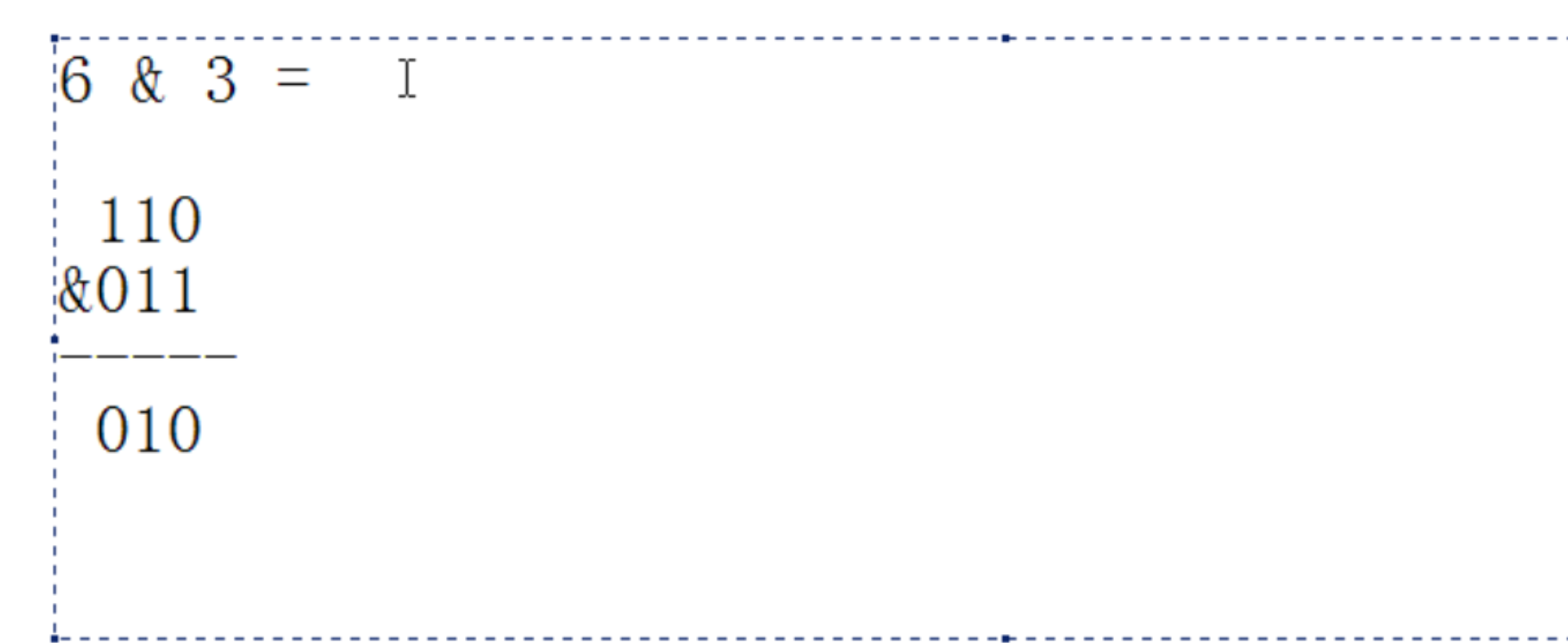
# &--与运算符，判断一个数是奇数还是偶数，下面有例子



0为假，1为真。先看最后一位计算，0与1，为0；只有1与1为1；其他的都是0。

上面的计算结果为2

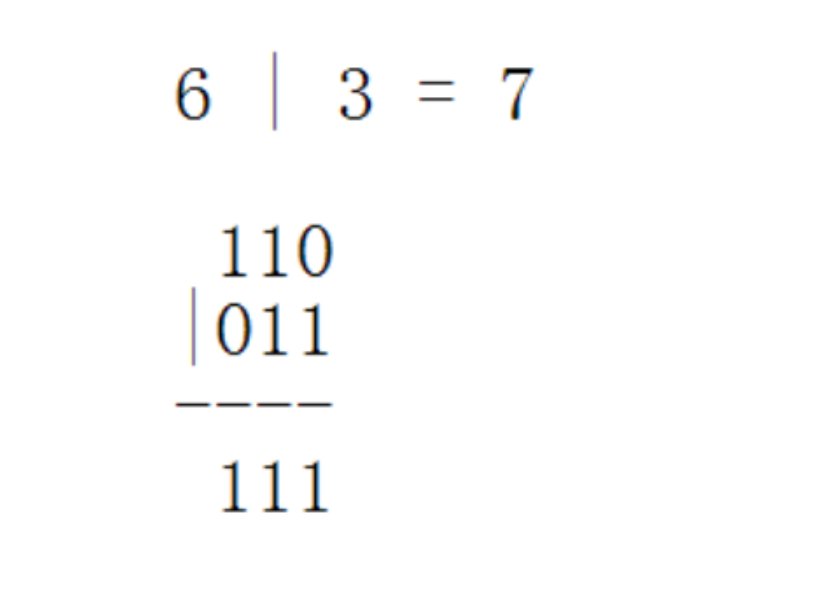
