



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione



# 1. Funzioni Ricorsive in C

3 esercizi risolti + 1 extra

Tutorato di  
Programmazione Avanzata

RELATORE  
Imberti Federico

SEDE  
Dalmine, BG

# Key-points della teoria

1. Una funzione può essere **iterativa**, **ricorsiva** senza tail recursion (**TR**) o ricorsiva con TR;
2. TR è un'ottimizzazione fatta dal compilatore per riutilizzare lo stesso stack frame risparmiando in spazio ed efficienza;
3. Affinché una funzione sia TR è necessario che non ci siano altre istruzioni dopo la chiamata ricorsiva, altrimenti il compilatore non può scartare lo **stack frame** i-esimo e deve per forza tenerlo in memoria creandone uno per ogni chiamata;
4. Ci sono due modi per passare parametri a una funzione:
  - a. **Pass-by-value**: copia del right-value;
  - b. **Pass-by-reference**: left-value



**UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO**

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione

# Domandine?

# Esercizio 1/3

Scrivi una funzione `COUNT_GT` in C che dato in ingresso due array di interi `a1` e `a2`, di lunghezza anche diversa, conta quanti elementi del primo array (`a1`) sono maggiori dell'elemento in uguale posizione in `a2`. Se in `a2` non c'è l'elemento, viene considerato 0.

Per esempio

`COUNT_GT ({2,3},{3,1})`  $\rightarrow 2 < 3$  e  $3 > 1 \implies > 1$

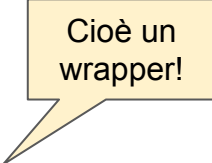
`COUNT_GT ({2,3,4},{3,1})`  $\rightarrow 2 < 3$  e  $3 > 1$  e  $4 > 0 \implies > 2$

Scrivi tre versioni:

- Una iterativa non ricorsiva
- Una ricorsiva senza tail call
- Una ricorsiva con tail call.

Scrivi anche un main di esempio in cui chiami le funzioni con qualche array.

Non usare alcuna variabile globale. Cerca di tenere il più semplice possibile la segnatura delle funzioni ricorsive, ma se non riesci fai una funzione con segnatura semplice che chiami quella ricorsiva.



Cioè un wrapper!

## Esercizio 2/3

Scrivi una funzione **mix\_strings** che date in ingresso due stringhe A e B, **restituisce** una stringa che fa il mix di A e B mettendo un carattere di A e uno di B alternati. Se una stringa delle due è più lunga, la parte avanzata viene tagliata.

Ad esempio

**mix\_strings("cane","nero") = "cnaenreo"**

**mix\_strings("ciao","marianna") = "cmiaaroi"**

Scrivi tre versioni:

- Una iterativa non ricorsiva
- Una ricorsiva senza tail call
- Una ricorsiva con tail call.

Scrivi anche un main di esempio in cui chiami le funzioni con almeno le stringhe di cui sopra.

Non usare alcuna variabile globale. Cerca di tenere il più semplice possibile la segnatura delle funzioni ricorsive, ma se non riesci fai una funzione con segnatura semplice che chiami quella ricorsiva.

## Esercizio 3/3

Scrivi una funzione **sort** che riceve un array di interi e **restituisce** una copia dell'array ordinato.

(a) Scrivi tre versioni:

- Una iterativa non ricorsiva
- Una ricorsiva senza tail call
- Una ricorsiva con tail call.

(b) Scrivi anche un main di esempio in cui chiami le funzioni con almeno gli array di cui sopra.

Non usare alcuna variabile globale. Cerca di tenere il più semplice possibile la segnatura delle funzioni ricorsive, ma se non riesci fai una funzione con segnatura semplice che chiami quella ricorsiva.

# Esercizio Extra

Scrivi una funzione in C che dato in ingresso un array di int restituisce il numero di elementi pari contenuti nell'array. Scrivi tre versioni: una non ricorsiva, una ricorsiva senza tail recursion e una ricorsiva con tail recursion. Specifica esattamente i parametri che passi alla procedura, il tipo di passaggio utilizzato e il loro significato. Descrivi anche le assunzioni che fai (ad esempio zero terminated o cose simili).

Scrivi anche un main di esempio in cui chiami le funzioni con un array di tua scelta con 10 elementi. L'array deve essere dichiarato come variabile nel main.

Non usare variabili globali.