## 进制转换

(1)进制：简单的说，就是计数的方式。x进制就是逢x进一。

(2)java语言中，对整数有3中表现形式

A：十进制

B：八进制

C：十六进制

(3)进制的由来：进制是由电信号来的。

C：十六进制

\*\*逢16进1

\*\*由0-9，A-F组成

\*\*A-F对应是：10-15。并且A到F大小写都行

\*\*用0x或者0X开头表示十六进制

D：规律：进制越大，表现形式越短

A：二进制

\*\*逢2进1

\*\*由0,1组成

B：八进制

\*\*逢8进1

\*\*由0-7组成

\*\*用0开头表示八进制

(4)不同进制的数据演示。

(5)进制转换

A：其他进制到十进制

\*\*规律：把系数\*基数的权次幂相加即可。

\*\*系数

\*\*基数

\*\*权

\*\*举例

\*\*\*二进制到十进制：1101 对应 13 =1\*23+1\*22+0\*21+1\*20

\*\*\*八进制到十进制：056 对应 46 =0\*82+5\*81+6\*80

\*\*\*十六进制到十进制：0xF4 对应244 =15\*161+4\*160

B：十进制到其他进制转换

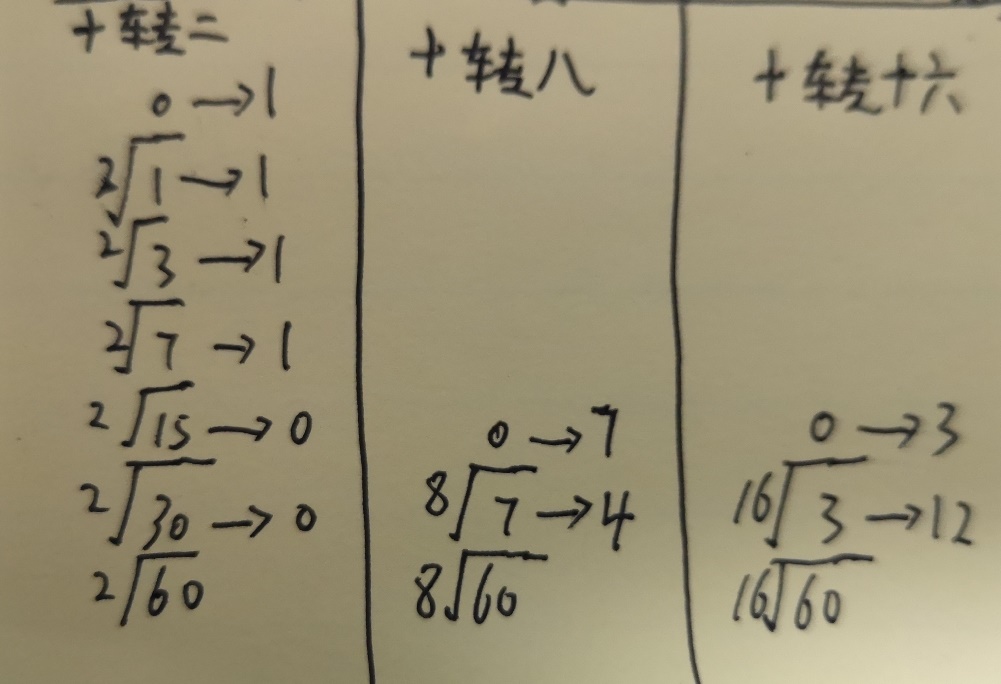
\*\*规律：除基取余，直到商为0，最后，余数反转就是结果。

\*\*举例

\*\*\*十进制到二进制：60 对应 111100

\*\*\*十进制到八进制：60 对应 074

\*\*\*十进制到十六进制：60 对应 0x3C



\*\*技巧

\*\*\*十进制到八进制

\*\*\*\*十进制--**二进制**--八进制

\*\*\*\*13 -- 1101 -- 15 (015)

\*\*\*十进制到十六进制

\*\*\*\*十进制--**二进制-**-十六进制

\*\*\*\*13 -- 1101 -- 13 (0xD)

\*\*\*二进制到八进制：

\*\*\*\*从右往左，每三位一组合，左边不够用0补。

\*\*\*\*把组合后的每一位换算成十进制数据。

\*\*\*\*最后，从左往右，把结果进行组合即可。

\*\*\*二进制到十六进制：

\*\*\*\*从右往左，每四位一组合，左边不够用0补。

\*\*\*\*把组合后的每一位换算成十进制数据。

\*\*\*\*最后，从左往右，把结果进行组合即可。

\*\*\*\*对应位上的数据，应该是十六进制的。

(6)代码示例

十进制到十六进制

(7)其他

## 位运算符 &,|,^,~,

**位运算，要求我们先把数据转换成二进制。**

**A、位异或运算（^）**

运算规则是：两个数转为二进制，然后从高位开始比较，如果相同则为0，不相同则为1。

比如：8^11.

8转为二进制是1000，11转为二进制是1011.从高位开始比较得到的是：0011.然后二进制转为十进制，就是Integer.parseInt("0011",2)=3;

\*\*一个数对同一个数异或两次，该数不变。

\*\*应用：加密。

**B、位与运算符（&）**

运算规则：两个数都转为二进制，然后从高位开始比较，如果两个数都为1则为1，否则为0。

比如：129&128.

129转换成二进制就是10000001，128转换成二进制就是10000000。从高位开始比较得到，得到10000000，即128.

**C、位或运算符（|）**

运算规则：两个数都转为二进制，然后从高位开始比较，两个数只要有一个为1则为1，否则就为0。

比如：129|128.

129转换成二进制就是10000001，128转换成二进制就是10000000。从高位开始比较得到，得到10000001，即129.

**D、位非运算符（~）**

运算规则：如果位为0，结果是1，如果位为1，结果是0.

比如：~37

在Java中，所有数据的表示方法都是以补码的形式表示，如果没有特殊说明，Java中的数据类型默认是int,int数据类型的长度是8位，一位是四个字节，就是32字节，32bit.

8转为二进制是100101.

补码后为： 00000000 00000000 00000000 00100101

取反为： 11111111 11111111 11111111 11011010

因为高位是1，所以原码为负数，负数的补码是其绝对值的原码取反，末尾再加1。

因此，我们可将这个二进制数的补码进行还原： 首先，末尾减1得反码：11111111 11111111 11111111 11011001 其次，将各位取反得原码：

00000000 00000000 00000000 00100110，此时二进制转原码为38

所以~37 = -38.

## 位运算符>>,>>>,<<

A：<< 左移运算符，num << n,相当于num乘以2n。右边空位用0补齐

B：>> 右移运算符，num >> n,相当于num除以2 n。左边高位是什么就用什么补齐。移出的部分将被抛弃

例如：

0000 1111(15)右移2位的结果是0000 0011(3)，0001 1010(18)右移3位的结果是0000 0011(3)。

C：>>>   无符号右移，忽略符号位，无论最高位是1还是0，左边被移空的高位都填入0。