UNIVERSITÀ
DI TORINO

Laboratorio di Programmazione I

Lezione n. 8: Matrici

Alessandro Mazzei

Slides; prof. Elvio Amparore

Outline

- Dimensionamenti di array e matrici
- rettangolari.c
- Lab08-Es1 Tris
- Lab08-Es2 Righe e colonne pari e non negative
- Matrici ragged
- ragged.c
- iris.c
- Lab08-Es3 Prefissi di righe
- Lab08-Es4 Cambio treni

Dimensionamento di array

Dimensione decisa in fase di compilazione (*compile-time*):

Definizione:

```c
#define LEN  4
⋮
int arr[LEN];
```

Passaggio come argomento:

```c
// Prototipo
void fn(int a[LEN]);
// Chiamata
fn(arr);
```

Dimensione decisa in fase di esecuzione (*runtime*):

Definizione:

```c
size_t len = 4;
int arr[len];
```

Passaggio come argomento:

```c
// Prototipo
void fn(const size_t len, int a[len]);
// Chiamata
fn(len, arr);
```

len deve apparire come argomento prima di a[len]

Formato delle matrici

Consideriamo due **formati** che può assumere una matrice:

- **matrice rettangolare:** struttura dati bidimensionale in cui tutte le righe hanno la stessa lunghezza (cioè lo stesso numero di elementi) e tutte le colonne hanno lo stesso numero di righe.
- **matrice ragged (irregolare):** struttura dati bidimensionale in cui le righe possono contenere un numero diverso di elementi.

Matrice rettangolare

- Dimensione matrice: **ROWS** × **COLS**
- Indici righe: $i \in [0, \mathbf{ROWS})$
- Indici colonne: $j \in [0, \mathbf{COLS})$

		Indici colonne					**COLS**
		0	1	2	3	4	5
Indici righe	0	2	-1	-3	-3	3	NON VALIDO
	1	4	3	2	2	0	NON VALIDO
	2	-2	7	2	2	12	NON VALIDO
	3	6	8	-4	-4	1	NON VALIDO
ROWS	4	NON VALIDO	NON VALIDO	NON VALIDO	NON VALIDO	NON VALIDO	NON VALIDO

Matrice rettangolare: sintassi

Dimensione definita in fase di compilazione (*compile-time*):

Definizione:

```
#define ROWS  4
#define COLS  5
⋮
int mat[ROWS][COLS];
```

Passaggio come argomento:

```
// Prototipo
void fn(int m[ROWS][COLS]);
// Chiamata
fn(mat);
```

Dimensione definita in fase di esecuzione (*runtime*):

Definizione:

```
size_t rows = 4, cols = 5;
int mat[rows][cols];
```

Passaggio come argomento:

```
// Prototipo
void fn(const size_t rows, const size_t cols,
        int m[rows][cols]);
// Chiamata
fn(rows, cols, mat);
```

Array delle righe

Dato una matrice **mat** e un indice di riga **i**, l'espressione **mat[i]** è l'array della riga.

Può quindi essere trattato come un normale array.

```
#define ROWS  4
#define COLS  5
void fn_su_array(int arr[COLS]);
⋮
int mat[ROWS][COLS];
fn_su_array(mat[i]); // i ∈ [0, ROWS)
```

```
void fn_su_array(const size_t cols, int arr[cols]);
⋮
size_t rows = 4, cols = 5;
int mat[rows][cols];
fn_su_array(cols, mat[i]); // i ∈ [0, rows)
```

Array delle righe

Oppure come un puntatore a dati (stessa sintassi per array di dimensione fissa o VLA).

```
void fn_su_ptr(const size_t lenA, int* pA);
```

```
#define ROWS  4
#define COLS  5
⋮
int mat[ROWS][COLS];
fn_su_ptr(ROWS, mat[i]); // i ∈ [0, ROWS)
```

```
size_t rows = 4, cols = 5;
int mat[rows][cols];
fn_su_ptr(cols, mat[i]); // i ∈ [0, rows)
```

Esercizi per matrici rettangolari

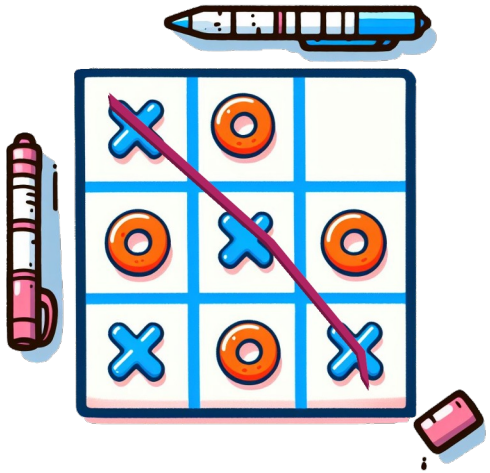
Aprire il file **rettangolari.c** fornito nel codice iniziale, leggere il contenuto ed infine implementare le funzioni dichiarate, seguendo la specifica.

Tris!

Sulla pagina Moodle trovate un esercizio con nome

Lab08-Es1 Tris

Leggere la specifica ed implementare il programma che determina se c'è un giocatore che vince i tris proposti.



