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

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

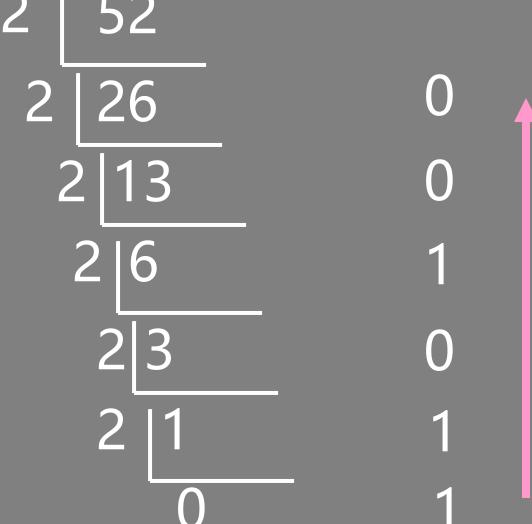
类型	标识符	字节	数值范围
整型	int	4	-2^{31} 到 2^{31} -1 (-2147483648 ~ 2147483647)
无符号整型	unsigned int	4	0 到 2 ³² -1 (0 ~ 4294967295)
短整型	short	2	-2^{15} 到 2^{15} -1 ($-32768 \sim 32767$)
无符号短整型	unsiged short	2	0到2¹⁶-1 (0 ~ 65535)
单精度实数	float	4	(正负) 3.4x10 ⁻³⁸ 到3.4x10 ³⁸
双精度实数	double	8	(正负) 1.7x10 ⁻³⁰⁸ 到1.7x10 ³⁰⁸

二进制转换为十进制

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

= $(1x2^5 + 0x2^4 + 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0)_{10}$

十进制转换为十进制



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 x=25; 原码 0000 0000 0001 1001

short y=-25; 原码 1000 0000 0001 1001

y的**反码**: **1**111 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

= 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 short占两个字节,共16个二进制位

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。

观察下列数字对应的二进制形式

• 2 • 10

4100

6110

16
 10000

32
 100000

• 1 1

• 3 11

151111

• 31 1

11111

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

什么叫位运算?

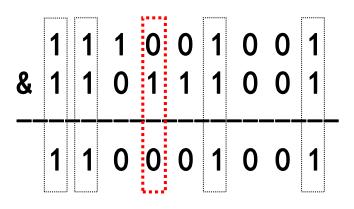
int y; y=25 & 18; cout < < y; 结果是1 0表示false,非0的数表示true

int x; x=25 <u>&</u> 18; cout < < x; 结果是16

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

11001 & 10010

10000



都是1才取1



趁热打铁 偶数&1=? 奇数&1=?

$$1&0 = = 0$$

$$0 = 0.80$$

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

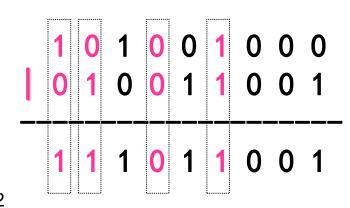
"|"的意思是"或"

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

11001

10010

11011



只要有1就取1

Λ

"^"的意思是异或

位运算里面最难也最微妙的操作

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

11001

^ 10010

01011

^	1 0	1	0	0	0	1	0	0	0
	1	0	1	0	1	0	0	0	1

同为0, 异为1

$$0^{0} = 0$$

顺便一提

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

-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

```
位运算符
 逻辑运算符
                      与 25 && 18==1 25 & 18==16
              &
   88
                      或
                          25 | 18==1 25 | 18==16
                      取反
                           !25==0
                                        ~25 == -26
                      异或 25 ^ 18==11
                                         同为0, 异为1
              Λ
 \sim 11001 = =00110
(25)_{10} = (11001)_2 (18)_{10} = (10010)_2
                                        25 ^ 18 = 11
   25 & 18 = 16
                     25 | 18 = 27
                                           11001
       11001
                        11001
                                         ^ 10010
     & 10010
                        10010
                        11011
                                           01011
        10000
                     (0或1) | 1 = 1
                                        (0或1) ^ 1 =取反
   (0或1) & 1 = 自己
                     (0或1) | 0=自己
                                        (0或1) ^ 0 =自己
   (0或1) & 0=0
```

移位运算符"<<"与">>>"

$$x=25$$
; 00011001
 $x<<2$ 01100100 ==(100)₁₀

$$X << 2$$
 $(01100100) == (100)_{10}$ $X << Y == X*2^{Y}$ $X >> Y == X/2^{Y}$

$$x>>2$$
 00000110 ==(6)₁₀

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: & , | , ^ , ~ , < < , > >

•
$$x < < 1 = x*2$$

•
$$x > 1 = x/2$$

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 & (2ⁱ)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为为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为为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 (2i)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 ^ (2i)作用是? 把x的右起往左第i+1位强制变反2i==1<<ii

位运算符的应用,有整数x

$$x^{(1<< k)-1)}$$

```
把x的右边连续的1取出来
                           x & (x^{(x+1)})
((100101111->000001111)
把x的末k位变为0
                         x & (x \wedge ((1 < < k) - 1))
(101001 -> 100000, k=4)
把x中与y相交的部分删除
                         x ^ (x & y)
        x = 001101
       y = 000101
处理结果: x= 001000 )
```

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^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2³²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

unsigned int n; cin>> n; cout<<((n >> 16) | (n << 16));

0100。它即是十进制的249036820。

筷子问题

有n(n<=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^{\circ}2^{\circ}5^{\circ}6^{\circ}5^{\circ}2^{\circ}6^{\circ}3^{\circ}6=? 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)

 $x \mod 2^y = ?$

相当于取x的后y位

x & ((1<<y)-1)

```
例如 13 \mod 2^3 == 5
1101 \& ((1<<3)-1)
1101 \& ((1000)-1)
1101 \& 0111 = 0101 = (5)_{10}
```

```
x是一个不大于2*10°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。
                    int main()
   例如
   输入: 10
                       int x,i,last,total=0;
   输出: 2
                      CIN > > X; // 2*10°以内的数的二进制不超过32位
                      for(i=1;i<=32;i++)
while ((x & -x)!=0)
                        last=x & 1; //取x的二进制的最后─位
                        if(last==1) total++;
  total++;
                        X=X>>1; //x的二进制数右移一位
  x = x - (x \& -x);
                    cout < < total;
                    return 0;
```

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 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<=n<=10⁶ 0<=Ai<=2⁶³-1 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, 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