第 3章　表　达　式

3.1 表达式基础

3.1.1 表达式的组成

1. 运算符

运算符又称操作符，是一个符号，它指示在一个或多个操作数上完成某种运算操作或动作。

C语言中，将除了输入、输出及程序流程控制操作以外的所有基本操作都作为运算处理，如赋值运算 “=”、逗号运算“，”、括号运算“( )”。

C语言的运算符按完成的运算操作可分为算术运算符、关系运算符、逻辑运算符、赋值运算符与其它运算符等，按参与运算符的操作数个数可分为单目运算符、双目运算符与三目运算符。

(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：=、+=、−=、*=、/=、%=。

(6) 条件运算符：？：。

(7) 逗号运算符：，。

(8) 指针运算符：*、&。

(9) 求字节数运算符：sizeof。

(10) 强制类型转换运算符：(类型)。

(11) 其它运算符；·、→、( )、[ ]等。

2. 操作数

操作数是运算符的操作对象, 可以是常量、变量、函数与表达式。

常量、变量、函数本身就是简单表达式, 从一般意义上讲, C语言中所有操作数都是表达式。复杂表达式由运算符连接简单表达式形成。

3.1.2 表达式的书写

C语言的表达式虽然来源于数学表达式, 是数学表达式在计算机中的表示, 但在书写时应该注意遵循C语言表达式书写的原则:

(1) C语言的表达式只能采用线性的形式书写。 例如:

① $\frac{1}{3}+i+j^3$ 应写成1/3+i+j*j*j。

② $\frac{a+b}{c+d}*e+f$ 应写成（a+b）/（c+d）*e+f。

(2) C语言的表达式只能使用C语言中合法的运算符和操作数, 对有些操作必须调用库函数完成, 而且运算符不能省略。例如:

① πr^2应写成3.14159*r*r。

② $\frac{1}{2}ab\sin a$ 应写成0.5*a*b*sin（alph）。

③ |x－y|应写成fabs（x－y）。

④ $y+2^x$ 应写成y+pow（2, x）。

3.1.3 表达式的分类

C语言表达式种类很多, 有多种分类方法。 我们一般依据运算的特征将表达式分为:

① 算术表达式, 如a+b*2.0-3.0/5.0。

② 关系表达式, 如x>=y, 关系表达式可以认为是逻辑表达式的特殊情况。

③ 逻辑表达式, 如（x>=2）&&（x<=8）。

④ 赋值表达式, 如a=b=c=1。

⑤ 条件表达式, 如（a>b）?a: b。

⑥ 逗号表达式, 如a=2, b=a*a, c=sqrt（b）。

1. 运算的优先级

运算的优先级是指运算执行的先后顺序。C语言将运算的优先级划分为15级，第1级优先级最高，第15级优先级最低。

初等运算( )、[ ]、→、·的优先级最高；

单目运算！、~、++、--、-、(类型)、*、&、sizeof的优先级次高；

算术运算+、-、*、/、%再次之；

然后是关系运算<、<=、==、!=、>、>=；

再然后是逻辑运算&&、||，条件运算式？：，赋值运算=、+=、-=、*=、/=、%=；

逗号运算优先级最低；

位运算优先级比较分散。

可将优先级顺序简单记忆为算术、关系和逻辑，移位、位逻辑在中间。请参阅附录B。

表达式求值时按运算的优先级别从高到低顺序进行，优先级相同的运算顺序由运算的结合性规定。

通过圆括号运算可以改变运算的优先顺序，先圆括号内，后圆括号外。

2. 运算符的结合性

运算符的结合性是指, 优先级相同的运算从左到右进行（左结合性）还是从右至左进行（右结合性）, 左结合性是人们习惯的计算顺序。

C语言中, 只有单目运算（！、 ～、 ++、 --、 -、 *、 &）、条件运算（？：）、赋值运算（=、 +=、 -=、 *=、 /=、 %=）的结合性是右结合, 其余运算为左结合。

