

# 位运算

## 复习：计算机的最小存储单位比特

计算机通过**二进制**编码的形式对信息进行存储。

二进制数：它由两个基本字符**0**，**1**组成，二进制数运算规律是**逢二进一**。

1个二进制位称为1比特(bit)

**1Byte == 8bit**

$$\begin{array}{r} 1001 \\ + 0101 \\ \hline 1110 \end{array}$$

1个二进制位在计算机中占的空间称为 1bit 或者叫做1位

1个英文字母由8个二进制位表示，称为1 Byte或 1字节

1Byte (字节) = 8bit (二进制位)

1024Byte = 1KB

1024KB = 1MB

1024MB = 1GB


1024GB = 1TB

1024TB = 1PB

类型	标识符	字节	数值范围
整型	int	4	-2 <sup>31</sup> 到2 <sup>31</sup> -1 (-2147483648 ~ 2147483647)
无符号整型	unsigned int	4	0到2 <sup>32</sup> -1 (0 ~ 4294967295)
短整型	short	2	-2 <sup>15</sup> 到2 <sup>15</sup> -1 (-32768 ~ 32767)
无符号短整型	unsigned short	2	0到2 <sup>16</sup> -1 (0 ~ 65535)
单精度实数	float	4	(正负) 3.4x10 <sup>-38</sup> 到3.4x10 <sup>38</sup>
双精度实数	double	8	(正负) 1.7x10 <sup>-308</sup> 到1.7x10 <sup>308</sup>

## 二进制转换为十进制

$$(101011)_2 = (43)_{10}$$


$$= (1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0)_{10}$$

十进制转换为二进制

$$(52)_{10} = (110100)_2$$

	余数
2   52	
2   26	0
2   13	0
2   6	1
2   3	0
2   1	1
0	1



# 什么叫位运算？ 概念：原码，补码，反码

有符号的数据类型(可表示正负数)，最高位是符号位

short占两个字节，共16个二进制位

short x=25; 原码 0000 0000 0001 1001

short y=-25; 原码 1000 0000 0001 1001

原码就是这个数字原始的二进制形式，只不过最高位表示符号。

y的反码：1111 1111 1110 0110  
将原码除符号位外其余的按位取反

y的补码,反码加1 1111 1111 1110 0111

x+y= 0000 0000 0001 1001 + 1111 1111 1110 0111  
= 0000 0000 0000 0000

计算机中，数字用补码的形式表示  
正数的补码就是它的原码  
负数的补码等于反码加1

## 观察下列数字对应的二进制形式

• 2	• 10	• 1	1
• 4	• 100	• 3	11
• 6	• 110	• 15	1111
• 16	• 10000	• 31	11111
• 32	• 100000		

告诉你一个小秘密，一般人我都不告诉它

**偶数的最后一位:0**

**奇数的最后一位:1**

想一想为什么呢

提示:逢2进1

int y; y=25 && 18; cout<<y; 结果是1 0表示false,非0的数表示true

```
int x;    x=25 & 18;    cout<<x;    结果是16
```

**"&"是位运算符，意思是“与”**      **"&"跟逻辑运算符"&&"不同**

$$(25)_{10} = (11001)_2 \qquad (18)_{10} = (10010)_2$$

$$25 \text{ \& } 18 = 16$$

$$\begin{array}{r} 11001 \\ \text{\textcolor{red}{\&}} 10010 \\ \hline 10000 \end{array}$$

Diagram illustrating the XOR operation. It shows three rows of binary digits: 1 1 1 0 0 1 0 0 1, 1 1 0 1 1 1 0 0 1, and 1 1 0 0 0 1 0 0 1. A red dashed box highlights the third column (the third bit from the left) across all three rows, showing the values 0, 1, and 0 respectively. A horizontal line is drawn below the third row.

都是1才取1



&

(0或1) & 1 =? 自己

(0或1) & 0 =? 0

**趁热打铁**

**偶数&1=?**

**奇数&1=?**

1 & 1 == 1

1 & 0 == 0

0 & 0 == 0

**判断奇偶以后不用写if(x%2==1)了吧?**

**if(x&1)cout<<"奇"; else cout<<"偶";**

"|"跟逻辑运算符"|"不同

"|"的意思是“或”

$$(25)_{10} = (11001)_2$$

$$(18)_{10} = (10010)_2$$

$$25 \mid 18 = 16$$

$$\begin{array}{r} 11001 \\ \mid 10010 \\ \hline 11011 \end{array}$$

$$(0 \text{ 或 } 1) \mid 1 = 1$$

$$(0 \text{ 或 } 1) \mid 0 = \text{自己}$$

	1	0	1	0	0	1	0	0	0
	0	1	0	0	1	1	0	0	1
<hr/>									
	1	1	1	0	1	1	0	0	1

只要有1就取1

$$1 \mid 1 == 1$$

$$1 \mid 0 == 1$$

$$0 \mid 0 == 0$$



"^"的意思是异或

位运算里面最难也最微妙的操作

$$(25)_{10} = (11001)_2 \quad (18)_{10} = (10010)_2$$

$$25 \wedge 18 = 16$$

$$\begin{array}{r} 11001 \\ \wedge 10010 \\ \hline 01011 \end{array}$$

(0或1) ^ 1 = 取反

(0或1) ^ 0 = 自己

$$\begin{array}{ccccccc} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \wedge & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ \hline 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}$$

