第3章

数据运算

计算机的基本功能是进行数据运算，而数据运算主要是通过对表达式的计算完成的。表达式是将运算量用运算符连接起来组成的式子，其中运算量可以是常量、变量或函数。由于运算量划分为不同的数据类型，每一种数据类型都规定了自己特有的运算或操作，这就形成了对应于不同数据类型的运算符集合及其相应的求值规则。本章主要介绍 C 语言运算符及各种表达式的构造与求值规则。

3.1 算术运算

算术运算，也叫数值运算，是程序设计中使用最多的一种数据运算。

3.1.1 算术运算符

算术运算符包括加(+)、减(-)、乘(*)、除(/)、求整数余数(%)、加 1(++)、减 1(--)及正负号(+、-)，其中，+、-、*、/、%运算符必须连接两个运算量，称为二元运算符；++、--、正负号(+、-)运算符只能连接一个运算量，称为一元运算符，参见附录 B。

1. 求余运算

求余运算也叫求模运算，只能用来求两个整数(int 型或 char 型)的余数。假设两个整数分别为 a 和 b，a%b 表示求 a 整除 b 后的余数，基本规则如下：

```
a%b=a-a/b*b
```

其中，a/b 得到 a 除以 b 的整数商，例如：

```
14%(-6)=14-14/(-6)*(-6)=14-(-2)*(-6)=2
-14%6=-14-(-14/6)*6=-14-(-2)*6=-2
```

2. 加 1 和减 1 运算

加 1 和减 1 运算也叫自加和自减运算，是对变量自身增加 1 或减少 1 的运算。例如：

++i 表示 i=i+1

--i 表示 i=i-1

(1) ++或--只能对变量施加运算,不能对常量或表达式施加运算。如

```
++(25),++(x+y),--(x-3),++i--
```

等,是错误的。

(2) ++或--可以写在变量之前,称为前缀,也可以写在变量之后,称为后缀。它们之间的区别在于:

n++表示先取 n 的值,然后令 n=n+1

++n 表示先令 n=n+1,然后取 n 的值

例如,令 i=5,j=5,则

```
x1=++i;
```

先计算 i=i+1,得到 i 的值是 6,然后将 6 赋给 x1;

```
x2=j++;
```

先将 j 的值 5 赋给 x2,然后计算 j=j+1,得到 j 的值是 6。

【例 3.1】 加 1 和减 1 运算。

由于加 1 和减 1 运算既有计算的功能,又有赋值的功能,因此将改变变量的值。

```
#include <stdio.h>
void main( )
{   int a=100;
    printf("%d\n",a);
    printf("%d\n",++a);          //先计算 a 的值为 101,再输出 a 的值
    printf("%d\n",a--);          //先输出 a 的值 101,再计算 a=a-1 得 a 的值为 100
    printf("%d\n",a);
}
```

程序运行结果如下:

```
100
101
101
100
```

3.1.2 算术表达式及其求值规则

算术表达式是用算术运算符连接数值型的运算量构成的式子,用来完成数值计算的功能。如:

```
-5*3+6%4/2-1,(a+b)/(a-b),a+(b++)*c,(i++)+j,y+x++-(++i)
```

等。

1. 算术表达式的书写规则

正确书写算术表达式对正确计算该表达式的值非常重要，应注意以下几点。

(1) 乘法运算符“*”在表达式中既不能省略，也不能用“·”或“×”代替；除法运算符“/”也不能用其他符号代替。

(2) 正确使用运算符的优先级和结合性来决定表达式的求值顺序。

优先级指的是当几个不同的运算符同时出现在表达式中时，各个运算符的优先次序。例如，在表达式 a/b+c 中，由于除法的优先级高于加法，因此，应先计算 a/b 然后与 c 相加。

结合性指的是当同一个优先级别的运算符同时出现在表达式中时，其运算的优先次序。例如，在表达式 a/b*c 中，由于乘除法处于同一优先级，它们的优先次序由结合性确定。乘除法的结合性为自左至右，因此，应先计算 a/b 然后再乘 c；又如，在表达式 -++i 中，取负号和减 1 运算处于同一优先级，根据它们自右至左的结合性，应先计算 ++i 然后再取负。

(3) 恰当地使用括号可以改变优先次序或防止出现二义性。例如，表达式 a/(b*c) 中，使用括号可将乘法运算提前，从而改变了运算顺序，因此，该表达式与 a/b/c 是等价的。又如，对诸如 ---i 这样的式子，其中连续三个减号的含义可以有多种解释，为防止二义性，应使用括号将后两个减号作为减 1 运算符，才能形成正确的表达式 -(--i)；否则，编译系统就会报错。