3．类型转换

一般地，相应的运算只有相应类型的数据才能进行，不同类型数据进行运算时，要进行类型转换。

类型转换有自动转换与强制转换两种方式。

1) 自动转换

自动转换(又称隐含转换)由系统自动完成，转换的规则如图3-1所示。

图3-1中，横向箭头方向表示必定要进行的转换，即short型、char型数据必定先转换成int型，float型数据必定先转换为double型，再进行运算。纵向箭头方向表示类型自动转换的方向，请注意转换的方向是由低向高转换的，int型最低，double型最高。int型与unsigned型数据进行运算，int型转换成unsigned型；int型与long 型数据进行运算，int型转换成long型；int型与double型数据进行运算，int型转换成double型。其它类推。

这种转换是一种保值映射，在转换中数据的精度不受损失。

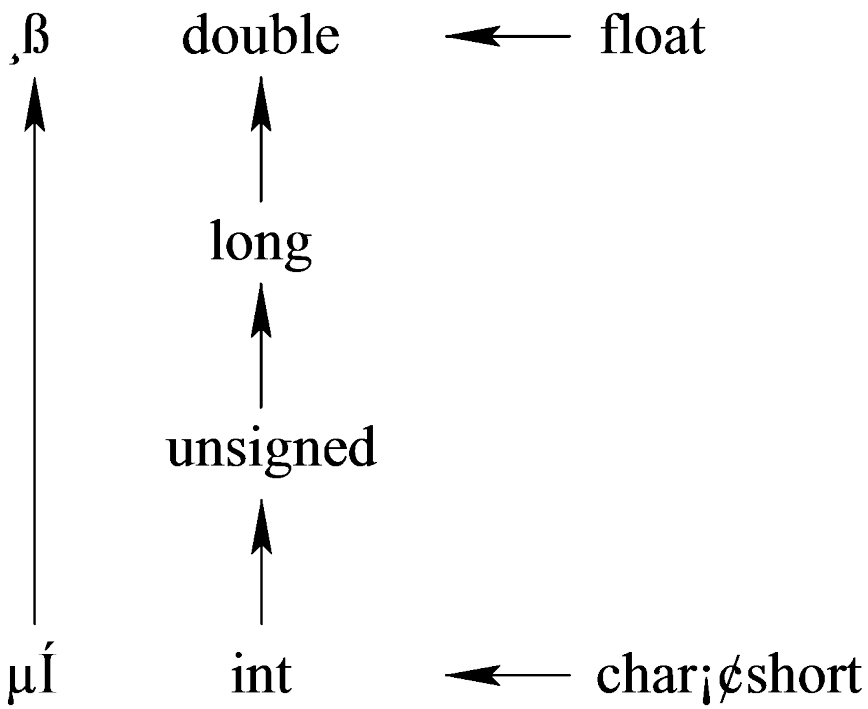

图3-1　类型自动转换规则

2) 强制转换

强制类型转换通过类型转换运算完成。

格式：

(类型名) (表达式)

强制转换将表达式的类型强制转换成类型名指定的类型。例如x、y为float，则(int) (x+y)将实型表达式x+y强制转换成整型。

值得注意的是，强制类型转换在将高类型转换为低类型时，数据精度会有损失，是一种不安全的转换。同时强制类型转换是一次性的、暂时性的，并不能永久改变所转换表达式的类型。例如a为int，则(double) a为双精度型，a依然为整型。

例3 –1 表达式计算。

(1) float x=2.5, y=4.7; int a=7;

计算x + a%3*（int）（x+y)%2/4

⑦ ① ④ ③ ② ⑤ ⑥

① a%3等于1。

② x+y等于7.2。

③（int）（x+y）等于7 。

④ a%3*（int）（x+y）等于7。

⑤ a%3*（int）（x+y）%2等于1。

⑥ a%3*（int）（x+y）%2/4等于0。

⑦ x+a%3*（int）（x+y）%2/4等于2.5+0, 结果为2.5。

(2) int a=2, b=3; float x=3.5, y=2.5;

