Tecniche di Programmazione (2018/19)

Esercitazione 5

Argomento: Matrici e File

Scaricare il file mat.c e i file di esempio per le matrici mat_1.txt, mat_2.txt, mat_3.txt e mat4.txt. Aggiungere nel file mat.c la definizione delle funzioni indicate negli esercizi seguenti. Modificare opportunamente la funzione main per effettuare delle verifiche di funzionamento delle funzioni scritte. Modificare o aggiungere altri file di matrici per ulteriori verifiche.

Esercizio 5.1

Scrivere la funzione C

```
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 C

```
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]
```

avra' la seguente forma:

```
2 3
1.1 2.2 3.3
4.4 5.5 6.6
```

Esercizio 5.3

Scrivere la funzione C

```
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 C

```
void Mat_write(const char *filename, Mat *m);
```

che, dati in ingresso il nome di un file e una struttura Mat m contenente una matrice, salvi la matrice m nel file filename. La matrice deve essere scritta sul file seguendo la formattazione indicata nell'esercizio 4.4.

Esercizio 5.5

Scrivere la funzione C

```
void Mat_free(Mat *m);
```

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

ALTRI ESERCIZI PROPOSTI

Esercizio 5.6

Scrivere la funzione C

```
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 X_ij e' uguale all'elemento X_ji.

Esercizio 5.7

Scrivere la funzione C

```
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 5.8

Scrivere la funzione C

```
Mat* Mat_clone(Mat *m);
```

che, data in ingresso una struttura Mat m contenente una matrice, allochi e restituisca una copia della matrice m.

Esercizio 5.9

Scrivere la funzione C

```
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.

le dimensioni delle due matrici di input non combacino

Esercizio 5.10

Scrivere la funzione C

```
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.