## 位运算（&、|、^、~、>>、<<）

### 分类 [编程技术](https://www.runoob.com/w3cnote_genre/code" \o "编程技术)

### **1.位运算概述**

从现代计算机中所有的数据二进制的形式存储在设备中。即 0、1 两种状态，计算机对二进制数据进行的运算(+、-、\*、/)都是叫位运算，即将符号位共同参与运算的运算。

口说无凭，举一个简单的例子来看下 CPU 是如何进行计算的，比如这行代码：

int a = 35;int b = 47;int c = a + b;

计算两个数的和，因为在计算机中都是以二进制来进行运算，所以上面我们所给的 int 变量会在机器内部先转换为二进制在进行相加：

35: 0 0 1 0 0 0 1 147: 0 0 1 0 1 1 1 1————————————————————82: 0 1 0 1 0 0 1 0

所以，相比在代码中直接使用(+、-、\*、/)运算符，合理的运用位运算更能显著提高代码在机器上的执行效率。

### **位运算概览**

| **符号** | **描述** | **运算规则** |
| --- | --- | --- |
| & | 与 | 两个位都为1时，结果才为1 |
| | | 或 | 两个位都为0时，结果才为0 |
| ^ | 异或 | 两个位相同为0，相异为1 |
| ~ | 取反 | 0变1，1变0 |
| << | 左移 | 各二进位全部左移若干位，高位丢弃，低位补0 |
| >> | 右移 | 各二进位全部右移若干位，对无符号数，高位补0，有符号数，各编译器处理方法不一样，有的补符号位（算术右移），有的补0（逻辑右移） |

### **3.按位与运算符（&）#**

**定义**：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

总结：两位同时为1，结果才为1，否则结果为0。

例如：3&5 即 0000 0011& 0000 0101 = 0000 0001，因此 3&5 的值得1。

注意：负数按补码形式参加按位与运算。

**与运算的用途：**

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）取一个数的指定位

比如取数 X=1010 1110 的低4位，只需要另找一个数Y，令Y的低4位为1，其余位为0，即Y=0000 1111，然后将X与Y进行按位与运算（X&Y=0000 1110）即可得到X的指定位。

3）判断奇偶

只要根据最未位是0还是1来决定，为0就是偶数，为1就是奇数。因此可以用if ((a & 1) == 0)代替if (a % 2 == 0)来判断a是不是偶数。

4.按位或运算符（|）

定义：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

总结：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 0000 0011| 0000 0101 = 0000 0111，因此，3|5的值得7。

注意：负数按补码形式参加按位或运算。

或运算的用途：

1）常用来对一个数据的某些位设置为1

比如将数 X=1010 1110 的低4位设置为1，只需要另找一个数Y，令Y的低4位为1，其余位为0，即Y=0000 1111，然后将X与Y进行按位或运算（X|Y=1010 1111）即可得到。

5.异或运算符（^）

定义：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

总结：参加运算的两个对象，如果两个相应位相同为0，相异为1。

异或的几条性质:

* 1、交换律
* 2、结合律 (a^b)^c == a^(b^c)
* 3、对于任何数x，都有 x^x=0，x^0=x
* 4、自反性: a^b^b=a^0=a;

异或运算的用途：

1）翻转指定位

比如将数 X=1010 1110 的低4位进行翻转，只需要另找一个数Y，令Y的低4位为1，其余位为0，即Y=0000 1111，然后将X与Y进行异或运算（X^Y=1010 0001）即可得到。

2）与0相异或值不变

例如：1010 1110 ^ 0000 0000 = 1010 1110

3）交换两个数

## 实例

void Swap(int &a, int &b){  
    if (a != b){  
        a ^= b;  
        b ^= a;  
        a ^= b;  
    }  
}

6.取反运算符 (~)

定义：参加运算的一个数据，按二进制进行"取反"运算。

运算规则：

~1=0~0=1

总结：对一个二进制数按位取反，即将0变1，1变0。

异或运算的用途：

1）使一个数的最低位为零

使a的最低位为0，可以表示为：a & ~1。~1的值为 1111 1111 1111 1110，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。

7.左移运算符（<<）

定义：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）。

设 a=1010 1110，a = a<< 2 将a的二进制位左移2位、右补0，即得a=1011 1000。

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。

8.右移运算符（>>）

定义：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。

例如：a=a>>2 将a的二进制位右移2位，左补0 或者 左补1得看被移数是正还是负。

操作数每右移一位，相当于该数除以2。

10.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。