第3章 顺序结构程序设计

程序里要对数据进行各种操作，其中进行各种运算操作是最基本的操作之一。在C语言程序中，使用表达式，也就是通常所说的计算式子，描述各种运算。表达式是由参与运算的数据和表示运算的符号，按照一定的规则组成的式子。描述运算的符号称为运算符，由一个或两个特定符号表示一种运算。

C语言具有丰富的运算符，可分为多种类型，包括如下：

(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(=、+=、-=、*=、/=、%=等)。

(6) 条件运算符(?:)。

(7) 逗号运算符(,)。

(8) 指针运算符(*和&)。

(9) 求字节数运算符(sizeof)。

(10) 强制类型转换运算符((类型))。

(11) 分量运算符(.、→)。

(12) 下标运算符([])。

(13) 其他(如函数调用运算符())。

C语言的运算符按照参与运算的数据个数，分为单目运算符(一个运算数据)、双目运算符(两个运算数据)和三目运算符(三个运算数据)；按照运算的先后次序，分为15个优先等级(见附录B)；另外还规定了运算符的结合性，即在相同优先级的运算符相邻时，是先计算左边的还是先计算右边的，先计算左边的是左结合，先计算右边的是右结合。除赋值运算符以外的双目运算符都是左结合的，赋值运算符单运算符和三目运算符都是右结合的。

本章主要讨论算术运算、赋值运算、自增自减运算的运算符，以及由它们构成的表达式，同时介绍各种运算符的优先级和结合性，最后讲解顺序结构程序设计方法。

3.1 算术运算和算术表达式

算术运算是最常用的运算，在大多数程序里都要进行。表3.1列出了C语言提供的算术运算符。

表 3.1　算术运算符

C语言中的操作	算术运算符	C语言中的操作	算术运算符
加法运算	+	除法运算	/
减法运算	-	取模运算(求余数)	%
乘法运算	*		

需要注意的是,C语言中使用的特殊符号,星号(*)表示乘号,斜杠(/)表示除号,百分号(%)表示求整数相除的余数。另外,加号(+)除了可以表示两个数相加外,还表示正号,例如,+5。类似地,减号(-)除了可以表示两个数相减外,还表示负号,例如,-12。

3.1.1　整数算术运算

如果参与运算的操作数都是整数,例如:

```
3+5,5-7,4*3,6/4,7%4
```

C语言规定,运算的结果一定是整数。也就是说,3+5 的结果是 8,5-7 的结果是-2,4*3的结果是 12,6/4 的结果是 1,7%4 的结果是 3。特别需要注意的是,整数除法运算和求余数运算与数学中的规定有所不同。

例如,3/5,结果为 0。

3%5,结果为 3(商 0 余 3)。

特别注意:

(1) 运算符"%"只能用于整数运算。

(2) a%b 的值为 0,则 a 能被 b 整除。

(3) a%b 的值在 0～b-1 之间。

(4) a%b 结果的符号与 a 相同。

这些性质在以后的程序中会经常用到。

思考:1+1/2+1/3+1/4+1/5 的运算结果是多少?

3.1.2　实数算术运算

如果参与运算的操作数都是实数,例如:

```
3.4+5.7,5.1-7.3,4.7*3.2,6.5/4.6
```

C语言规定,运算的结果一定是实数。也就是说,3.4+5.7 的结果是 9.1,5.1-7.3 的结果是-2.2,4.7*3.2 的结果是 15.04,6.5/4.6 的结果是 1.41。特别需要注意的是,实数不能使用运算符"%"。

思考:1.0+1.0/2.0+1.0/3.0+1.0/4.0+1.0/5.0 的运算结果是多少?

3.1.3 混合算术运算

如果参与运算的操作数一个是整数，另一个是实数，例如：

```
3+5.7,5.1-7,4.7*3,6/4.6
```

C 语言规定，运算的结果一定是实数。也就是说，3＋5.7 的结果是 8.7，5.1－7 的结果是－1.9，4.7 * 3 的结果是 14.1，6/4.6 的结果是 1.3。这种情况也不能使用运算符“%”。

思考：1＋1.0/2＋1.0/3＋1.0/4＋1.0/5 的运算结果是多少？

3.1.4 算术表达式

算术表达式是由参与算术运算的操作数(可以是常量、变量、函数等)、算术运算符和圆括号组成的符合 C 语言语法规则的式子。形式和数学中的算术表达式类似，但 C 语言中的算术表达式必须写成一行的形式，例如，数学中的$\frac{3}{5}$，在 C 语言中必须写成 3/5 的形式。

