第3章 chapter 3

运算符与表达式

本章重点

- 算术运算符和算术表达式。
- 赋值运算符和赋值表达式。
- 各种运算符的优先级。

3.1 运算符与表达式概述

在C语言中，经常需要对数据进行运算或操作，最基本的运算形式通常由一些表示特定的数学或逻辑操作的符号描述，这些符号称为运算符或操作符，被运算的对象（数据）称为运算量或操作数。由运算符连接的运算对象组成的符合C语言语法规则的式子称为表达式，表达式描述了对哪些对象按什么顺序进行什么样的操作。

C语言提供了丰富的运算符，能构成多种表达式，表达式把基本操作都作为运算符处理，使得C语言处理问题的功能强、范围广。

C语言的运算符可以分为以下13类。

（1）算术运算符　　（+、-、*、/、%、++、--）
（2）关系运算符　　（>、<、>=、<=、==、!=）
（3）逻辑运算符　　（!、&&、||）
（4）位运算符　　（~、&、|、^、<<、>>）
（5）赋值运算符　　（=及其扩展赋值运算符）
（6）条件运算符　　（? :）
（7）强制类型转换运算符　（类型）
（8）逗号运算符　　（,）
（9）指针运算符　　（*、&）
（10）求字节运算符　　（sizeof）
（11）分量运算符　　（.、->）
（12）下标运算符　　（[ ]）
（13）其他　　（如圆括号、函数调用运算符()）

本章主要介绍算术运算符、赋值运算符、逗号运算符和强制类型转换运算符等。其

他运算符将在后面的章节中陆续介绍。

在学习运算符时要掌握运算符的几个属性，包括运算符的目数、优先级和结合性等，同时要注意运算符要求的运算量类型及结果类型。

3.2 基本算术运算符与算术表达式

1. 基本算术运算符

算术运算符中，加或正号(+)、减或负号(-)、乘(*)、除(/)、求余(%)为基本算术运算符，功能和数学中的类似，但用法有很大的不同，以下几点需要特别注意。

(1) +(加法)、-(减法)、*(乘法)、/(除法)、%(求余)为双目运算符，即参加运算时要求有两个运算量。运算过程中先乘除、后加减，即乘、除(*，/，%)的优先级高于加减(+，-)。若优先级相同，则按自左至右的结合方向计算，这一点和数学中的运算规则一致。C 语言中，当一个运算量两侧的运算符优先级相同时，则根据运算符的结合性进行运算。运算符的结合性分为两种，即左结合性(自左至右)和右结合性(自右至左)，这 5 个算术运算符都是左结合的，例如，运算 2-3*4 表达式时，3 的两侧的运算符为-和*，按优先级先乘后加，而运算 1+2-3 时，2 的两侧的运算符的优先级相同，则根据运算符的左结合性按从左到右的方向运算。

(2) +(正号)和-(负号)为单目运算符，即只要求有一个运算量。C 语言中，所有单目运算符的优先级均高于双目运算符，所有单目运算符都是右结合的，即优先级相同时按从右到左的方向运算。

(3) +、-、*、/的运算量可以是整型或浮点型数据，结果的类型由运算对象的类型决定，而“%”要求两个运算量都必须为整型数据，结果也是整型数据，是两个整数相除后的余数。另外，运算结果的正负号与被除数的符号一致。例如，7%4 的值为 3，9%10 的值为 9，5/2%3 的值为 2，9%(-2)的值为 1。

(4) 两个整数相除的结果为整数，舍弃小数部分，如 5/3 的值是 1。但是如果被除数或除数中有一个负数，则舍入的方向是不固定的。例如，-5/3 的运算结果在有的系统中是-1，而在有的系统中是-2。但多数系统均采用“向 0 取整”的规则，即取整数时向 0 靠拢。例如，5/3 的结果是 1，-5/3 的结果是-1。

2. 算术表达式

用算术运算符和圆括号将运算对象连接起来，且符合 C 语言语法规则的式子称为 C 算术表达式，运算量包括常量和变量等。将数学中的表达式写成 C 算术表达式时，必须满足 C 语言的语法规则。例如，$\frac{1}{2}$ab 写成 C 算术表达式应为 1.0/2*a*b 或 1/2.0*a*b，请考虑为什么不能写成 1/2*a*b。

3. C 算术表达式的常用规则

(1) 乘号不能省。如：ab 应写成 a*b，否则系统会认为 ab 为一个变量名。

(2) C 算术表达式中可以使用圆括号改变运算顺序,因为圆括号是 C 语言中优先级最高的运算符之一,在一个表达式中会最先执行圆括号这个运算符,圆括号可以嵌套使用,但左右括号必须匹配。