(4) 在表达式中只允许使用括号，不允许使用方括号或花括号，多重括号可以嵌套配对使用，如((a+b)-(c+d))/(b+d)。

(5) C 语言不含乘方运算，当需要进行乘方运算时，可以通过连乘的方式实现乘方运算，也可以使用 C 编译系统提供的数学函数，如 pow(x,y)表示 x^y，pow10(n)表示 10^n。

2. 算术表达式的求值规则

(1) 优先级和结合性规则。计算机在进行表达式的计算时，通常是严格按运算符的优先级和结合性进行的，就算术表达式而言，括号最优先，其次是一元运算符，然后是乘、除和求模，最后是加、减；当同一优先级的运算符同时出现时，按它们的结合性确定其优先次序。

【例 3.2】 算术表达式计算中的优先级和结合性规则。

```
#include <stdio.h>
void main( )
{   int a=3,b=5,c=15,d=2;
    printf("%d\n",6-a*b/c%d);
    printf("%d\n",++a-a++);
    printf("%d\n",a);
}
```

在第一个 printf( )语句中，由于乘、除、求模的优先级高于减法，因此先计算 a*b/

c%d,在这一步计算中,根据从左到右的原则依次进行乘、除和求模的运算,得到中间结果1,然后再计算 6－1,结果为 5。在第二个 printf( )语句中,由于＋＋优先于－,因此,先计算第一个＋＋a,得 a=4(注意前后两个 a 是同一个变量,它们具有相等的值),然后计算4－4,输出 0,输出之后还要进行 a＋＋运算,得 a 的值为 5,并在第三个 printf( )语句中输出 5。

(2) 自动类型转换规则。算术表达式中可能含有不同类型数值量的混合运算,除了遵循优先级和结合性规则外,还要遵循如下自动类型转换规则。

① 如果表达式中含 char 型,全部转换成 int 型。例如,表达式'a'＋'3'＋15 中,字符常量'a'和'3'将被转换成 int 型的 97 和 51,然后再进行计算,其结果为 163。

② 如果表达式中含 float 型,全部转换成 double 型。

③ 同类型的运算量运算后,结果仍为同类型。例如,表达式 3/2 的值是 1 而不是1.5,因为其中的两个运算量均为 int 型,计算结果也应是 int 型。

④ 不同类型的运算量运算前,先将低精度类型转换成高精度类型,运算结果为高精度类型。这种类型转换称为自动类型转换。

类型精度的高低取决于两点:一是看类型的长度,长度越长,精度越高,例如,double 型的长度为 8,而 float 型的长度为 4,因此,double 型的精度高于 float;二是看该类型所能表示的数的范围,能表示的数越大,精度越高。例如,int 型能表示的最大数为 32767,而 unsigned int 型能表示的最大数为 65535,因此 unsigned int 的精度高于 int。

【例 3.3】 混合运算的自动类型转换求值规则。

```
#include <stdio.h>
void main( )
{   char ch='a';
    int i=5;
    unsigned int j=6;
    long int k=12;
    float f=3.0;
    double d=6.0;
    printf("%lf\n",ch/i+i*k-(j+k)*(f*d)/(f+i));
}
```

表达式 ch/i＋i＊k－(j＋k)＊(f＊d)/(f＋i)中含有多种类型的变量,根据优先级和结合性规则及自动类型转换规则进行计算,计算过程如图 3.1 所示。图中凡标有类型关键字的,表示对应的数据进行了自动类型转换,带圆圈的数字表示计算的顺序号。

和变量一样,C 语言的表达式不仅有值,也分为不同的类型。表达式值的类型就是表达式的类型。本例中,由于表达式 ch/i＋i＊k－(j＋k)＊(f＊d)/(f＋i)的值是 double 型的,该表达式也是 double 型的。程序输出结果如下:

```
38.500000
```