再例如，数学表达式$\frac{x1+x2+x3+x4}{5}$的 C 语言表达式是(x1＋x2＋x3＋x4)/5。

数学表达式 b^2-4ac 的 C 语言表达式是 b * b－4 * a * c。

数学表达式$\frac{a+b}{c-d}$的 C 语言表达式是(a＋b)/(c－d)。

数学表达式 πr^2 的 C 语言表达式是 3.1415926 * r * r(π 是常数，不可以写成符号 π)。

数学表达式$\frac{a}{x}+by$ 的 C 语言表达式是 a/x＋b * y。

3.1.5 算术表达式的计算规则

算术表达式按照运算符的优先规则从左到右计算，C 语言中算术运算符的优先规则和数学中的规定是一样的，即：

```
高：*   /   %
低：+-
```

例如，算术表达式 8－13/5＋4 * 8－7＋6%3，先从左到右计算 13/5、4 * 8、6%3，得到 8－2＋32－7＋0，再从左到右计算，得到 31。

和数学中的运算一样，算术表达式中可以有括号，但无论多少层，都只使用小括号。括号中的表达式优先级别是最高的，要先计算括号中的表达式。

例如，((8－13)/5＋4) * 8－(7＋6%3)，先计算(8－13)和(7＋6%3)，其中(7＋6%3)，先计算 6%3，再计算 7＋0，结果是 7；再计算(－5/5＋4)，再计算 3 * 8，再计算 24－

7，结果是 17，如图 3.1 所示。

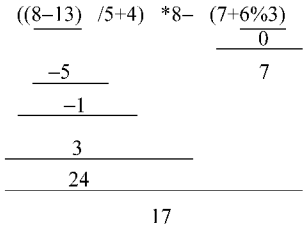

图 3.1　表达式计算过程示意图

3.2　赋值运算和赋值表达式

3.2.1　赋值运算符

1. 基本赋值运算

C 语言的赋值运算符是“=”，一般表达形式为

```
变量=表达式
```

其中，表达式可以是常量、变量、函数等。赋值运算是优先级很低的运算，赋值运算过程是：先计算赋值运算符(=)右边的表达式的值，然后将计算结果赋给赋值运算符(=)左边的变量。

这里一定要注意，“=”号左边一定要是一个变量，否则语法错误，赋值运算是改变变量取值的一个重要手段。

例如：

```
a=3                              //将 3 赋值给 a
b=sum/30                         //将 sum 的值除以 30 再赋值给 b
```

2. 赋值运算的类型转换问题

经常会遇到赋值运算符两侧的数据类型不一致的情况，这时在执行赋值运算时就要进行类型转换。转换时，以赋值运算符左侧的变量的类型为准。例如：

有定义

```
int a;
float x;
```

执行 a=45.78 时，a 的取值是 45。

执行 x=623 时，x 的取值是 623.000000。

3. 复合的赋值运算符

C 语言允许将形式为

变量=变量 算术运算符 表达式

的表达式简洁地写成

变量 算术运算符=表达式

注意：在双目算术运算符与赋值号之间不允许有任何空格，共有如表 3.2 所示的5 种算术复合赋值运算符。

表 3.2　算术复合赋值运算符

C 语言中的操作	复合赋值运算符	C 语言中的操作	复合赋值运算符
加赋值运算	+=	除赋值运算	/=
减赋值运算	-=	取余赋值运算	%=
乘赋值运算	*=		

例如：

```
a+=1    等同于 a=a+1                    //将 a 的当前值加上 1 后,再赋值给 a
x-=y+1  等同于 x=x-(y+1)
a*=b    等同于 a=a*b
x/=n+1  等同于 x=x/(n+1)
x%=10   等同于 x=x%10
```

C 语言还提供可以使用的 5 种位逻辑复合赋值运算符：<<=、>>=、&=、^=、|=。

3.2.2　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。

例如："x=5+6*a"是一个赋值表达式。如果 a 的值是 10，则 C 语言规定，表达式 5+6*a 的值为 65，变量 x 赋值后的值为 65，表达式 x=5+6*a 的值也为 65。

赋值运算符右侧的表达式，可以是任意合法的表达式，当然也可以是一个赋值表达式。例如：x=(y=15)，赋值运算符右侧括号内的"y=15"就是一个赋值表达式，它的值等于 15。执行表达式"x=(y=15)"，相当于执行"y=15"得到 15 后，再执行为 x 赋值的操作，结果 x 也取值 15。