(3) 应避免两个运算符并置。如:a * b/－c 应写为 a * b/(－c)。

(4) 由于两个整数相除的结果仍为整数,因此若想让得到的结果为实数,就必须把除数或被除数转换成实数。如:5/12 应写成 5.0/12 或 5/12.0。

(5) 表达式中的所有符号要写成一行。如:$\frac{1}{2}$应写成 1/2.0。

(6) 上下角标不能直接写,需要转换。如:x^2→x * x,x_1→x1。

(7) 调用标准数学函数时,自变量应写在一对括号内。如:|－216|应写成 fabs(－216),sin12 应写成 sin(12),e^x 应写成 exp(x)。常用的标准数学库函数见附录 D。

(8) 三角函数的自变量应使用弧度。如:sin50°应写成 sin(50 * 3.1415926/180)。

【例 3.1】 把下面的数学表达式写成 C 算术表达式。

$$\frac{2a+\sin(45^\circ)}{|x-y|\,e^{2.3}}$$

解:

(2 * a＋sin(45 * 3.1415926/180))/(fabs(x－y) * exp(2.3))

【例 3.2】 编写程序,区分整型数据和浮点型数据参与除法运算的不同。

```
#include <stdio.h>
int main(){
    int a=10;
    float x,y;
    x = a/20;
    y = a/20.0;
    printf("x=%f\n",x);
    printf("y=%f\n",y);
    return 0;
}
```

运行结果:

```
x=0.000000
y=0.500000
```

当参与除法运算的数据都是整型数据和浮点型数据时,结果是完全不同的,如同我们在学习、生活和工作中要严格依法依章办事,讲规则,守规则,做任何事情都要一丝不苟,要做遵纪守法的文明人。

3.3 赋值运算符与赋值表达式

1. 赋值运算符

C 语言中,“＝”称为赋值运算符,用来连接两个操作数,其一般形式为

变量=表达式

其作用是将“=”右面的表达式的值赋给左面的变量，赋值运算符是有方向的，意义和数学中的等号不同，C 语言中用“==”表示相等。

例如：

```
a=3;            /* 把整数 3 赋给变量 a,即存储到 a 中 */
b=4;
a=b;            /* 把 b 的值赋给 a 后,a 和 b 的值都是 4 */
b=a;            /* 假设赋值之前,还是 a=3,b=4,把 a 的值赋给 b 后,a 和 b 的值都是 3 */
```

赋值运算符是双目运算符，是右结合的，赋值运算符的优先级比较低，仅高于逗号运算符，例如：

```
a=a+1;          /* 先计算 a+1 的值,然后赋值给变量 a,变量 a 的存储单元中的原值被覆盖 */
```

2. 赋值表达式

由赋值运算符将一个变量和一个表达式连接起来的式子称为赋值表达式。既然是表达式，就应该有一个确定的值，这个值就是被赋值后左边变量的值，例如，表达式 a=3+2 的值就是 5，可以将一个赋值表达式再赋给一个变量，例如表达式：

```
b=a=3+2
```

在 a 的两侧有两个赋值运算符，先运算哪一个由运算符的结合性决定，赋值运算符是右结合的，所以运算顺序为从右至左，先运算右边的赋值表达式 a=3+2，前面的表达式相当于 b=(a=3+2)，先运算 a=3+2，a 的值是 5，赋值表达式 a=3+2 的值也是 5，再将 5 赋给 b，变量 b 的值为 5，同时表达式 b=a=3+2 的值也是 5。

3. 复合的赋值运算符

赋值运算符与算术运算符和位运算符的结合形成了 10 个复合赋值运算符，包括：

+=，-=，*=，/=，%=（与算术运算结合）

<<=，>>=，&=，^=，|=（与位运算符结合）

后 5 种复合赋值运算符与位运算有关，将在第 9 章介绍，本章主要介绍前 5 种。例如：

a+=3 等价于 a=a+3。

x*=y+8 等价于 x=x*(y+8)。

```
a+=a-=a*=a          /* 若 a 的初值为 3,则表达式的值为 0 */
```

此表达式为复合的赋值表达式，它的求解过程如下。

① 按结合方向用加括号的方法确定计算顺序：a+=(a-=(a*=a))。

② 改写成常规的写法：a=a+(a=a-(a=a*a))。

③ 依次计算：a=a+(a=a-(a=9))；a=a+(a=a-9)；a=a+(a=0)；a=a+0；

a=0+0；a=0。

在上面的复合赋值表达式的求解过程中，当运算表达式 a=9 时，a 中的数据 3 被替换成了 9，因此在运算后面的表达式 a-9 时，参与运算的 a 的值是 9，而不再是 3。

