Lecture4-1 位运算符

1. 运算符优先级表

操作符按照从上到下的优先级递减顺序显示，同一组中的运算符具有相同的优先级，表中显示了它们的结合性

Operator	*Name*	*Associativity*
`()`	Parentheses	Left to right
`()`	Function call	Left to right
`[]`	Array subscript	Left to right
`.`	Object member access	Left to right
`++`	Postincrement	Left to right
`--`	Postdecrement	Left to right
`++`	Preincrement	Right to left
`--`	Predecrement	Right to left
`+`	Unary plus	Right to left
`-`	Unary minus	Right to left
`!`	Unary logical negation	Right to left

Operator	*Name*	*Associativity*
`(type)`	Unary casting	Right to left
`new`	Creating object	Right to left
`*`	Multiplication	Left to right
`/`	Division	Left to right
`%`	Remainder	Left to right
`+`	Addition	Left to right
`-`	Subtraction	Left to right
`<<`	Left shift	Left to right
`>>`	Right shift with sign extension	Left to right
`>>>`	Right shift with zero extension	Left to right
`<`	Less than	Left to right
`<=`	Less than or equal to	Left to right
`>`	Greater than	Left to right
`>=`	Greater than or equal to	Left to right
`instanceof`	Checking object type	Left to right

Operator	*Name*	*Associativity*
==	Equal comparison	Left to right
!=	Not equal	Left to right
&	(Unconditional AND)	Left to right
^	(Exclusive OR)	Left to right
|	(Unconditional OR)	Left to right
&&	Conditional AND	Left to right
||	Conditional OR	Left to right
?:	Ternary condition	Right to left
=	Assignment	Right to left
+=	Addition assignment	Right to left
-=	Subtraction assignment	Right to left
*=	Multiplication assignment	Right to left
/=	Division assignment	Right to left
%=	Remainder assignment	Right to left
op=	Assignment with binary operator (*op* is one of +, -, *, /, % , &, |, ^, <<, >>, >>>)	Right to left

2. 整型

在内存中以位存储的任何整型

其中 b_n是符号位

整型类型	大小	n
byte	8 bits (1 byte)	n = 7
short	16 bits (2 bytes)	n = 15
int	32 bits (4 bytes)	n = 31
long	64 bits (8 bytes)	n = 63

进制

- 十进制：-23 是 int，1L 或 1l 是long，123_456 是 int
- 八进制：023 是 int，每个数字表示 3 个 bit
- 十六进制：0x2F，0XAF 是 int，以 0x 或 0X 开头，每个数字表示 4 个 bit
- 二进制：0b0011,0b1010_1111 是 int，每个数字表示 1 个 bit

整型转字符串

```
Integer.toString(100,8)   // get "144"     --octal representation
Integer.toString(100,2)   // get "1100100" --binary representation
Integer.toString(100,16)  // get "64"      --Hex representation
```

3. 按位运算符 Bitwise operators

操作符	描述
`|`	按位或
`&`	按位与
`~`	按位补
`^`	按位异或 XOR
`<<`	左移
`>>`	右移
`>>>`	无符号右移

按位运算示例

```
// get b23..b16 from rgb.
int r = rgb>>16 & 0xFF;

// set b23..b16 of rgb with b7..b0 of r,
// set b15..b8 of rgb with b7..b0 of g,
// set b7..b0 of rgb with b7..b0 of b.
rgb = r<<16 + g<<8 + b;

// x = A and y = B Pre-condition
x = x ^ y;
y = x ^ y;
x = x ^ y;
// x = B and y = A Post-Condition
```

4. 布尔运算符 Boolean related operators

Operator	*Name*	*Associativity*
!	Unary logical negation	Right to left
==	Equal comparison	Left to right
!=	Not equal	Left to right
&	(Unconditional AND)	Left to right
^	(Exclusive OR)	Left to right
|	(Unconditional OR)	Left to right
&&	Conditional AND	Left to right
||	Conditional OR	Left to right
?:	Ternary condition	Right to left

布尔运算示例

```
1  if(isConfirmed() == false)
2  if(!isConfirmed()) // elegant
3
4  if(a > b) return true; else return false;
5  return a > b ? true : false;
6  return a > b; // elegant
7
8  if(x != 0 && y/x > 5)... // 如果能得出结论，其余的子项均不计算
9  if(x != 0 & y/x >5)... // 所有的子项都计算
```