# Operazioni Matrici

L'intero programma si basa su un file sorgente principale all'interno del quale, facendo riferimento a due librerie di gestione apposite chiamate "OpMatrix" e "ReadWriteFoo", verrà data la possibilità mediante un opportuno menù di selezione multipla di scegliere l'operazione da svolgere con le matrici.

## Astrazione Funzionale

Dovendo precisamente lavorare con operazioni che effettuano modifiche sulle matrici è stato necessario creare una opportuna struttura che contenesse i dati che garantissero una chiara e facile gestione delle stesse.
L'intero programma pertanto lavora sulla struttura si fatta:

```
record matrice
{
    array bidimensionale con valori di tipo reale
    righe // variabile intera che contiene il numero di righe della matrice
    colonne // variabile intera che contiene il numero di colonne della matrice
    id // variabile intera che rappresenta un indentificativo che contraddistingue ogni singola matrice
}
```

Per poter garantire la gestione all'interno del programma di un numero di matrici che varia durante il corso del programma, viene creata una struttura dati opportunamente allocata che rappresenta un elenco di strutture di tipo matrice.

## Funzioni d'uso generale

---- OpMatrix Library -----

Tale prima libreria contiene unicamente funzioni d'uso comune che vengono adoperate per poter svolgere correttamente tutte le operazioni sulla struttura dati di tipo matrice.

------ Trasponi Matrice -------
trasposta( matrice dest, matrice src )
Procedura che consente di creare la matrice trasposta di quella rappresentata dal parametro src, la quale verrà salvata nella matrice dest.

----- Somma Matrici -----------
somma ( matrice dest, matrice src1, matrice src2 )
Procedura di somma che consente di effettuare la somma della matrici passate come parametri formali( passaggio per riferimento ) src1 e src2 e di salvare il risultato di tale operazione nella matrice rappresentata dal parametro formale dest, parametro, anch'esso, passato per riferimento.

----- Differenza Matrici -----------
differenza ( matrice dest, matrice src1, matrice src2 )
Procedura di differenza che consente di effettuare la differenza della matrici passate come parametri formali( passaggio per riferimento ) src1 e src2 e di salvare il risultato di tale operazione nella matrice rappresentata dal parametro formale dest, parametro, anch'esso, passato per riferimento.

-------- Prodotto Scalare ----------
scalare( matrice dest, matrice src, a )

Procedura che effettua il prodotto di uno specifico valore reale passato come parametro e rappresentato da a per ogni singolo valore presente nella matrice rappresentata dalla struttura passata per riferimento e rappresentata da src.
Il risultato dell'operazione di prodotto viene salvato nella matrice rappresentata dalla variabile dest.

---------- Prodotto Vettoriale ----------
prodotto( matrice dest, matrice src1, matrice src2 )
Procedura che effettua il prodotto vettoriale delle matrici src1 e src2 dando la possibilità di creare la matrice risultante che sarà rappresentata dalla struttura matrice rappresentata dalla variabile dest.

------- ReadWriteFoo library ----------
Per poter garantire una maggiore riusabilità dell'intero codice, all'interno di tale libreria vengono implementate delle funzioni che provvedono a settare opportuni valori di una matrice oppure consentono di acquisirli per poterli adoperare durante una specifica operazione.
Tali funzioni hanno la caratteristica di essere le uniche a poter accedere ai campi della struttura matrice.

------ Inizializza nuova Matrice -------
creaMatrice( matrice m, righe, colonne )
Procedura che riceve un unico parametro passato per riferimento che rappresenta la struttura matrice che si intende inizializzare con i parametri passati per valore righe e colonne, adoperando le funzioni che modificano le righe e le colonne scriviRighe() e scriviColonne().

------ Leggi Righe ----------
leggiRighe( matrice )
Ritorna il numero di righe della matrice passata come parametro alla funzione stessa.

---- Leggi Colonne ----------
leggiColonne( matrice )
Riceve come parametro una matrice e ritorna un valore intero che rappresenta il numero di colonne della stessa.

----- Scrive valore in una posizione della matrice ------
scriviValore( matrice, i, j )
Provvede a salvare un valore letto opportunamente dallo standard input e che appartiene rigorosamente all'insieme dei reali. Tale valore, viene memorizzato nella posizione della matrice specificata dai valori i e j passati come parametri alla funzione.

----- Legge un valore presente in una posizione della matrice ----
leggiElemento( matrice , i, j )
Provvede a ritornare un valore reale presente nella posizione i,j della matrice passata come parametro.

----- Modifica il valore delle righe della matrice -----
scriviRighe( matrice )
Ricevendo come parametro d'uscita la matrice permette di modificare il valore delle righe in essa contenuto.