同为0，异为1

$$1 \wedge 1 == 0$$

$$1 \wedge 0 == 1$$

$$0 \wedge 0 == 0$$

# 顺便一提

- 算一算：
- $2^1 = (10)_2^1 = ?$
- $6^1 = (110)_2^1 = ?$
- $10^1 = (1010)_2^1 = ?$
  
- $3^1 = (11)_2^1 = ?$
- $5^1 = (101)_2^1 = ?$
- 找到规律了吗

将偶数的最后一位强制变为1  
也就是变为与它相邻的奇数

将奇数的最后一位强制变为0  
也就是变为与它相邻的偶数

取反 “~”

short x=25; 0000 0000 0001 1001

~x=~25     1111 1111 1110 0110     恰好是-26的补码

-26的补码表示:    -26的原码=1000 0000 0001 1010

                     -26的反码=1111 1111 1110 0101

                     -26的补码= 1111 1111 1110 0110

所以~25 == -26

结论: ~x=-x-1

逻辑运算符

位运算符

&&

&

与 25 && 18 == 1

25 & 18 == 16

||

|

或 25 || 18 == 1

25 | 18 == 16

!

~

取反 !25 == 0

~25 == -26

^

异或 25 ^ 18 == 11

同为0, 异为1

~ 11001 == 00110

$(25)_{10} = (11001)_2$   $(18)_{10} = (10010)_2$

25 & 18 = 16

25 | 18 = 27

25 ^ 18 = 11

11001

& 10010

10000

11001

| 10010

11011

11001

^ 10010

01011

(0或1) & 1 = 自己

(0或1) & 0 = 0

(0或1) | 1 = 1

(0或1) | 0 = 自己

(0或1) ^ 1 = 取反

(0或1) ^ 0 = 自己

## 移位运算符"<<"与">>"

x=25; 00011001  
x<<2 01100100 == (100)<sub>10</sub>

$$X \ll Y == X * 2^Y$$

$$X \gg Y == X / 2^Y$$

x=25; 00011001  
x>>2 00000110 == (6)<sub>10</sub>

**对于正整数:**

**左移**

x<<y

把x的每个二进制位向左移动y位，移动造成的最右边的空位由0补足，最左边的数溢出。

**右移**

x>>y

把x的每个二进制位向右移动y位，移动造成的最左边的空位由0补足，最右边的数溢出。

位运算符: &, |, ^, ~, <<, >>

**结论:**

- $x \ll 1 = x * 2$

- $x \gg 1 = x / 2$

# 位运算符的应用

一个整数  $x \& 1$  的结果就是取  $x$  二进制的最末位。

一个数  $x \& (2^i)$  作用是？  
结果为0，表明  $x$  的右起往左第  $i+1$  位为0  
结果非0，表明  $x$  的右起往左第  $i+1$  位为1

$(x >> i) \& 1$  作用是？  
结果为0，表明  $x$  的右起往左第  $i+1$  位为0  
结果非0，表明  $x$  的右起往左第  $i+1$  位为1

"|" 运算通常用于二进制特定位上的无条件赋值，

例如一个数  $x | 1$  的结果就是把其二进制最末位强行变成1。

一个数  $x | (2^i)$  作用是？  
把  $x$  的右起往左第  $i+1$  位强制变成1

"^" 运算通常用于对二进制的特定一位进行取反操作

因为异或这样定义：0和1异或0都不变，异或1则取反。

一个数  $x \wedge (2^i)$  作用是？  
把  $x$  的右起往左第  $i+1$  位强制变反

$2^i = 1 << i$



## 位运算符的应用，有整数x

把x的末k位变为1

(10**1001** -> 10**1111**, k=4)

$$x \mid ((1 \ll k) - 1)$$

把x的末k位取反

(10**1001** -> 10**0110**, k=4)

$$x \wedge ((1 \ll k) - 1)$$

把x的右边连续的1变为0

((10010**1111** -> 10010**0000**)

$$x \& (x + 1)$$

把x的右边连续的0变为1

((10011**0000** -> 10011**1111**)

$$x \mid (x - 1)$$

把x的右边连续的1取出来  
((10010**1111**->00000**1111**)

$$x \& (x \wedge (x+1))$$

把x的末k位变为0  
(10**1001**->10**0000**,k=4)

$$x \& (x \wedge ((1 \ll k) - 1))$$

把x中与y相交的部分删除  
(

$$x \wedge (x \& y)$$

x=001101

y= 000101

处理结果: x= 001**000** )

