Operatori del C per lavorare sui bit

Il C mette a disposizione gli operatori booleani fondamentali per operare sui bit. Anche se le operazioni booleane che questi operatori implementano sono le stesse degli operatori logici, occorre non confondere le due categorie in quanto le entità su cui lavorano sono diverse.

Gli operatori logici che lavorano sui bit esaminano ogni bit dei due operandi (nel caso di operatori binari AND, OR e XOR) o dell'operando (nel caso dell'operatore unario NOT) e applicano le regole qui sotto riportate **ordinatamente a ogni coppia di bit**.

AND (&)

A	**B**	**A & B**
0	0	0
0	1	0
1	0	0
1	1	1

OR (|)

A	**B**	**A | B**
0	0	0
0	1	1
1	0	1
1	1	1

exclusive OR: XOR (^)

A	**B**	**A ^ B**
0	0	0
0	1	1
1	0	1
1	1	0

NOT (~) – operatore UNARIO

A	**~A**
0	1
1	0

Manipolazione di bit con gli operatori del C

assegnamento

```
i=0;
```

risultato

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

alzare uno specifico bit: OR con maschera opportuna

```
i = i | 1;
```

maschera usata: 1 (0x0001)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

risultato

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

moltiplicare per 2^N: shift a sinistra di N posizioni

```
i = i << 2;
```

risultato

0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

dividere per 2^N: shift a destra di N posizioni

```
i = i >> 1;
```

risultato

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Mantenere invariati gli 8 bit bassi e azzerare tutti gli altri: **AND** con maschera opportuna

```
i = i & 255;
```

maschera usata: 255 (0x00ff)

0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

risultato

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Mantenere invariati gli 8 bit alti e azzerare tutti gli altri: **AND** con maschera opportuna

```
i = i & (255 << 8);
```

maschera usata: 255 << 8 (255, cioe' 0x00ff, spostato a sin. di 8 pos., ossia 0xff00)

1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

risultato

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

```
i = 14;
```

risultato

0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

OR esclusivo (exclusive OR: abbreviato "XOR")

```
i = i ^ 2;
```

maschera usata: 2 (0x0002)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

risultato

0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

Scrivere un valore V di 2 bit nei bit 6 e 7 di un intero X a 16 bit

```
int v=3;
int x=0;
x = x | (v << 6);
```

v

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

x prima dell'operazione

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

x dopo l'operazione

0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Come estrarre da x il valore di 2 bit contenuto nei bit 6 e 7 senza influenzare gli altri bit di X?

Un possibile procedimento:

1) fare una copia di X e lavorare su essa
2) azzerare tutti i bit che non c'entrano (da 0 a 5 e da 8 a 15)
3) shift a destra di 6 posizioni

y (copia di x)

0	0	1	0	1	1	0	1	1	0	0	1	0	1	1	0

```
y = y & 0x00C0;
```

maschera usata: 0x00C0

0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

i bit che non interessano sono stati azzerati

0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

```
y = y >> 6;
```

i bit che interessano sono stati portati a destra

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

oppure:

1) fare una copia di X e lavorare su essa
2) shift a destra di 6 posizioni
3) azzerare i bit da 2 a 15

y (copia di x)

0	0	1	0	1	1	0	1	1	1	0	1	0	1	1	0

```
y = y >> 6;
```

i bit che non interessano sono stati azzerati

0	0	0	0	0	0	0	0	1	0	1	1	0	1	1	1

```
y = y & 0x0003;
```

maschera usata: 0x0003

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

i bit dal 2 in su sono stati azzerati

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1