赋值运算符是右结合性的运算符，即按照"自右而左"的结合顺序，因此，x=(y=15)与 x=y=15 等价。

赋值表达式也可以包含复合的赋值运算符。例如：a+=a*=a，假设 a 的初始值为 10，按照"自右而左"的结合顺序，先进行"a*=a"的运算，它相当于 a=a*a，a*a 的值为 10*10 等于 100，赋值给赋值运算符左侧的 a，a 的新值成为 100，也就是表达式 a*=a 的值是 100；再进行"a+=100"的运算，相当于 a=a+(100)，a 的最终值为 100+100=200。

作为表达式的一种，赋值表达式还可以出现在其他语句(如输出语句、循环语句等)

中。例如："printf("%f",x=y);"如果 y 的值为 23,先完成表达式 x=y 的运算,x 的值为 23,再输出表达式 x=y 的值,其值为 23。这样在一个语句中可以完成赋值和输出两种操作功能。

3.3 自增自减运算

在程序设计中经常会用到给变量加 1 或减 1 的运算,C 语言为此专门提供了两个运算符:自增运算符++和自减运算符--,如表 3.3 所示。

表 3.3 自增运算符和自减运算符用法表

C 语言中的操作	运算符	用法	含 义
自增运算	++	++n	先将 n 的值加 1,后使用 n 的值
		n++	先使用 n 的值,后将 n 的值加 1
自减运算	--	--n	先将 n 的值减 1,后使用 n 的值
		n--	先使用 n 的值,后将 n 的值减 1

自增运算符和自减运算符只对**单个变量**进行操作,称为单目运算符。常用于循环语句中使循环控制变量自动加 1 或减 1,也用于指针变量,使指针指向上一个或下一个地址。

对于++n 和 n++,单独使用时,意义相同,执行运算后,都是使变量 n 的值加 1。如果用在赋值语句中,意义有所不同,例如,假设有变量定义"int n,x,y;",并对变量 n 赋初始值 n=4,执行语句 x=++n 与执行语句 y=n++,变量 x 和 y 的值是不同的。

执行 x=++n:先将 n 的值加 1,再将 n 的值赋值给 x,因此 x 的值为 5。

执行 y=n++:先将 n 的值赋值给 y,再将 n 的值加 1,因此 y 的值为 4。

例 3.1 自增运算符的前置、后置对比。

```
#include "stdio.h"
int main()
{
    in   n,x,y;
    n=4;
    x=++n;
    printf("n=%d\tx=%d\n",n,x);
    n=4;
    y=n++;
    printf("n=%d\ty=%d\n",n,y);
    return 0;
}
```

程序的运行结果为

```
n=5     x=5
n=5     y=4
```

类似地，对于－－n 和 n－－，单独使用时，意义也相同，都是使变量 n 在原值的基础上减 1。但如果用在赋值语句中，意义就有所不同。

例 3.2 自减运算符的前置与后置对比。

```
#include "stdio.h"
int main()
{
    int n,x,y;
    n=4;
    x=--n;
    printf("n=%d\tx=%d\n",n,x);
    n=4;
    y=n--;
    printf("n=%d\ty=%d\n",n,y);
    return 0;
}
```

程序的运行结果为

```
n=3     x=3
n=3     y=4
```

如果用于 printf 函数的输出项，意义也有差别。

例 3.3 自增自减运算用在输出语句中。

```
#include "stdio.h"
int main()
{
    int n;
    n=4;
    printf("%d\t",n);
    printf("%d\t",++n);
    printf("%d\n\n",n);
    n=4;
    printf("%d\t",n);
    printf("%d\t",n++);
    printf("%d\n\n",n);
    n=4;
    printf("%d\t",n);
    printf("%d\t",--n);
    printf("%d\n\n",n);
    n=4;
    printf("%d\t",n);
    printf("%d\t",n--);
    printf("%d\n\n",n);
    return 0;
}
```

程序的运行结果为

```
4    5    5

4    4    5

4    3    3

4    4    3
```

3.4 优先级和类型转换

如果一个表达式中有多个运算符，那么运算的先后顺序就显得很重要。在C语言中，优先级顺序由符合标准数学用法的一系列排列规则决定，称为优先级法则。如果一个表达式中参与运算的数据类型不同，在运算前就需要进行类型转换，在C语言中有自动转换和强制转换两种方法。

3.4.1 优先级

在3.1节中已经介绍了由多个算术运算符构成的算术表达式中，运算符的优先级别由高到低是：圆括号、双目运算符乘除取余(*、/、%)、双目加减(+、-)，结合性是由左向右。

