

# 22. Operatori

Corso di Informatica

Programmazione

Corso di Laurea in Matematica (D.M. 270/04) - A.A. 2020/2021

Angelo Cardellicchio

[angelo.cardellicchio@uniba.it](mailto:angelo.cardellicchio@uniba.it)

18/11/2020

# Outline

- Definizioni e regole generali
- Operatore di assegnazione
- Operatori matematici
- Cenni di logica booleana
- Operatori binari
- Operatori relazionali
- Operatori logici
- Conversione di tipo

# Definizioni e regole generali

- Un **operatore** agisce su coppia di dati (**binario**) o su un dato singolo (**unario**)
- Un'**espressione** è una sequenza di operatori regolata dai principi di precedenza ed associatività
- La **precedenza** vale solo in caso di più operatori, e va da sinistra verso destra
  - *È comunque assicurata nel caso si usino delle parentesi tonde*

a op\_1 b op\_2 c  
a op\_1 (b op\_2 c)

- L'**associatività** indica l'ordine in cui sono valutati gli operandi (i dati)
  - *Si va quasi sempre da sinistra (**l-value**) verso destra (**r-value**)*

# Operatore di assegnazione

- L'operatore di **assegnazione** ci permette di assegnare un valore ad una variabile
- Viene usato in fase di **inizializzazione**

**a = 10;**

**c = 'A';**

- **Non** serve per valutare il valore di una variabile, ma solo per assegnarne uno nuovo!
  - *Ergo, un'espressione del tipo `if (a = 10)` non funzionerà!*

# Operatori matematici (1)

- Sono gli operatori fondamentali coinvolti nelle operazioni di tipo aritmetico
  - **Esercizio 1:** scriviamo un programma che calcoli il quadrato di un numero
  - **Esercizio 2:** scriviamo un programma che determini se un numero è pari
  - Suggestimento: usare `==` per valutare il valore di una variabile
  - Suggestimento: usare l'header `stdbool.h` per i valori booleani

Operatore	Descrizione
+	Somma di due numeri
-	Differenza tra due numeri
*	Moltiplicazione di due numeri
/	Rapporto tra due numeri
%	Modulo

# Operatori matematici (2)

- **Esercizio 1:** scriviamo un programma che calcoli il quadrato di un numero

```
// quadrato
#include <stdio.h>

int quadrato(int n) {
    return n * n;
}

int main() {
    int a = 3;
    printf("Il quadrato di %d è %d\n", a, quadrato(a));
}
```

# Operatori matematici (3)

- **Esercizio 2:** scriviamo un programma che determini se un numero è pari

```
// pari
#include <stdbool.h>
#include <stdio.h>

bool numero_pari(int n) {
    if (n % 2 == 0) {
        return true;
    }
    return false;
}

int main() {
    int pari = 4;
    int dispari = 5;
    printf("Il valore di %d è pari? %d\n", pari,
        numero_pari(pari));
    printf("Il valore di %d è pari? %d\n", dispari,
        numero_pari(dispari));
}
```

# Cenni di logica booleana

- Governa le interazioni tra i valori booleani (**true** e **false**)
  - Ricordare che `true == 1`, e `false == 0`
- Sono operatori **binari**
- Alcune regole:
  - L'**AND** restituisce **true** se i due operandi sono **true**
  - L'**OR** restituisce **true** se almeno un operando è **true**
  - Lo **XOR** (**exclusive OR**) restituisce **true** se solo un operando è **true**

A	B	AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0



# Operatori binari (1)

- Operano a livello di **bit**
- Possono:
  - Effettuare uno **shift** (scorrimento) verso sinistra o destra
  - Effettuare operazioni binarie bit a bit
- Supponiamo di avere una variabile di tipo **byte**
  - $a = 4 = 2^2 = (00000010)_2$
- Applicando gli operatori di scorrimento:
  - $a \gg 1 \Rightarrow a = (00000001)_2 = 2^1 = \frac{a}{2^1}$
  - $a \ll 1 \Rightarrow a = (00000100)_2 = 2^3 = a \cdot 2^1$

Operatore	Descrizione
$\gg$	Right shift
$\ll$	Left shift
$\&$	AND bit a bit
$ $	OR bit a bit
$\wedge$	XOR bit a bit

# Operatori binari (2)

- Supponiamo ora di avere due **char**
- Vediamo cosa succede applicando gli operatori binari logici

