第3章

运算符及表达式

思考题

1. C语言的算术表达式与数学中的算式一样吗？
2. 如何判断一个数是不是偶数？
3. 如何表示一个数比其他两个数都大？
4. 如何判断一个二进制数的次低位是1？

3.1 基本概念

计算机的基本功能就是计算，其他判断、智能都是基于计算的。在程序中，表示各种计算都离不开运算符或操作符(operator)和操作数(operand)。运算符表示进行何种运算，操作数是参与运算的数据。

例如，在下面计算圆周长的程序语句中：

```
c=2*PI*r;
```

其中“=”“*”是运算符，“c”“2”“PI”“r”是操作数。

C语言的运算符如表3-1所示。

表3-1 C语言的运算符

序号	运算符功能类别	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	=、+=、-=、*=、/=、%=、++、--、<<=等
6	条件运算符	?、:
7	逗号运算符	,
8	指针运算符	*、&

续表

序号	运算符功能类别	运　算　符
9	求字节数运算符	sizeof
10	强制类型转换运算符	(int)、(double)、(char)等
11	分量运算符	.、－＞
12	下标运算符	[]
13	其他	函数调用运算符()等

运算符、操作数、函数和圆括号按一定的规则顺序构成了表达式，表达式最终会计算得到一个数据结果。往往将这个结果赋值给一个变量，或者输出。

当程序中有多个运算符相连出现时，C 语言总是从左至右尽量多地将若干字符组成一个运算符。例如计算机会将 a＝b＋＋＋c 理解为 a＝(b＋＋)＋c 而不是 a＝b＋(＋＋c)。建议我们编程时加上括号以避免歧义，使程序更易读。

3.1.1　运算符分类

运算符分为单目、双目和三目运算符，表示参与进行该运算的操作数数量。绝大多数是双目运算符，有少数几个单目运算符和一个三目运算符。

三目运算符有 d1?d2:d3。由“?”和“:”共同构成，它们间隔开 3 个操作数。该运算符被称为条件运算符，根据第一个操作数 d1 的值，该运算得到不同的运算结果：如果 d1 是非 0，则运算结果是操作数 d2；如果 d1 是 0，则运算结果为操作数 d3。

单目运算符有：－(求反)、！(逻辑非)、～(位逻辑非)、＋＋(自加 1)、－－(自减 1)、sizeof(求长度)、指针运算符、强制类型转换运算符等。

运算符根据功能分为赋值运算符、算术运算符、关系运算符、逻辑运算符、位运算符等，下面将按此分类详细介绍主要的运算符。

3.1.2　运算符与数据类型

运算符对参与运算的操作数数量有要求，少数运算符对操作数的数据类型也有要求。如求余运算符(％)和位运算符，要求两个操作数都必须是整型；间接寻址单目运算符＊要求操作数是指针类型。

多数运算符对操作数无类型要求，当一个运算符的几个操作数类型不同时，不同运算符有不同的处理：

(1) 赋值运算符会将右边表达式的结果自动转换为赋值运算符左边的变量类型。

(2) 双目算术运算符会将操作数自动转换为整型或实型(大致是将精度低的类型转换为精度高的类型)，参见 2.7 节及图 2-6；三目运算符对操作数 2 和操作数 3 也进行这种转换。

(3) 其他运算符不作转换。

运算符的运算结果根据运算符的不同也具有特定的数据类型：

(1) 赋值运算结果为赋值运算符左边的变量类型；

(2) 双目算术运算结果为整型或实型(大致是二操作数中精度高的类型)，单目算术运算的结果类型是操作数的类型；

(3) 逻辑运算、关系运算和求长度运算结果的数据类型都是整型；

(4) 逗号运算的结果数据类型是最右边操作数的类型；

(5) 强制类型转换结果为要求强制转换到的类型；

(6) 取地址运算符 &. 的运算结果为指针，取值运算符 * 的运算结果为指针所指类型。

3.1.3 运算符的优先级与结合性

在一个表达式中有多个不同的运算符出现时，先进行哪个运算取决于各运算符的优先级。传统算术中有“先乘除后加减”的运算优先级规则。括号的优先级最高，它也是一种强制改变优先级的符号。