如果加入单目算术运算符++、--、+(正号)、-(负号)，则运算符的优先级别由高到低变为圆括号、单目算术运算符(++、--、+、-)、双目算术运算符乘除取余(*、/、%)、双目算术运算符加减(+、-)，其中单目算术运算符(++、--、+、-)的结合性是由右向左。

赋值运算符的优先级别低于所有算术运算符，结合性是由右向左。

上述规则用表格总结如表3.4所示。

表3.4 运算符的优先级和结合性

运 算 符	优先级别	结合性	类 型
()	高 ↑ 低	由左向右	圆括号
++、--、+、-		由右向左	单目算术运算符
*、/、%		由左向右	双目算术运算符乘、除、取余
+、-		由左向右	双目算术运算符加、减
=、+=、-=、*=、/=、%=		由右向左	赋值运算符

3.4.2 类型转换

1. 自动类型转换

在3.1、3.2节中都提到不同类型数据参与运算时，C语言采用**自动**类型转换的方式处理，转换处理隐含在计算过程中，将一种类型的数据转换为另一种兼容的类型。

例如，表达式2+4.5的计算，将整型数据2转换为double类型数2.0，再与4.5进行加法运算，这个转换是C编译系统自动完成的，不需程序员关心。

在有赋值运算符参与的表达式中，也可以进行自动转换。例如，如果有变量定义：double x，则赋值运算 x=45 的结果是变量 x 取值为 45.0，整型数据 45 自动转换为 double 类型数据。

注意：这时如果是将一个 double 类型数据赋值给一个整型变量，会出现截尾情况，例如，如果有变量定义：int a，执行赋值运算：a=123.756 的结果是变量 a 取值 123，小数部分被截去。

C语言自动类型转换规则如图3.2所示，当不同数据进行混合运算时，基本原则就是表达数据范围小的类型会自动转换为表达数据范围大的类型进行运算。

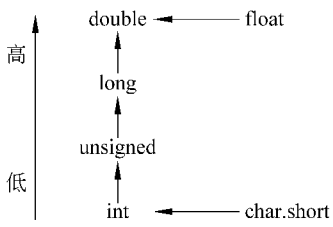

图 3.2 自动类型转换规则

2. 强制类型转换

C语言也允许根据需要按与自动类型转换不同的方式进行强制类型转换。

例 3.4 自动转换效果示例。

```
#include "stdio.h"
int main()
{
    int x,y;
    float ave;
    x=12 ;y=25 ;
    ave= (x+y)/2;
    printf("ave=%f\n",ave);
    return 0;
}
```

程序的运行结果为

```
ave=18.000000
```

其中表达式(x+y)/2，由于变量 x、y 都为整数，因此要依据整数除法的规则进行，按照C语言的运算规则，表达式的值为整数18，赋值给 float 型变量 ave，结果为18.0，所以在输出时得到了 ave=18.000000 的结果(Visual C 默认有6位小数)。

例 3.5 使用强制类型转换示例。

```
#include "stdio.h"
int main()
{
    int x,y;
    float ave;
    x=12 ;y=25 ;
    ave=(float)(x+y)/ 2;
    printf("ave=%f\n",ave);
    return 0;
}
```

程序的运行结果为

```
ave=18.500000
```

其中表达式(float)(x+y)/2 中,将整型表达式(x+y)强制转换为 float 类型,再按照自动转换规则进行实数除法的运算,表达式的值为 18.5。为了得到 18.5 的实型结果,程序中也可不对(x+y)进行强制转换,而将整数 2 写成 2.0,请读者自行尝试。

比如程序:

```
#include "stdio.h"
int main()
{
    double sum;
    sum=1+1/2+1/3+1/4+1/5;
    printf("sum=%f\n",sum);
    return 0;
}
```

中,由于所有的分数计算 1/2、1/3 等分子小于分母,分数结果为 0(整数除法结果),若 sum 计算改为如下形式:

```
sum=1+1/(double)2+1/(double)3+1/(double)4+1/(double)5;
```

虽然可以得到结果:sum=2.283333,但书写稍显烦琐,为此可将计算 sum 的语句写为

```
sum=1+1.0/2+1.0/3+1.0/4+1.0/5;
```

此外对于 x/y 这样的分数表达式,如果 x 和 y 都是整型变量,要想得到实型的商,可以使用下列表达之一:

```
1.0*x/y      x/1.0/y
```

强制类型转换的一般形式为

(类型名)表达式