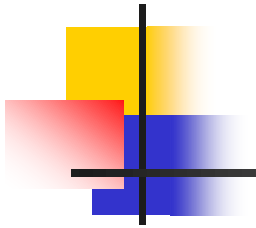
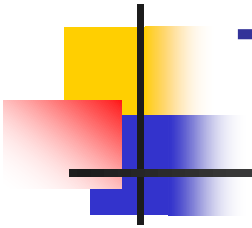
Espressioni in Linguaggio C



Concetti chiave

- Espressioni
- Conversioni Implicite
- Operatori aritmetici
- Operatore di assegnamento
- Operatori relazionali
- Operatori logici
- Gerarchia e associatività degli operatori
- Espressioni condizionali con l'operatore ?:
- Parentesi tonde
- Cast



Tipologie di espressioni

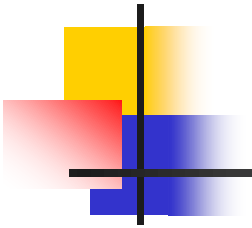
- Espressione: combinazione di variabili ed operatori
  - Aritmetici, Relazionali e Logici, Assegnazione, Incremento/Decremento unitario, Condizionali e Virgola
- Caratteristiche:
  - precedenza
  - associatività (sin->des o des->sin)
  - e arità (numero operandi:1, 2 o 3)
- Attenzione alle conversioni implicite !

Operatori aritmetici

- Due operandi, tranne la negazione (-) che è unario
- La priorità degli operatori può essere alterata mediante le parentesi tonde.
- Gli operatori aritmetici sono associativi da sinistra verso destra.
- L'**operatore di modulo**, % restituisce il resto della divisione intera. **Si applica solo ad interi e produce un intero**
  - Esempio: se `x` vale 5, `34 % x` vale 4

Operatori aritmetici

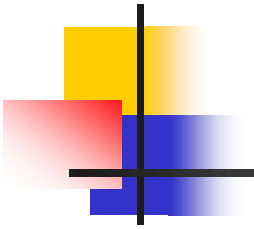
- +, -, *, / si applicano a qualunque tipo di dato (overloading)
- **Risultato dell'espressione dipende dal tipo degli operandi**
- Se gli operandi appartengono allo stesso tipo, allora il risultato è dello stesso tipo
- **Se gli operandi appartengono a tipi diversi, si ha una promozione di tipo (conversione implicita)**
- Esempio: 3/2=1, 3/2.0=1.5



Operatori aritmetici- Conversione Implicita

Esiste una priorità di tipi:

- Lunghezza maggiore
- Virgola mobile
- Senza segno



Operatori aritmetici- Conversione Implicita

1. Tutti i char e short int sono "promossi" ad int
2. Se un operando è long double, l'altro viene convertito in long double
3. Se un operando è double, l'altro viene convertito in double
4. Se un operando è float, l'altro viene convertito in float
5. Se un operando è unsigned long int, l'altro viene convertito in unsigned long int
6. Se un operando è long int, l'altro viene convertito in long int
7. Se un operando è unsigned int, l'altro viene convertito in unsigned int
8. Altrimenti entrambi sono int e il risultato è int

Esempi Conversione Implicita

```
char g = 'F';
short int s = 345;
int i_res = g + s; // g e s promossi in int

int a = 11;
double d = 33.455;
double d_res = a + d; // a promosso in double

unsigned long int ul = 1234567UL;
unsigned int ui = 213213U;
unsigned long int ul_res = ul + ui; // ui è promosso in unsigned long

unsigned long int ul_2 = 3234567UL;
long int li = 2400000L;
unsigned long int ul_res_2 = ul_2 + li; // li è promosso in unsigned long

long int li_2 = 4678989L;
unsigned int ui_2 = 5678999U;
long int li_res = li_2 + ui_2; // ui_2 è promosso in long int
```

Esempi Conversione Implicita

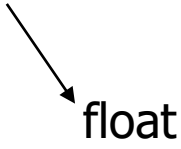
- unsigned long int x=100;
- int y=-101;
- x+=y


```
#include<stdio.h>

unsigned long int x=100;
int y=-101;

int main(void){
   x+=y;
   printf("\nEspressione %u  ",x);
}
```

- Risultato: unsigned long 4.294.967.295

Esempi Conversione Implicita

- int x=2;
- float y=3;
- y/x



- Risultato: float = 1.5

```
#include<stdio.h>

int x=335, y=10;
float z,k=335;

int main(void){
    z=x/y;
    printf("\nDivisione z=x/y=%f  ",z);
    z=k/y;
    printf("\nDivisione z=k/y=%f  ",z);
}
```