在一个表达式中有两个相同优先级的运算符相邻出现时，先进行左边的还是右边的运算符的运算取决于各运算符的结合性。如加法、减法运算符“＋”“－”是左结合，赋值运算符“＝”是右结合。因此表达式“3－2＋1”的结果是 2 而不是 0(假如是右结合，就先运行 2＋1)。例如运行下面两个语句后

```
int a,b=1;
a=b=2;
```

变量 a 的值将是 2，而不是 1(假如是左结合，就先运行 a＝b)。

所有运算符的优先级、结合性见附录 C。准确记住运算符的优先级、结合性对于分析程序是必修的，但对于编程又不是必修的。记住主要的运算符优先级可以使程序简洁，编程时记不清了可以通过括号准确规定优先级。

3.2 算术运算符

算术运算符有：＋(加)、－(减)、*(乘)、/(除)、%(求余)，它们在 C 语言中表示最基本的算术运算。乘、除、求余的优先级比加、减高，都是左结合。求余运算要求两个操作数都是整型数据。

由算术运算符连接起来组成的表达式是算术表达式。

注意：

(1) C 语言中没有幂运算符(乘方运算符)，幂运算可通过函数 power 实现。

(2) C 语言编程中处理除法运算容易出错。

当两个整数进行除法运算时，若不能整除，其结果也一定为整数而舍去小数部分，采取“向零取整”的方式舍去小数部分。例如：1/2 的结果为 0 而不是 1 或 0.5，－2/3 的结果为 0 而不是－1。

当两个数相除，若除数和被除数中有一个是实型，则进行实型数除法。因此建议编程时

遇到整型常量除法，一定将一个操作数加上小数点写为实型常量，如“1./2”而不是“1/2”。

比较：1/3*3.的结果为0，而1./3*3的结果近似为1。请自己分析原因。

(3) 求余运算“%”结果的正负，规定与被除数符号相同。

例如：5%3和5%-3的结果都是2，-5%3和-5%-3的结果都是-2。

3.3 赋值运算符

赋值运算符就是等号“=”，它是C语言程序中使用最多的运算符。它表示将其右边的表达式的结果赋值给其左边的变量，是一种数据传递。

赋值运算符优先级很低，仅高于简单列举的逗号运算符“,”。

注意：赋值运算符“=”不是比较是否相等，比较是否相等是关系运算符“==”。

赋值运算符的左边必须是变量或指针表示的内存空间(统称为左值操作数)。

注意理解语句

```
a=a+2;
```

它表示将变量a的值取出送到CPU，加上2后再赋值给变量a。即保存回原空间。

C语言中为这一类运算专门设计了运算符，称为复合赋值运算符，就是将算术运算符或位运算符与赋值运算符结合起来，表示将左值操作数的值取出进行算术或位运算后再赋值回原变量。复合赋值运算符有：+=、-=、*=、/=、%=、>>=、<<=、&=、^=、|=。

复合赋值运算符仍然是赋值运算符，优先级与赋值运算符一样。

例如下面语句执行后

```
01    int a=2,b=3;
02    a*=b+4;
```

变量a的值是14，而不是6，因为复合赋值运算“*=”的优先级比算术运算“+”要低。

更进一步设计了“++”和“--”表示自加1和自减1运算，但它们是单目运算符，优先级要高得多。如语句

```
01    i++;
02    ++a;
03    b--;
04    --c;
```

a++与++a在单独的语句中没有区别，但混合其他运算符时有区别：++a执行“a=a+1”，表达式的结果是执行“a=a+1”后的结果；a++是先将a的值取出暂存，最后将作为表达式的结果，然后执行“a=a+1”，最后将暂存值作为表达式的结果。例如，运行下面程序段

```
01    int a=1,b=1,c,d;
02    c=a++;          //表达式(a++)的值与变量a的值不同
03    d=++b;          //表达式(++b)的值与变量b的值相同
```

程序运行后，变量 a、b、c、d 的值分别为 2、2、1、2。

例 3.1 在下面的程序执行中，分析各变量中数据的变化。

```
#include<stdio.h>
main()
{ shorta=-1;                    //a 为-1,内存中是 11111111 11111111
  unsigned short b;
  int c=1,d=5,m=3,n;
  unsigned k;
  double x;
  n=3.7;                        //n 为 3
  x=1/2*100.;                   //x 为 0.0
  x=1./2*100;                   //x 为 500.0
  n=4/3.*100;                   //n 为 133
  n=(int)a;                     //e 为-1,占 4 字节
  b=(unsigned short)a;          //b 为 65535,内存中是 11111111 11111111
  k=(unsigned)a;                //k 为 4294967295,内存中是 ffff ffff
  m=c++,n=++d;                  //m 为 1,c 为 2,n 为 6,d 为 6
  m=5%-3,n=-5%-3;               //m 为 2,n 为-2
  a=b;                          //a 为-1,b 为 65535,内存都是 ffff
  m=b;                          //m 为 65535,4 字节;b 为 65535,2 字节
}
```

