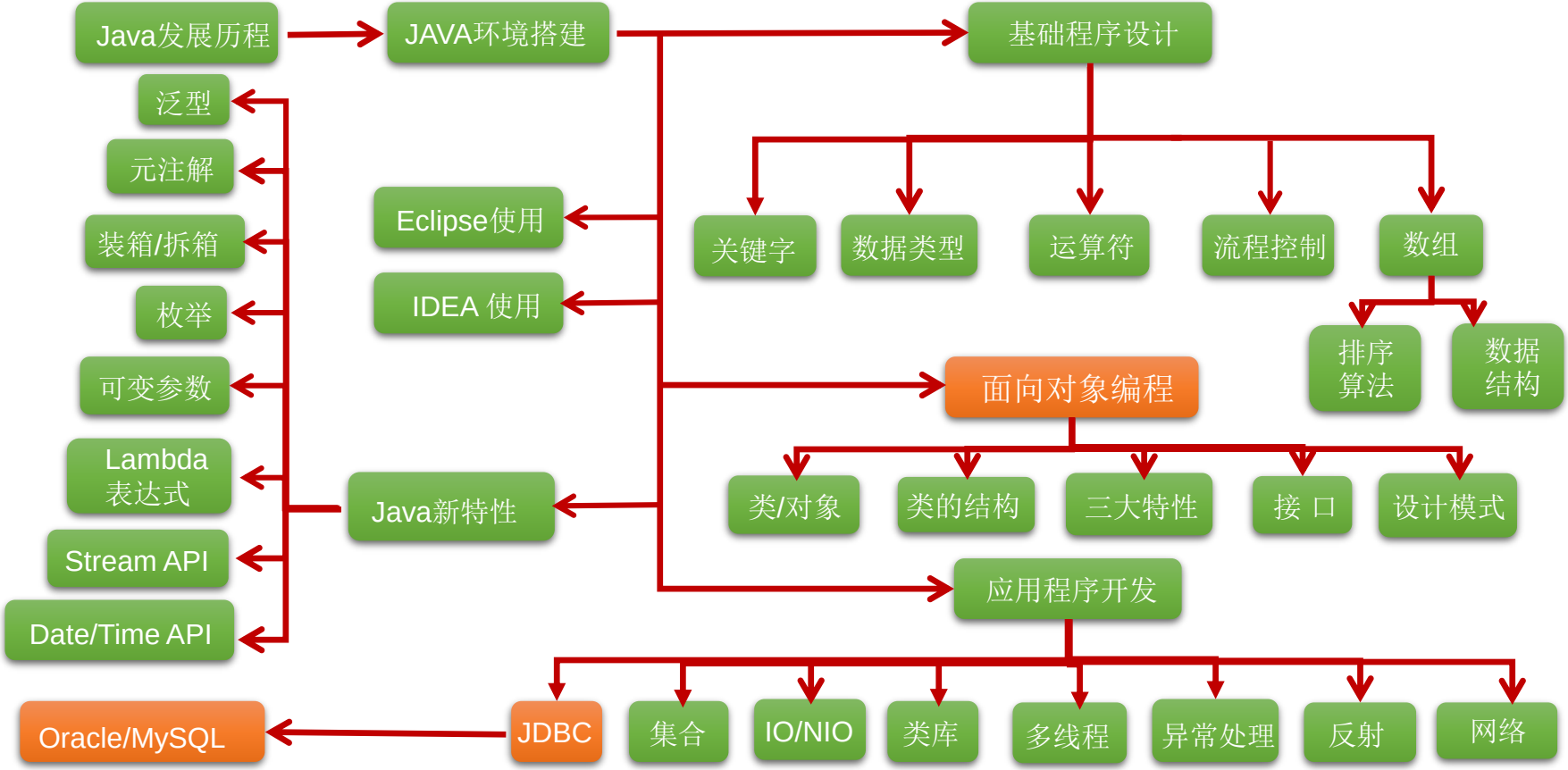
第2章
Java基本语法(上)：
变量与运算符

讲师：宋红康
新浪微博：尚硅谷-宋红康

Java基础知识图解



目录

尚硅谷

2-1 关键字与保留字

2.1 关键字与保留字

- 关键字(keyword)的定义和特点
  - 定义：被Java语言赋予了特殊含义，用做专门用途的字符串（单词）
  - 特点：关键字中所有字母都为小写
  - 官方地址： https://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html

用于定义数据类型的关键字				
class	interface	enum	byte	short
int	long	float	double	char
boolean	void			
用于定义流程控制的关键字				
if	else	switch	case	default
while	do	for	break	continue
return				
用于定义访问权限修饰符的关键字				
private	protected	public		

2.1 关键字与保留字

用于定义类，函数，变量修饰符的关键字				
abstract	final	static	synchronized	
用于定义类与类之间关系的关键字				
extends	implements			
用于定义建立实例及引用实例，判断实例的关键字				
new	this	super	instanceof	
用于异常处理的关键字				
try	catch	finally	throw	throws
用于包的关键字				
package	import			
其他修饰符关键字				
native	strictfp	transient	volatile	assert
*** 用于定义数据类型值的字面值**				
true	false	null		

保留字(reserved word)

- Java保留字：现有Java版本尚未使用，但以后版本可能会作为关键字使用。自己命名标识符时要避免使用这些保留字

  goto 、const

2-2 标识符(Identifier)

2.2 标识符(Identifier)

- 标识符：
  - Java 对各种变量、方法和类等要素命名时使用的字符序列称为标识符
  - 技巧：凡是自己可以起名字的地方都叫标识符。
- 定义合法标识符规则：
  - 由26个英文字母大小写，0-9 ，_或 $ 组成
  - 数字不可以开头。
  - 不可以使用关键字和保留字，但能包含关键字和保留字。
  - Java中严格区分大小写，长度无限制。
  - 标识符不能包含空格。

练习：miles, Test, a++, --a, 4#R, $4, #44, apps, class, public, int, x, y, radius

Java中的名称命名规范

- Java中的名称命名规范：
  - **包名**：多单词组成时所有字母都小写：xxxyyyzzz
  - **类名、接口名**：多单词组成时，所有单词的首字母大写：XxxYyyZzz
  - **变量名、方法名**：多单词组成时，第一个单词首字母小写，第二个单词开始每个单词首字母大写：xxxYyyZzz
  - **常量名**：所有字母都大写。多单词时每个单词用下划线连接：XXX_YYY_ZZZ

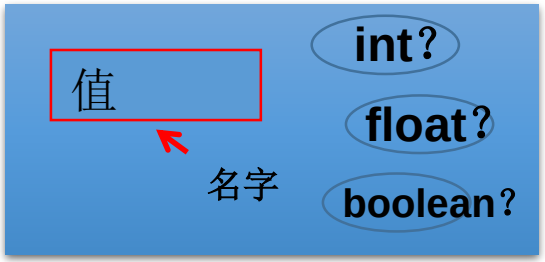
- 注意1：在起名字时，为了提高阅读性，要尽量有意义，“见名知意”。
- 注意2：java采用unicode字符集，因此标识符也可以使用汉字声明，但是不建议使用。

更多细节详见《代码整洁之道.pdf》

2-3 变量

- 基本数据类型
- 基本数据类型变量间转换
- 基本数据类型与String间转换
- 进制与进制间的转换

2.3 变 量

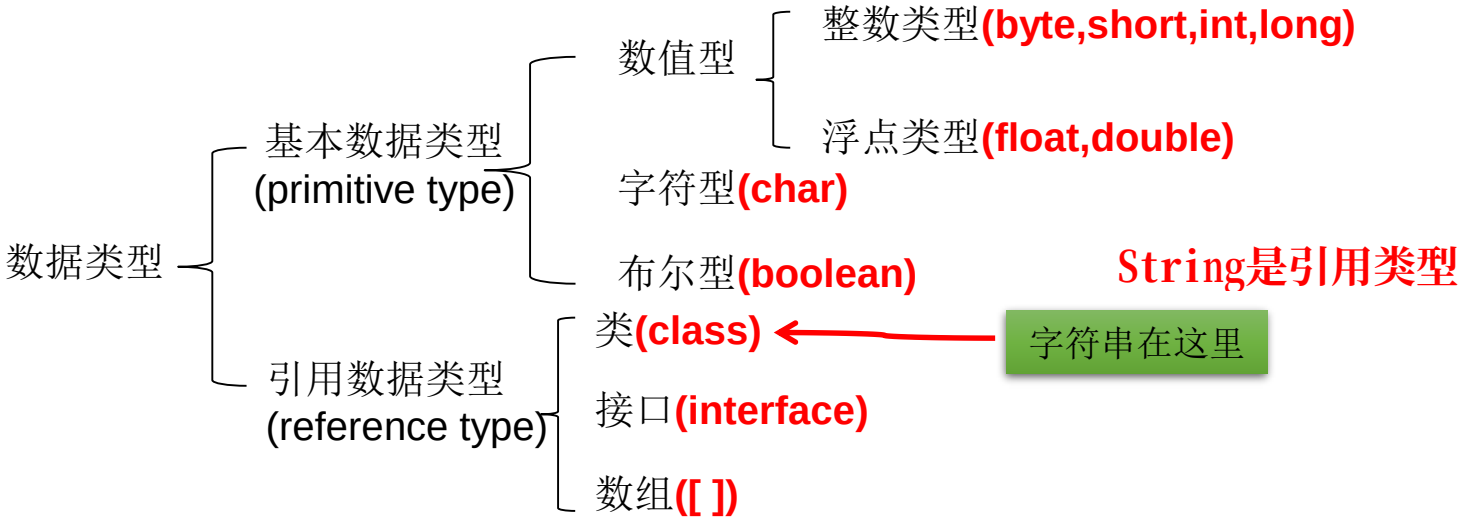


- **变量的概念：**
  - 内存中的一个存储区域
  - 该区域的数据可以在同一类型范围内不断变化
  - 变量是程序中最基本的存储单元。包含变量类型、变量名和存储的值
- **变量的作用：**
  - 用于在内存中保存数据
- **使用变量注意：**
  - Java中每个变量必须先声明，后使用
  - 使用变量名来访问这块区域的数据
  - 变量的作用域：其定义所在的一对{ }内
  - 变量只有在其作用域内才有效
  - 同一个作用域内，不能定义重名的变量

