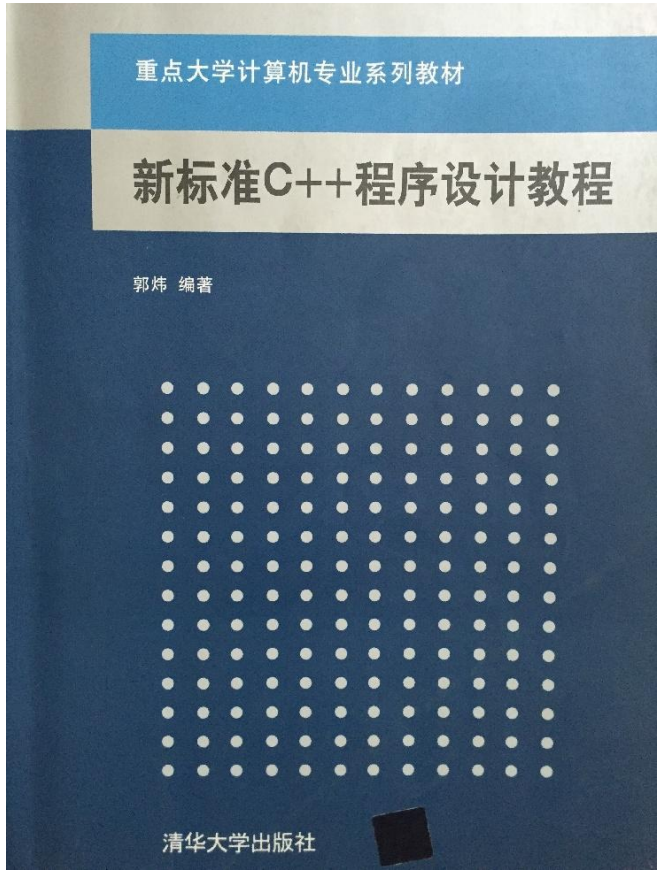
程序设计与算法(一)

李文新 郭炜

主讲教师互动微博：
http://weibo.com/guoweiofpku

指定教材:



位运算

(教材P28)

基本概念

位运算：

用于对整数类型（int,char, long 等）变量中的某一位(bit),
或者若干位进行操作。比如：

基本概念

位运算：

用于对整数类型（int,char, long 等）变量中的某一位(bit),
或者若干位进行操作。比如：

1）判断某一位是否为1

基本概念

位运算：

用于对整数类型（int,char, long 等）变量中的某一位(bit),
或者若干位进行操作。比如：

1）判断某一位是否为1
2）只改变其中某一位，而保持其他位都不变。

基本概念

位运算：

用于对整数类型（int,char, long 等）变量中的某一位(bit)，或者若干位进行操作。比如：

1）判断某一位是否为1

2）只改变其中某一位，而保持其他位都不变。

C/C++语言提供了六种位运算符来进行位运算操作：

&	按位与(双目)
|	按位或(双目)
^	按位异或(双目)
~	按位非(取反)(单目)
<<	左移(双目)
>>	右移(双目)

按位与“&”

将参与运算的两操作数各对应的二进制位进行与操作，只有对应的两个二进位均为1时，结果的对应二进制位才为1，否则为0。

按位与“&”

例如：表达式“21 & 18 ”的计算结果是16
(即二进制数10000)，因为：

按位与“&”

例如：表达式“21 & 18 ”的计算结果是16
(即二进制数10000)，因为：

21 用二进制表示就是：
0000 0000 0000 0000 0000 0000 0001 0101
18 用二进制表示就是：
0000 0000 0000 0000 0000 0000 0001 0010
二者按位与所得结果是：
0000 0000 0000 0000 0000 0000 0001 0000

按位与“&”

通常用来将某变量中的某些位清0且同时保留其他位不变。
也可以用来获取某变量中的某一位。

例如，如果需要将int型变量n的低8位全置成0，而其余位不变，则可以执行：

按位与“&”

通常用来将某变量中的某些位清0且同时保留其他位不变。
也可以用来获取某变量中的某一位。

例如，如果需要将int型变量n的低8位全置成0，而其余位不变，则可以执行：

```
n = n & 0xffffff00;
```

按位与“&”

通常用来将某变量中的某些位清0且同时保留其他位不变。
也可以用来获取某变量中的某一位。

例如，如果需要将int型变量n的低8位全置成0，而其余位不变，则可以执行：

```
n = n & 0xffffff00;
```

也可以写成：

```
n &= 0xffffff00;
```

按位与“&”

