
OPERATORI DI BITWISE (su singoli bit)

Gli operatori di bitwise (che operano sui singoli bit) sono i seguenti:

- "**&**" AND
- "**|**" OR
- "**^**" XOR
- "**~**" Complemento a 1 (0=>1, 1=>0)
- "**<<**" shift a sinistra
- "**>>**" shift a destra

Nota: fare attenzione a non confondere & con && (& e' "bitwise and", mentre && e' "logical and"); la stessa cosa vale per | e ||.

"~" e' un operatore unario, cioe' opera su un solo argomento indicato a destra dell'operatore.

Nelle operazioni di shift **l'operando destro** deve essere **positivo:**

```
x<<5;
y>>2
```

L'operatore **<<** è l'operatore di shift (**scorrimento**) **a sinistra**, cioè scorre tutti i bit dell'operando a sinistra di tante posizioni quanto indicato dall'operando destro:

```
int x=1;   //in binario 0000 0000 0000 0000 0000 0000 0000 0001
int y=x<<2           // 0000 0000 0000 0000 0000 0000 0000 0100 = 4
```

L'operatore **>>** effettua invece lo **shift a destra**, inserendo a sinistra dei bit, con la cosiddetta estensione di segno: nel caso in cui il valore dell'operando da scorrere sia **positivo vengono inseriti dei bit 0**, in caso contrario, **valore negativo, vengono inseriti degli 1**:

```
int x=15;      //in binario 0000 0000 0000 0000 0000 0000 0000 1111
int y=x>>2;              // 0000 0000 0000 0000 0000 0000 0000 0011 = 3
x=-8;          // 1111 1111 1111 1111 1111 1111 1111 1000
y=x>>2;        // 1111 1111 1111 1111 1111 1111 1111 1110 = -2
```

```
z<<2 shift  dei bit in z di due posti verso sinistra
     cosi',          se z=00000010 (binario) o 2 (decimale)
     allora,  z=z>>2 => z=00000000 (binario) o 0 (decimale)
     inoltre, z=z<<2 => z=00001000 (binario) o 8 (decimale)
```

Quindi, uno shift a sinistra e' equivalente ad una moltiplicazione per 2; similmente, uno shift a destra equivale ad una divisione per 2.

Nota: l'operazione di shift e' molto piu' veloce della reale moltiplicazione o divisione.

Altri appunti al link: http://www.ateneonline.it/hyperbook/j_book/J0204_4.htm