赋值表达式可以包括在其他表达式中。如“printf("%d",a=b);”，执行此输出语句时，首先计算赋值表达式 a=b 的值，然后把表达式的值输出。

3.4 不同数据类型的转换

当把一个常量赋值给一个变量或不同的数据类型一起参与运算时，通常要求它们的类型是一致的，若它们的类型不一致，但都是数值型或字符型，就要先把数据转换成相同类型后再执行相应的运算。这个转换过程包括两种方式：自动转换和强制转换。

3.4.1 自动转换

自动转换是由 C 语言的编译器自动执行的，赋值时的自动转换主要包括以下 6 方面：

(1) 将浮点型数据(包括单、双精度)赋给整型变量时，舍弃实数的小数部分。例如：

```
int i;
i=3.54;
```

当执行上述语句时，变量 i 中存储的值是右端数据的整数部分 3。

(2) 将整型数据赋值给浮点型变量时，数值不变，但以实数形式存储到变量中。例如：

```
float a=12;
```

当上述语句执行结束后，变量 a 中存储的值为 12.0，按%f 格式输出时会得到 12.000000。

(3) 将 double 型数据赋给 float 变量时，截取其前 7 位有效数字，放到 float 变量的存储单元中；将一个 float 型数据赋给一个 double 型数据，数值不变，有效位扩展到 16 位。

(4) 将字符型数据赋给整型变量时，将字符型数据在内存中的存储内容(占 1 字节)放到整型变量存储单元的低 8 位(1 字节)；对于高 8 位，不同的系统有不同的处理，有的补 0，有的补 1。将整型数据赋给一个 char 型变量时，只将其内存中的低 8 位原样赋给 char 型变量，高 8 位截去。例如：

```
char c;
int i=321;
c=i;
```

当把整型变量 i 中的 321 赋给字符型变量 c 时，只把低 8 位的数据赋给变量 c，即 c 的值是 65，用%c 输出时，输出的结果是字符'A'。赋值情况如图 3.1 所示。

(5) 将带符号的 int 型数据赋给 long 型变量时，将 int 型数据放入 long 型变量的低 16 位；高 16 位实行符号位扩展，即若 int 型数据为正数，则 long 型变量的高 16 位补 0，否

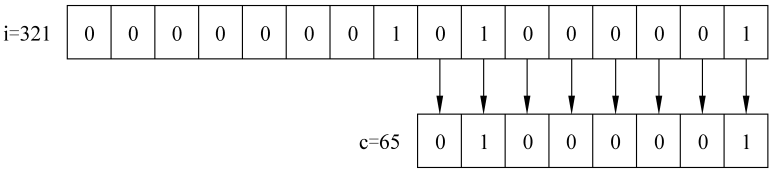

图 3.1　整型数据赋给字符型变量

则补 1。将 long 型数据赋给 int 型变量时，只需要将 long 型数据的低 16 位放入 int 型变量，高 16 位截去，此时数据可能发生变化。

例如：

```
long a=65536;
int b;
b=a;
```

将长整型变量 a 中的 65536 赋给整型变量 b 时，只把低 16 位赋给了 b，即 b 的值是 0，赋值情况如图 3.2 所示。

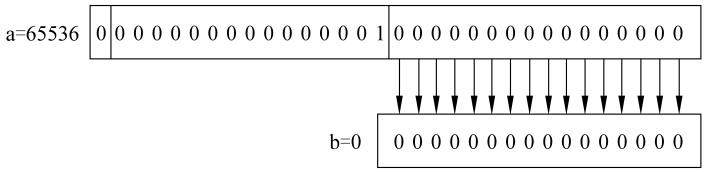

图 3.2　长整型数据赋给整型变量

(6) 将 unsigned int 型数据赋给 long int 型变量时，把 unsigned 型数据放入 long 型数据的低 16 位，高 16 位补 0。将 unsigned 型数据赋给一个所占字节相同的整型变量时，数据将原样放入非 unsigned 型变量，但可能会因超出范围而出错，这是由于 unsigned 型数据的范围要比相应的非 unsigned 型数据的范围大。将非 unsigned 型数据赋给长度相同的 unsigned 型变量时，也是原样照赋(符号位的数字作为数值一起传送)。

从以上几方面的赋值情况可以看出，不同类型的整型数据之间的赋值实质上就是按存储单元中的存储形式直接传送。将占用字节数少的整型数据赋给占用字节数多的变量时，要对高位部分补 0 或补 1；相反，将占用字节数多的整型数据赋给占用字节数少的变量时，只截取低位部分赋值，高位部分舍掉，因为内存单元中的数据是按补码存放的，这里涉及补码的知识，不再详述。