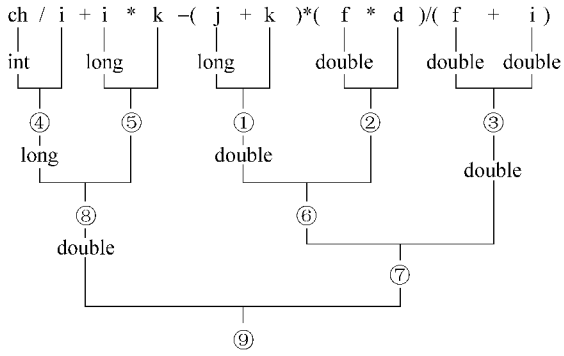

图 3.1　表达式 ch/i+i * k-(j+k) * (f * d)/(f+i)的求值过程

自动类型转换总是将低精度数转换为高精度数，目的是在转换过程中不损失数的精度。要注意，当负整数转换成无符号整数时，由于符号位变成了数值位，其值将会发生变化。例如，short int a=-1 转换成 unsigned 型时将变成 65535。

3. 强制类型转换规则

在算术表达式的计算中，如果要违反自动类型转换规则，可以采用强制类型转换。例如，根据自动类型转换规则，表达式 3/2 的值是 1 而不是 1.5。如果将整数 3 强制转换成 float 型，写成(float)3/2，其值就是 1.5 而不是 1。强制类型转换的一般格式如下：

```
(类型关键字)(表达式)
```

强制也是一种一元运算符，且与其他一元运算符具有相同的优先级和结合性，它的功能是将指定的表达式的值强制转换成指定的类型。例如：

```
(double)a
```

将变量 a 的值强制转换成 double 型

```
(int)(x+ y)
```

将表达式(x+y)的值强制转换成 int 型

```
(int)x+ y
```

将变量 x 的值强制转换成 int 型，然后与 y 相加。

【例 3.4】 强制类型转换。

```
#include <stdio.h>
void main( )
{   int a=7,x=3,y=2;
    float b=2.5f,c=4.7f,z=3.5f;
    printf("%f\n",b+a%3 * (int)(b+c)%2/4);
    printf("%d\n",(x+y)%2+(int)b/(int)z);
}
```

程序运行结果如下：

```
2.500000
1
```

程序运行时，在第一个 printf( )语句中，b＋c 的值为 7.2，强制转换成 int 型后变成 7，整个表达式变成 2.5＋7％3＊7％2/4，最后计算结果为 2.5，float 型，因此要用"％f"控制输出；第二个 printf( )语句中，变量 b 和 z 的值分别被强制转换成 int 型的 2 和 3，整个表达式变成(3＋2)％2＋2/3，最后计算结果为 1(注意，在计算 2/3 时，其值是 0 而不是 0.67)，int 型，因此要用"％d"控制输出。

在程序设计实践中，强制类型转换有时非常有用。例如，运算符％要求连接两个整型量，设 x 是 float 型变量，则表达式 x％3 就是错误的，用(int)x％3 就允许了。第 5 章将要介绍的数组，当用 float 型或 double 型变量作下标时，也需要使用强制类型转换。还有，C 编译系统提供的数学函数大多数是 double 型的，通过强制类型转换可将它们转换成需要的类型。

使用强制类型转换应注意以下几点。

(1) 在对一个表达式进行强制类型转换时，整个表达式应该用括号括住。例如，(float)(a＋b)若写成(float)a＋b 就只对变量 a 进行了强制类型转换。

(2) 在对变量或表达式进行了强制类型转换后，并不改变变量或表达式原来的类型。例如，设 x 为 float 型，y 为 double 型，则(int)(x＋y)是将 x＋y 的值强制转换成 int 型，而表达式 x＋y 本身仍为 double 型。

(3) 高精度整数转换成低精度整数时，由于会丢掉一些数位，因而会造成值的改变。例如，int 型整数 65536 转换成 short int 型整数时会变成 0；int 型整数 65535 转换成 short int 型整数时会变成－1。

3.2 赋值运算

在 C 语言中，赋值运算与赋值语句紧密相关，赋值表达式的末尾加上分号就是赋值语句。通过赋值表达式有助于揭示 C 语言特有的一些语言规则。

形如

```
v=e
```

的表达式称为赋值表达式。其中，“＝”称为赋值号或赋值运算符，“＝”右边的 e 是一个表达式，左边的 v 是一个变量或数组元素。赋值表达式的功能是计算 e 的值并存放在 v 中。

1. 赋值运算符

(1) 赋值运算符“＝”是一种二元运算符，必须连接两个运算量，其左边只能是变量或数组元素，右边则可以是任何表达式。例如，x＝y＋2，x＝x＋1 等是允许的，而 y＋2＝x，5＝x 等是错误的。