Operatori relazionali e logici

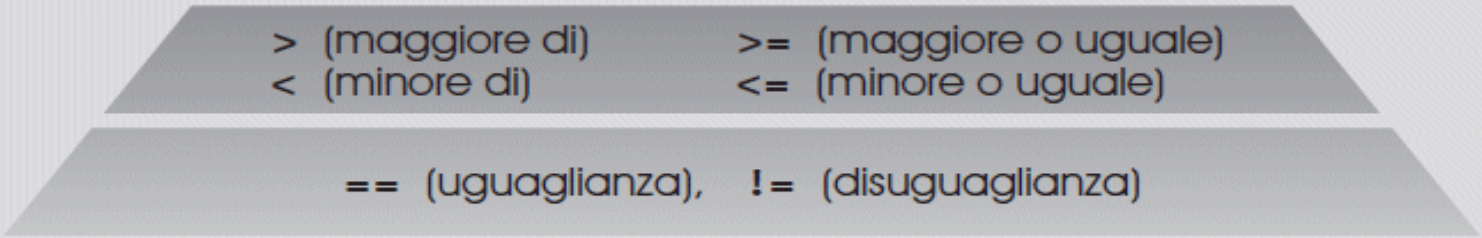


Figura 8.2 *Gerarchia degli operatori relazionali.*

- 2 operandi, da sinistra a destra
- L'operatore di uguaglianza == è diverso da quello di assegnamento =
- La priorità di >, >=, < e <= è la stessa e maggiore di quella di == e !=
- In: **`x>y==z>t`**, vengono valutati prima `x>y` e `z>t`, successivamente viene verificata l'uguaglianza tra i due risultati come se l'espressione fosse `(x>y)==(z>t)`.

Operatori relazionali e logici

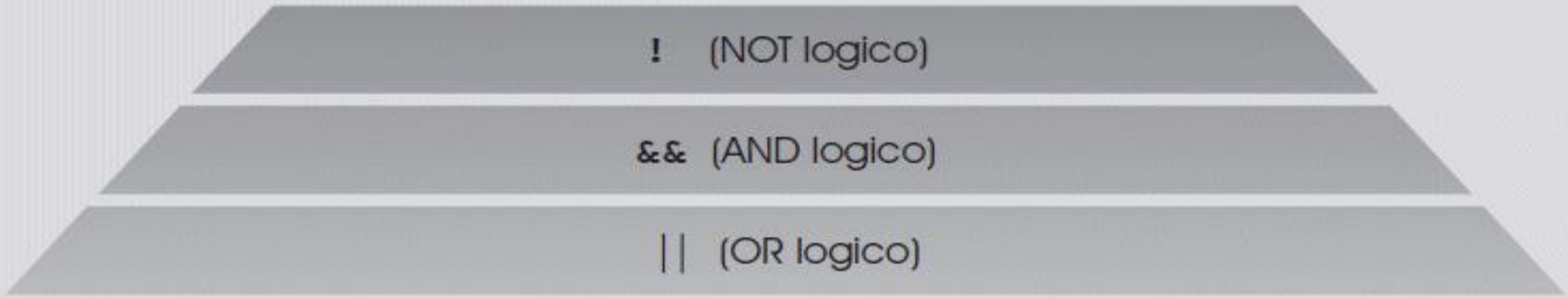


Figura 8.3 *Gerarchia degli operatori logici.*

x	y	x&&y	x||y	!x
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Tavola di verità degli operatori logici, dove `0` corrisponde a **falso** e `1` a **vero**

