TUTORATO 09

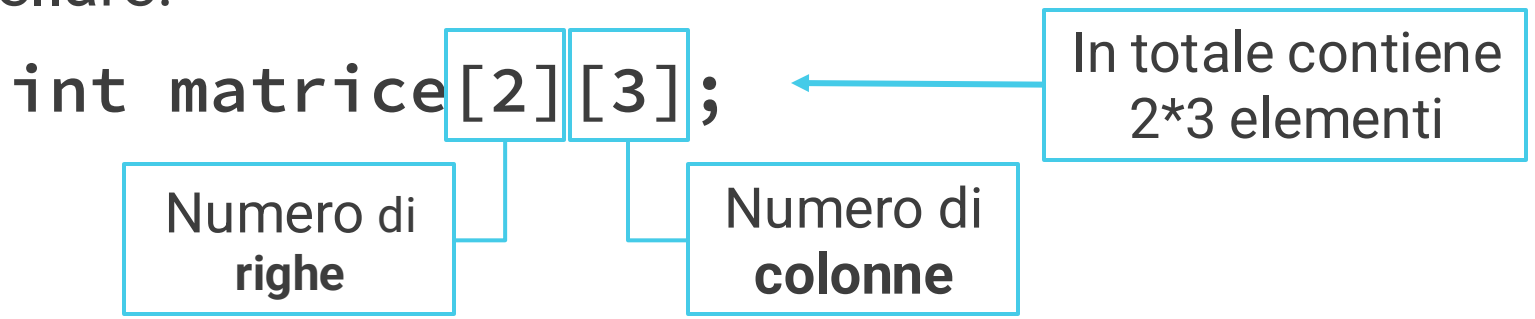
PROGRAMMAZIONE E LABORATORIO A.A 2025-2026


Dipartimento
di Matematica
e Informatica

Tutor: Dott. Dominik Miotla

ARRAY MULTIDIMENSIONALI

Gli array multidimensionali sono array di array, utili a memorizzare dati dello stesso tipo in formato tabellare:



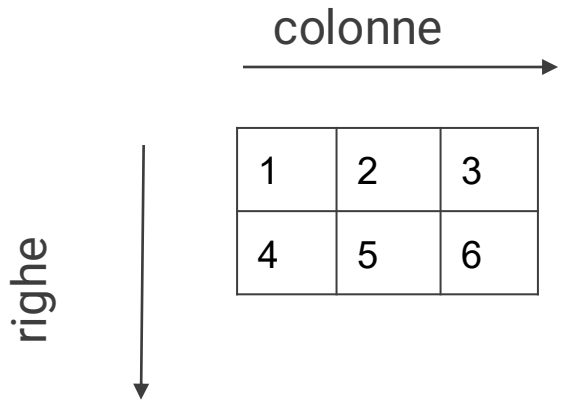
Per accedere agli elementi:

```
for(int rowIndex=0; rowIndex<2;rowIndex++){
    for(int colIndex=0; colIndex<3;colIndex++){
        // ...
    }
}
```

ARRAY MULTIDIMENSIONALI

La dichiarazione di un array multidimensionale in C corrisponde ad una matrice.

```
int array[2][3] = {
    {1, 2, 3},
    {4, 5, 6}
    };
```



Ricorsione

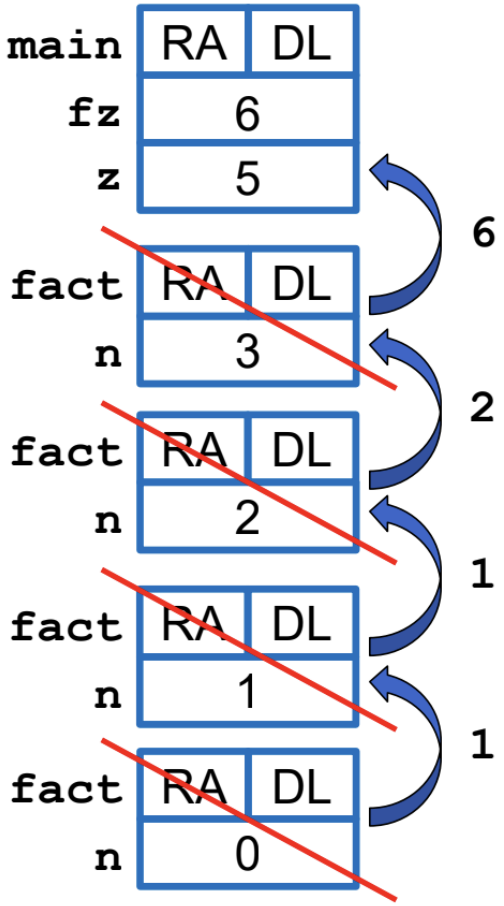
Risolvere un problema con un approccio ricorsivo comporta:

1. identificare un “caso base” la cui soluzione sia nota;
2. riuscire a esprimere la soluzione al caso generico ***n*** in termini dello stesso problema in uno o più casi più semplici ***(n-1, n-2, etc)***.