- 声明变量
  - 语法：<数据类型> <变量名称>
  - 例如：int var;
- 变量的赋值
  - 语法：<变量名称> = <值>
  - 例如：var = 10;
- 声明和赋值变量
  - 语法： <数据类型> <变量名> = <初始化值>
  - 例如：int var = 10;

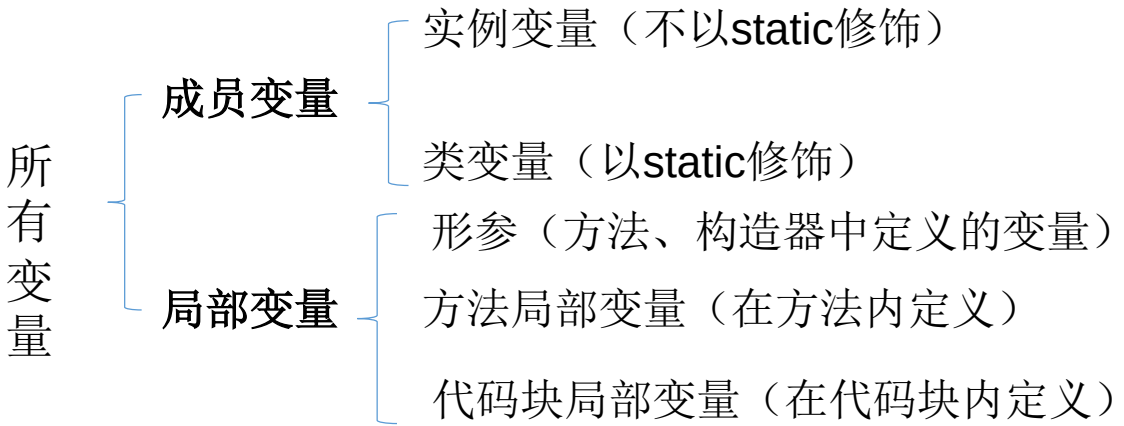
变量的分类-按数据类型

- 对于每一种数据都定义了明确的具体数据类型（强类型语言），在内存中分配了不同大小的内存空间。



补充：变量的分类-按声明的位置的不同

- 在方法体外，类体内声明的变量称为成员变量。
- 在方法体内部声明的变量称为局部变量。



- **注意：二者在初始化值方面的异同:**

  同：都有生命周期　　**异：**局部变量除形参外，需显式初始化。

整数类型：byte、short、int、long

- Java各整数类型有固定的表数范围和字段长度，不受具体OS的影响，以保证java程序的可移植性。
- java的整型常量默认为 int 型，声明long型常量须后加‘l’或‘L’
- java程序中变量通常声明为int型，除非不足以表示较大的数，才使用long

类 型	占用存储空间	表数范围
byte	1字节=8bit位	-128 ~ 127
short	2字节	-2^{15} ~$2^{15}-1$
int	4字节	-2^{31} ~ $2^{31}-1$ (约21亿)
long	8字节	-2^{63} ~ $2^{63}-1$

如果不加L，那么会被当做int看待，如果变量类型为long还好，可以转换为long，如果是数字范围小于int的类型就会出现转换错误

500MB 1MB = 1024KB 1KB= 1024B B= byte ? bit?

bit: 计算机中的最小存储单位。byte:计算机中基本存储单元。

```
1  public class VariableTest {
2      public static void main(String[] args) {
3          int number1;
4          number1 = 10;
5
6          int number2;
7          number2 = 20;
8
9          int number3;
10         number3 = number1 + number2;
11         System.out.println("Number3 = " + number3);
12
13         int number4 = 50;
14         int number5 = number4 - number3;
15         System.out.println("Number5 = " + number5);
16     }
17 }
```

浮点类型：float、double

- 与整数类型类似，Java 浮点类型也有固定的表数范围和字段长度，不受具体操作系统的影响。

- 浮点型常量有两种表示形式：
  - 十进制数形式：如：5.12　　512.0f　　.512　(必须有小数点)
  - 科学计数法形式:如：5.12e2　　512E2　　100E-2

通常使用double，而不是float，一个是因为double精度更高，而且没有后缀。

- float:单精度，尾数可以精确到7位有效数字。很多情况下，精度很难满足需求。

  double:双精度，精度是float的两倍。通常采用此类型。

float表示的数字范围比long更大

- **Java 的浮点型常量默认为double型，声明float型常量，须后加‘f’或‘F’。**

类　型	占用存储空间	表数范围
单精度float	4字节	-3.403E38 ~ 3.403E38
双精度double	8字节	-1.798E308 ~ 1.798E308

如果不加F，那么会被当做double看待，如果设定的变量类型为float，就会出现double -> float转换错误

转义字符	说明
\b	退格符
\n	换行符
\r	回车符
\t	制表符
\"	双引号
\'	单引号
\\	反斜线

字符类型：char

- char 型数据用来表示通常意义上“字符”(2字节)
- Java中的所有字符都使用Unicode编码，故一个字符可以存储一个字母，一个汉字，或其他书面语的一个字符。
- 字符型变量的三种表现形式：
  - 字符常量是用单引号(‘ ’)括起来的单个字符。例如：char c1 = 'a'； char c2 = '中'; char c3 = '9';
  - Java中还允许使用转义字符‘\’来将其后的字符转变为特殊字符型常量。例如：char c3 = ‘\n’; // '\n'表示换行符
  - 直接使用 Unicode 值来表示字符型常量： ‘\uXXXX’。其中，XXXX代表一个十六进制整数。如：\u000a 表示 \n。
- char类型是可以进行运算的。因为它都对应有Unicode码。

了解：ASCII 码

- 在计算机内部，所有数据都使用二进制表示。每一个二进制位（bit）有 0 和 1 两种状态，因此 8 个二进制位就可以组合出 256 种状态，这被称为一个字节（byte）。一个字节一共可以用来表示 256 种不同的状态，每一个状态对应一个符号，就是 256 个符号，从 0000000 到 11111111。

- ASCII码：上个世纪60年代，美国制定了一套字符编码，对英语字符与二进制位之间的关系，做了统一规定。这被称为ASCII码。ASCII码一共规定了128个字符的编码，比如空格“SPACE”是32（二进制00100000），大写的字母A是65（二进制01000001）。这128个符号（包括32个不能打印出来的控制符号），只占用了一个字节的后面7位，最前面的1位统一规定为0。

- 缺点：
  - 不能表示所有字符。
  - 相同的编码表示的字符不一样：比如，130在法语编码中代表了é，在希伯来语编码中却代表了字母Gimel (ג)

了解： Unicode 编码

- 乱码：世界上存在着多种编码方式，同一个二进制数字可以被解释成不同的符号。因此，要想打开一个文本文件，就必须知道它的编码方式，否则用错误的编码方式解读，就会出现乱码。

- Unicode：一种编码，将世界上所有的符号都纳入其中。每一个符号都给予一个独一无二的编码，使用 Unicode 没有乱码的问题。

- Unicode 的缺点：Unicode 只规定了符号的二进制代码，却没有规定这个二进制代码应该如何存储：无法区别 Unicode 和 ASCII：计算机无法区分三个字节表示一个符号还是分别表示三个符号。另外，我们知道，英文字母只用一个字节表示就够了，如果unicode统一规定，每个符号用三个或四个字节表示，那么每个英文字母前都必然有二到三个字节是0，这对于存储空间来说是极大的浪费。

了解： UTF-8

- UTF-8 是在互联网上使用最广的一种 Unicode 的实现方式。
- UTF-8 是一种变长的编码方式。它可以使用 1-6 个字节表示一个符号，根据不同的符号而变化字节长度。
- UTF-8的编码规则：
  - 对于单字节的UTF-8编码，该字节的最高位为0，其余7位用来对字符进行编码（等同于ASCII码）。
  - 对于多字节的UTF-8编码，如果编码包含 n 个字节，那么第一个字节的前 n 位为1，第一个字节的第 n+1 位为0，该字节的剩余各位用来对字符进行编码。在第一个字节之后的所有的字节，都是最高两位为"10"，其余6位用来对字符进行编码。