Righe e colonne pari

Sulla pagina Moodle trovate un esercizio con nome

Lab08-Es2 Righe e colonne pari e nonnegative

Leggere la specifica ed implementare il programma che determina se ci sono delle righe/colonne interamente pari e nonnegative (≥0).

r_1, c_1		r_2, c_2			

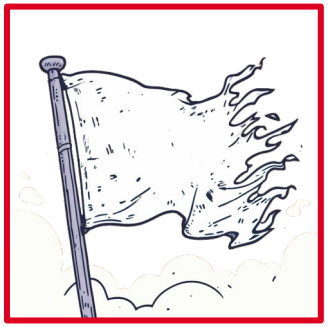
Matrice ragged (irregolare)

- Dimensione matrice: **ROWS** × **COLS**
- Indici righe: $i \in [0, \textbf{ROWS})$
- Indici colonna: $j \in [0, \textbf{rags}[i])$

  (ogni riga i ha un numero indipendente rags[i] di colonne inizializzate)

  $\forall i \in [0, \textbf{ROWS})$ vale che $\textbf{rags}[i] \in [0, \textbf{COLS})$

rags[]		Indici righe		Indici colonne 0	1	2	3	4	COLS 5
4	0		0	2	-1	-3	-3	??	NON VALIDO
0	1		1	??	??	??	??	??	NON VALIDO
5	2		2	-2	7	2	2	12	NON VALIDO
2	3		3	6	8	??	??	??	NON VALIDO
NON VALIDO	4	ROWS	4	NON VALIDO	NON VALIDO	NON VALIDO	NON VALIDO	NON VALIDO	NON VALIDO



Matrice ragged: sintassi

Dimensione definita in fase di compilazione (*compile-time*):

Definizione:

```
#define ROWS  4
#define COLS  5
⋮
int mat[ROWS][COLS];
size_t rags[ROWS];
```

Passaggio come argomento:

```
// Prototipo
void fn(int m[ROWS][COLS], size_t rags[ROWS]);
// Chiamata
fn(mat, rags);
```

Dimensione definita in fase di esecuzione (*runtime*):

Definizione:

```
size_t rows = 4, cols = 5;
int mat[rows][cols];
size_t rags[rows];
```

Passaggio come argomento:

```
// Prototipo
void fn(const size_t rows, const size_t cols,
        int m[rows][cols], size_t rags[rows]);
// Chiamata
fn(rows, cols, mat, rags);
```

Matrice ragged (irregolare)

Aprire il file **ragged.c** e implementare le funzioni richieste.
Verificare che l'output generato rispetti i valori attesi.

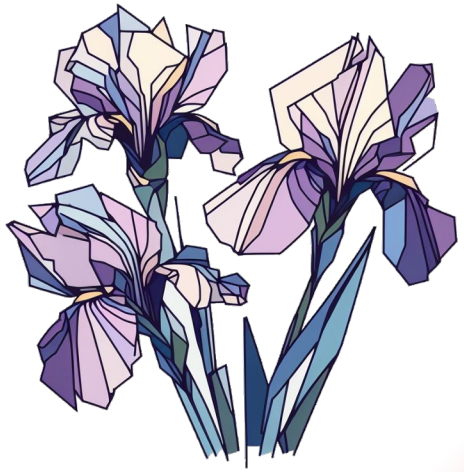
Iris

Completare il file **iris.c**, seguendo i prototipi delle funzioni da implementare. Fare attenzione alla dichiarazione della matrice ragged.

L'output atteso è:

```
Specie              Campioni Media  Estremi
Iris Setosa            23    1.43   (min: 1.00 max: 1.70)
Iris Versicolor        18    4.26   (min: 3.30 max: 4.90)
Iris Virginica         31    5.62   (min: 4.50 max: 6.90)

Dopo eliminazione valori estremi:
Specie              Campioni Media
Iris Setosa            19    1.41
Iris Versicolor        16    4.28
Iris Virginica         29    5.61
```



Prefissi di riga

Sulla pagina Moodle trovate un esercizio con nome

Lab08-Es3 Prefissi di righe

Leggere la specifica ed implementare il programma che determina se ci sono righe della prima matrice che sono prefissi di righe della seconda matrice.

Cambio treni

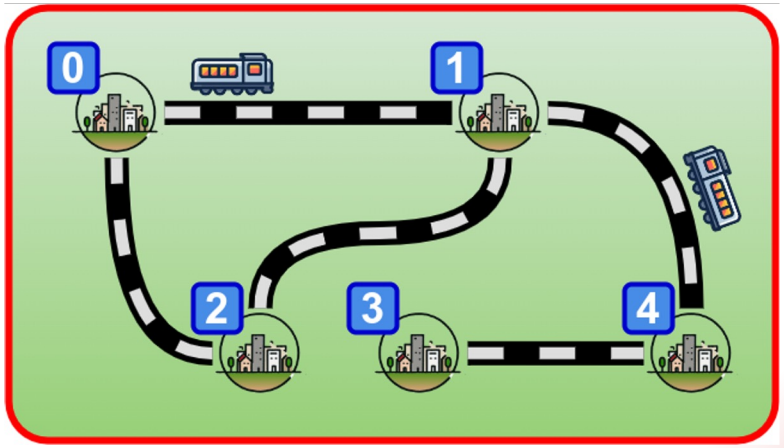
ESERCIZIO EXTRA

Leggere l'esercizio su Moodle.

Lab08-Es4 Cambio treni

Usiamo una **matrice irregolare** per rappresentare le connessioni ferroviarie tra un insieme di città.

Una città **i** è connessa con una città **j** se sulla riga **mat**[**i**] appare il valore **j**. Determinare se coppie di città hanno delle connessioni dirette o con 1 cambio.



Esempi:

- le città **0** raggiunge la città **1** con una connessione diretta;
- la città **2** può raggiungere la città **4** effettuando un cambio;
- la città **0** non raggiunge la città **3** né con un collegamento diretto, né con un cambio.