整型、浮点型和字符型数据可以进行混合运算，不同类型的数据在同一表达式中参与运算时也会发生自动转换，将不同类型的数据先转换成同一类型，然后进行运算。87ANSI C 的转换规则如图 3.3 所示。

在图 3.3 中，向左的箭头表示必定的转换。例如，若表达式中出现了 short 型或 char 型，则必定转换为 int 型进行运算；若为 float 型，则必定转换为 double 型，以提高运算精度。

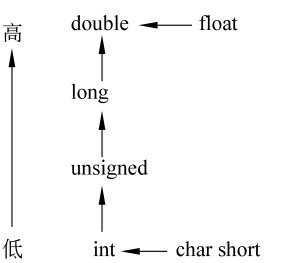

图 3.3　不同类型的数据转换规则

向上的箭头表示当运算对象为不同类型时的转换方向。例如，int 型和 float 型数据进行运算时，首先 float 型必定转换为 double 型，int 型直接转换为 double 型，然后两个 double 型进行运算，结果为 double 型。

从图 3.3 中可以看出，表达式中只要有一个 float 型或 double 型数据，结果就必定是 double 型。例如，若已知："int x; float y; double m; long n;"，则计算表达式 1＋'b'＋x/y－m＊n 时，按各运算符的优先级和结合性，表达式的运算顺序如下。

① 进行 x/y 运算，将 x 和 y 都转换成 double 型，然后相除，结果为 double 型。

② 将 n 转换成 double 型，然后进行 m＊n 运算，结果为 double 型。

③ 进行 1＋'b'的运算，将'b'转换为 int 型数据 98，然后相加，结果为 int 型数据 99。

④ 将 1＋'b'的结果转换成 double 型后与 x/y 的运行结果相加，结果为 double 型。

⑤ 将 1＋'b'＋x/y 的结果与 m＊n 的结果相减，最后表达式的结果类型为 double 型。

3.4.2 强制类型转换

强制类型转换通过强制类型转换运算符实现。

一般形式为

(类型标识符)表达式

例如：

```
int a;
float b,c;
(int)b                          /*将 b 中的值转换为 int 型*/
(int)(b+c)                      /*先计算 b+c,然后将 b+c 的值转换成 int 型*/
```

【说明】

(1)"(类型标识符)"称为强制类型转换运算符，作用是将其后面的表达式的值转换为括号中指定的类型。C 语言的运算符非常丰富，优先级各不相同，强制类型转换运算符的优先级高于算术运算符，在执行表达式(int)b＋c 时，首先把 b 的值转换成 int 型，然后进行加法运算。

(2) 在对变量进行强制类型转换时，得到的结果并没有存放到原来的变量中，即不改变原来变量的类型，只是得到一个中间值。

【例 3.3】 强制类型转换。

程序如下：

```
#include "stdio.h"
int main()
{   float a;
    int b;
    a=3.56;
    b=(int)a;
    printf("a=%f,b=%d",a,b);
}
```

运行结果：

```
a=3.560000,b=3
```

【说明】 根据运行结果可以看出，被强制类型转换的变量 a 中的值并没有改变，因此可以说明转换时只是得到了一个中间结果。

3.5 自增、自减运算符

自增运算符和自减运算符都是单目运算符，它们只能和一个单独的变量组成表达式。自增运算符和自减运算符的符号表示分别为“＋＋”和“－－”，它们的作用分别是使变量的值增加 1 和减少 1。例如，设 i 为 int 型变量，若想使 i 的值增加 1，则可以用表达式＋＋i 或 i＋＋，此时＋＋i 和 i＋＋的结果是相同的。

当＋＋i 和 i＋＋作为其他表达式的一部分时，结果是不同的。这是由于＋＋i 是先使 i 的值加 1，然后 i 参与表达式的运算；而 i＋＋是 i 先参与表达式的运算，然后使 i 的值加 1，这样就会使整个表达式的运算结果不同。自减运算与自增运算类似，只是把加 1 改成减 1。

【例 3.4】 自增运算符的应用。

程序如下：

```
#include "stdio.h"
int main()
{   int i,j,m,n;
    i=8;j=8;
    m=i++;                    /*相当于 m=i; i=i+1;的组合*/
    n=++j;                    /*相当于 j=j+1; n=j;的组合*/
    printf("i=%d,m=%d\n",i,m);
    printf("j=%d,n=%d",j,n);
}
```