Operatore	a = 10001011 b = 01011010
&	r = 00001010
	r = 11011011
^	r = 11010001

Operatore	Descrizione
>>	Right shift
<<	Left shift
&	AND bit a bit
	OR bit a bit
^	XOR bit a bit

# Operatori relazionali (1)

- Confrontano due operandi di tipo primitivo
- Restituiscono un valore booleano (**true** o **false**)
  - **Esercizio 3:** scriviamo un programma che confronta due intervalli di valori del tipo  $[a, b]$  e  $[c, d]$ , con  $a, b, c$  e  $d$  numeri interi. Il programma deve stampare a schermo il maggiore tra gli estremi inferiori  $a$  e  $c$ , il minore tra gli estremi superiori  $b$  e  $d$  e stabilire se il numero di elementi nei due intervalli è lo stesso.

Operatore	Descrizione	Esempio
>	True se $a$ è maggiore di $b$	$a > b$
>=	True se $a$ è maggiore od uguale a $b$	$a >= b$
<	True se $a$ è minore di $b$	$a < b$
<=	True se $a$ è minore od uguale a $b$	$a <= b$
==	True se $a$ è uguale a $b$	$a == b$
!=	True se $a$ è diverso da $b$	$a != b$

## Operatori relazionali (2)

- **Esercizio 3:** scriviamo un programma che confronta due intervalli di valori del tipo  $[a, b]$  e  $[c, d]$ , con  $a, b, c$  e  $d$  numeri interi. Il programma deve stampare a schermo il maggiore tra gli estremi inferiori  $a$  e  $c$ , il minore tra gli estremi superiori  $b$  e  $d$  e stabilire se il numero di elementi nei due intervalli è lo stesso.

# Operatori relazionali (3)

```
void compara_intervalli(int a, int b, int c, int d) {  
    int max_estremo_inferiore = a;  
    int min_estremo_superiore = b;  
    if (c > a) {  
        max_estremo_inferiore = c;  
    }  
    if (d < b) {  
        min_estremo_superiore = d;  
    }  
    printf("Il maggiore tra gli elementi e' %d\n",  
        max_estremo_inferiore);  
    printf("Il minore tra gli elementi e' %d\n",  
        min_estremo_superiore);  
    if ((b - a) != (d - c)) {  
        printf("Gli intervalli non hanno lo stesso numero  
            di elementi\n");  
    } else {  
        printf("Gli intervalli hanno lo stesso numero di  
            elementi\n");  
    }  
}
```

# Operatori logici

- Agiscono su operandi booleani e restituiscono un valore logico
- L'AND e l'OR sono operatori binari, il NOT è unario
- Da **non** confondere con gli operatori binari
  - **Esercizio 4:** dati gli intervalli visti nell'esercizio 3, scrivere un programma che indichi se i due intervalli hanno lo stesso numero di elementi e gli estremi degli stessi coincidono, oppure se solo una di queste condizioni è verificata. Usare solo operatori logici.

Operatore	Descrizione	Esempio
&&	AND logico tra a e b	a && b
	OR logico tra a e b	a    b
!	NOT logico	!a

# Conversione di tipo

- La conversione di tipo può essere **implicita** ed **esplicita**
- La conversione **implicita** avviene:
  - *con gli operatori matematici, semplificando i tipi più complessi (ad esempio, un float viene convertito in int, un int in char, e così via);*
  - *con gli operatori di assegnazione, nei quali l'operando di sinistra assume il tipo dell'operando di destra;*
  - *eventuali errori di saturazione (overflow) **non** vengono segnalati!*
- La conversione **esplicita** avviene mediante l'operatore di **casting**
  - *Questo indica il nuovo tipo tra parentesi, davanti al nome della variabile da trasformare*

# Conversione di tipo

- Alcuni esempi di conversione implicita:

```
int a = 3.2 + 4.2;           // il risultato sarà convertito da float ad int
int b = "c"                  // il risultato sarà convertito da char ad int
short c = 9999999           // il valore di c sarà comunque 32767!
```

- Alcuni esempi di conversione esplicita:

```
int a = 3;
int b = 4;
float c;
c = (float) b/a;
```

- In generale, **evitare le conversioni di tipo implicite...**
  - ...e ricorrere a quelle esplicite solo quando necessario!*



# Domande?

42