小技巧：例如与011，那么相当于只有最后两位有效，那么就取110的最后两位。

# |--或运算符

与上面的刚好相反

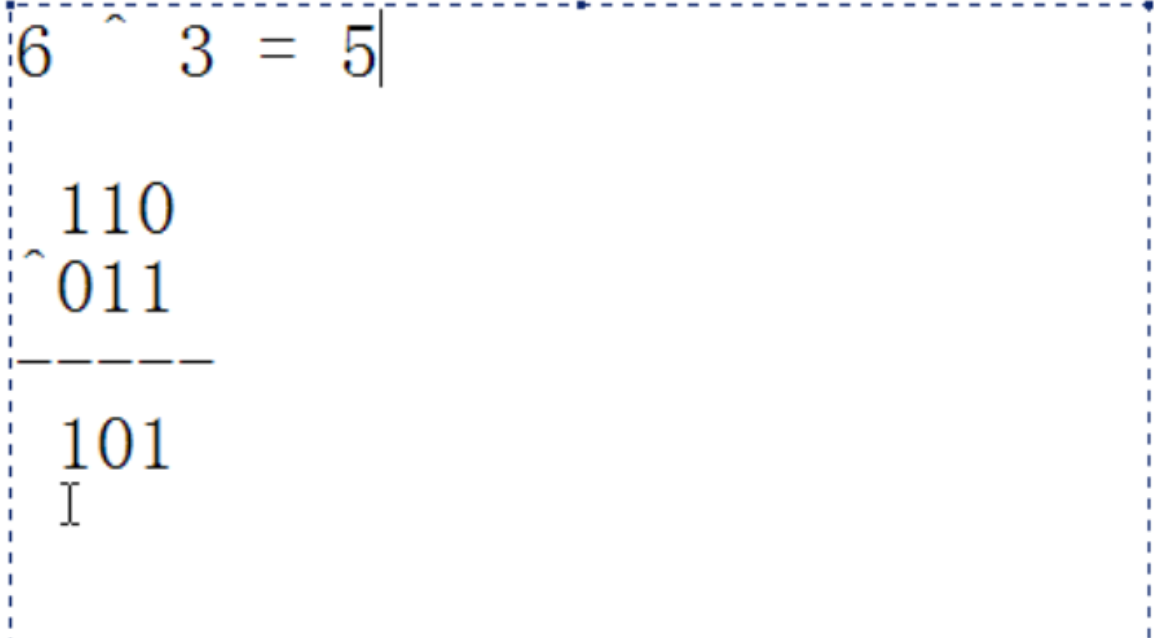
只有都为0才为0，只要有一个是1结果就是1.