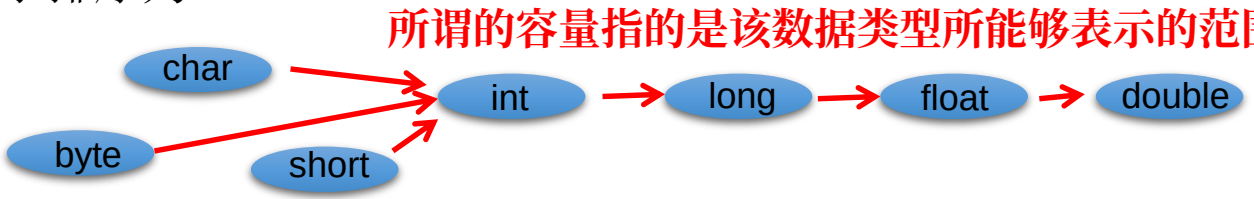
详见《尚硅谷_宋红康_计算机字符编码.docx》

布尔类型：boolean

- boolean 类型用来判断逻辑条件，一般用于程序流程控制：
  - if条件控制语句；
  - while循环控制语句；
  - do-while循环控制语句；
  - for循环控制语句；
- boolean类型数据只允许取值true和false，无null。
  - 不可以使用0或非 0 的整数替代false和true，这点和C语言不同。
  - Java虚拟机中没有任何供boolean值专用的字节码指令，Java语言表达所操作的boolean值，在编译之后都使用java虚拟机中的int数据类型来代替：true用1表示，false用0表示。———《java虚拟机规范 8版》

基本数据类型转换

- **自动类型转换**：容量小的类型自动转换为容量大的数据类型。数据类型按容量大小排序为：



- 有多种类型的数据混合运算时，系统首先自动将所有数据转换成容量最大的那种数据类型，然后再进行计算。
- byte,short,char之间不会相互转换，他们三者在计算时首先转换为int类型。
- boolean类型不能与其它数据类型运算。
- 当把任何基本数据类型的值和字符串(String)进行连接运算时(+)，基本数据类型的值将自动转化为字符串(String)类型。

字符串类型：String

- String不是基本数据类型，属于引用数据类型
- 使用方式与基本数据类型一致。例如：String str = “abcd”;
- 一个字符串可以串接另一个字符串，也可以直接串接其他类型的数据。例如：

```
str = str + “xyz” ;
int n = 100;
str = str + n;
```

除了boolean类型，其他类型都可以和String做连接运算，结果还是String类型

示 例—StringTest类

```
1  public class StringTest {
2      public static void main(String[] args) {
3          int no = 10;
4          String str = "abcdef";
5          String str1 = str + “xyz” + no;
6
7          str1 = str1 + "123";
8          char c = '国';
9
10          double pi = 3.1416;
11          str1 = str1 + pi;
12          boolean b = false;
13          str1 = str1 + b;
14          str1 = str1 + c;
15
16         System.out.println("str1 = " + str1);
17     }
18 }
```

练习1

String str1 = 4;　　//判断对错：no

String str2 = 3.5f + “”;　　//判断str2对错：yes

System.out.println(str2);　　//输出：”3.5”

System.out .println(3+4+“Hello!”);　　//输出：7Hello!

System.out.println(“Hello!”+3+4);　　//输出：Hello!34

System.out.println(‘a’+1+“Hello!”);　　//输出：98Hello!

System.out.println(“Hello”+‘a’+1);　　//输出：Helloa1

强制类型转换

- 自动类型转换的逆过程，将容量大的数据类型转换为容量小的数据类型。使用时要加上强制转换符：()，但可能造成精度降低或溢出,格外要注意。
- 通常，字符串不能直接转换为基本类型，但通过基本类型对应的包装类则可以实现把字符串转换成基本类型。
  - 如： String a = “43”; int i = Integer.parseInt(a);
  - boolean类型不可以转换为其它的数据类型。

练习2

判断是否能通过编译

```
1）short  s = 5;
    s = s-2;                    //判断：no
2）  byte b = 3;
     b = b + 4;                 //判断：no
     b = (byte)(b+4);           //判断：yes
3）char c = ‘a’;
    int  i = 5;
    float d = .314F;
    double result = c+i+d;      //判断：yes
4）  byte b = 5;
     short s = 3;
     short t = s + b;           //判断：no
```

s-2结果默认是int类型，因此需要转换类型

进 制

世界上有10种人 ， 认识和不认识二进制的。

关于进制

- 所有数字在计算机底层都以**二进制**形式存在。

- 对于整数，有四种表示方式：
  - **二进制(binary)**：0,1 ， 满2进1.以0b或0B开头。
  - **十进制(decimal)**：0-9 ， 满10进1。
  - **八进制(octal)**：0-7 ， 满8进1. 以数字0开头表示。
  - **十六进制(hex)**：0-9及A-F，满16进1. 以0x或0X开头表示。此处的A-F不区分大小写。
    如：0x21AF +1= 0X21B0

十进制	十六进制	八进制	二进制
0	0	0	0
1	1	1	1
2	2	2	10
3	3	3	11
4	4	4	100
5	5	5	101
6	6	6	110
7	7	7	111
8	8	10	1000

十进制	十六进制	八进制	二进制
9	9	11	1001
10	A	12	1010
11	B	13	1011
12	C	14	1100
13	D	15	1101
14	E	16	1110
15	F	17	1111
16	10	20	10000
17	11	21	10001

二进制

- Java整数常量默认是int类型，当用二进制定义整数时，其第32位是符号位；当是long类型时，二进制默认占64位，第64位是符号位
- 二进制的整数有如下三种形式：
  - 原码：直接将一个数值换成二进制数。最高位是符号位
  - 负数的反码：是对原码按位取反，只是最高位（符号位）确定为1。
  - 负数的补码：其反码加1。

- 计算机以二进制补码的形式保存所有的整数。
  - 正数的原码、反码、补码都相同
  - 负数的补码是其反码+1

●为什么要使用原码、反码、补码表示形式呢？

计算机辨别“符号位”显然会让计算机的基础电路设计变得十分复杂! 于是人们想出了将符号位也参与运算的方法. 我们知道, 根据运算法则减去一个正数等于加上一个负数, 即: 1-1 = 1 + (-1) = 0 , 所以机器可以只有加法而没有减法, 这样计算机运算的设计就更简单了。

$$1-1 = 1 + (-1) = [0000\ 0001]_{原} + [1000\ 0001]_{原} = [0000\ 0001]_{补} + [1111\ 1111]_{补} = [0000\ 0000]_{补} = [0000\ 0000]_{原}$$

