第二章 C 语言程序设计基础

【学习目标】

掌握 C 语言的数据类型，常量和变量的使用。
掌握常用的运算符和表达式的使用。
掌握库函数 scanf()、printf()、putchar()、getchar()的使用。
能够编写程序解决一些简单的数学计算问题。

【学习内容】

C 语言的常量和变量以及数据类型。
C 语言的各类运算符和表达式。
几个基本输入/输出函数。

【学习重点和难点】

重点掌握 C 语言的基本数据类型的定义和使用方法。
转义字符的理解和使用。
重点掌握算术、逻辑、自增和自减运算符及其运算。
scanf()函数和 printf()函数中的格式控制，各种不同数据类型的混合输入和输出。

【问题的提出】

数据是程序加工处理的对象，也是加工的结果，例如对微分方程求解、资料检索、事务管理等，所以数据是程序设计中所要涉及和描述的主要内容。那么，在解决实际的问题中，程序如何描述相关的数据（包括数据结构、数据表示范围、数据在内存中的存储分配等）？这就是数据类型的定义。数据类型指把待处理的数据对象划分成一些集合，属于同一集合的各数据对象都具有同样的性质，例如对它们能够做同样的操作。

程序语言中把数据划分的不同类型与计算机硬件有密切关系，计算机硬件把被处理的数据分成一些类型（如有定点数、浮点数等），CPU 对不同的数据类型提供了不同的操作指令。实际上，数据类型是计算机领域中一个非常重要的概念之一。在学习程序设计的过程中，将要不断地与数据类型打交道。

数据类型除了决定数据的存储形式及取值范围外，同时决定了数据能够进行的运算。例如，在 C 程序里可以写出下面的一个表示了某种计算过程的“表达式”，其中包含了一些数据，如整数和实数等：x-(3.24*5+sin(2,3))/4*6.24

问题是：写程序时如何写出所需表达式， C 语言对各种数据的表达有什么规定，在表达式里可以写什么？它们表示什么意思？写出的表达式表示了什么计算过程？有关计算的结果是什么？

实际上，回答这些问题也就是学习并掌握 c 语言中数据的基本数据类型，各种运算符、表达式以及运算时的相关规定。

【教学要点】

1. 常量和变量

在 C 语言中，任何数据的形式有两种：常量或变量。常量就是程序运行过程中其值不能改变的量，而无论常量还是变量，都必须属于各种不同的数据类型。

【例 4.1】：定义一个变量，赋值并输出该数。

源程序如下：

```
main( )
{ int x;        /*   定义 x 为一个整型变量  */
  clrscr( );    /*   清屏    */
  x=5;          /*   将整型常量 5  赋值给变量 x ，“=”表示赋值运算    */
  printf(" x=%d ",x);     /*  利用输出库函数 printf()输出结果并显示到屏幕上    */
}
```

说明：

1. 常量也有多种类型，如整数 12，实数 134.55，字符’a’等。它常用在变量初始化及各种语句中。
2. 一个变量可以看作一个容器，程序运行中可以将有关的数据存入其中,使容器中的值发生变化。这个容器实际上是编译系统根据变量的类型，自动为定义的每一个变量在内存中分配相应的存储单元。变量具有保持值的性质，也就是说：如果在某时刻给某变量赋了一个值，此后用该变量的值时，每次得到的总是那个值。这种情况一直延续到下次再给这个变量赋值为止。
3. 程序里的每个变量都有一个名字，变量名必须以字母、下画线开始，见名知意。不能用关键字做变量名，一般采用小写字母。
4. 一个变量包含四方面属性：
   变量的名字，它提供了在程序里访问变量的基本途径；
   变量的类型，它规定了变量的可能使用方式，可能存储的值，可能使用的各种操作；
   变量的存储位置，这是变量在计算机里的具体实现；
   存储在变量中的数据值。

【例 4.2】：输出一个圆的面积。

源程序如下：

```
#define PI 3.14159     /*  定义符号常量 PI，表示实型常量 3.14159    */
main( )
{ int r=3;              /*    定义 r 为一个整型变量  */
   float area;          /*    定义 area 为一个实型变量    */
   area=PI*r*r;         /*  表达式中的“*”表示乘法    */
  printf("area=%f ",area);     /*  利用输出库函数 printf()输出结果并显示到屏幕上    */
}
```

运行结果为：

```
  area=28.274309
```

说明：

1. 在程序中，常用一个标识符表示一个常量，并称为符号常量。如本例中用符号 PI 代表 3.14159 ，其目的在于简化程序的书写和便于程序的修改。
2. 注意本例中有实数与整数的混合运算。如果将变量 area 定义成整型，同时将 printf("area=%f ",area); 改为 printf("area=%d ",area); 其结果会为：area=28 。因此，在定义变量时，要根据实际问题中数据的要求来决定为何种类型。