Ricorsione esempio

```
int fact(int n){
    if (n==0)
        return 1;
    else
        return n*fact(n-1);
}

main(){
    int fz, z = 5;
    fz = fact(z-2);
}
```



malloc: Allocazione dinamica di memoria

1. Cos'è malloc? Una funzione della libreria standard C (<stdlib.h>) che permette di **allocare memoria dinamicamente** durante l'esecuzione del programma. Restituisce un **puntatore a void** (void*) che può essere convertito al tipo desiderato.

2. Perché usarla? Utile quando la quantità di memoria necessaria **non è nota a tempo di compilazione**. La memoria allocata **rimane valida** finché non viene liberata esplicitamente.

- Sintassi : int* arr = (int*) malloc(10 * sizeof(int));

- Controllo dell'allocazione if (arr == NULL) { // errore: memoria non disponibile }

free: Deallocazione della memoria

3. Perché serve free? La memoria allocata con malloc **non viene liberata automaticamente**. Usare free evita **memory leaks**, cioè porzioni di memoria che rimangono allocate inutilmente.

- **Sintassi:** free(arr);

4. Regole fondamentali:

- Liberare **solo memoria allocata dinamicamente**.
- Non chiamare free due volte sullo stesso puntatore (**double free**).
- Dopo free, è buona pratica impostare il puntatore a NULL: free(arr); arr = NULL;

free: Esempio completo

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    int* arr = malloc(10 * sizeof(int));

    if (arr != NULL) {
        // uso dell'array
        free(arr);
        arr = NULL;
    }
}

```

ESERCIZIO 1: SOMMA DI MATRICI

1) Scrivere una funzione in C che dati due array multidimensionali ne restituisca la somma
2) scrivere una funzione *stampaSomma* che stampi l'array multidimensionale finale
3) separare l'esecuzione del programma nei file matrici.c matrici.h main.c
4) scrivere il makefile ed eseguire il programma

2	3	6
5	1	2
5	5	5

+

9	9	9
2	3	4
8	7	3

=

11	12	15
7	4	6
13	12	8

ESERCIZIO 2: ORARIO

Supponendo di avere 4 slot da 2 ore ciascuno dalle 9 alle 18 (con pausa pranzo dalle 13 alle 14) per 5 giorni, scrivere un programma che permetta di gestire il proprio orario settimanale delle lezioni (stringhe). L'orario viene innanzitutto inizializzato a "vuoto":

GIORNO	SLOT 1	SLOT 2	SLOT 3	SLOT 4
Lunedì	Vuoto	Vuoto	Vuoto	Vuoto
Martedì	Vuoto	Vuoto	Vuoto	Vuoto
Mercoledì	Vuoto	Vuoto	Vuoto	Vuoto
Giovedì	Vuoto	Vuoto	Vuoto	Vuoto
Venerdì	Vuoto	Vuoto	Vuoto	Vuoto

In un ciclo che va avanti finché l'utente non sceglie 'q', il programma stampa il menu:

a. Stampare l'orario completo (come nell'immagine)
b. Stampare l'orario di un solo giorno (chiesto in input all'utente)
c. Modificare l'orario (inserendo una lezione in un determinato slot di un determinato giorno, chiesti entrambi in input all'utente)

ESERCIZIO 3: Somma

Si scriva una funzione ricorsiva:

int somma(int a[], int n);

che calcola la somma degli elementi dell'array a dall'indice 0 fino all'indice n

ESERCIZIO 4: Massimo

Si scriva una funzione ricorsiva

int massimo(int a[], int n);

che calcola il massimo degli elementi dell'array a dall'indice 0 fino all'indice n

ESERCIZIO 5: Allocazione dinamica array

Scrivi un programma che:

1. Chieda all'utente la dimensione N di un array (intero positivo);
2. Allochi dinamicamente un array di interi di dimensione N;
3. Popoli l'array con i primi N numeri pari (0, 2, 4...);
4. Stampi l'array a video.
5. Liberi correttamente la memoria.

ESERCIZIO 6: Strutture e allocazione dinamica

1. Definisci una struct Prodotto con:
    - codice (int)
    - prezzo (float)
2. Nel main, chiedi all'utente quanti prodotti vuole inserire K.
3. Alloca dinamicamente un array di struct Prodotto.
4. Usa un ciclo per far inserire all'utente i dati di ogni prodotto.
5. Calcola e stampa la media dei prezzi.
6. Libera la memoria.