二进制→十进制

1 * 2^3 + 1 * 2^2 + 1* 2^1 = 14

符号位
0：正数
1：负数

-14的原码

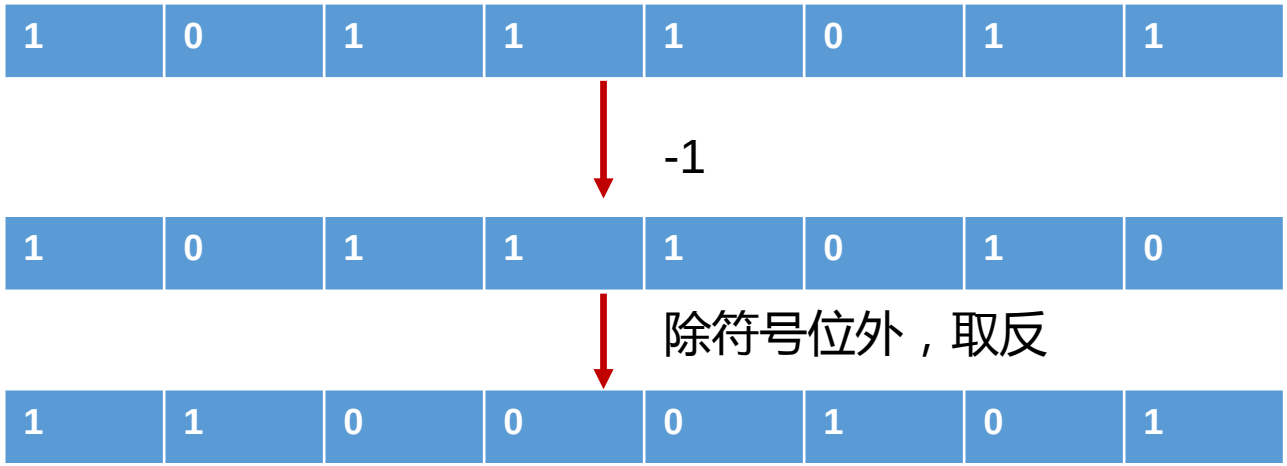
除符号位外，各个位取反

-14的反码

反码+1

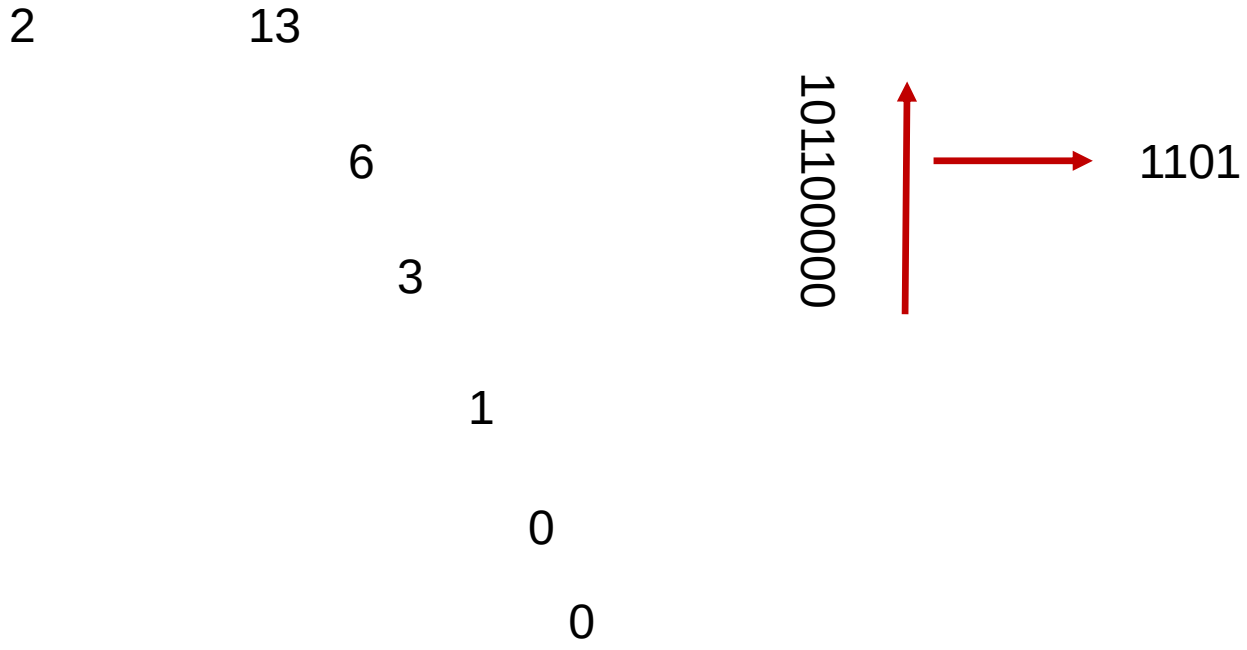
-14的补码

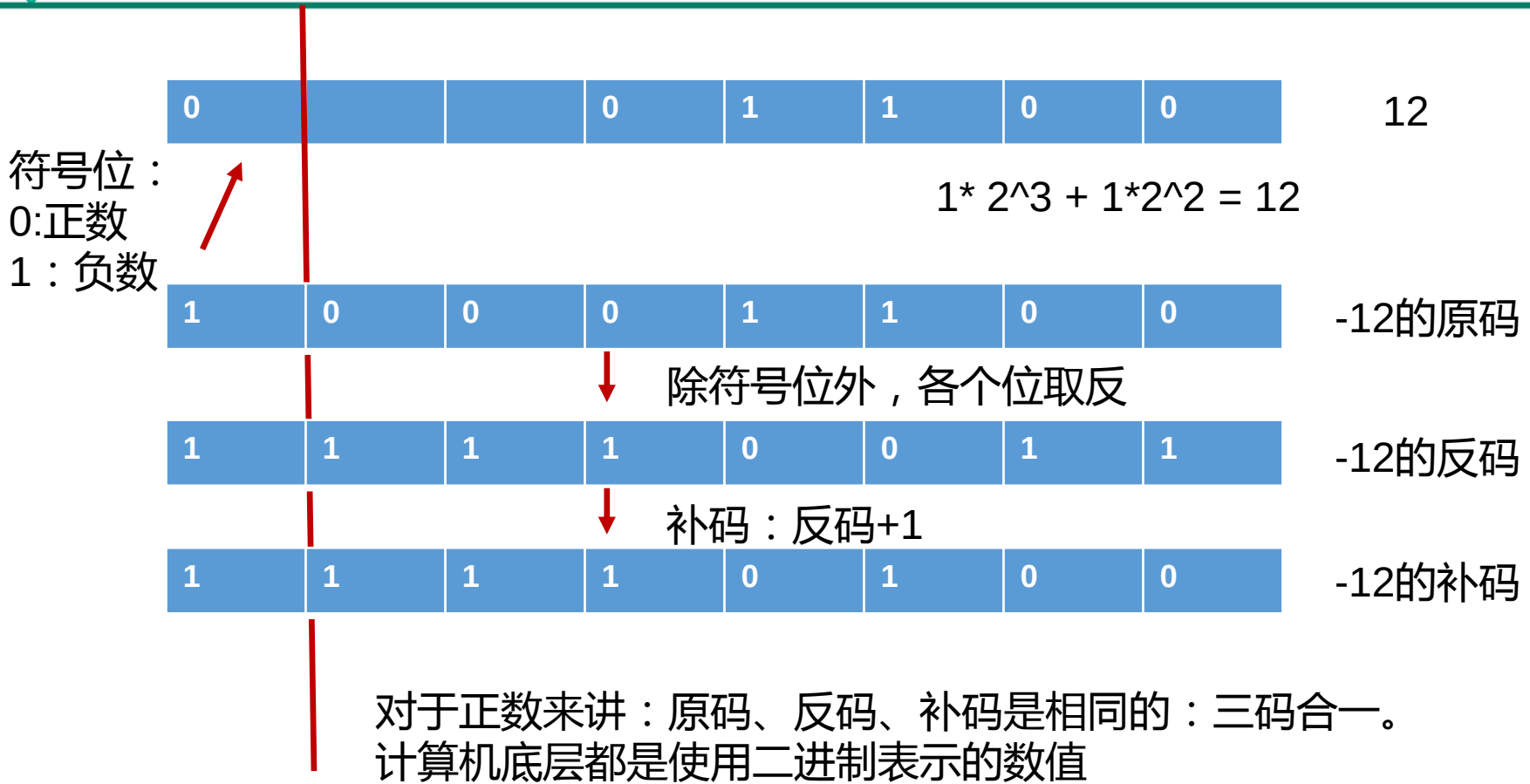
计算机底层都以补码的方式来存储数据！



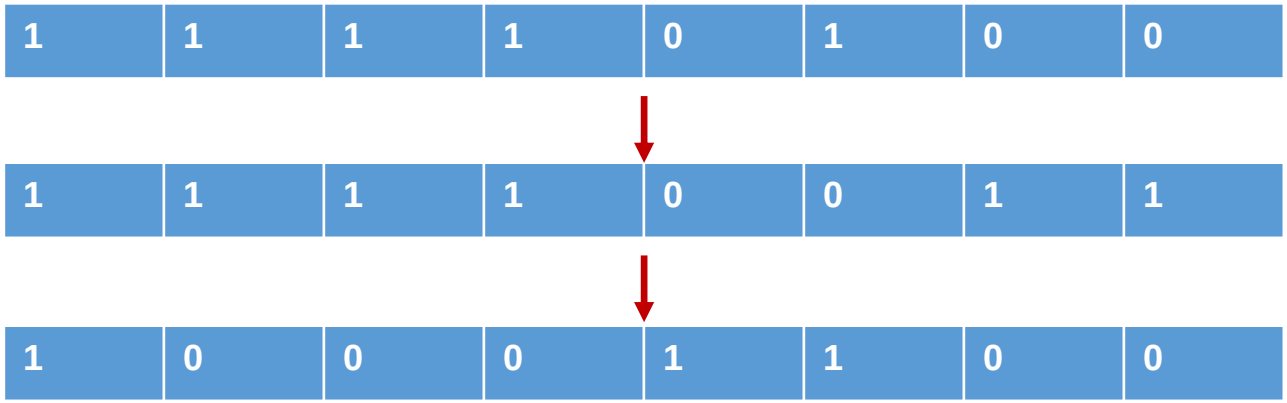
0	1	1	1	1	1	1	1	+127
1	1	1	1	1	1	1	1	-127的原码
1	0	0	0	0	0	0	0	-127的反码
1	0	0	0	0	0	0	1	-127的补码
1	0	0	0	0	0	0	0	-128的补码

