文件添加可执行权限。

```
chmod +x a.out
```

输入下面的命令运行这一个程序。

```
./a.out
```

运行程序时，光标等待输入内容。输入内容以后，按“Enter”键继续向后执行。程序的运行结果如下所示。

```
please enter a char:
```

这时从键盘输入一个字母 a，然后按“Enter”键。这时程序的结果如下所示。

```
the char is a.
please enter a number:
```

这时输入一个数字 5，然后按“Enter”键，程序显示结果如下所示。

```
please enter another number:
```

这时输入一个数字 3，然后按“Enter”键，程序显示结果如下所示。

```
5 + 3 = 8
```

2.4　运算符

运算符指的是用来完成各种运算的符号。C 程序中有着丰富的运算符，很多运算都是通过运算符来实现的。本节将讲解常用运算符的操作。

2.4.1 算术运算符

算术运算符是指进行简单数学计算的运算符，是程序中最简单最常用的运算符。算术运算符的种类和使用方法如表 2.3 所示。

表 2.3 算术运算符

运 算 符	含 义
+ -	加号和减号
* /	乘号和除号
++	自加，在原来的值上面加 1
--	自减，在原来的值上面减 1
+= -=	相加赋值和相减赋值，对原有值加或者减去一个值
*= /=	相乘赋值或相除赋值，对原有值乘或除一个值
%	求余运算，第一个数除以第二个数求余数

例如下面的代码是使用这些运算符进行简单的数学运算的例子。

```
#include <stdio.h>
int main()
{
   int i =3 ,j = 7 ,k;                    /*定义变量和赋值。*/
   float m=2.5 , n = 3.8,t;
   k= 3+7;                                /*做加法运算。*/
   printf("%d\n",k);
   k*=5;                                  /* *=运算，表示把 k 乘以 5 再赋值给 k。*/
   printf("%d\n",k);
   k=k%j;                                 /*求余运算。*/
   printf("%d\n",k);
   printf("%d\n",i+j);                    /*可以直接在参数列表中进行运算。*/
   t= 3.8/2.5;
   printf("%f\n",t);                      /*以浮点数的形式输出。*/
   printf("%f\n",m*n+5);                  /*直接在参数列表中进行运算。*/
}
```

输入下面的命令编译这个程序。

```
gcc 2.2.c
```

输入下面的命令，对编译后的程序添加可执行权限。

```
chmod +x a.out
```

输入下面的命令，运行这个程序。

```
./a.out
```

程序的运行结果如下所示。

```
10
50
1
10
```

```
1.520000
14.500000
```

对于++、--运算的使用，将符号写在前面和写在后面的含义是不同的。++写在后面时，表示取得值然后再自加，写在前面时，表示先自加再取值。例如下面的代码，就是自加和自减运算符使用的例子。

```
#include <stdio.h>
int main()
{
   int i =3 ,j = 7;
   i++;                                    /*i 的值为 4。*/
   printf("%d\n",i);
   printf("%d\n",i++);                     /*输出 i 的值为 4，然后自加得 5。*/
   printf("%d\n",i++);                     /*输出 i 的值为 5，然后自加得 6。*/
   printf("%d\n",++i);                     /*自加得 7，输出结果得 7。*/
   printf("%d\n",--i);                     /*自减得 6，输出结果为 6。*/
   printf("%d\n",i++ + j++);               /*先取值相加得 13，然后分别自加。*/
}
```

输入下面的命令编译这个程序。

```
gcc 2.7.c
```

输入下面的命令，对编译后的程序添加可执行权限。

```
chmod +x a.out
```

输入下面的命令，运行这个程序。

```
./a.out
```

程序的运行结果如下所示。

```
4
4
5
7
6
13
```

2.4.2　关系运算符

所谓关系运算符，指的是比较两个数的大小关系或相等关系的符号。关系运算符的种类与使用方法如表 2.4 所示。

表 2.4　关系运算符

运 算 符	含　义
>	大于。第一个数大于第二个数时返回真
>=	大于等于。第一个数大于等于第二个数时返回真
<	小于。第一个数小于第二个数时返回真
<=	小于等于。第一个数小于等于第二个数时返回真

==	等于。两个数相等时返回真
!=	不等于。两个数不相等时返回真

关系运算符的返回值是真（true）和假（false）的概念。在 C 语言中，真值 true 可以是不为 0 的任何值。而假值 false 则为 0。使用关系运算符，若表达式为真则返回 1，否则表达式为假则返回值为 0。例如下面的关系运算符的使用。

```
6>5                    返回 1
10=9                   返回 0
```

2.4.3 逻辑运算符

所谓逻辑运算符，指的是用形式逻辑原则来建立数值间关系运算的符号。逻辑运算有与、或、非三种，使用方法如表 2.5 所示。

表 2.5 逻辑运算符

运 算 符	含 义	返 回 值
&&	逻辑与	所有的条件为真则返回真
||	逻辑或	只要有一个条件为真则返回真
!	逻辑非	真值取非返回假，假值取反返回真

例如下面的代码，是关系运算符与逻辑运算符的使用实例。

```
#include <stdio.h>
int main()
{
   int i=5,j=3;
   printf("%d\n",i>j);                    /*判断 i 是否大于等于 j，返回真，输出 1。*/
   printf("%d\n",i==j);                   /*判断 i 是否等于 j，返回假，输出 0。*/
   printf("%d\n",i<j);
   printf("%d\n",i!=j);
   printf("%d\n",i!=j);
   printf("%d\n",(i>j)&&(i>=j));          /*判断两个条件，然后做逻辑与运算。*/
   printf("%d\n",(i>j)&&(i<j));
   printf("%d\n",1&&1&&1);                /*所有条件为真则返回真。*/
   printf("%d\n",1&&0&&1);                /*有一个条件为假则返回假。*/
   printf("%d\n",0||0||1);                /*有一个条件为真则返回真。*/
   printf("%d\n",!0);                     /*非假返回真。*/
}
```

输入下面的命令编译这个程序。

```
gcc 2.8.c
```

输入下面的命令，对编译后的程序添加可执行权限。

```
chmod +x a.out
```

输入下面的命令，运行这个程序。

```
./a.out
```

程序的运行结果如下所示。

```
1
0
0
1
1
1
0
1
0
1
1
```

2.5　小结

本章讲解了数据类型、变量赋值与输出、常用运算符等知识。这些知识是编写 C 程序的基础，通过这些知识的学习，可以理解 C 程序的一些概念，编写简单的 C 语言程序。在本章的学习中，数据类型与运算符的使用是重点，需要详细地理解数据类型的概念和输出方式。