(2) 赋值运算符除了“=”,还有另外 10 种复合赋值运算符,它们由赋值号“=”和另外一个二元运算符组成,具有计算和赋值双重功能,这 10 种复合赋值运算符是+=、-=、*=、/=、%=、&=、|=、^=、<<=和>>=。其中,前 5 种复合赋值运算符具有算术运算和赋值的双重功能:

x+=e 等价于 x=x+e;

x-=e 等价于 x=x-e;

x*=e 等价于 x=x*e;

x/=e 等价于 x=x/e;

x%=e 等价于 x=x%e。

后 5 种复合赋值运算符则具有位运算和赋值的双重功能。

(3) 赋值运算符的优先级在所有的 C 运算符集中处于倒数第二,参见附录 B,结合性均为从右到左。

(4) 在使用复合赋值运算符时,要将右边的表达式作为一个整体与左边的变量进行运算,例如,x*=y+3 表示 x=x*(y+3)而不是 x=x*y+3。

2. 赋值表达式

(1) 赋值表达式作为一种表达式,同样也有值和类型的问题。C 语言规定:被赋值的变量或数组元素 v 的值和类型就是赋值表达式的值和类型。在赋值过程中,当赋值号两边的 v 和 e 类型相同时,直接赋值;当 v 和 e 类型不一致时,计算机系统会自动将 e 的值转换成与 v 相同的类型后再赋给变量 v。赋值表达式中的类型转换既有可能要将 e 的值由低精度转换成高精度,也有可能要由高精度转换成低精度,以满足与 v 的类型一致。例如,赋值表达式 x=10.8-4.3 执行后,假设 x 是 int 型的变量,则根据类型转换规则,变量 x 得到的值是 6,那么,该赋值表达式的值为 6,类型为 int。

(2) 上节介绍的++和--运算施加于变量后也构成赋值表达式,例如,i++表示 i=i+1,所以 i++也是赋值表达式。但要特别注意++或--只能施加于变量,而不能施加于表达式。例如,++i++就不是正确的赋值表达式,因为根据++运算符从右到左的结合性,它表示++(i=i+1),++被施加在赋值表达式 i=i+1 上是不允许的。

(3) 赋值表达式本身也可以作为一个表达式赋给另一个变量,例如,b-=b*b 是一个正确的赋值表达式,可以将它赋给变量 a,写成 a=b-=b*b,也构成一个正确的赋值表达式。这时,由于赋值运算符从右向左的结合性,先计算 b-=b*b,然后再将这个值赋给 a。但是,要特别注意,赋值号左边必须是变量或数组元素,诸如 a=b+c=d、x=i++=y、u=2=v 这样的表达式就不是正确的赋值表达式。

【例 3.5】 赋值表达式中的类型转换。

```
#include <stdio.h>
void main( )
{   int a; char b; float c;
    c=2.5;
    b=c;
```

```
    a=b;
    printf("%d,%d,%.1f",a,b,c);
}
```

在计算数值型的赋值表达式时，要将赋值号右边的表达式转换成与左边变量或数组元素相同的类型。这种转换既可能将高精度类型转换成低精度类型，也可能相反。在本例中，计算赋值表达式 c=2.5 时，先要将双精度型常数 2.5 转换成 float 型，再赋给变量 c；在计算赋值表达式 b=c 时，先要将 c 中的 float 型数转换成 char 型，然后赋给变量 b；在计算赋值表达式 a=b 时，先要将 b 中的 char 型值转换成 int 型，然后赋给变量 a。前两个赋值是将高精度转换成低精度，第三个赋值则将低精度转换为高精度。程序运行结果如下：

```
2,2,2.5
```

【例 3.6】 赋值表达式中类型转换引起的数值变化。

```
#include <stdio.h>
void main( )
{   int a=-1;
    unsigned int b;
    b=a;
    printf("%d %u\n",a,b);
}
```

在 Visual C++ 6.0 中，数值是按补码形式存储的，int 型的－1 的二进制存储格式为 32 个“1”，在进行 b=a 的赋值时，要将－1 转换成 unsigned int 型，这时，最高位的符号位也被视为数值位，32 个“1”代表的数值是 4294967295。因此，程序输出结果如下：

```
-1  4294967295
```

【例 3.7】 复合赋值运算符和变量的连续赋值。

```
#include <stdio.h>
void main( )
{   int a=12;
    a+=a-=a*a;
    printf("%d\n",a);
}
```