十进制 →二进制：除2取余的逆



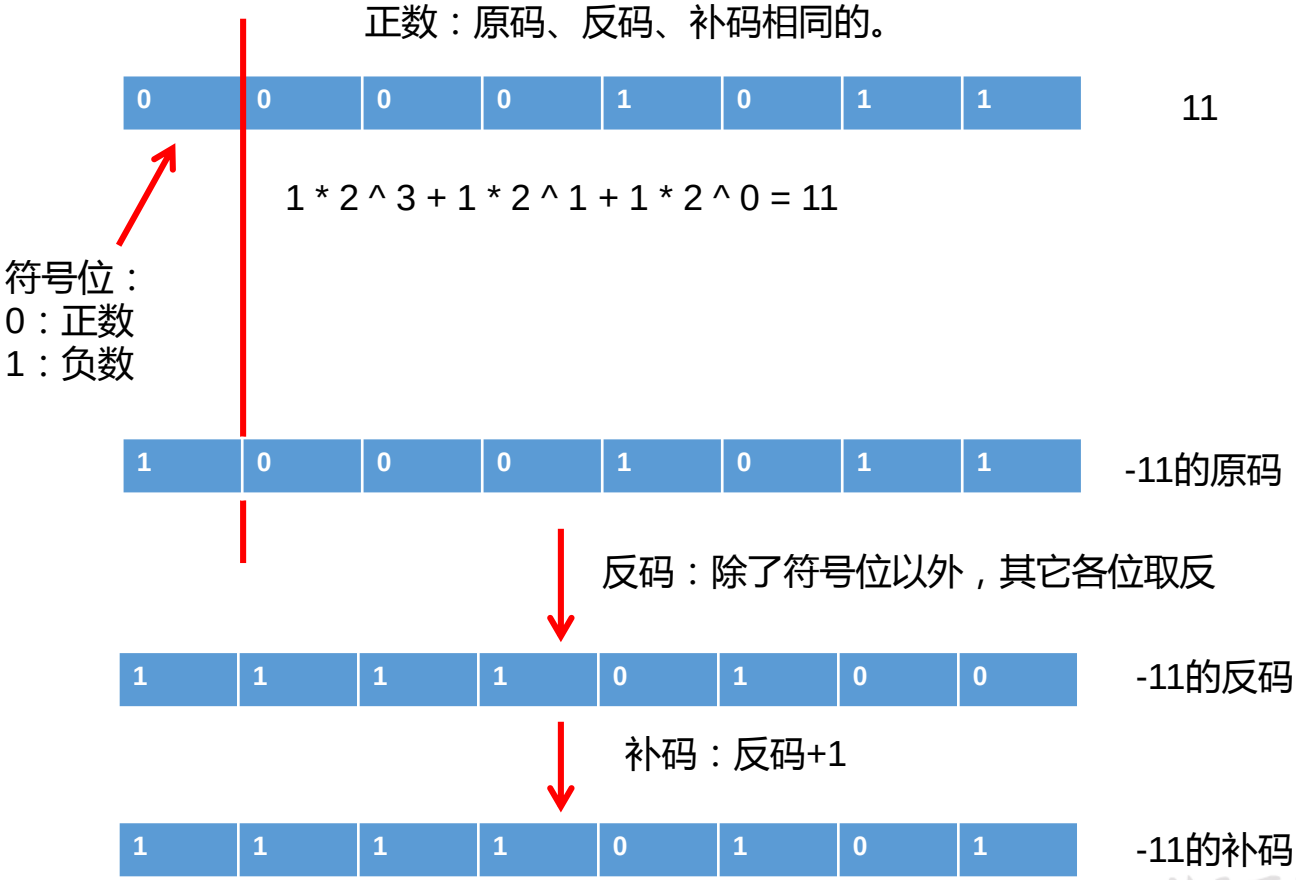


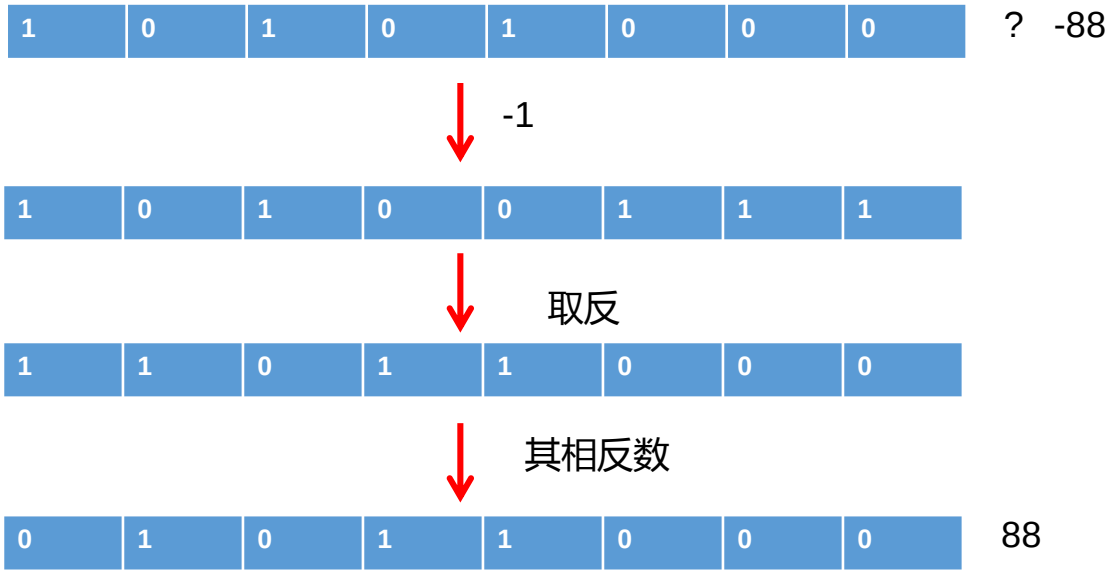
对于正数来讲：原码、反码、补码是相同的：三码合一。
计算机底层都是使用二进制表示的数值
计算机底层都是使用的数值的补码保存数据的。



2.3 变量之进制

在计算机底层，都是以补码的方式存储数值的。
对于正数来说：原码、反码、补码是相同的，三码合一


正数：原码、反码、补码相同的。
0 0 0 0 1 0 1 1
11
1 * 2 ^ 3 + 1 * 2 ^ 1 + 1 * 2 ^ 0 = 11
符号位：
0：正数
1：负数
1 0 0 0 1 0 1 1
-11的原码
反码：除了符号位以外，其它各位取反
1 1 1 1 0 1 0 0
-11的反码
补码：反码+1
1 1 1 1 0 1 0 1
-11的补码