运行结果：

```
i=9,m=8
j=9,n=9
```

【说明】

(1) 自增运算符和自减运算符只能用于变量，不能用于常量或表达式。

(2) “＋＋”和“－－”是单目运算符，优先级高于所有的双目运算符，仅次于圆括号运算符，结合方向是右结合的。

3.6 逗号运算符与逗号表达式

在 C 语言中，逗号不仅可以作为分隔符，还可以作为一种运算符，用来把两个或多个表达式连接起来。逗号表达式的一般形式为

表达式 1,表达式 2,…,表达式 n

逗号运算符也称顺序求值运算符,逗号表达式的运算过程是按从左到右的顺序逐个运算每个表达式,即先运算表达式 1,再运算表达式 2,以此类推,直到表达式 n 运算结束,整个逗号表达式的值和类型就是最右边的表达式 n 的值和类型。

逗号运算符是双目运算符,优先级最低,其结合方向是从左到右。

例如:

```
a=3*5,a*4          /*先计算赋值表达式的值,结果是 a 的值为 15,逗号表达式的值为 60*/
(a=4*5,a*3),a+10          /*先进行圆括号内的表达式运算,结果 a=20,括号内的逗
                            号表达式的值为 60,整个逗号表达式的值为 30*/
```

3.7 小 结

本章主要介绍了一些常用的运算符,包括赋值运算符、算术运算符、自增自减运算符、强制类型转换运算符和逗号运算符。其中,赋值运算符的符号是“=”,它的作用是将“=”右面的数据赋给左面的一个变量,它是有方向的,它不是数学中的等号。算术运算符中,注意“/”和“%”的特别之处,其中,除法运算符“/”两边的运算量若都为整型,则结果为整型,若有一边为浮点型,则结果为浮点型。而求余运算符“%”要求运算符两边的运算量必须为整型。自增和自减运算符“++”和“--”在使用时有很多的细节问题,一些写法的运行结果可能会出乎你的预料,在不知道确切的结果时不要随意使用,例如当 i=2 时,经过赋值运算 j=i+++i+++i++之后,j=6,i=5。所以在使用这两个运算符时,应该仅使用其最简单的形式。

学习运算符必须掌握各个运算符的相关属性,如运算符的目数、优先级和结合性。例如,自增自减运算符是单目运算符;逗号运算符的优先级最低,其次是赋值运算符。所有单目运算符都是右结合的,所有单目运算符的优先级都高于双目运算符等。

在 C 程序中,当涉及计算数学中的表达式的值时,要注意写成符合 C 语言语法规则的 C 算术表达式,要保证值不变,包括分数的处理、添加一些必要的圆括号等。

习 题 3

一、选择题

1. 若有以下类型的说明语句:

```
char w; int x; float y; double z;
```

则表达式 w*x+z-y 的结果为_____类型。

A. float　　B. char　　C. int　　D. double

2. 若 x 为 int 型变量，x=6;则执行以下语句后，x 的值为_____。

```
x+=x-=x*x;
```

A. 36　　B. -60　　C. 60　　D. -24

3. 若题中变量已正确定义并赋值，则下列符合 C 语言语法的表达式是_____。

A. a%=7.6　　B. a++,a=7+b+c

C. int(12.3)%4　　D. a=c+b=a+7

4. 已知 a 和 b 是 int 型变量，x 和 y 是 float 型变量，且各变量已经被有效赋值。正确的赋值语句是_____。

A. a=1,b=2　　B. y=(x%2)/10;　　C. x*=y+2;　　D. a+b=x;

二、阅读程序，写出运行结果

```
#include "stdio.h"
int main()
{   int m=1,n=10,a,b;
    a=m++;
    b=++n;
    printf("a=%d,b=%d,m=%d,n=%d\n",a,b,m,n);
}
```

运行结果：_____。

三、编程题

1. 将华氏温度转换为摄氏温度和绝对温度(下式中，c 表示摄氏温度，f 表示华氏温度，k 表示绝对温度)。

$$c=\frac{5}{9}(f-32)$$

$$k=273.16+c$$

在程序中使用 scanf 输入 f 的值，然后计算 c 和 k 的值。程序运行两次，为 f 输入的值分别是 34 和 100(注意程序中数据类型和输入/输出语句的格式)。

2. 已知整型变量 a=6，请编程计算并输出下列赋值表达式的值。

```
a+=a-=a*=a
```

3. 输入三角形的三边长，求三角形的面积。求三角形面积的方法为$\sqrt{s(s-a)(s-b)(s-c)}$，其中，a,b,c 为三边长，s=(a+b+c)/2。