位运算符

基础规定

1. 位是什么? 就是二进制数字的每一“位”，一个整数数字，是由32位构成。
2. 位运算符是仅仅针对整数进行的运算符
3. 位运算符有如下几个

& 按位 与

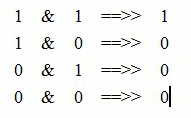
| 按位 或

~ 按位 非 按位取反

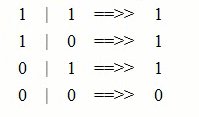
^ 按位 异或：

1. 位运算符的基本语法规则

按位与基本规则：



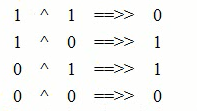
按位或基本规则



按位非基本规则



按位异或基本规则



可见，规则是相同为0，不同为1.

### 整数的按位与运算（&)

形式：

$n1 & $n2; //n1，n2是2个任意整数；

含义：

将该2个整数的二进制数字形式（注意，都是32位）的每一个对应位上的数字进行基本按位与运算之后的结果！

注意：他们运算的结果，其实仍然是一个普通的数字（10进制）。

示例图示（只用8个位来演示）：

$r1 = 10 & 20;

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10的2进制 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 20的2进制 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| &运算结果： | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

代码验证：



### 整数的按位或运算：

形式：

$n1 | $n2; //n1，n2是2个任意整数；

含义：

将该2个整数的二进制数字形式（注意，都是32位）的每一个对应位上的数字进行基本按位或运算之后的结果！

注意：他们运算的结果，其实仍然是一个普通的数字（10进制）。

示例图示（只用8个位来演示）：

$r1 = 10 | 20;

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10的2进制 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 20的2进制 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| |运算结果： | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

则结果该数据值大小为： 1\*24 + 1 \* 23 + 1\*22 + 1\* 21 + 0 = 16 + 8 + 4 + 2 = 30

代码验证：



### 整数的按位左移运算

形式：

$n1 << $m

含义：

将十进制数字n1的二进制数字形式（也是32位的）的每一个位上的数字都一次性往左边移动m位，

并将右边空出来的位置补0，左边冒出去的不管，这样操作之后得到的结果。

示例图示（只用8个位来演示）：

$r1 = 10 << 2;

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10的2进制 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 左移2位后 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 则结果为： |  |  | 25 | 0 | 23 | 0 | 0 | 0 |

可见，结果为：25 + 23 = 32 + 8 = 40

代码验证：