例 3.1 中表达式“k＝(unsigned)a;”是先将短整型 a 的 2 字节 16 位有符号整型数据进行符号位扩展到 4 字节，再按无符号数读出后赋值给无符号整型变量 k。“m＝b;”是先将无符号整型 b 的 2 字节 16 位无符号整型数据进行 0 扩展到 4 字节，再赋值给有符号整型变量 a。具体转换方法规定见 2.7 节。

变量和 const 常量可以在定义时赋值，称为初始化。const 常量不能在此之外再被赋值。

可以连续赋值，但此时必须是所有变量已被定义。

例 3.2 指出下面程序中的语法错误。

```
#include<stdio.h>
main()
{ int a=b=c=1;                  //错误,变量 b、c 未定义
  int m,n;
  int k=m=n=8;                  //正确,但不规范,不如全部定义后,再连续赋值
  double x,y,z;
  x=y=z=0.;                     //正确且规范
  m=x%3;                        //错误,求余运算%的操作数必须是整型,x 不是
  x/2++;                        //错误,x/2 不是左值操作数,是一个实型数据
  2=m;                          //错误,2 不是左值操作数,是一个整型常量
  y/2=z;                        //错误,y/2 不是左值操作数,是一个实型数据
}
```

3.4 关系运算符

计算机程序能够进行判断是因为可以进行比较，这种比较运算就是关系运算符。C 语言设计了 6 种关系运算符，都是双目运算符，都是左结合。

C 语言的 6 种关系运算符，按优先级可分为两组：

(1) 优先级较高的 4 种：>(大于)、>=(大于或等于)、<(小于)、<=(小于或等于)；

(2) 优先级相对低的 2 种：==(等于)、!=(不等于)。

由关系运算符连接起来的表达式即关系表达式，关系表达式的结果都是整型，只有两种结果：0 或 1，分别代表比较的逻辑结果假或真。

例 3.3 分析下面程序中各关系表达式的结果。

```
01    #include<stdio.h>
02    main()
03    { int a=1,b=5,c=-8,m=0;
04      double x=0.01;
05      a>=0;                  //关系表达式的结果为 1
06      m==0;                  //关系表达式的结果为 1
07      b==0;                  //关系表达式的结果为 0
08      b>=c;                  //关系表达式的结果为 1
09      x>m;                   //关系表达式的结果为 1
10      a>b+c;                 //关系表达式的结果为 1,算术优先
11    }
```

使用关系运算符要注意以下几点：

(1) 无论参与关系运算的操作数的类型如何，关系运算的结果只能是两个整型值之一：0(假，false)或 1(真，true)。C 语言没有逻辑类型，用整型表示。

(2) 关系运算符的“==”运算符(相等的比较)由两个等号“=”组成，注意与赋值运算符“=”区别开。

(3) 字符型数据按其 ASCII 码值的大小进行比较，但字符串常量不能直接用关系运算符比较。比如："ABCD"与"AAA"不能直接比较大小，应使用相关的字符串函数进行比较。

(4) 不要对实型数据作是否相等或不相等的比较，因为两个实型数据的有效数字位数不一定相同，难以作精确比较。否则可能出现无法预知误差而影响比较结果或出现与预期值背离的结果。应根据两个实型数据的差的绝对值是否小于某个比较小的数来近似判断其是否相等。

(5) 在 C 语言中，实现数学中的 $3 \leqslant x \leqslant 5$，需要分解为 $3 <= x$ 和 $x <= 5$ 两个关系运算，然后进行逻辑与运算。表达式“3<=x<=5”是合法的，但不是与期望的数学逻辑一致，而是先判断“3<=”，结果可能是 0 或 1，然后再与 5 进行比较，0 或 1 都小于 5，结果永远是 1(真)。反映数学逻辑 $3 \leqslant x \leqslant 5$ 正确的 C 语言表达式应该是“3<=x&&x<=5”。