十进制→二进制：除2取余的逆

2

13

6

3

1

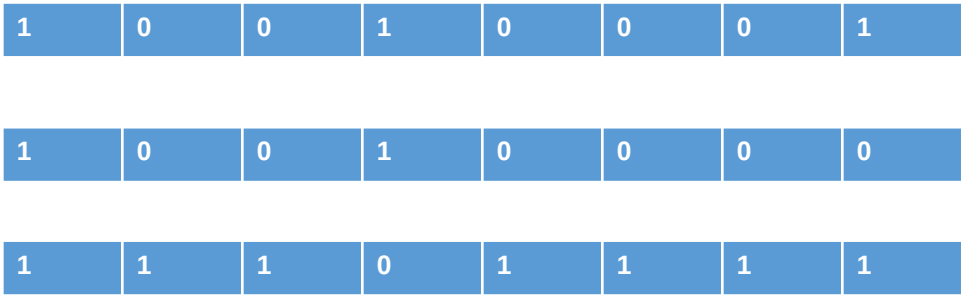
0

0

1011000000

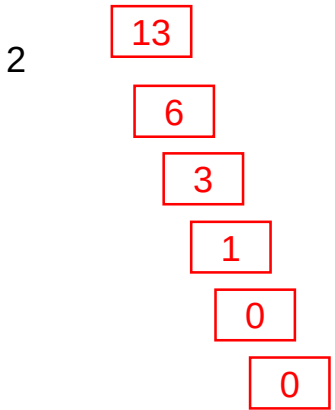
1101

2.3 变量之进制

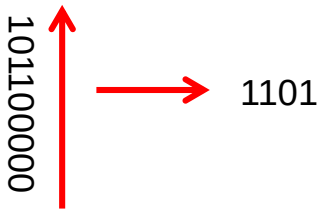


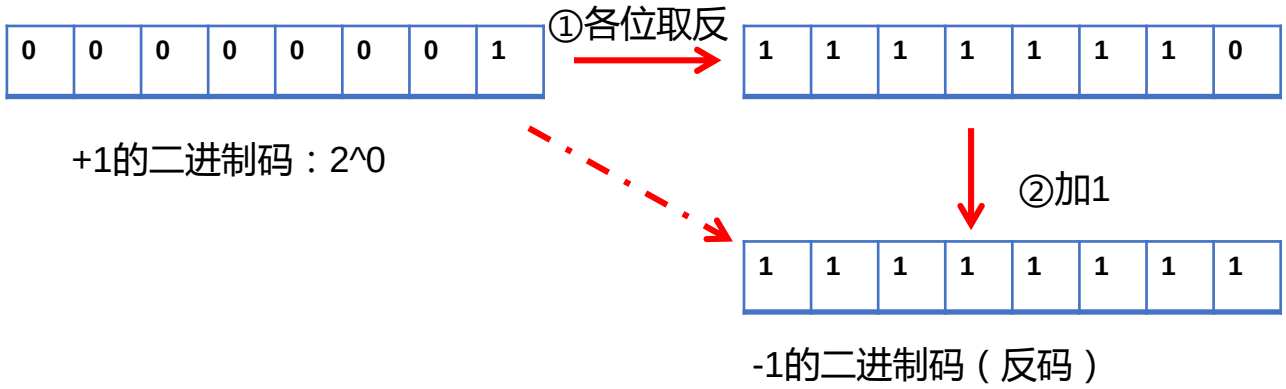
? -111

13 = 1 + 4 + 8 = 2^0 + 2 ^ 2 + 2 ^ 3



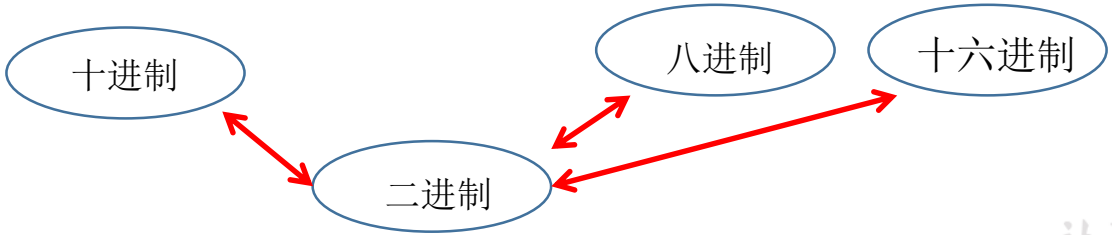
十进制→二进制：除2取模

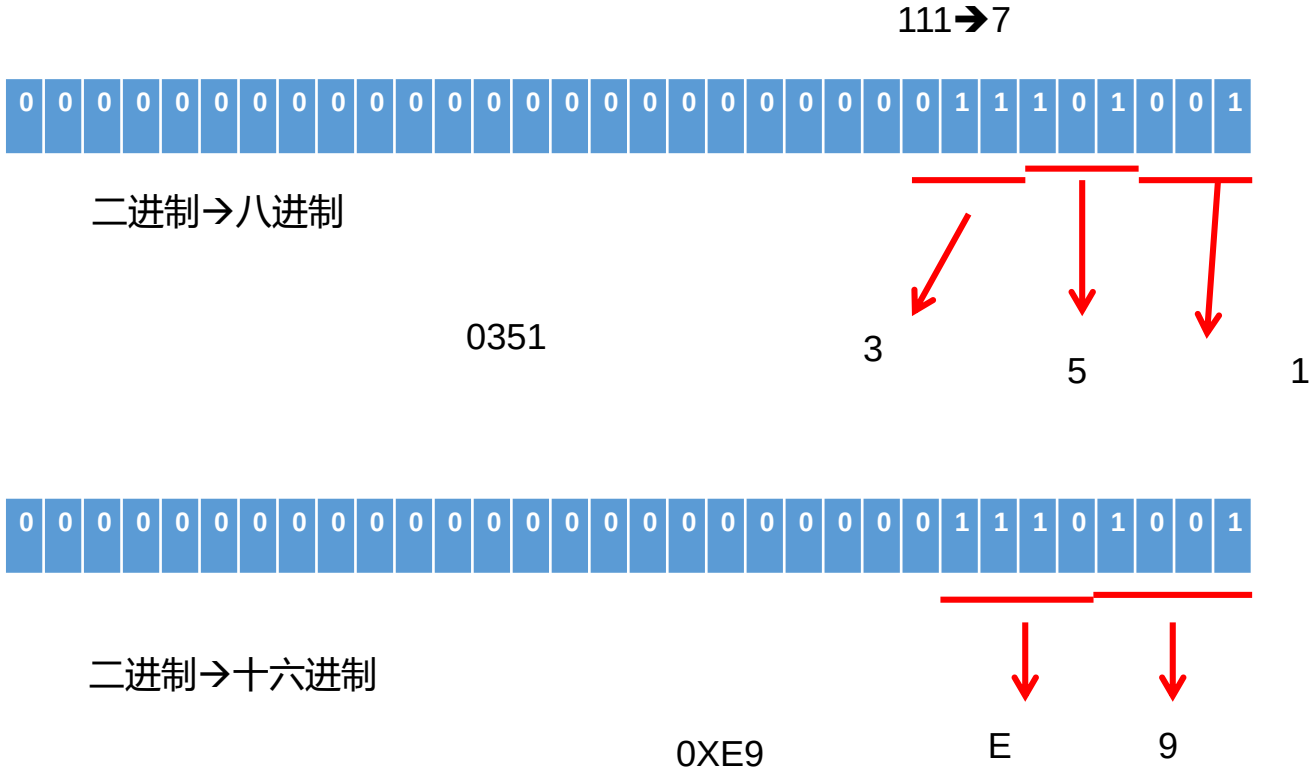




进制间转化

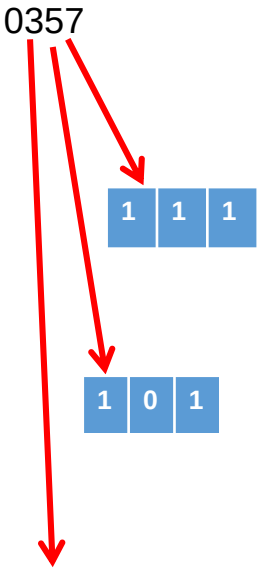
- 进制的基本转换
  - 十进制 二进制互转
    - ✓ 二进制转成十进制 乘以2的幂数
    - ✓ 十进制转成二进制 除以2取余数
  - 二进制 八进制互转
  - 二进制 十六进制互转
  - 十进制 八进制互转
  - 十进制 十六进制互转




111➔7
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 0 1
二进制➔八进制
0351
3
5
1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 0 1
二进制➔十六进制
0XE9
E
9

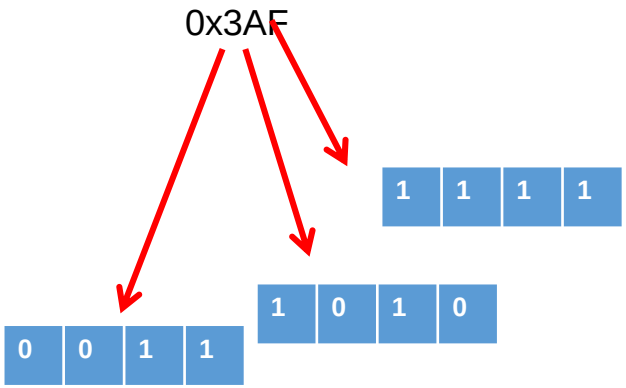
八进制：

0357



十六进制

0x3AF



二进制：

练 习

1. 将以下十进制数转换为十六进制和二进制

   123　256　87　62

2. 将以下十六进制数转换为十进制和二进制

   0x123　0x25F　0x38　0x62

2-4 运算符

运算符是一种特殊的符号，用以表示数据的运算、赋值和比较等。

- 算术运算符
- 赋值运算符
- 比较运算符（关系运算符）
- 逻辑运算符
- 位运算符
- 三元运算符

2.4.1 运算符：算术运算符

运算符	运算	范例	结果
+	正号	+3	3
-	负号	b=4; -b	-4
+	加	5+5	10
-	减	6-4	2
*	乘	3*4	12
/	除	5/5	1
%	取模(取余)	7%5	2
++ ++	自增（前）：先运算后取值 自增（后）：先取值后运算	a=2;b=++a; a=2;b=a++;	a=3;b=3 a=3;b=2
- - - -	自减（前）：先运算后取值 自减（后）：先取值后运算	a=2;b=- -a a=2;b=a- -	a=1;b=1 a=1;b=2
+	字符串连接	“He”+”llo”	“Hello”

算术运算符的注意问题

- 如果对负数取模，可以把模数负号忽略不记，如：5%-2=1。但被模数是负数则不可忽略。此外，取模运算的结果不一定总是整数。

模运算的结果符号与分子的符号相同

- 对于除号“/”，它的整数除和小数除是有区别的：整数之间做除法时，只保留整数部分而舍弃小数部分。 例如：int x=3510;x=x/1000*1000; x的结果是？

因此，如果要获取小数的结果，就要将除数或者被除数设置为浮点数float，double

- “+”除字符串相加功能外，还能把非字符串转换成字符串.例如：System.out.println(“5+5=”+5+5); //打印结果是？ 5+5=55 ？

练习1：算术运算符：自加、自减

```
public class SignTest{
	public static void main(String[] args){
		int i1 = 10;
		int i2 = 20;
		int i = i1++;
		System.out.print(“i=”+i);
		System.out.println(“i1=”+i1);
		i = ++i1;
		System.out.print(“i=”+i);
		System.out.println(“i1=”+i1);
		i = i2--;
		System.out.print(“i=”+i);
		System.out.println(“i2=”+i2);
		i = --i2;
		System.out.print(“i=”+i);
		System.out.println(“i2=”+i2);
	}
}
```

输出：

i=　　i1=

i=　　i1=

i=　　i2=

i=　　i2=

练习2

随意给出一个整数，打印显示它的个位数，十位数，百位数的值。
格式如下：
数字xxx的情况如下：
个位数：
十位数：
百位数：

例如：
数字153的情况如下：
个位数：3
十位数：5
百位数：1

- 符号：=
  - 当“=”两侧数据类型不一致时，可以使用自动类型转换或使用强制类型转换原则进行处理。
  - 支持连续赋值。

- 扩展赋值运算符： +=, -=, *=, /=, %=

思考1：

```
short s = 3;
s = s+2;  ①
s += 2;   ②
```

①和②有什么区别？

思考2：

```
int i = 1;
i *= 0.1;
System.out.println(i);//
i++;
System.out.println(i);//
```

思考3：

```
int m = 2;
int n = 3;
n *= m++;
System.out.println("m=" + m);
System.out.println("n=" + n);
```

思考4：

```
int n = 10;
n += (n++) + (++n);
System.out.println(n);
```

2.4.3 运算符：比较运算符

运算符	运算	范例	结果
==	相等于	4==3	false
!=	不等于	4!=3	true
<	小于	4<3	false
>	大于	4>3	true
<=	小于等于	4<=3	false
>=	大于等于	4>=3	true
instanceof	检查是否是类的对象	“Hello” instanceof String	true

- 比较运算符的结果都是boolean型，也就是要么是true，要么是false。
- **比较运算符“==”不能误写成“=”**。

思考1：

```
boolean b1 = false;
//区分好==和=的区别。
if(b1==true)
        System.out.println("结果为真");
else
       System.out.println("结果为假");
```

2.4.4 运算符：逻辑运算符

短路的意思就是如果左边的判断成功，那么右边就不需要继续判断了

&—逻辑与　　|—逻辑或　　！—逻辑非

&& —短路与　　|| —短路或　　^ —逻辑异或

a	b	a&b	a&&b	a|b	a||b	!a	a^b
true	true	true	true	true	true	false	false
true	false	false	false	true	true	false	true
false	true	false	false	true	true	true	true
false	false	false	false	false	false	true	false

2.4.4 运算符：逻辑运算符

- 逻辑运算符用于连接布尔型表达式，在Java中不可以写成3<x<6，应该写成x>3 & x<6 。

- “&”和“&&”的区别：
  - 单&时，左边无论真假，右边都进行运算；
  - 双&时，如果左边为真，右边参与运算，如果左边为假，那么右边不参与运算。

- “|”和“||”的区别同理，||表示：当左边为真，右边不参与运算。

- 异或(^)与或(|)的不同之处是：当左右都为true时，结果为false。

  理解：异或，追求的是“异”！

练习：请写出每题的输出结果

```
int x = 1;
int y=1;

if(x++==2 & ++y==2){
                x =7;
}
System.out.println("x="+x+",y="+y);
```

```
int x = 1,y = 1;

if(x++==2 && ++y==2){
                x =7;
}
System.out.println("x="+x+",y="+y);
```

```
int x = 1,y = 1;

if(x++==1 | ++y==1){
                x =7;
}
System.out.println("x="+x+",y="+y);
```

```
int x = 1,y = 1;

if(x++==1 || ++y==1){
                x =7;
}
System.out.println("x="+x+",y="+y);
```