计算（float）（a+b）/2 +（int）x%（int）y

②　①　③ ⑦ ④　⑥　⑤

① a+b等于5。

②（float）（a+b）等于5.0, 强制转换成float型。

③（float）（a+b）/2等于5.0/2.0, 结果为2.5。

④（int）x等于3, 强制转换成int型。

⑤（int）y等于2, 强制转换成int型。

⑥（int）x%（int）y等于1。

⑦（float）（a+b）/2+（int）x%（int）y等于2.5+1.0, 结果为3.5（双精度型）。

(3) int i=3; float f=1.0; double d=2.3;

计算　10 +'a'+i*f-d

①　　③②　④

① 'a'转换成97, 10+'a'等于107。

② i, f转换成双精度型, i*f等于3.0。

③ 107转换成双精度型, 10+'a'+i*f等于110.0。

④ 10+'a'+i*f-d等于107.7。

3.2 算术表达式

1. 自增运算

自增运算符：++。

自增运算是单目运算，操作数只能是整型变量，有前置、后置两种方式：

++i，在使用i之前，先使i的值增加1，俗称先增后用。

i++，先使用i的值，然后使i的值增加1，俗称先用后增。

例如：

i=1999；

j=++ i；/*先将i的值增1，变为2000后赋给j，j的值也为2000*/

j=i++；/*先将i的值赋给j，j的值为1999。然后将i的值增1，变为2000*/自增运算的优先级处于第2级，具右结合性。

2. 自减运算

自减运算符: --

自减运算与自增运算一样也是单目运算, 操作数也只能是整型变量。同样有前置、后置两种方式:

--i, 在使用i 之前 , 先使i 的值减1, 俗称先减后用。

i--, 先使用i 的值, 然后使i 的值减1, 俗称先用后减。

如：　i=2000;

j=--i; /*先减, 将i 的值减1, 变为1999。 后使用, j的值也为1999*/

j=i--; /*先使用, j的值为2000。 后减, 将i 的值减1, i的值变为1999*/自减运算的优先级、 结合性同自增运算。

请特别注意：

(1) 自增、自减运算只能用于变量，不能用于常量或表达式。

(2) 自增、自减运算比等价的赋值语句生成的目标代码更高效。

(3) 自增、自减运算常用于循环语句中，使循环控制变量自动加或减1；还可用于指针变量，使指针指向下一个或上一个地址。

(4) 表达式中包含有自增、自减运算时，特别容易出错，务请小心。

例如，i=1，(++i)+(++i)+(++i)的值为多少呢？可能计算出是9(=2+3+4)，其实这是错误的。实际上计算时先对整个表达式扫描，i 先自增3次，由1→2→3→4，因此计算结果应为12(=4+4+4)。

(i++)+(i++)+(i++)的值又是多少呢？分析同上，应为3，当然表达式计算完成后i的值同样改变为4。

3．运算符的组合问题

C语言的运算符一般为一个字符，有的由两个字符组成。在表达式中出现多个字符时如何组合呢？C编译系统在处理时，原则上尽可能自左至右将若干字符组合成一个运算符。

如将i+++j解释为(i++)+j，而不是i+(++j)。如要表示i+(++j)，必须加括号。标识符、关键字也按同样的原则进行处理。

例3 – 2 自增自减运算。

```
/*程序3 - 2 , 自增自减运算*/
# include "stdio.h"
main( )
{int i, j;
 i=j=5;
 printf（"i++=%d, j－－=%d\n", i++, j－－）；
 printf（"++i=%d, －－j=%d\n", ++i, －－j）；
 printf（"i++=%d, j－－=%d\n", i++, j－－）；
 printf（"++i=%d, －－j=%d\n", ++i, －－j）；
 printf（"i=%d, j=%d\n", i, j）；
}
```

运行结果：

i++=5，j--=5

++i=7，--j=3

i++=7，j--=3

++i=9，--j=1

i=9，j=1

3.3 赋值表达式

3.3.1 赋值运算

赋值运算符：=。

赋值运算是双目运算，一个操作数为变量，另一个操作数为表达式。赋值运算符左边的操作数只能是变量，右边可以是表达式。

进行赋值运算时先计算右边表达式的值，然后将右边表达式的值赋给左边变量，即送给左边变量对应的存储单元，并以此作为整个赋值表达式的值。例如i为int，i=3+5%2，则先计算右边表达式3+5%2的值，得到4，然后将4 赋给左边变量i 。