3.5 逻辑运算符

仅靠关系运算无法表达复杂的逻辑关系，如大于 3 且小于 5、小于 3 或大于 5 等。C 语言为此设计了逻辑运算符。C 语言有 3 种逻辑运算符，优先级、结合性均有差别。由逻辑运算符连接起来组成的表达式是逻辑表达式，逻辑表达式的结果与关系表达式一样只能是两个整型值之一：0(假)或 1(真)。

(1) 逻辑与运算符：&&。例如 a&&b，若 a、b 均为非 0(真)，则 a&&b 结果为 1(真)；若 a、b 之一为 0(假)，则 a&&b 结果为 0(假)。优先级在 3 种逻辑运算中居中，左结合。

(2) 逻辑或运算符：||。例如 a||b，若 a、b 之一为非 0(真)，则 a||b 的结果为 1(真)；若 a、b 均为 0(假)，则 a||b 的结果为 0(假)。优先级在 3 种逻辑运算中最低，左结合。

(3) 逻辑非运算符：!。例如!a，若 a 为非 0(真)，则!a 的结果为 0(假)；若 a 为 0(假)，则!a 的结果为 1(真)。优先级在 3 种逻辑运算中最高，右结合。

在 C 语言中，逻辑运算的操作数并不要求是逻辑值，允许各种类型的数据参与逻辑运算。对各种数据的逻辑判断规则是：数据值为 0 判断为“假”，非 0 都判断为“真”。逻辑运算的结果“真”则保存为 1。

例如“3||2”合法，其结果为 1(true)。“8&&0”也合法，其结果为 0(false)。

若 a=4，b=5，则 a&&b 的值为 1(true)，因为 a 和 b 均为非 0，被当作“真”。

若 a=4，则!a 的结果为 0(false)，因为 a 的值为非 0 被当作“真”，对“真”进行非运算结果为“假”。

注意：在逻辑运算中，C 语言规定了两种快速处理：

(1) 对于逻辑表达式“(表达式 1) &&(表达式 2)”，如果表达式 1 的值已经是 0，则直接得到逻辑表达式的结果为 0，而不运行计算表达式 2。

(2) 对于逻辑表达式“(表达式 1) ||(表达式 2)”，如果表达式 1 的值已经是 1，则直接得到逻辑表达式的结果为 1，而不运行计算表达式 2。

以上两种快速处理对于表达式 2 中有赋值运算时会影响运行结果，分析程序时需要注意，编程则建议避免在表达式 2 中赋值。

例 3.4 将下列逻辑条件用 C 语言的表达式表示：

(1) 3 个边长数据 a、b、c 若能够构成三角形，表达式为真值，否则为假值。

解：能构成三角形的充要条件是三角形三条边中的任意两条边的边长之和大于第三条边。即

```
a+b>c && a+c>b && b+c>a
```

更清晰的写法：

```
(a+b)>c && (a+c)>b && (b+c)>a
```

(2) 判别一个字符(ch)是否是一个字母。

解：语句如下。

```
ch>='A' && ch<='Z'||ch>='a' && ch<='z'
```

(3) 判别某一年(year)是否为闰年。闰年的条件是符合下面两者之一：①能被 4 整除，但不能被 100 整除；②能被 400 整除。例如 2008、2000 年是闰年，2005、2100 年不是闰年。

解：语句如下。

```
(year % 4==0 && year % 100!=0) ||year % 400==0
```

当给定 year 为某一整数值时，如果上述表达式值为 1，则 year 是闰年，否则 year 不是闰年。

还可以用“!”运算作用于上述表达式，直接判别不是闰年：

```
!((year % 4==0 && year % 100!=0) ||year % 400==0)
```

这时，若表达式值为 1，则 year 不是闰年，否则 year 是闰年。

3.6 位运算符

前面介绍的运算符都是针对 1 字节以上长度的数据处理的，但在对开关量的控制中，二值数据的 1 位即可采集一个开关量的状况进行输出控制，这样需要对某 1 位或几位数据进行判断或处理，为此，C 语言设计了位运算符。位运算符又分为位逻辑运算符和移位运算符，其操作数要求必须是整型，结果也是整型。

