六种位运算符

C语言提供了六种位运算符:

& 按位与
| 按位或
^ 按位异或
~ 取反
<< 左移，相当与*2
>> 右移，正数高位补0，负数由计算机决定

循环左移k次 (x<<k) | (x >> (32-k)),

循环右移k次 (x>>k) | (x << (32-k))

当然常常应为优先级问题而犯错~~~

优先级及口诀如下

优先级别	运算符	记忆口诀

1	() [] . ->	括号成员第一； //括号运算符[]() 成员运算符. ->
		全体单目第二； //所有的单目运算符比如++、 --、 +(正)、 -(负) 、指针运算*、&乘除余三,加减四； //这个"余"是指取余运算即%
		移位五，关系六； //移位运算符：<< >> ，关系：> < >= <= 等
		等于(与)不等排第七； //即== 和!=
		位与异或和位或； //这几个都是位运算: 位与(&)异或(^)位或(|)
		"三分天下"八九十;
		逻辑或跟与； //逻辑运算符:|| 和 &&
		十二和十一； //注意顺序:优先级(||) 底于 优先级(&&)
		条件高于赋值， //三目运算符优先级排到13 位只比赋值运算符和","高
		逗号运算级最低! //逗号运算符优先级最低

2	！~ -（负号） ++ -- &（取变量地址） * （type）(强制类型) sizeof	
3	* / %	
4	+ -	
5	>> <<	
6	> >= < <=	
7	== !=	
8	&	
9	^	
10	|	
11	&&	
12	||	
13	?:	
14	= += -= *= /= %= |= ^= &= >>= <<=	
15	,	

按位与运算

按位与运算符"&"是双目运算符。其功能是参与运算的两数各对应的二进位相与。只有对应的两个二进位均为1时，结果位才为1，否则为0。参与运算的数以补码方式出现。
例如：9&5可写算式如下：
00001001 (9的二进制补码)
&00000101 (5的二进制补码)
00000001 (1的二进制补码)
可见9&5=1。
按位与运算通常用来对某些位清0或保留某些位。例如把a 的高八位清 0 ，保留低八位，可作a&255运算（255 的二进制数为0000000011111111）。

按位或运算

按位或运算符“|”是双目运算符。其功能是参与运算的两数各对应的二进位相或。只要对应的二个二进位有一个为1时，结果位就为1。参与运算的两个数均以补码出现。
例如：9|5可写算式如下：
00001001
|00000101
00001101 (十进制为13)
可见9|5=13

按位异或运算

按位异或运算符“^”是双目运算符。其功能是参与运算的两数各对应的二进位相异或，当两对应的二进位相异时，结果为1。参与运算数仍以补码出现，例如9^5可写成算式如下：
00001001
^00000101
00001100 (十进制为12)

求反运算

求反运算符~为单目运算符，具有右结合性。其功能是对参与运算的数的各二进位按位求反。例如~9的运算为：
~(0000000000001001)
结果为：1111111111110110

左移运算

左移运算符“<<”是双目运算符。其功能把“<< ”左边的运算数的各二进位全部左移若干位，由“<<”右边的数指定移动的位数，高位丢弃，低位补0。例如：
a<<4
指把a的各二进位向左移动4位。如a=00000011(十进制3)，左移4位后为00110000(十进制48)。

右移运算

右移运算符“>>”是双目运算符。其功能是把“>>”左边的运算数的各二进位全部右移若干位，“>>”右边的数指定移动的位数。例如：
设 a=15,
a>>2
表示把000001111右移为00000011(十进制3)。

注意：对于有符号数，在右移时，符号位将随同移动。当为正数时，最高位补0，而为负数时，符号位为1，

最高位是补0或是补1 取决于编译系统的规定。Turbo C和很多系统规定为补1。

简单运用

一：交换两个数（字符）,不用第三个变量就可以交换两个变量的值了：

用异或^,原理：两次异或能还原，即a = (a^b) ^ b

二：判断一个数是不是2的幂次：

原理：2的幂次的二进制表示中只有一位是1，其他位为0

x = x&(x-1)是让x的二进制码最右侧的1置为0，如果结果为0就表示原先x只有1位是1，其他位为0

inline bool is2pow(int x) { return (x&(x-1)==0 && (x!=0)); }

inline bool is2pow(int x) { return ((x&-x)==x); }

三：求一个整数有多少位是0：

原理同上。用x&(x-1)

```
int  count = 0;
while(x)
{
   ++count;
   x &= (x-1);
}
```

四：二进制快速求幂：

```
long pow(int x, unsigned int n){
    long p = 1;
    while (n){
        if (n & 1) p *= x;
        x *= x;
        n >>= 1;
    }
    return p;
}
```

五：判断奇偶数：

原理：奇数最后一位为1，偶数为0

inline bool odd(int x) { return x&1; }

inline bool even(int x) {return !(x&1); }

n%2 = n&1

n%4 = n&3

n%8 = n&7

……

六：求x绝对值：

原理：x为正数时不做改变，为负数时取反加1

x为正数时y = 0 = 0000 0000 0000 0000

x为负数时y = -1 = 1111 1111 1111 1111

跟0异或是本身，跟1异或是取反

```
1 inline int abs(int x){
2    int y = x >> 31;
3    return ( x^y-y);
4 }
```

七：对2的幂次取模：

原理：x&y取出x和y二进制位1的所有位。x^y>>1取出x,y只有一个二进制位1的并除以2

return (x&y) + (x^y)>>1);

不用位运算时注意 (x+y)/2,有可能会溢出。

x向上取整到y，其中y=2^n (字节对齐用)：

#define rund(x,y) (((x)+(y)-1)&~((y)-1))

八：其他：

只有第k位为1的数 1 << (k-1)

后k位为均为1的数 (1<<k)-1

x 的第k+1位 x >> k &1

x的第k+1位置1： x >> k |(1 << k)

x的第k+1位置0： x >> k &~(1 << k)

注意：左移1位再右移1位不一定时原先的值

至于高深用法可以戳戳这里：

如果这是你所爱的，就不要让自己后悔~~~