运算符

2019年11月2日 11:18

在java中通过符号来表示指定的运算规则

分类：

算术运算符

赋值运算符

关系运算符（比较运算符）

逻辑运算符

位运算符

三目运算符

算术运算符：

+ - * / % ++ --

+:求和

如果两个元素的数据类型一致，结果的数据类型和元素保持一致；

如果两个元素的数据类型不一致，结果的数据类型和大的数据类型保持一致；

注意：

byte short char存在自动提升

经典面试题：

```
byte b1 = 3 ;
byte b2 = 4;
3  byte b3 = b1+ b2;
4   byte b4 = 3+4;
第三行、第四行代码编译是否会报错?
byte b3 = b1+ b2;//编译会报错 byte会自动提升int  运算结果为int类型
//从大转小
//在编译期，变量的值是无法获取到的；
byte b4 = 3+4;//编译不会报错  在编译期程序会进行优化 byte b4 = 7;
//检测7在不在byte类型范围之内，如果在的话，可以正常赋值；
//如果不在的话，编译就会报错；
```

-：求差

如果两个元素的数据类型一致，结果的数据类型和元素保持一致；

如果两个元素的数据类型不一致，结果的数据类型和大的数据类型保持一致；

注意：

byte short char存在自动提升

*：求积

如果两个元素的数据类型一致，结果的数据类型和元素保持一致；

如果两个元素的数据类型不一致，结果的数据类型和大的数据类型保持一致；

注意：

byte short char存在自动提升

/:求商

如果两个元素的数据类型一致，结果的数据类型和元素保持一致；

如果两个元素的数据类型不一致，结果的数据类型和大的数据类型保持一致；

注意：

byte short char存在自动提升

注意：

1、除数不能为0，如果除数为0，被除数为整数，程序编译没有问题，但是运行会抛出异常

Exception in thread "main" java.lang.ArithmeticException: / by zero

at OperatorDemo.main(OperatorDemo.java:42)

异常如果没有处理，当抛出异常时，程序会停止运行

2、正浮点数/0: Infinity(无穷大)

负浮点数/0:- Infinity(无穷小)

0.0/0:NaN(Not a Number)

%:取余 模余

num1%num2: num1/num2=商（整数） num1-num2*商=余数

6.1%2.2=6.1-2.2*2=1.7

余数是正负取决于num1元素的正负号

++：自增运算符 1

单目运算符，只能对单个变量元素作运算；

i++;<=>i = (type)(i+1);

++可以在变量前边也可以放在变量后边；

如果++在变量的前边，则先给变量+1，再参与运算

如果++在变量的后边，则使用变量的值参与运算，变量的值再+1

注意：尽量不要使用++ --来构造太过复杂的表达式，否则程序的可读性会非常差

--：自减运算符 1

单目运算符，只能对单个变量元素作运算；

i--;<=>i = (type)(i-1);

--可以在变量前边也可以放在变量后边；

如果--在变量的前边，则先给变量-1，再参与运算

如果--在变量的后边，则使用变量的值参与运算，变量的值再-1

注意：尽量不要使用++ --来构造太过复杂的表达式，否则程序的可读性会非常差

补充知识点：

字符串：

“abc”

String 类

格式：String str = "hello";

输出语句：

System.out.println(str);//先输出数据再换行

System.out.print(str);//直接输出数据 不再换行

+：

求和

字符串拼接

+： 通过+拼接会产生一个新的字符串

1、字符串拼接字符串

2、字符串拼接数值型

3、字符串拼接字符型

4、字符串拼接布尔型

赋值运算符

= += -= *= /= %=

=: 直接赋值

变量 常量

支持连等赋值 a = b =3

+=:

左侧变量的值和右侧的值求和，将得到结果再赋值给左侧变量

+=默认包含了数据类型的强制转换

注意：

+=符号左边不能为数值和常量，只能是变量

-=：

左侧变量的值和右侧的值求差，将得到结果再赋值给左侧变量

-=默认包含了数据类型的强制转换

注意：

-=符号左边不能为数值和常量，只能是变量

关系运算符

比较运算符

>(大于) >=(大于等于) <(小于) <=(小于等于) ==(等于) !=(不等于)