3.6.1 位逻辑运算符

位逻辑运算符有按位求反(～)、按位与(&)、按位异或(^)和按位或(|)4 个。其中按位求反(～)是单目运算符，其余 3 个是双目运算符。

双目位逻辑运算是将两个整型操作数的每个对应二进制位分别进行逻辑运算，得到结果。单目位逻辑运算是对整型操作数的每个二进制位分别求反得到结果。

注意区别逻辑运算：逻辑运算对操作数类型没有要求，将操作数当作一个整体进行逻辑运算处理，根据操作数的值是 0 或非 0 得到结果。

例 3.5 分析下面程序的输出结果：

```
01   #include<stdio.h>
02   main()
03   { short a=-1,b=5;
04     unsigned short m=65535,n=0;
05     int k=3;
06     printf("%d,%d\n",(a&b),(a&&b));          //5,1
07     printf("%d,%d\n",(a|b),(a||b));          //-1,1
08     printf("%d,%d\n",(m&n),(m&&n));          //0,0
09     printf("%d,%d\n",(a&m),(a&&m));          //65535,1
10     printf("%d,%d\n",(a&k),(a&&k));          //3,1
11     printf("%d,%d\n",～a,!a);                //0,0
12     printf("%d,%d\n",～b,!b);                //-6,0
13     printf("%d,%d\n",～m,!m);                //-65536,0
14     printf("%d,%d\n",～n,!n);                //-1,1
15     printf("%d,%d\n",～k,!k);                //-4,0
16   }
```

解：分析 5 个变量的二进制值，位逻辑运算时，对短整型、无符号短整型先分别进行符号位扩展、0 扩展到 4 字节 32 位，然后按位进行逻辑运算，最后的结果按整型读出即得到结果。如图 3-1 所示是 5 个变量存储的二进制数据。

图 3-1 只画出了将整型变量 k 的低 16 位数、高 16 位数都设为 0。

用十六进制数表示，短整型变量 a 的数据－1 进行符号位扩展为 ffffffff，m0 扩展为 0000ffff，则 a&m 的结果为 0000ffff，按整型十进制读出为 65 535。

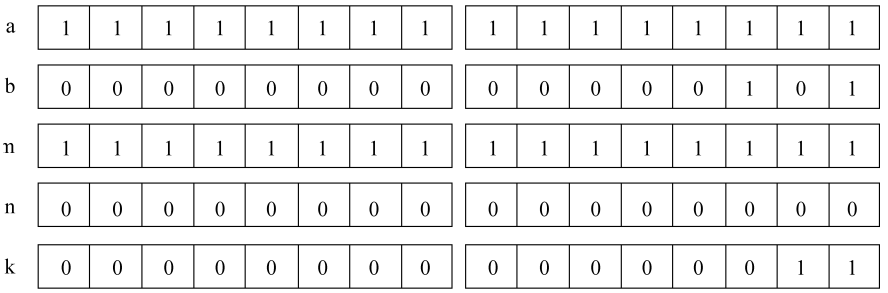

图 3-1　例 3.5 中 5 个变量存储的二进制数据

～m 的结果为 ffff0000，按整型读取这是补码，原码（对负数补码符号位不变，其余位求反，然后加 1）为 80010000，十进制结果为－65 536。

3.6.2　移位运算符

移位运算符有左移位运算符（<<）和右移位运算符（>>）两种，是双目运算符，要求两个操作数都是整型或无符号整型，结果也是整型或无符号整型。

a<<n，表示将整型数据 a 的二进制数各位依次左移 n 位，左边移出后舍弃，右边移入 0。如果 n<0，相当于左移 32＋n 位。效果大致相当于 a 乘以 2^n。

a>>n，表示将整型数据 a 的二进制数各位依次右移 n 位，右边移出后舍弃，整型数左边移入符号位，无符号数左边移入 0。如果 n<0，相当于右移 32＋n 位。效果大致相当于 a 除以 2^n，但小数部分向下取整。

3.6.3　位运算的应用

在自动控制中，经常需要根据某 1 位或几位的状态判断是否需要进行某种处理，而 C 语言的控制语句只能根据总体逻辑值（0 或 1）进行处理，因此需要将位状态转换为总体逻辑值。

