

Programmazione 1 - Modulo C

Java: Operatori logici lazy vs bitwise

Marco Beccuti

Università degli Studi di Torino

Dipartimento di Informatica

Ottobre 2020



Operatori logici lazy

- Gli operatori **&&** e **||** sono operatori lazy (pigri) o short-cut:

valutano il secondo argomento solo se é necessario

- Per esempio:

▶ **P && Q** \Rightarrow Q viene valutato solo se P vale true;

▶ **P || Q** \Rightarrow Q viene valutato solo se P vale false;

P	Q	P AND Q
1	1	1
1	0	0
0	1	0
0	0	0

P	Q	P OR Q
1	1	1
1	0	1
0	1	1
0	0	0

Operatori logici lazy

- **$(a \neq 0) \ \&\& \ (b/a > 100)$** \Rightarrow non dà errore quando ad **a** è assegnato 0;
- **$(b/a > 100)$** \Rightarrow viene valutato solo quando **a** è diverso da zero.

Operatori logici bitwise

- Gli operatori **&** e **|** sono operatori bitwise:

valutano sempre i due argomenti

- Per esempio:

- ▶ **P & Q** \Rightarrow P e Q sono sempre valutati;
- ▶ **P | Q** \Rightarrow P e Q sono sempre valutati;

Operatori logici bitwise

- $(a \neq 0) \ \& \ (b/a > 100)$ da errore quando ad **a** é assegnato 0;
- $(b/a > 100)$ viene valutato sempre!!!.

Operatori logici bitwise

- L'operatore AND bitwise `&` se applicato a due variabili intere effettua AND dei bit:

```
int a=60 // = 0011 1100
```

```
int b=13 // = 0000 1101
```

```
c=a&b // = 0000 1100 ossia c=12
```

a	0	0	1	1	1	1	0	(60)
b	0	0	0	0	1	1	0	(13)
<hr/>								
a&b	0	0	0	0	1	1	0	(12)

a	b	a&b
0	0	0
1	0	0
0	1	0
1	1	1

Assumiamo per semplicità che gli interi siano codificati con 8bit

Operatori logici bitwise

- L'operatore OR bitwise | se applicato a due variabili intere effettua OR dei bit:

```
int a=60 // = 0011 1100
```

```
int b=13 // = 0000 1101
```

```
c=a|b // = 0011 1101 ossia c=61
```

a	0011 1100	(60)
b	0000 1101	(13)
<hr/>		
a b	0011 1101	(61)

a	b	a b
0	0	0
1	0	1
0	1	1
1	1	1

Assumiamo per semplicità che gli interi siano codificati con 8bit

Operatori logici bitwise

- L'operatore NOT bitwise \sim esegue la negazione dei bit:

```
int a=60 // = 0011 1100
```

```
c=~a // = 1100 0011 ossia c=-61
```

a	0 0 1 1	1 1 0 0	(60)
<hr/>			
$\sim a$	1 1 0 0	0 0 1 1	(61)

a	$\sim a$
0	1
1	0

Assumiamo per semplicità che gli interi siano codificati con 8bit

Operatori logici bitwise

- L'operatore shift `<<` scorre n bit a sinistra.
Gli spazi vuoti a destra sono riempiti con degli zeri, mentre quelli che escono a sinistra sono eliminati.

```
int a=60 // = 0011 1100
```

```
c=a<<2 // = 1111 0000 ossia c=240
```

a 00 11 1100 (60)

a<<2 1100 0000 (240)

a<<n

n=2

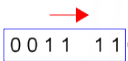
Assumiamo per semplicità che gli interi siano codificati con 8bit

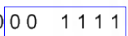
Operatori logici bitwise

- L'operatore shift `>>` scorre n bit a destra.
Gli spazi vuoti a sinistra sono riempiti con degli zeri mentre quelli che escono a destra sono eliminati.

```
int a=60 // = 0011 1100
```

```
c=a>>2 // = 0000 1111 ossia c=15
```

a  (60)

a>>2 00  (15)

a>>n

n=2

Assumiamo per semplicità che gli interi siano codificati con 8bit

Operatori logici bitwise

- Un esempio di utilizzo degli operatori bitwise per controllare gli attributi (8bit) di un testo visualizzato:

Italic	Bold	Blinking	Underline	Dbi-Underline	Future Use	Future Use	Future Use	Attribute
7	6	5	4	3	2	1	0	Byte Position
128	64	32	16	8	4	2	1	Value

Operatori logici bitwise

- Un esempio di utilizzo degli operatori bitwise per controllare gli attributi (8bit) di un testo visualizzato:

```
short text1=0;
System.out.print("Inserisci un valore per text1:");
text1=(short)SIn.readLineInt();
System.out.println("text1: "+Integer.toBinaryString(text1));
short MASKCORSIVO=1<<7; //128
short MASKGRASSETTO=1<<6; //64
short MASKBLINK=1<<5; //32
short MASKBSOTTOLINEATO=1<<4; //16

if ((text1 & MASKCORSIVO)>0)
    System.out.println("CORSIVO ON");
else
    System.out.println("CORSIVO OFF");
if ((text1 & MASKGRASSETTO)>0)
    System.out.println("GRASSETTO ON");
else
    System.out.println("GRASSETTO OFF");

if ((text1 & MASKBLINK)>0)
    System.out.println("LAMPEGGIANTE ON");
else
    System.out.println("LAMPEGGIANTE OFF");

if ((text1 & MASKBSOTTOLINEATO)>0)
    System.out.println("SOTTOLINEATO ON");
else
    System.out.println("SOTTOLINEATO OFF");
```