2. 数据类型和变量的定义、使用

【例 4.3】：定义三个变量，求和值并输出结果。

源程序如下：

```
main( )
{ int x;                /*   定义 x 为一个整型变量  */
  float y,z,sum;     /*   定义三个实型变量 y,z,sum*/
   clrscr( );          /*   清屏    */
   x=2;                /*  变量的赋值    */
   y=5.5;
   z=10.0;
   sum=x+y+z;          /*   求三数之和并赋值给变量 sum      */
 printf(" sum=%f ",sum);     /*  利用输出库函数 printf()输出结果并显示到屏幕上    */
}
```

运行结果为：

```
  sum=17.500000
```

说明：

1. C 语言为每个类型定义了一个标识符，通常把它们称为类型名。例如整数型用关键字 int 标识，实型用 float 标识，字符型用 char 标识。一个类型名由一个或几个关键字组成，它与前面讲的"名字"不尽相同。类型名仅用于说明数据属于哪一种类型。
2. C 语言要求程序里所用的每个变量都必须先定义，然后才能使用。若把本例中的语句 int x; 放到求和语句 sum=x+y+z; 之后，编译时系统就会报错。因此，在初学程序设计时，要注意变量的相关规定和语句的先后顺序。
3. 允许在一个类型说明符后，说明多个相同类型的变量。各变量名之间用逗号间隔。类型说明符与变量名之间至少用一个空格间隔。如 float y,z,sum;
4. 每种类型都有相应的取值范围。如果把一个超过数据类型限制范围的数（太大或太小）存放到变量中，将会产生数据的溢出（上溢或下溢）。如将本例中 x=2; 改为 x=35000; 就会溢出，而编译系统一般是不会检查这种溢出的，从而造成计算结果的错误。因此，在定义变量时，要根据实际问题中数据的大小来决定为何种类型。
5. 本例中需注意不同数据类型的混合运算。C 语言规定，操作数的类型不一致时，则必须转换为同一类型的数据才能运算，这种转换由系统完成。

【例 4.4】：实型数据的舍入误差。

源程序如下：

```
main( )
{
   float a,b;
   a=333456.789e5;
   b=a+20;
   printf("a=%f,b=%f\n",a,b);
   printf("a=%e,b=%e\n",a,b);
}
```

运行结果为：

```
a=33345678848.000000,b=33345678848.000000
a=3.33457e+10,b=3.33457e+10
```

说明：

1. 实型变量是用有限的存储单元存储的，因此提供的有效数字是有限的，在有效位以外的数字将被舍去，由此可能会产生一些误差。本例中实型变量 a、b 只能保证 7 位

有效数字，后面的数字无意义）

2. 由于实数存在舍入误差，使用时要注意：不要试图用一个实数精确表示一个大整数，因为浮点数是不精确的；实数一般不判断“相等”，而是判断接近或近似；避免直接将一个很大的实数与一个很小的实数相加、相减，否则会“丢失”小的数；根据实际问题的要求选择单精度或双精度类型。

【例 4.5】：定义一个字符变量，运算后输出其值。

源程序如下：

```
main( )
{ int x=32;
   char c1='a';     /*  定义字符变量 c1,并同时赋初值为字符常量'a'    */
   c1=c1-x;
   printf(" c1=%c\n ",c1);      /*  利用输出 printf(),格式控制符%c 输出结果    */
   printf(" c1=%d\n ",c1);      /*  利用输出 printf(),格式控制符%d 输出结果  */
}
```

运行结果为：

```
c1=A
c1=65
```

说明：

1. 字符常量是用单引号括起来的一个字符，例如'a','b'。一个字符变量只能存放一个字符常量，即单个字符。字符变量的类型说明符是 char。
2. 本例中出现的\n 是一个转义字符常量，表示回车换行，常常用在输出格式中。转义字符是一种特殊的字符常量。它以反斜线"\"开头，后跟一个或几个字符。转义字符具有特定的含义，不同于字符原有的意义，故称“转义”字符。转义字符主要用来表示那些用一般字符不便于表示的控制代码。
3. 字符类型数据可以和整型数据进行混合运算（如加减、相互赋值、输出格式混用等）。一个字符数据减去一个整数，表示该字符所对应的 ASCII 值减去这个整数。如本例中的 c1=c1-x; 利用此特点很方便的实现了英文字母的大小写的转换。
4. 一个字符数据既可以以字符形式输出（ASCII 码对应的字符），也可以以整数形式输出（直接输出 ASCII 码）。格式控制符%c 的含义是：以字符形式输出；格式控制符%d 的含义是： 以十进制整数形式输出。