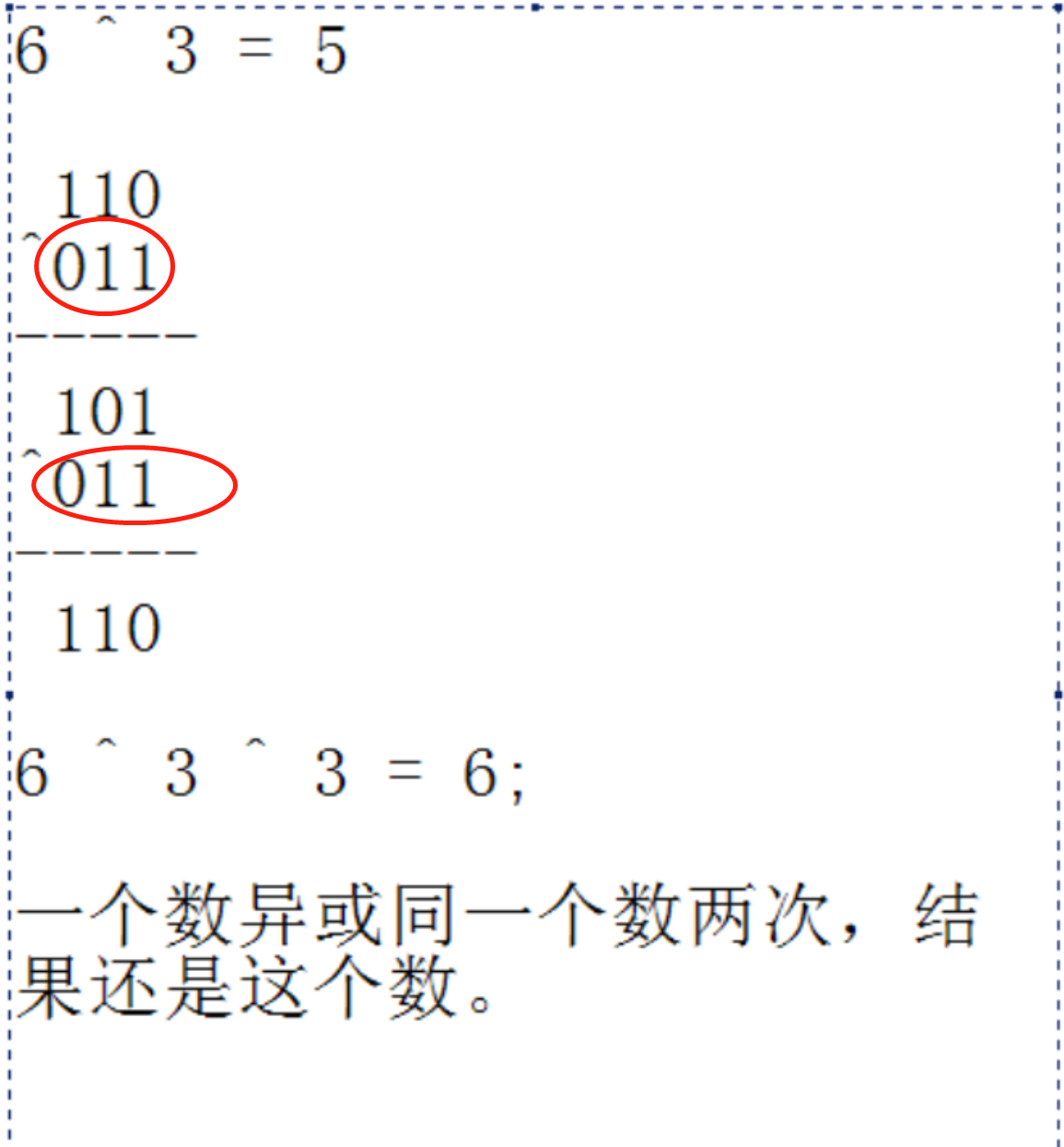
所以6|3=111（7），结果就是7。



# ^异或运算符，不使用第三方变量交换两个整数变量的值，下面有例子

相同为假，不同为真。相同为0，不同为1。





应用：加密算法

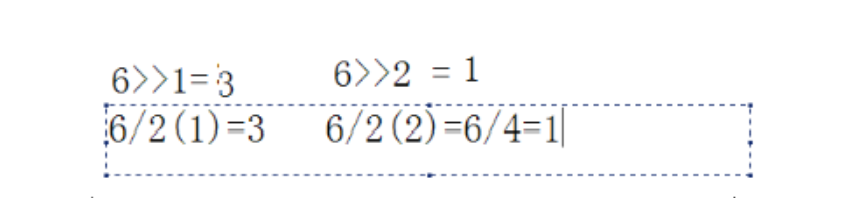
# <<左移运算符

就是将一个数转换成二进制后，向左移位，高位舍弃，低位补零。

A左移N位，结果是A\*（2的N次方），十进制左移一位相当于乘以10.