例 3.6　设输入的值在整型变量 a 中，设 a 的最低位称为 a_0，向上各位依次称为 a_1，a_2，…，a_{31}，用 C 语言表达如下：

（1）判断其 a_3 位是否为 1。

解：(a&0b1000)或(a&8)。当 a 的 a_3 位为 1 时，表达式的值为非 0；否则表达式的值为 0。

（2）判断其 a_3 位是否为 0。

解：(～a&0b1000)或(～a&8)。当 a 的 a_3 位为 0 时，表达式的值为非 0；否则表达式的值为 0。

（3）判断其 a_3 位和 a_5 是否均为 1。

解：((a&0b1000)&&(a&0b100000))或((a&8)&&(a&32))或(a&40==40)。当 a 的 a_3 位和 a_5 位均为 1 时，表达式的值为 1；否则表达式的值为 0。

（4）判断其 a_3 位是否为 1 且 a_5 为 0、a_7 位为 1、a_8 位为 1。

解：((a&0b110101000)==0b110001000)或(a&424==392)。当 a 的 a_3 位为 1 且 a_5

位为 0、a_7 位为 1、a_8 位为 1 时，表达式的值为 1；否则表达式的值为 0。

通过位逻辑运算，屏蔽了不需要的位，将某 1 位或几位的状态转换为逻辑值，以便于后续的选择控制。

3.7 运算符的优先级

3.1.3 节已经介绍了运算符的优先级和结合性的概念，附录 C 详细列出了所有运算符的优先级和结合性。

为便于记忆，可将运算符的优先级大致归纳，如图 3-2 所示。

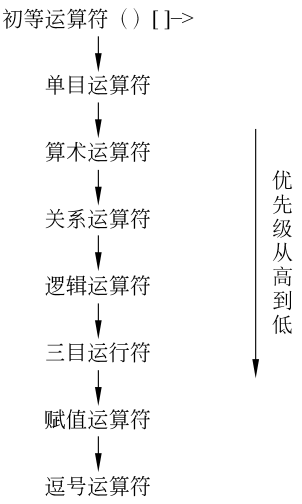

图 3-2 C 语言运算符优先级的粗略顺序

其中，逻辑非(!)、位求反(~)被认为是单目运算符的优先级，算术运算符中又细分先乘除后加减，算术运算符之后有一个移位运算符(<<或>>)，关系运算符中又细分先比较大小再比较相等，逻辑运算符前还有位逻辑运算符(先位与 &、再位异或^、最后位或|)，逻辑运算又细分为先与运算(&&)后或运算(||)。

运算符的结合性大多数是左结合，只有 3 类是右结合：单目运算符、三目运算符和赋值运算符。

3.8 小　　结

(1) 运算符是构成表达式、控制计算机进行何种运算或处理的核心。

(2) C 语言常用的运算符有：赋值、算术、关系、逻辑、位运算、条件、逗号运算符等。

(3) 运算符有优先级和结合性。

(4) 根据运算符构成表达式需要操作数的数目，分为单目、双目和三目运算符，多数是双目运算符。

(5) 运算符和操作数构成表达式,表达式完成运算后的结果是一个数据。

习 题 3

1. 分析下面 C 语言表达式的结果:

(1) 1/2 * 4.　(2) 4./8 * 100　(3) -5%-3　(4) 2&&1　(5) 2&1

(6) 2&6　(7) 2||1　(8) 2|1　(9) 2|6

2. 将下面各题的条件用 C 语言的关系或逻辑表达式表达:

(1) 整型变量 m 为偶数;

(2) 整型变量 n 为奇数;

(3) 变量 x 满足 3<x<5;

(4) 无符号整型变量 m 的最高位为 1 且最低位为 0;

(5) 三角形的三条边长分别为 a、b、c,该三角形是直角三角形。

3. 分别用一个 C 语言的表达式得到下面各题要求的数据:

(1) 无符号整型变量 d 的值的十进制数的个位数;

(2) 无符号整型变量 d 的值的十进制数的十位数;

(3) 无符号整型变量 d 的值的十进制数的百位数;

(4) 无符号整型变量 d 的值用八进制数表示时的最低位数字;

(5) 无符号整型变量 d 的值用八进制数表示时的次低位数字。

4. 指出下面程序语句中的错误:

```
01   main()
02   { int a,b,c;
03     double d1,d2,d3,d4,d5;
04     d1=4ac;
05     3+=a;
06     d2+d3=d4;
07     d2+d3==d4;
08     d5=d1%2;
09     a=d1&4;
10     d2=d3>>3;
11   }
```