说明：

(1) 赋值运算的优先级为第14级，具有右结合性。

(2) 赋值运算中的表达式最简单的形式是一个常量，这时变量得到一个初值。

例如：

```
int i；float x；char ch；

i=100；x=12.345；ch='A'；
```

(3) 赋值运算中的表达式又可以是赋值表达式，如此可辗转赋值。

例如：

```
int  x，y，z;
```

x=y=z=0.0，相当于x=(y=(z=0.0))，即x、y、z都得到值0.0。

③　②①

(4) 赋值运算符“=”不同于数学中的等号。例如，i=i+1在数学中绝对不成立，但在C语言中肯定成立，其含义是将i 的当前值加上1再赋给i。

3.3.2 赋值类型转换

在进行赋值运算时，一般要求赋值运算符两侧操作数的类型相同或相容。如果赋值运算符两侧操作数的类型不一致，要进行类型转换。赋值运算时的类型转换以赋值运算左边操作数的类型为主体进行。

(1) 将实型数据赋给整型变量时，舍弃实数的小数部分。例如i为int，若有i=1.23，则i的值为1。

(2) 将整型数据赋给单、双精度型变量时，数值大小不变，但以浮点形式存储到变量中。

(3) 字符型数据赋给整型变量时，由于字符数据只占一个字节，而整型变量为2个字节，因此将字符型数据放在整型变量的低8位中，而对整型变量的高8位要进行扩充。

(4) 将基本整型数据赋给长整型变量时，基本整型数据放在长整型变量的低16位，高16位用符号位扩充。反之，将长整型数据赋给基本整型变量时，只将长整型数据的低16位送给基本整型变量。

① int　i=－1;

　　long　　int　　j;

　　j=i;

i 的二进制形式: 0000000000000001

j 的二进制形式:00000000000000000000000000000001

② int　i=−1；

　long　int　j；

　j=i；

i的二进制形式：1111111111111111

　　j的二进制形式：11111111111111111111111111111111

③ long int i =−1；

int j；

j=i；

i的二进制形式：11111111111111111111111111111111

j的二进制形式：1111111111111111

(5) 将无符号整型数据赋给长整型变量时，不存在符号位扩展的问题，只需将高位补0即可。将无符号整型数据赋给一个占字节数相同的整型变量时，将无符号整型数据原样送整型变量中，并将最高位当作符号位，如果数据超出相应的整型范围，将产生数据错误。如果将整型数据赋给占字节数相同的无符号整型变量时，也是原样照赋，最高位作数值处理。

例如：

① unsigned int i=65 535；

int j；

j=i；/* j的值为−1*/

② int i =−1；

unsigned int j；

j=i；/*j 的值为65 535*/

3.3.3 复合赋值运算

在基本赋值运算符“=”之前加上任一双目算术运算符及位运算符可构成复合赋值运算符, 又称带运算的赋值运算符。

算术复合赋值运算符: +=、 —=、 *=、 /=、 %=

位复合赋值运算符: &=、 |=、 ^=、 >>=、 <<=

一般形式: 变量☆=表达式

等价于: 变量=变量☆表达式

☆代表任一双目算术运算符或位运算符。

复合赋值运算先进行所带运算, 再进行赋值运算。 复合赋值运算的优先级同赋值运算。

例如：

① int a=3；

a+=2等价于a=a+2，结果为5。

② float x=1.2，y=2.1；

y*=x+3.4等价于y=y*(x+3.4)，结果为9.66。

③ int a=3，b=2；

b/=a+=1等价于b=b/(a=a+1)，结果为0。

例3-3 赋值运算。

```
/*程序3-3，赋值运算*/
#include "stdio.h"
main( )
{int i，j；
float  x，y；
  i=j=1；
  x=y=1.1；
printf("i=%d，j=%d\n"，i，j)；
```