【面试题】程序输出：

```
1.    class  Test  {
2.       public static void main (String []  args)  {
3.            boolean x=true;
4.            boolean y=false;
5.            short z=42;
6.             //if(y == true)
7.            if((z++==42)&&(y=true))z++;
8.            if((x=false) || (++z==45))  z++;
9.
10.           System. out.println(“z=”+z);
11.           }
12.   }
```

结果为：

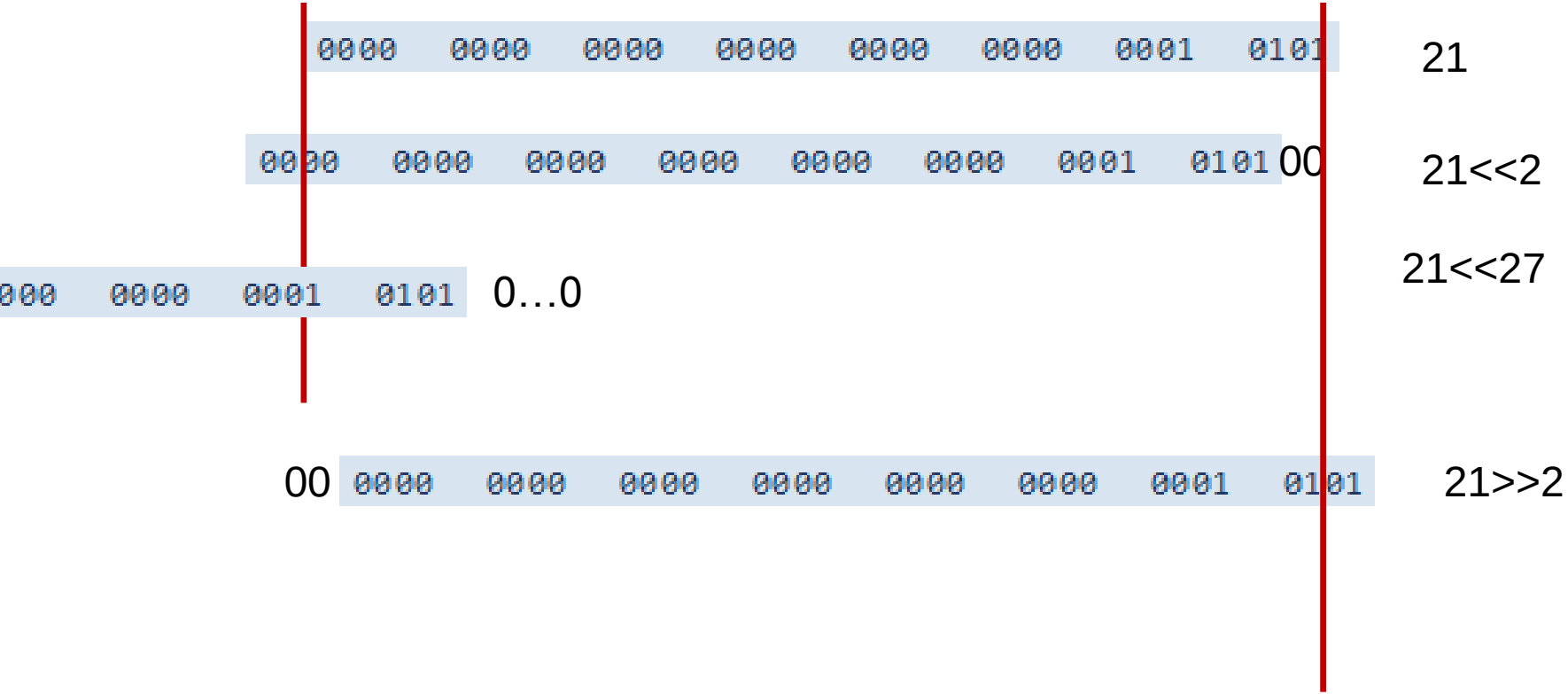
z= 46

2.4.5 运算符：位运算符

注意：无<<<

位运算符		
运算符	运算	范例
<<	左移	3 << 2 = 12 --> 3*2*2=12
>>	右移	3 >> 1 = 1 --> 3/2=1
>>>	无符号右移	3 >>> 1 = 1 --> 3/2=1
&	与运算	6 & 3 = 2
|	或运算	6 | 3 = 7
^	异或运算	6 ^ 3 = 5
~	取反运算	~6 = -7

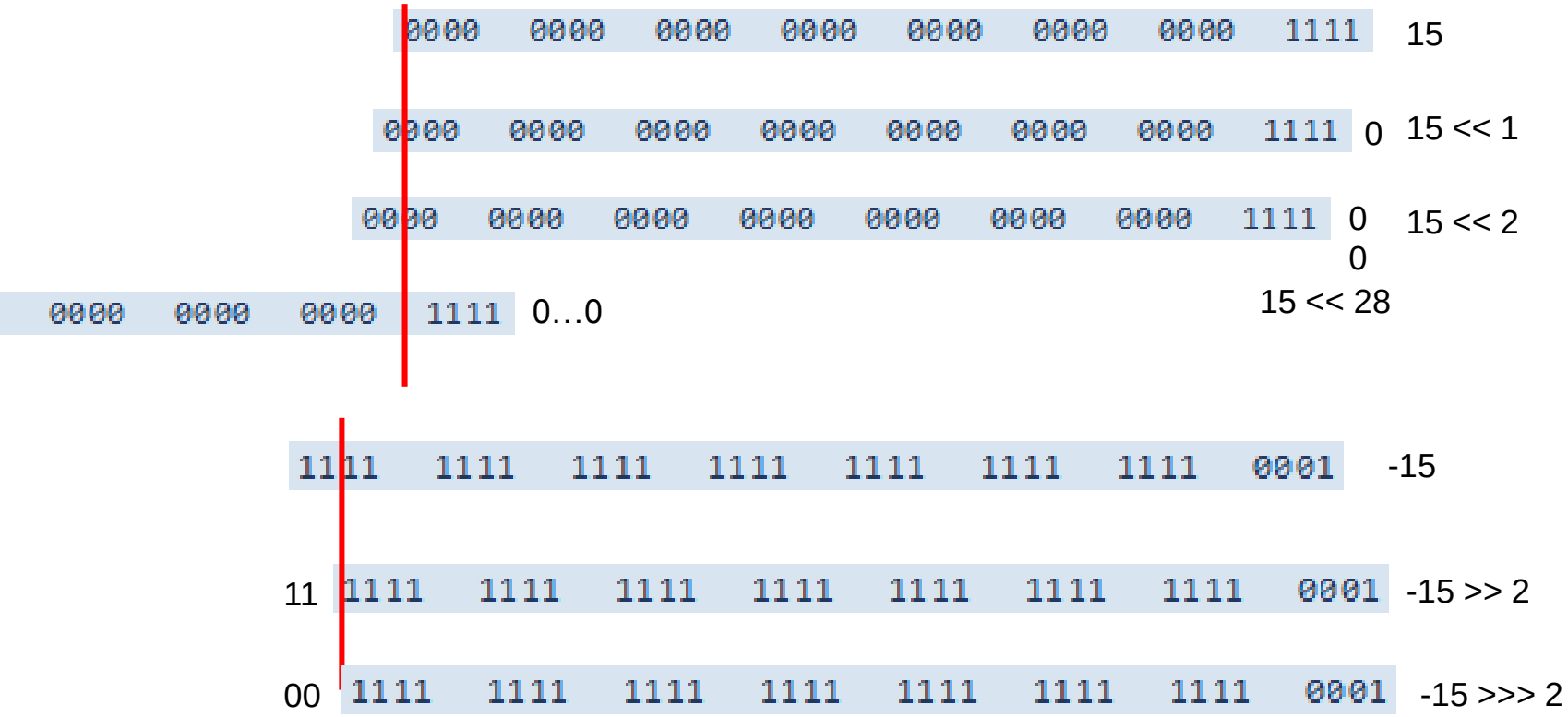
- 位运算是直接对整数的二进制进行的运算


0000 0000 0000 0000 0000 0000 0001 0101
21
0000 0000 0000 0000 0000 0000 0001 0101 00
21<<2
000 0000 0001 0101 0...0
21<<27
00 0000 0000 0000 0000 0000 0000 0001 0101
21>>2