## 位运算符的应用

```
int a=17,b=8;  
a=a ^ b;  
b=a ^ b;  
a=a ^ b;  
cout<<a<<" "<<b;
```

异或<sup>^</sup>的逆运算就是它本身，所以

```
a=a ^ b;  
b=a ^ b;  
a=a ^ b;
```

作用是交换a和b的值,只适用于正数

例：

给出一个小于 $2^{32}$ 的正整数。这个数可以用一个32位的二进制数表示（不足32位用0补足）。我们称这个二进制数的前16位为“高位”，后16位为“低位”。将它的高低位交换，我们可以得到一个新的数。试问这个新的数是多少（用十进制表示）。

例如，数1314520用二进制表示为0000 0000 0001 0100 0000 1110 1101 1000（添加了11个前导0补足为32位），其中前16位为高位，即0000 0000 0001 0100；后16位为低位，即0000 1110 1101 1000。将它的高低位进行交换，我们得到了一个新的二进制数0000 1110 1101 1000 0000 0000 0001 0100。它即是十进制的249036820。

```
unsigned int n;
```

```
cin>> n ;
```

```
cout<<( (n >> 16) | (n << 16) );
```

# 筷子问题

有 $n$  ( $n \leq 10,000,000$ ,  $n$ 为奇数)个整数，其中某个数字出现了奇数次，其他数字都出现了偶数次，请找出出现了奇数次的那个数字！ 空间限制  $O(1)$ , 时间限制  $O(n)$

例：

9

6 2 5 6 5 2 6 3 6

$$6 \wedge 2 \wedge 5 \wedge 6 \wedge 5 \wedge 2 \wedge 6 \wedge 3 \wedge 6 = ? \quad 3$$

```
int ans=0;
for(i=1;i<=n;i++)
{
    scanf("%d",&k);
    ans=ans^k;
}
printf("%d",ans);
```

# 对二的幂取模(求余)

$$x \bmod 2^y = ?$$

相当于取 $x$ 的后 $y$ 位

$$x \& ((1 \ll y) - 1)$$

例如  $13 \bmod 2^3 = 5$

$$1101 \& ((1 \ll 3) - 1)$$

$$1101 \& ((1000) - 1)$$

$$1101 \& 0111 = 0101 = (5)_{10}$$

## 练习

x是一个不大于 $2 \times 10^9$ 的整数，请将它转换为二进制数，统计该二进制数中1出现的个数。

例如

输入：10

输出：2

```
while ((x & -x) != 0)
```

```
{
```

```
    total++;
```

```
    x = x - (x & -x);
```

```
}
```

```
int main()
```

```
{
```

```
    int x, i, last, total = 0;
```

```
    cin >> x; //  $2 \times 10^9$ 以内的数的二进制不超过32位
```

```
    for(i = 1; i <= 32; i++)
```

```
    {
```

```
        last = x & 1; // 取x的二进制的最后一位
```

```
        if(last == 1) total++;
```

```
        x = x >> 1; // x的二进制数右移一位
```

```
    }
```

```
    cout << total;
```

```
    return 0;
```

```
}
```

```
while ((x & -x) != 0)
```

```
{
```

```
    total++;
```

```
    x = x - (x & -x);
```

```
}
```

x    0000 1010

x    0000 1000

-x   1111 0110

-x   1111 1000

&    0000 0010

&    0000 1000

**重要结论：**  $x \& -x$  作用是找出x中右起第一个1所在的位置  
也就是常说的lowbit(x)操作  
 $\text{lowbit}(x) = x \& -x$

$x = x - (x \& -x)$  相当于把x的右起第一个1减掉



# 趣味面试： NKOJ3099

有n个整数数列A，其中n-1个出现了3次，有1个出现了2次，找出出现1次那个数。

$1 \leq n \leq 10^6$        $0 \leq A_i \leq 2^{63}-1$

空间限制O(1)，时间限制O(n)

类似状态压缩，int a1=0,a2=0;

若a1=1010 表示右起第2和第4位上的1出现了1次。

若a2=0101 表示右起第3位上的1出现了2次。

(第3位上1出现的总次数为cnt,  $\text{cnt} \% 3 == 2$ )

# 趣味面试： NKOJ3099

```
int a1=0,a2=0;
for{int i=1; i<=n; i++}
{
    scanf ("%lld", &x) ;
    a2=(a2 | (a1&x) ) ;
    a1=(a1^x) ;
    int a3=(a1&a2) ;
    a2=(a2^a3) , a1=(a1^a3) ;
}
printf ("%lld", a1) ;
```

# 位运算习题

NKOI 3671,3672,3673,3674,2134,3099

思维: 3679, 3680