Operatori relazionali e logici

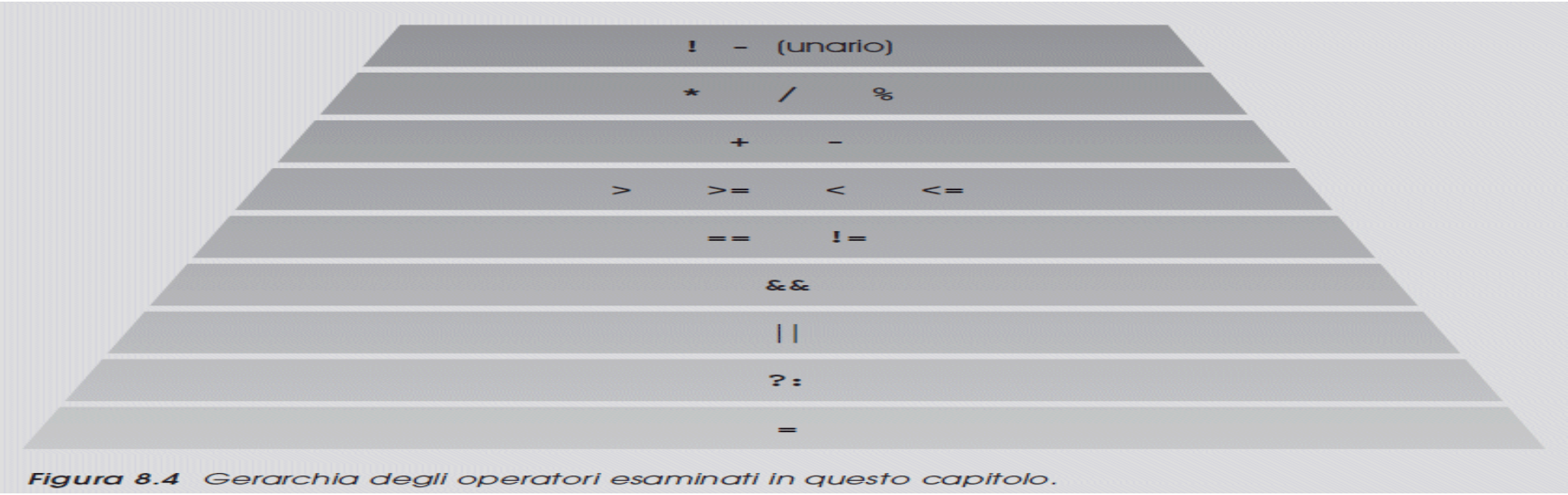


Figura 8.4 Gerarchia degli operatori esaminati in questo capitolo.

- **x==y && a>b** è vera se x è uguale a y e contemporaneamente a è maggiore di b
- **b<c || t!=r** è falsa solo se b non è minore di c e contemporaneamente t è uguale a r
- **!y** restituisce vero se y è falso e viceversa
- **i>n && x!=y** è vera se i>n e contemporaneamente x e diversa da y
- **!x && y>b || c!=3** è vera se x è 0 e contemporaneamente si verifica almeno una delle due: y è maggiore di b, OPPURE se c è diverso da 3
- **x && a>b || c <=d && f!=r** è vera se x && a>b OPPURE se c <=d && f!=r è vera

Operatore di assegnazione

- 2 operandi, da DESTRA verso SINISTRA
- **Conversione Implicita:** Valore dell'espressione=valore dell'espressione di destra convertito nel tipo di sinistra
- x=x+3
- x=y=z=5
- Esempio

```
#include<stdio.h>
float z=3.4, x;
int y;

int main(void){
        printf("\nEspressione x=y=z= %f  ",x=y=z);
}
```

Operatore di assegnazione

- E' possibile combinare l'assegnazione con gli operatori aritmetici:
  - +=, -=, /=, *=, %= x=x/z x/=z
  - X+=3, x-=y, x*=2
- Per gli incrementi e decrementi unitari esistono ++ e –
  - Forma prefissa ++x, --x
  - Forma postfissa x++, x --
  - Differenza: x=y++, x=++y
  - Esempio:

```
int x=4, y;
y=x++;      y=4 e x=5
y=++x;      y=5 e x=5
```

Espressioni condizionali

- Una *espressione condizionale* si ottiene con l'operatore ternario `?:`

  espr1 `?` *espr2* `:` *espr3*

  se *espr1* è vera, il risultato è *espr2*, altrimenti è *espr3*.

  Esempio: `x==y ? a : b`

  significa: "se `x` è uguale a `y`, allora `a`, altrimenti `b`"

- Si può utilizzare l'operatore ?: per assegnare un valore a una variabile

```
v= x==y ? a*c+5 : b-d;
```

espr1, *espr2*, *espr3* sono valutate prima di `?:`, l'assegnamento per ultimo

- Può essere inserita in qualsiasi posizione sia lecito scrivere un'espressione:

```
printf("%d maggiore o uguale a %d", (a>b?a:b), (a<=b?a:b));
```

Priorità

- Si può stravolgere l'ordine delle priorità utilizzando le parentesi rotonde
- a+b/3 oppure (a+b)/3
- x> 23 && y> 5 || z==3 oppure x>23 && (y>5 || z==3)

Operatore di Cast

- Si possono inibire le regole di conversione implicita utilizzando l'operatore di cast
- (tipo)espressione, l'espressione viene convertita nel tipo desiderato
- Esempio:

```
int x,y;

x/y -> intero

(double)x/y oppure x/(double)y -> reale
```

Operatore di Cast

- Esempio:

```
unsigned long x=100;
int y=-101;
```

x+y -> 4.294.967.295 (unsigned)

x+(float)y -> -1 (float)