3. 运算符和表达式的使用

描述数据运算的符号称为运算符。在表达式中运算符能够连接的运算对象的个数称为运算符的目。因此根据运算对象的多少，可以把 C 语言中的运算符分为单目运算符、双目运算符和三目运算符。

在 C 语言程序里，描述计算过程的最基本结构是表达式，表达式由被计算的对象和表示运算的特殊符号按照一定的规则构造而成。表达式的计算过程又称表达式求值。表达式的意义就是它所求出的值。

【例 4.6】：分析下列程序中运算的结果。

源程序如下：

```
main( )
{    int a=7,b=26,x,y,z;
      x= a++;    /* 后缀形式。先取 a 的值赋予 x 后，a 再进行++运算 */
      y= --b;    /* 前缀形式。先对 b 进行--运算后，再取其值赋予 y */
```

```
    printf("a=%d, x=%d\n", a, x);        /*  输出：a=8, x=7    */
    printf("b=%d, y=%d\n", b, y);        /*  输出：b=25, y= 25  */
    if(a%2==0)          /*  判断 a 是否为偶数，通过求余运算%，能被 2 整除是偶数  */
      printf("a=%d 为偶数 \n", a);    /* 输出：a=8 为偶数 */
    z=b/a;              /*  除法运算 25/8   */
    printf("z=%d \n", z);        /*    输出：z= 3   */
    x=0&&(a=b=50)             /* 逻辑运算   */
    printf("a=%d, b=%d, x=%d\n", a,b,x);    /*  输出：a=8, b=25, x=0    */
    a+=a*=a=3;        /*   重新对 a 变量赋值   */
    printf("a=%d\n", a);          /*  输出：a=18    */
}
```

说明：

1. 自增和自减运算符使用时容易出错，是一个学习难点。它们是单目运算符使用起来非常灵活,有前缀和后缀两种形式。如表达式 a++，是“先用 a 后加 1”；而表达式++a，是“先增加 1 后用 a”。
2. 对一个表达式求值，应该依据 C 语言规定的各类运算符的优先级和结合性进行计算。但在语句 x= a++; 的执行时，要注意虽然“++”的优先级高于“=”，仍应根据“++”的使用原则来计算：先取 a 的值赋予 x 后，a 再进行++运算。
3. 除法运算的两个操作数均为整数，其结果也是整数。如 4/7，结果为 0。因此，在实际问题中要注意是否需要得到实数的结果，若是，则至少有一个操作数应为实数。
4. 对于语句 x=0&&(a=b=50);的运算，等价于 x=(0&&(a=b=50)); 逻辑运算符&&的优先级高，结合方向从左向右，由于其左操作数为 0（表示假），整个逻辑表达式 0&&(a=b=50)的值即为假，无须再计算其右操作数，因此 a=b=50 是没有被执行的，a,b 的值保持原有的值不变。再执行赋值运算，x 的值为 0。
5. 语句 a+=a*=a=3;中涉及复合赋值运算，结合方向右从向左，计算过程是：首先执行 a=3，再执行 a*=a，等价于 a=a*a 则此时 a=3*3=9，最后执行 a+=a，等价于 a=a+a, 则此时 a=9+9=18。

4. 输入和输出函数的使用

【例 4.7】：利用不同的格式控制，使用输出函数 printf( )输出三变量的值。

源程序如下：

```
main( )
{int a=5;
 float b=6.0;
 char c='m';
 printf("%d,%f,%c\n",a,b,c);              /*  输出：5,6.000000,m    */
 printf("a=%d,b=%f,c=%c\n",a,b,c);   / *  输出：a=5,b=6.000000,c=m    */
 printf("%3d%6.2f%4c\n",a,b,c);          /*  输出：    5   6.00     m    *
 printf("a=%d\nb=%f\nc=%c\n",a,b,c); /*  输出：a=5
                                                  b=6.000000
                                                  c=m    */
  }
```

说明：

1. printf()函数用来向标准输出设备(屏幕)写数据; 可以用各种不同的格式写数据。从本例中可以看出，对三个变量的值的输出由于格式的不同，结果的输出形式就不相同，