2.4.5 运算符：位运算符

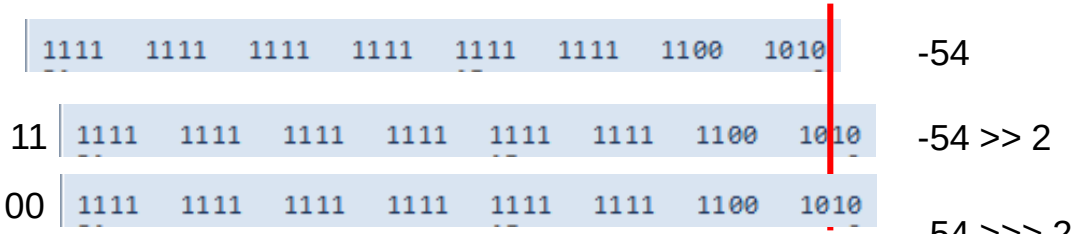
位运算符的细节	
<<	空位补0，被移除的高位丢弃，空缺位补0。
>>	被移位的二进制最高位是0，右移后，空缺位补0； 最高位是1，空缺位补1。
>>>	被移位二进制最高位无论是0或者是1，空缺位都用0补。
&	二进制位进行&运算，只有1&1时结果是1，否则是0;
|	二进制位进行 | 运算，只有0 | 0时结果是0，否则是1;
^	相同二进制位进行 ^ 运算，结果是0；1^1=0 , 0^0=0 不相同二进制位 ^ 运算结果是1。1^0=1 , 0^1=1
~	正数取反，各二进制码按补码各位取反 负数取反，各二进制码按补码各位取反

2.4.5 运算符：位运算符



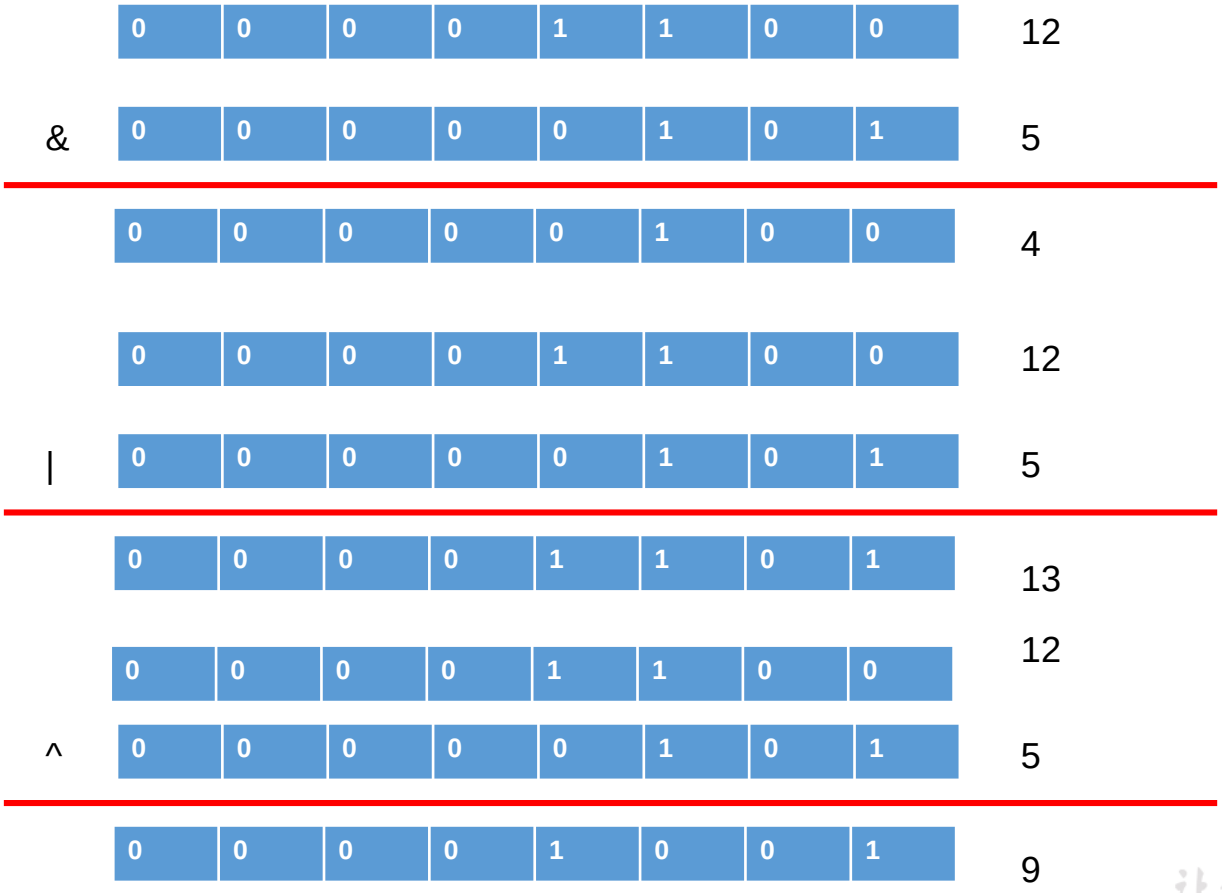
2.4.5 运算符：位运算符

对于正数来说，空出来的最高位拿0补



对于负数来说：
>>右移以后，最高空出来的位拿1去补
>>> 右移以后，高空出来的位拿0去补

2.4.5 运算符：位运算符



2.4.5 运算符：位运算符

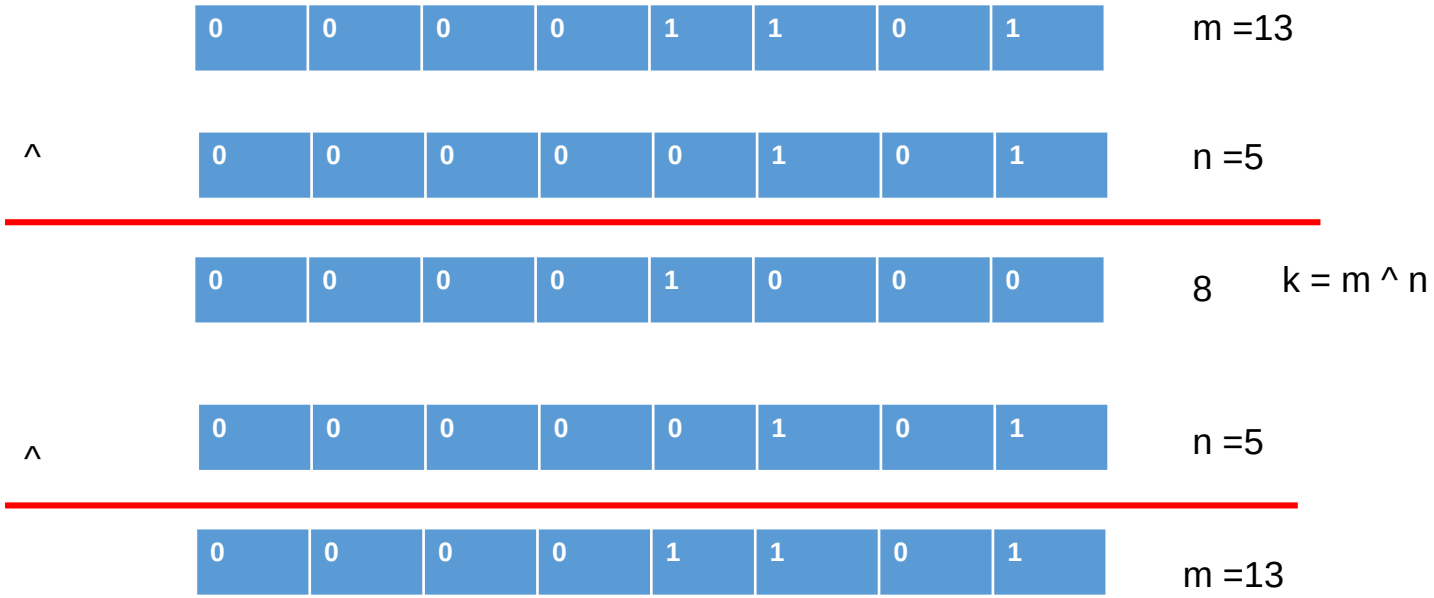
```
0000 0000 0000 0000 0000 0000 0000 0110
```
6

```
1111 1111 1111 1111 1111 1111 1111 1001
```
~6 = -7

交换两个值的三种方法

1. 定义临时变量tmp
2. 让一个值A变为A+B，然后B=A-B，A=A-B
3. 用异或运算，A=A^B，B = A^B，A = A^B，利用了异或运算的规则 A = (A^B)^B，两个相同的数做异或运算结果为0，而A与0做异或运算，结果为A

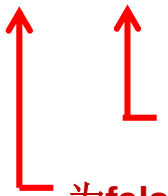
2.4.5 运算符：位运算符



m = k ^ n = (m ^ n) ^ n

2.4.6 运算符：三元运算符

- 格式:
  - (条件表达式)?表达式1：表达式2；



为true，运算后的结果是表达式1；

为false，运算后的结果是表达式2；

  - 表达式1和表达式2为同种类型
  - 三元运算符与if-else的联系与区别：
    1）三元运算符可简化if-else语句
    2）三元运算符要求必须返回一个结果。
    3）if后的代码块可有多个语句

练习： 获取两个数中的较大数
获取三个数中的较大数

运算符的优先级

- 运算符有不同的优先级，所谓优先级就是表达式运算中的运算顺序。如右表，上一行运算符总优先于下一行。
- 只有单目运算符、三元运算符、赋值运算符是从右向左运算的。

	. () {} ; ,
R—>L	++ -- ~ !(data type)
L—>R	* / %
L—>R	+ -
L—>R	<< >> >>>
L—>R	< > <= >= instanceof
L—>R	== !=
L—>R	&
L—>R	^
L—>R	|
L—>R	&&
L—>R	||
R—>L	? :
R—>L	= *= /= %=
	+= -= <<= >>=
	>>>= &= ^= |=


尚硅谷
让天下没有难学的的技术