# >>右移运算符

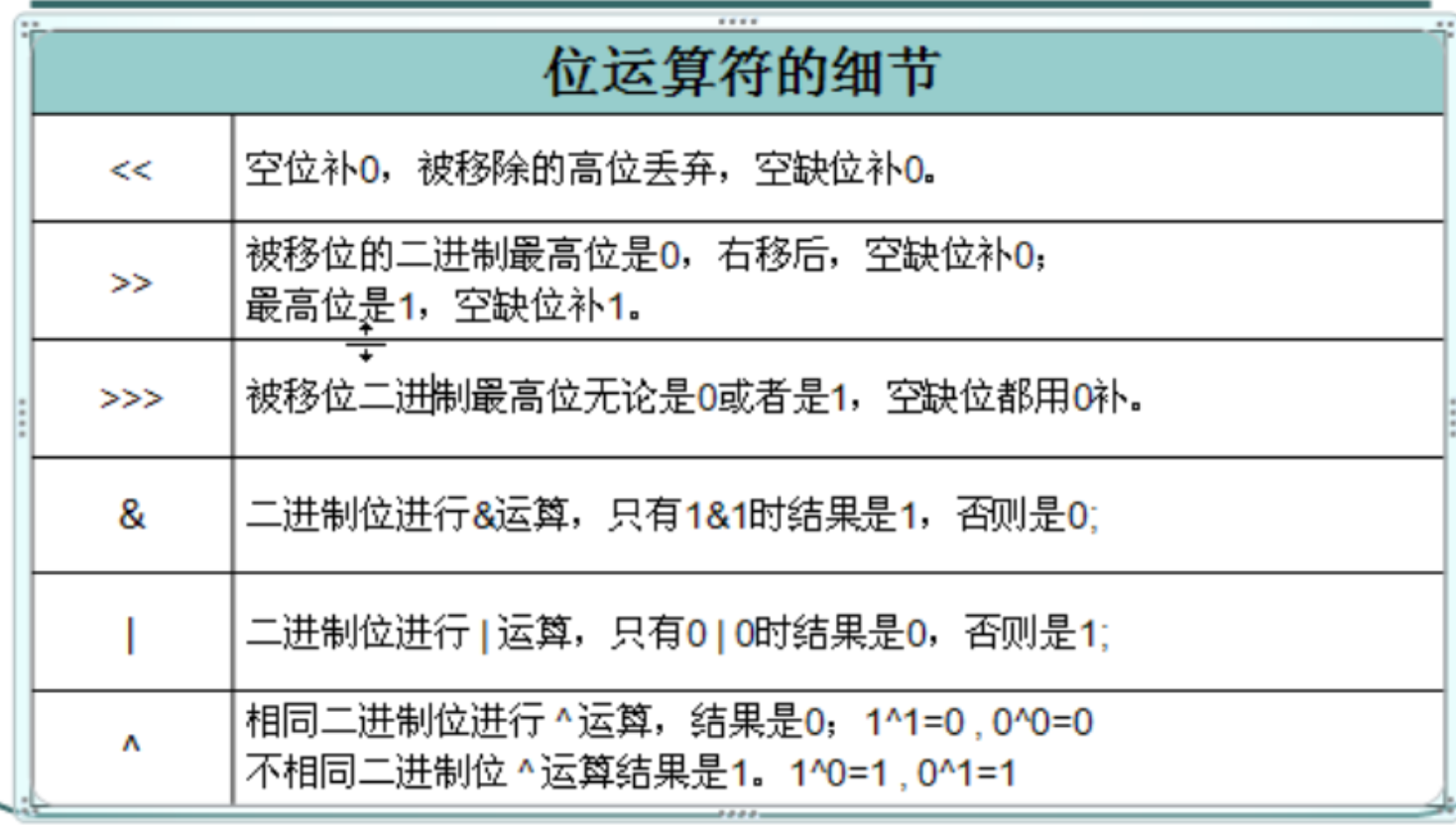
最高位补位，之前最高位是什么就补什么。



# >>>无符号右移

数据进行右移，无论原来的高位是什么，高位都用0补位。

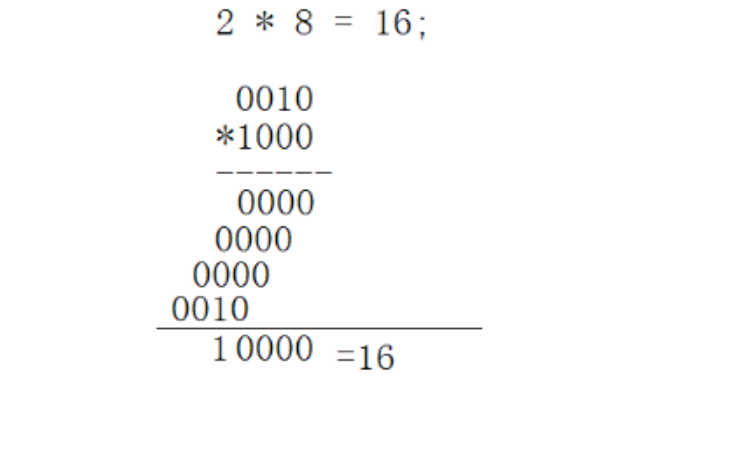
# 总结



# 实际计算

## 例一

2\*8



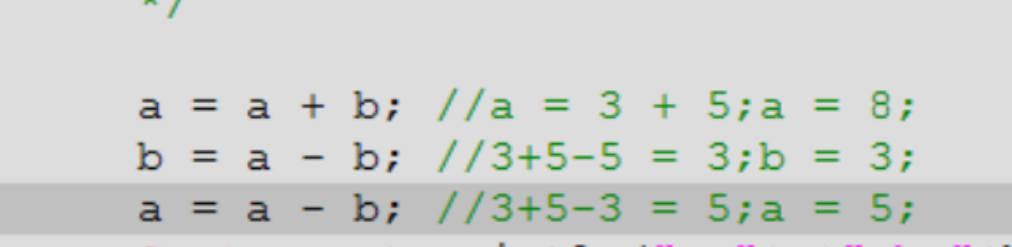
结果10000和0010相比，就是向左位移了三位。

## 例二

交换连个变量的值，不使用第三方变量

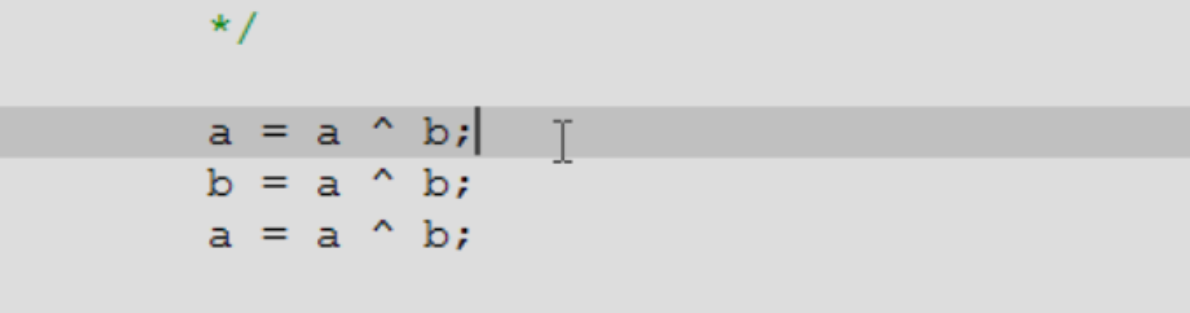
int a = 3，int b = 5

传统的方式需要第三方变量。



这种方式的缺点就是如果a，b的和超出int范围怎么办？？？？？

采用位运算方式：



这种思想和上面的第二种方式的思想是一样的。

## 例三

判断一个数是奇数还是偶数

|  |
| --- |
| /\*\*  \* 这只能解决正数和负数的问题。  \* @param i  \* @return  \*/  private static boolean isOdd3(int i) {  return i%2==1||i%2==-1;  }    /\*\*  \* 奇数和偶数转换成二进制的区别就是，奇数最后一位是1，偶数最后一位是0.  \* @param i  \* @return 是否为奇数  \*/  private static boolean isOdd4(int i) {  return (i&1)==1;  } |

任何一个数&1，结果要么是0，要么是1。是1说明是奇数，是0说明是偶数。

任何一个奇数，转成二进制，最后一位总是1，例如：0000-0000-0000-1101

1转成二进制为0000-0000-0000-0001，这两个做&运算。

|  |  |
| --- | --- |
| 0000-0000-0000-1101 | 0000-0000-0000-1100 |
| & 0000-0000-0000-0001 | & 0000-0000-0000-0001 |
| 0000-0000-0000-0001====1 | 0000-0000-0000-0000====0 |