注意：

运算的结果必然为boolean true false

逻辑运算符：

对于boolean类型作运算，结果也为boolean类型

& | ! ^ && ||

&: 与 且 AND

a&b a:true false b:true false

true&true = true

true&false = false

false&true = false

false&false = false

总结：

两个元素都为true，结果才为true；否则结果就为false;

|:或 OR

a | b a:true false b:true false

true|true = true

true|false = true

false|true = true

false|false = false

总结：

两个元素都为false，结果才为false；否则结果就为true;

！:非 取反

单目运算符

!true =false !false=true

^:异或

如果两个元素的值相等，则结果为false;如果两个元素的值不相等，则结果为true

true^true=false

false^false=false

true^false=true
false^true=true

经典的面试题：

交换两个变量的值；一个值异或同一个值两次结果还是该数本身

&&：双与 短路与

false&&?=false

运算规则：

如果&&左边是false，则右边的表达式不再执行，结果必然为false;

||:双或　短路或

true||?=true

运算规则：

如果||左边是true，则右边的表达式不再执行，结果必然为true;

优点：

提高了代码的执行效率、

位运算符（了解即可）

对整型数字作运算，基于二进制来运算的；

& | ^ ~ << >> >>>

&:按位与　1 true　0 false

0&1=0

0&0=0

1&0=0

1&1=1

2&3

```
    2    0000 0000          0010
&   3    0000 0000          0011
    ------------------------------------
         0000 0000          0010
```

|:按位或　1 true　0 false

0 | 1=1

0 | 0=0

1 | 1=1

1 | 0=1

2 | 3

	2	0000 0000	0010
|	3	0000 0000	0011
		0000 0000	0011 3

^：按位异或　1 true　0 false

1^1=0
0^0=0
1^0=1
0^1=1

2^3

	2	0000 0000	0010
^	3	0000 0000	0011
		0000 0000	0001 1

一个数异或另外一个数两次结果还是该数本身

交换两个整数变量的值

方式一：通过第三个变量　最好理解

方式二：求和的方式　不用创建第三个变量

方式三：^来实现　效率最高

~：按位取反

单目运算符

~0=1　~1=0

计算机底层存储数据以及运算都是基于二进制的补码来实现的；

原码　反码　补码

正数：原码=反码=补码

负数：

原码：十进制转二进制的方式　最高位 符号位

反码：符号位不变，其余各位取反0->1　1->0

补码：反码的基础+1

~5

~ 5 0000 0000 0101

补码： 1111 1111 1010 -6

反码： 1111 1111 1001

原码： 1000 0000 0110 -6

二进制的补码

~ -5 1000 0000 0101 原码

1111 1111 1010 反码

1111 1111 1011 补码

---

4 0000 0000 0100 补码=反码=原码

规律：~i=-i-1

<<

5<<2

<< 5 0000 0000 0101

0000 0000 010100 16+4=20

总结：

直接将数字的二进制的形式向左移动相应的位数，左侧移出去的直接舍弃，右侧空出来的补0；

符号位保持不变的；

a<<b: a*2^b

5放大2的3倍：5*2*2*2 5<<3

>>:

6>>2

>> 6 0000 0000 0110

000000 0000 0110

1

总结：

直接将数字的二进制的形式向右移动相应的位数，右侧移出去的直接舍弃，左侧

根据符号位来补；

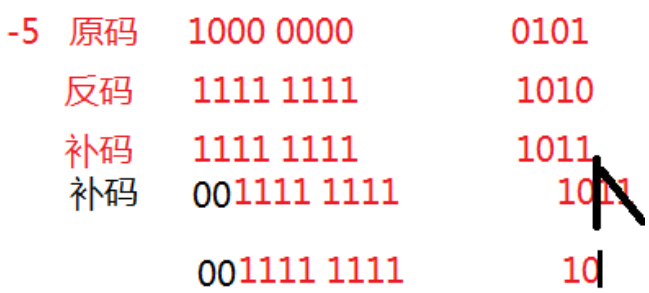
a>>b: a/(2^b)

>>>:

总结：

直接将数字的二进制的形式向右移动相应的位数，右侧移出去的直接舍弃，左侧补0；

-5>>>2



对于正数和右移的结果一致；负数做无符号右移后效果一样

三目运算符

?:

格式：

表达式？表达式1：表达式2

表达式：boolean值

执行流程：

执行表达式，如果值为true，则执行表达式1；如果值为false，则执行表达式2；

案例：

1、判断数字是奇数还是偶数

2、比较两个数的大小

3、比较三个数的大小