Tecniche di Programmazione

Esercitazione 5

Usare un main per testare il corretto funzionamento delle funzioni implementate.

La struttura Mat e’ la seguente:

typedef struct {

int rows;

int cols;

float \*\*rows\_pt;

} Mat;

# Matrici

### Esercizio 5.1

Scrivere la funzione

Mat\* mat\_alloc(int rows, int cols);

che, dato in ingresso il numero di righe rows ed il numero di colonne cols, allochi e restituisca una struttura Mat contenente una matrice di dimensione rows x cols. La matrice deve essere memorizzata come array di puntatori alle righe della matrice stessa.

### Esercizio 5.2

Scrivere la funzione

void mat\_free(Mat \*m);

che, data in ingresso una struttura Mat m contenente una matrice, deallochi completamente la matrice m.

### Esercizio 5.3

Scrivere la funzione

void mat\_print(Mat \*m);

che, data in ingresso una struttura Mat m contenente una matrice, stampi la matrice.

### Esercizio 5.4

Scrivere la funzione

Mat \* mat\_clone(Mat \*m);

che, data in ingresso una una una struttura Mat m contenente una matrice, allochi una nuova struttura ed una nuova matrice delle stesse dimensioni di m, vi copi il contenuto di m e ne restituisca il puntatore.

### Esercizio 5.5

Scrivere la funzione

bool mat\_is\_symmetric(Mat \*m);

che data in ingresso una struttura Mat m contenente una matrice, verifichi che m sia simmetrica o meno. Se m e' simmetrica la funzione deve restituire true in uscita, altrimenti deve restituire false. Si ricorda che una matrice e' simmetrica se ogni elemento m\_ij e' uguale all'elemento m\_ji.

NB: per usare il tipo bool in C includere la libreria <stdbool.h>.

### Esercizio 5.6

Scrivere la funzione

void mat\_normalize\_rows(Mat \*m);

che, data in ingresso una struttura Mat m contenente una matrice, modifichi m in modo da normalizzare le righe. Si ricorda che la normalizzazione di una riga si ottiene dividendo tutti gli elementi della riga per il modulo della riga stessa.

### Esercizio 5.7

Scrivere la funzione

Mat\* mat\_sum(Mat \*m1, Mat \*m2);

che, date in ingresso due strutture Mat m1 e Mat m2 contenenti due matrici, allochi e restituisca la somma delle suddette matrici. Nel caso non fosse possibile eseguire la somma (per esempio, se le dimensioni delle due matrici di input non sono uguali), la funzione deve stampare a schermo un messaggio di errore e ritornare NULL.

### Esercizio 5.8

Scrivere una funzione:

Mat\* mat\_product(Mat \*m1, Mat \*m2);

che, date in ingresso due strutture Mat m1 e Mat m2 contenenti due matrici, allochi e restituisca il prodotto delle suddette matrici. Nel caso non fosse possibile eseguire il prodotto (per esempio, se le dimensioni delle due matrici di input non consentono il prodotto), la funzione deve stampare a schermo un messaggio di errore e ritornare NULL.

### Esercizio 5.9

Scrivere una funzione:

Mat\* game\_of\_life(Mat\* mat);

Basata sull’[algoritmo di Conway](https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life) che implementa una iterazione del gioco. La funzione deve prendere in input una matrice al tempo i e restituire la matrice al tempo i+1. Indicare una cella viva con il float 1, e una cella morta con 0.

Scrivere un main per testare la funzione. Suggerimento: il main potrebbe usare mat\_alloc per creare una matrice iniziale, e mat\_free, mat\_print ad ogni chiamata di game\_of\_life per il normale avanzamento.

# Arrays

### Esercizio 5.10

Scrivere una funzione

void sort\_strings(char \*\*array);

che riceve in input un array di N stringhe. L’elenco è terminato da un puntatore NULL in posizione array[N]. La funzione deve modificare l’array, ordinando le stringhe per lunghezza: in array[0] ci deve essere la stringa più lunga e in array[len-1] la più corta.

# Files

### Esercizio 5.11

Scrivere una funzione:

Mat\* mat\_read(const char \*filename)

che, dato in ingresso il nome di un file filename, allochi e restituisca una struttura Mat contenente una matrice letta dal file filename. Il file contiene un primo numero che indica il numero di righe ed un secondo che indica il numero di colonne della matrice, seguiti dalla lista di elementi che la compongono.

Per esempio il file contenente la matrice

m =

[1.1 2.2 3.3]

[4.4 5.5 6.6]

avrà la seguente forma:

2 3

1.1 2.2 3.3

4.4 5.5 6.6

### Esercizio 5.12

Scrivere una funzione:

void mat\_write(const char \*filename, Mat \*m)

che, dati in ingresso il nome di un file e una struttura Mat contenente una matrice, salvi la matrice in un file al percorso filename. La matrice deve essere scritta sul file seguendo la formattazione indicata nell'esercizio 5.10. Provare a scrivere, rileggere e comparare la matrice letta per controllare il corretto funzionamento delle ultime due funzioni.