通常用来将某变量中的某些位清0且同时保留其他位不变。
也可以用来获取某变量中的某一位。

例如，如果需要将int型变量n的低8位全置成0，而其余位不变，则可以执行：

```
n = n & 0xffffff00;
```

也可以写成：

```
n &= 0xffffff00;
```

如果n是short类型的，则只需执行：

```
n &= 0xff00;
```

按位与“&”

如何判断一个int型变量n的第7位（从右往左，从0开始数）是否是1 ？

按位与“&”

如何判断一个int型变量n的第7位（从右往左，从0开始数）是否是1 ？

只需看表达式 “n & 0x80”的值是否等于0x80即可。

0x80： 1000 0000

按位或“|”

将参与运算的两操作数各对应的二进制位进行或操作，只有对应的两个二进位都为0时，结果的对应二进制位才是0，否则为1。

按位或“|”

例如：表达式“21 | 18 ”的值是23，因为：

按位或“|”

例如：表达式“21 | 18 ”的值是23，因为：

21：　0000 0000 0000 0000 0000 0000 0001 0101
18：　0000 0000 0000 0000 0000 0000 0001 0010
21|18：0000 0000 0000 0000 0000 0000 0001 0111

按位或“|”

按位或运算通常用来将某变量中的某些位置1且保留其他位不变。

例如，如果需要将int型变量n的低8位全置成1，而其余位不变，则可以执行：

按位或“|”

按位或运算通常用来将某变量中的某些位置1且保留其他位不变。

例如，如果需要将int型变量n的低8位全置成1，而其余位不变，则可以执行：

```
n |= 0xff;
```

0xff：1111 1111

按位异或“^”

将参与运算的两操作数各对应的二进制位进行异或操作，即只有对应的两个二进位不相同时，结果的对应二进制位才是1，否则为0。

例如：表达式“21 ^ 18 ”的值是7(即二进制数111)。

21:　　0000 0000 0000 0000 0000 0000 0001 0101
18:　　0000 0000 0000 0000 0000 0000 0001 0010
21^18: 0000 0000 0000 0000 0000 0000 0000 0111

按位异或“^”

按位异或运算通常用来将某变量中的某些位取反，且保留其他位不变。

例如，如果需要将int型变量n的低8位取反，而其余位不变，则可以执行：

```
n ^= 0xff;
```

0xff：1111 1111

按位异或 “^”

异或运算的特点是：

如果 a^b=c，那么就有 c^b = a以及c^a=b。（穷举法可证）

此规律可以用来进行最简单的加密和解密。

按位异或“^”

另外异或运算还能实现不通过临时变量，就能交换两个变量的值：

```
int a = 5, b = 7;
a = a ^ b;
b = b ^ a;
a = a ^ b;
```

即实现a,b值交换。穷举法可证。

按位非“~”

按位非运算符“~”是单目运算符。
其功能是将操作数中的二进制位0变成1，1变成0。

例如，表达式“~21”的值是整型数 0xffffffea

21:　0000 0000 0000 0000 0000 0000 0001 0101
~21:　1111 1111 1111 1111 1111 1111 1110 1010

左移运算符“<<”

表达式：

a << b

的值是：将a各二进位全部左移b位后得到的值。左移时，高位丢弃，低位补0。a 的值不因运算而改变。

左移运算符“<<”

例如：
9 << 4

9的二进制形式：
0000 0000 0000 0000 0000 0000 0000 1001
因此，表达式“9<<4”的值，就是将上面的二进制数左移4位，得：
0000 0000 0000 0000 0000 0000 1001 0000
即为十进制的144。

左移运算符“<<”

实际上，左移1位，就等于是乘以2，左移n位，就等于是乘以2^n。而左移操作比乘法操作快得多。

右移运算符“>>”

表达式：

a >> b

的值是：将a各二进位全部右移b位后得到的值。右移时，移出最右边的位就被丢弃。 a 的值不因运算而改变。

右移运算符“>>”

表达式：

a >> b

的值是：将a各二进位全部右移b位后得到的值。右移时，移出最右边的位就被丢弃。 a 的值不因运算而改变。

对于有符号数，如long, int, short, char类型变量，在右移时，符号位（即最高位）将一起移动，并且大多数C/C++编译器规定，如果原符号位为1，则右移时高位就补充1，原符号位为0，则右移时高位就补充0。

右移运算符“>>”

