**按位与运算符（&）**

参加运算的两个数据，按二进制位进行“与”运算。

运算规则：0&0=0;  0&1=0;   1&0=0;    1&1=1;

      即：两位同时为“1”，结果才为“1”，否则为0

例如：3&5  即 0000 0011& 0000 0101 = 00000001  因此，3&5的值得1。

另，负数按补码形式参加按位与运算。

“与运算”的特殊用途：

（1）清零。如果想将一个单元清零，即使其全部二进制位为0，只要与一个各位都为零的数值相与，结果为零。

（2）取一个数中指定位

方法：找一个数，对应X要取的位，该数的对应位为1，其余位为零，此数与X进行“与运算”可以得到X中的指定位。

例：设X=10101110，

   取X的低4位，用 X & 0000 1111 = 00001110 即可得到；

   还可用来取X的2、4、6位。

**按位或运算符（|）**

参加运算的两个对象，按二进制位进行“或”运算。

运算规则：0|0=0；  0|1=1；  1|0=1；   1|1=1；

     即 ：参加运算的两个对象只要有一个为1，其值为1。

例如:3|5　即 00000011 | 0000 0101 = 00000111  因此，3|5的值得7。

另，负数按补码形式参加按位或运算。

“或运算”特殊作用：

（1）常用来对一个数据的某些位置1。

方法：找到一个数，对应X要置1的位，该数的对应位为1，其余位为零。此数与X相或可使X中的某些位置1。

例：将X=10100000的低4位置1 ，用X | 0000 1111 = 1010 1111即可得到。

**异或运算符（^）**

参加运算的两个数据，按二进制位进行“异或”运算。

运算规则：0^0=0；  0^1=1；  1^0=1；   1^1=0；

   即：参加运算的两个对象，如果两个相应位为“异”（值不同），则该位结果为1，否则为0。

“异或运算”的特殊作用：

（1）使特定位翻转找一个数，对应X要翻转的各位，该数的对应位为1，其余位为零，此数与X对应位异或即可。

例：X=10101110，使X低4位翻转，用X ^0000 1111 = 1010 0001即可得到。

（2）与0相异或，保留原值 ，X ^ 00000000 = 1010 1110。

从上面的例题可以清楚的看到这一点。

**取反运算符（~）**

参加运算的一个数据，按二进制位进行“取反”运算。

运算规则：~1=0；  ~0=1；

     即：对一个二进制数按位取反，即将0变1，1变0。

使一个数的最低位为零，可以表示为：a&~1。

~1的值为1111111111111110，再按“与”运算，最低位一定为0。因为“~”运算符的优先级比算术运算符、关系运算符、逻辑运算符和其他运算符都高。

**左移运算符（<<）**

将一个运算对象的各二进制位全部左移若干位（左边的二进制位丢弃，右边补0）。

例：a = a<< 2将a的二进制位左移2位，右补0，

左移1位后a = a \*2;

若左移时舍弃的高位不包含1，则每左移一位，相当于该数乘以2。

**右移运算符（>>）**

将一个数的各二进制位全部右移若干位，正数左补0，负数左补1，右边丢弃。

操作数每右移一位，相当于该数除以2。

例如：a = a>> 2 将a的二进制位右移2位，

左补0 or 补1得看被移数是正还是负。

>> 运算符把*expression1* 的所有位向右移 *expression2*指定的位数。*expression1*的符号位被用来填充右移后左边空出来的位。向右移出的位被丢弃。

例如，下面的代码被求值后，*temp* 的值是 -4：

  -14 （即二进制的 11110010）右移两位等于 -4（即二进制的 11111100）。

 var temp = -14 >> 2

**无符号右移运算符（>>>）**

**>>>运算**符把 *expression1* 的各个位向右移*expression2* 指定的位数。右移后左边空出的位用零来填充。移出右边的位被丢弃。

例如：var temp = -14 >>>2

变量 *temp*的值**为** -14 （即二进制的 11111111 11111111 1111111111110010），向右移两位后等于 1073741820 （即二进制的 00111111 11111111 1111111111111100）。

**复合赋值运算符**

**位运算**符与赋值运算符结合，组成新的复合赋值运算符，它们是：

&=   例：a &=b       相当于a=a& b

|=   例：a |=b       相当于a=a |b

>>=  例：a >>=b      相当于a=a>> b

<<= 例：a<<=b      相当于a=a<< b

^=   例：a ^= b      相当于a=a^ b

运算规则：和前面讲的复合赋值运算符的运算规则相似。

**不同长度的数据进行位运算**

如果两个不同长度的数据进行**位运算**时，系统会将二者按右端对齐，然后进行**位运算**。

以“与”运算为例说明如下：我们知道在C语言中long型占4个字节，int型占2个字节，如果一个long型数据与一个int型数据进行“与”运算，右端对齐后，左边不足的位依下面三种情况补足，

（1）如果整型数据为正数，左边补16个0。

（2）如果整型数据为负数，左边补16个1。

（3）如果整形数据为无符号数，左边也补16个0。

如：long a=123;int b=1;计算a& b。

如：long a=123;int b=-1;计算a& b。

如：long a=123;unsigned intb=1;计算a & b。