在计算赋值表达式 a＋＝a－＝a＊a 时，按从右到左的顺序先计算 a＊a，得 144，然后进行 a－＝144 的赋值运算，得到 a＝12－144＝ －132，该值作为赋值表达式 a－＝a＊a 的值，参加下一个赋值表达式的计算，得 a＝a＋(－132)＝－264。因此，当同一变量被连续赋值时，要注意其值在不断变化。本例程序的运行结果如下：

```
-264
```

3.3 逗号运算(顺序运算)

逗号运算符的作用是连接多个数据项,从而将它们能作为一个整体来处理。前面介绍的用一个类型关键字定义多个变量时,各变量之间要用逗号隔开,在 printf( )函数中多个输出项之间用逗号隔开,在 scanf( )函数中多个输入项之间用逗号隔开等等,目的都是将多个变量、输出项或输入项视为一个整体。本节讨论用逗号运算符将多个表达式作为一个整体时的处理原则。

1. 逗号表达式

用逗号运算符将几个表达式连接在一起就构成逗号表达式。例如:

```
a=3*5,a*4,a+5
```

通过使用两个逗号运算符将一个赋值表达式和两个算术表达式连接起来,构成一个逗号表达式。

(1) 逗号运算符的优先级和结合性在所有的运算符中是最低的,处于赋值运算符之后。其结合性为从左到右。

(2) 逗号运算符的功能是从左到右依次计算各个表达式的值,因此,逗号运算符也称顺序运算符。

(3) 逗号表达式的值是逗号表达式中最右边的一个表达式的值。例如,设 a 是 int 型变量,其初值为 5,则

```
a=3*5,a*4,a+5
```

中第一个表达式的值是 15,第二个表达式的值是 60,第三个表达式的值是 20,那么整个逗号表达式的值就是第三个表达式的值,即 20,而 a 的值为 15。

如果用括号改变计算顺序,把上式改写成

```
a=(3*5,a*4,a+5)
```

就要先计算逗号表达式(3*5,a*4,a+5)的值,再将该逗号表达式的值赋给变量 a,结果得到 a=10。

【例 3.8】 逗号运算符及逗号表达式。

```
#include <stdio.h>
void main( )
{   int c=5;
    printf("%d,%d,%d\n",c+=c++,c+8,++c);
    c=5;
    printf("%d\n",(c+=c++,c+8,++c));
    c=5;
    printf("%d\n",c+=c++,c+8,++c);
}
```

请读者结合前面所学的知识，分析该程序的运行结果。在不同的 C 编译环境下，其运行结果有所差异，在 Visual C++ 6.0 中的运行结果如下：

2. 逗号表达式的应用

(1) 用一个逗号表达式语句可代替多个赋值语句，如

```
a=0,b=1,c=2;
```

在语法上作为一条语句，它相当于下面的三条语句：

```
a=0;b=1;c=2;
```

(2) 用一个逗号表达式语句可得到多个计算结果，如

```
y=10,x=(y=y-5,60/y);
```

执行后，可同时得到 x 的值为 12，y 的值为 5。

(3) 当某些语法位置只允许出现一个表达式时，用逗号表达式可实现多个表达式的运算，例如第 4 章中将要介绍的 for 循环：

```
for(i=0,j=0;i<8,j<10;i++,j++)
```

中的 3 个语法位置：循环变量赋初值、循环终止条件判断和循环变量增值都只允许一个表达式，用逗号表达式可实现两个或多个表达式的运算。

3.4 关系运算和逻辑运算

在程序设计中，关系运算和逻辑运算通常用在程序流程控制中的 if、switch、for 或 while 语句中，目的是进行条件判断，以便程序能按照指定的流程运行。

1. 关系运算符

(1) 关系运算符有 6 种，即＞、＜、＞＝、＜＝、＝＝和！＝，分别表示大于、小于、大于或等于、小于或等于、等于和不等于，其中，表示大于或等于、小于或等于、等于和不等于的 4 种符号在书写形式上与人们日常使用的数学比较符略有不同。

(2) 关系运算符分为两个优先级，＞、＜、＞＝和＜＝处于同一优先级，＝＝和！＝处于同一优先级，前者的优先级高于后者，它们在整个 C 语言运算符集合中的运算优先顺序参见附录 B，结合性都是从左到右。

(3) 关系运算符主要用来对两个算术表达式或赋值表达式进行比较运算。

2. 关系表达式

(1) 用 6 个关系运算符中的一个连接两个算术表达式或赋值表达式，就构成了关系