实际上，右移n位，就相当于左操作数除以2^n，并且将结果往小里取整。

−25 >> 4 = −2
−2 >> 4 = −1
18 >> 4 = 1

```
#include <stdio.h>
int main()
{
	int n1 = 15;
	short n2 = -15;
	unsigned short n3 = 0xffe0;
	char c = 15;
	n1 = n1>>2;
	n2 >>= 3;
	n3 >>= 4;
	c >>= 3;
	printf( "n1=%d,n2=%x,n3=%x,c=%x",n1,n2,n3,c);
}
```

```
#include <stdio.h>
int main()
{
        int n1 = 15;
        short n2 = -15;
        unsigned short n3 = 0xffe0;
        char c = 15;
        n1 = n1>>2;
        n2 >>= 3;
        n3 >>= 4;
        c >>= 3;
        printf( "n1=%d,n2=%x,n3=%x,c=%x",n1,n2,n3,c);
} //输出结果是：n1=3,n2=fffffffe,n3=ffe,c=1
```

n1: 0000 0000 0000 0000 0000 0000 0000 1111

n1 >>= 2: 变成3

0000 0000 0000 0000 0000 0000 0000 0011

n2: 1111 1111 1111 0001

n2 >>= 3: 变成 fffe, 即-2

1111 1111 1111 1110

n3: 1111 1111 1110 0000

n3 >>= 4: 变成 ffe

0000 1111 1111 1110

c: 0000 1111

c >>= 3; 变成1

0000 0001

```
int n1 = 15;
short n2 = -15;
unsigned short n3 = 0xffe0;
char c = 15;
```

思考题：

有两个int型的变量a和n(0 <= n <= 31)，
要求写一个表达式，使该表达式的值和a的第n位相同。

思考题：

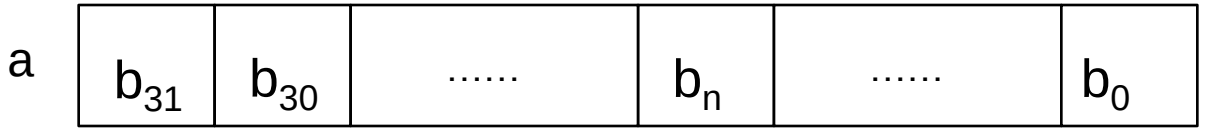
有两个int型的变量a和n(0 <= n <= 31)，
要求写一个表达式，使该表达式的值和a的第n位相同。

答案：

```
( a >> n ) & 1
```

思考题：

有两个int型的变量a和n(0 <= n <= 31)，
要求写一个表达式，使该表达式的值和a的第n位相同。



答案：

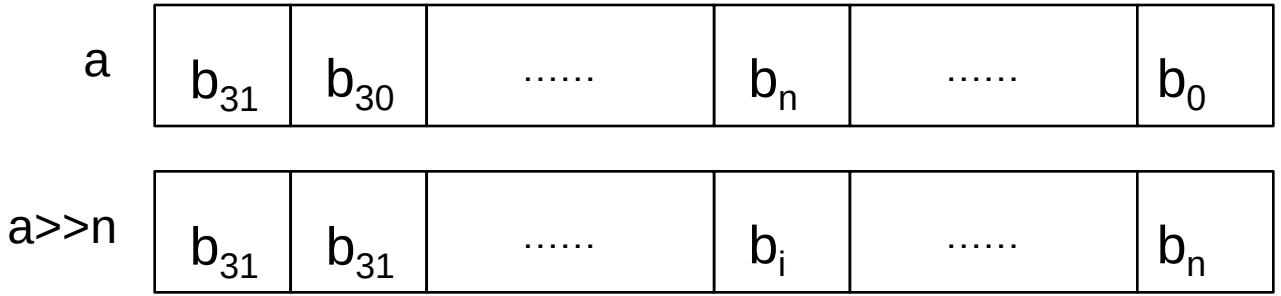
(a >> n) & 1

思考题：

有两个int型的变量a和n(0 <= n <= 31)，
要求写一个表达式，使该表达式的值和a的第n位相同。

答案：

(a >> n) & 1



思考题：

有两个int型的变量a和n(0 <= n <= 31)，
要求写一个表达式，使该表达式的值和a的第n位相同。

答案：

(a >> n) & 1

a	b_{31}	b_{30}	……	b_n	……	b_0
a>>n	b_{31}	b_{31}	……	b_i	……	b_n
(a>>n) & 1	0	0	……	0	……	b_n

思考题：

有两个int型的变量a和n(0 <= n < 31)，
要求写一个表达式，使该表达式的值和a的第n位相同。

另一答案：

```
(a & (1 << n )) >> n
```