在实际的问题中要根据问题的要求来编写适合的输出语句。

2. printf()函数的调用格式为: printf("<格式控制字符串>",<参数表>)，其中格式字符串包括两部分内容：一部分是普通字符,这些字符将按原样输出;另一部分是格式控制规定字符,以"%"开始,后跟一个或几个规定字符,用来确定输出内容格式。参数表是需要输出的一系列参数,其个数必须与格式字符串所说明的输出参数个数一样多,各参数之间用","分开,且顺序一一对应,否则将会出现意想不到的错误。
3. 格式控制字符在使用时，可以在"%"和格式符号之间插进数字表示最大输出宽度。如:%3d 表示输出 3 位整型数,不够 3 位右对齐。%6.2f 表示输出宽度为 6 的浮点数,其中小数位为 2,整数位为 3,小数点占一位,不够 6 位右对齐。
4. 如果字符串的长度、或整型数位数超过说明的输出宽度,将按其实际长度输出。但对浮点数,若整数部分位数超过了说明的整数位宽度,将按实际整数位输出;若小数部分位数超过了说明的小数位宽度,则按说明的宽度以四舍五入输出。
5. 若想在输出值前加一些 0,就应在宽度前加个 0。例如:%04d 表示在输出一个小于 4 位的数值时,将在前面补 0 使其总宽度为 4 位。
6. 可以控制输出左对齐或右对齐,即在"%"和字母之间加入一个"-"号可说明输出为左对齐,否则为右对齐。例如: %-7d 表示输出 7 位整数左对齐。

【例 4.8】：利用不同的格式控制，使用输入函数 scanf ( )输入三变量的值。

源程序如下：

```
main( )
{
   int a;
   float b;
   char c;
   printf("input a b c: ");   /* printf()在屏幕上将按原样输出 input a,b,c: ,此用法常常起到提示功能*/
   scanf("%d%f%c",&a,&b,&c);    /*   输入三个数   */
   printf("1.a=%d,b=%f,c=%c\n",a,b,c);
   printf("input a,b,c agin:");
   scanf("%d,%f,%c",&a,&b,&c);   /*   输入三个数，用逗号分隔   */
   printf("2.a=%d,b=%f,c=%c\n",a,b,c);

}
```

运行：

```
input a b c: 6 7.5m   /* 输入时 6 与 7.5 之间用空格分隔，m 紧跟在后，不用空格分隔   */
1.a=6,b=7.500000,c=m
input a,b,c agin:12,8.6,g        /*   输入的三个数，都用逗号分隔   */
2.a=12,b=8.600000,c=g
```

说明：

1. scanf 函数是一个标准库函数，它的函数原型与 printf 函数相同都在头文件“stdio.h”中。scanf 函数的一般形式为：scanf(“格式控制字符串”，地址表列) 。其中，格式控制字符串的作用与 printf 函数相同，但不能显示非格式字符串，也就是不能显示提示字符串。地址表列中给出各变量的地址。地址是由地址运算符“&”后跟变量名组成的。
2. 使用 scanf 函数在输入时，如在格式串中出现有普通字符，则必须照原样、在原位置输入；如果格式串与格式串间无普通字符，则数值与数值间用空格或回车键分隔，

数值与字符间不用分隔符，字符与字符间中不用分隔符。

3. 在 scanf 函数的输入控制字符串中，若采用下列方式

scanf("a=%d,b=%f,c=%c",&a,&b,&c);

scanf("%3d%4f%2c",&a,&b,&c); 也是可以的，但输入时容易出现输入的方式不正确，因此，不提倡使用。

【小结】

任何计算机语言都有一系列的语言规定和语法规则，本章主要介绍了 C 语言中有关数据与数据计算的基本概念和规则，需要在理解的基础上记忆和熟练。本章的要点是：

1、 各种数据类型及其类型说明，其中涉及到的重要概念有：整型、实型、字符型数据的表示、存储、取值范围、数值有效位及各种类型说明形式。例如，单精度实型数据的有效位只有 7 位；字符常数用单引号括起来，每个字符只占一个字节，而字符串常数用双引号括起来，其存储长度总比字符串多一个字节，用于表识字符串的结束；实型数据表示的是近似值；两个整数相除时有可能造成误差；C 中没有逻辑型数据，用 0 表示逻辑假，非 0 表示逻辑真。

2、 各种运算符与表达式，其中涉及到的重要概念有：运算对象的个数、运算优先级、结合型、类型转换等。例如，单目运算符、双目运算符和三目运算符的使用；赋值表达式、逗号表达式、条件表达式和组合运算表达式的值；将一个实型数据赋值给整型变量时将产生误差。关系运算和逻辑运算的结果是数值 1 或 0，表示逻辑真或假；运算时的类型转换是由低级到高级转换。

3、注意一些特殊运算符的使用，例如：-、++、--、*、&等，求负与减的区别，自增、自减与加 1 减 1 的区别，指针运算符*与乘号的区别，取地址运算符&与按位与运算的区别，&&与&的区别，||与|的区别等等。

4、要熟练的掌握输入和输出函数的正确使用。