------ Modifica il valore dell'id della struttura --------
scriviId( matrice, id )

Riceve come parametro una struttura di tipo matrice ed un intero che rappresenta l'identificativo che verrà assegnato alla matrice.
---------- Legge il valore dell'id che rappresenta la struttura matrice ----------
leggiId( matrice )
Riceve come parametro una struttura di tipo matrice e ritorna il corrispettivo id che la identifica.

------ Modifica il valore delle colonne della matrice -----
scriviColonne( matrice )
Ricevendo come parametro d'uscita la matrice permette di modificare il valore delle colonne in essa contenuto.

----- Modifica il valore di un elemento in una specifica posizione della matrice -----
scriviElemento( matrice, i, j, n )
Permette la modifica di un elemento collocato in posizione i, j della matrice passata come parametro che verrà sostituito con il valore contenuto in n.

## Matrice Trasposta

Data una matrice M = $\begin{matrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{matrix}$, calcolare la matrice trasposta.

Mt = $\begin{matrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{matrix}$

Analisi Dati Input

m (righe,colonne) – matrice valori reali
r, c variabili intere 0 < r <= MAXR; 0 < c <= MAXC
Analisi Dati Output
mt( c, r ) - matrice trasposta di Matx
Verifica Dati Input
Viene richiesto all'utente l'ID specifico della matrice sulla quale intende effettuare l'operazione di trasposizione.
Dopodichè viene richiamata la funzione creaMatrice() alla quale vengono passati come parametri la matrice dell'elenco che deve essere inizializzata adoperando quelle che sono le colonne della matrice scelta per inizializzare le righe della nuova matrice e adoperando le righe della matrice scelta per inizializzare le colonne della nuova matrice.

Verifica Dati Output

Richiamando la funzione stampaMatrice() e passando come parametro la matrice appena creata viene dapprima verificata la correttezza della matrice( dimensioni proprie della matrice in numero diverso da zero) ed in seguito viene visualizzata a video mostrando il relativo ID che la identificherà.

Implementazione algoritmo

Pseudo Codice

------- Trasposta --------

itera per i che varia da 1 a righe

  itera per j che varia da 1 a colonne

mt[i][j] ← m[j][i]

Interfaccia Programma

----------OPERAZIONI MATRICI----------

Powered by A. Suglia & N. Chekalin

-----------------MENU'----------------

1 - Inserire Matrice

2 - Stampa Matrice

3 - Sovrascrivi

4 - Matrice Trasposta

5 - Somma Matrici

6 - Differenza Matrici

7 - Prodotto Scalare Matrici

8 - Prodotto Vettoriale Matrici

0 - Esci

Scelta : 4

Inserire l'ID della matrice sulla quale operare

MATRICI INIZIALIZZATE: 2

Scelta ID -> 1

Matrice 1

3 2

4 5

Matrice Trasposta

3 4

2 5

**Prodotto matrice scalare**

Dati uno scalare a e una matrice M= $\begin{matrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{matrix}$, calcolare il prodotto tra a e M.

Ma= $\begin{matrix} aM_{11} & aM_{12} & aM_{13} \\ aM_{21} & aM_{22} & aM_{23} \\ aM_{31} & aM_{32} & aM_{33} \end{matrix}$

Analisi Dati di Input

M ( righe, colonne ) – matrice di valori reali
a – variabile reale
Verifica Dati Input
Viene richiesto all'utente l'ID specifico della matrice sulla quale intende effettuare l'operazione di prodotto scalare e ovviamente il valore scalare per il quale si intende moltiplicare l'intera matrice che ovviamente verrà acquisito facendo in modo che sia un valore che appartenga ai reali.
Dopodichè viene richiamata la funzione creaMatrice() alla quale vengono passati come parametri la matrice dell'elenco che deve essere inizializzata adoperando quelle che sono le righe e le colonne della matrice selezionata acquisite attraverso leggiRighe() e leggiColonne().

Implementazione algoritmo
------- Pseudo Codice --------
"Prodotto":
itera per i che varia da 1 a r
    itera per j che varia da 1 a c
        ma[i][j] ← a * m[i][j]
Interfaccia
----------OPERAZIONI MATRICI----------
Powered by A. Suglia & N. Chekalin
----------MENU'----------
1 - Inserire Matrice
2 - Stampa Matrice
3 - Sovrascrivi
4 - Matrice Trasposta
5 - Somma Matrici
6 - Differenza Matrici
7 - Prodotto Scalare Matrici
8 - Prodotto Vettoriale Matrici
0 - Esci
Scelta : 7
Inserire l'ID della matrice sulla quale operare
MATRICI INIZIALIZZATE: 2
Scelta ID -> 0
Inserisci valore scalare : 2
----Matrice 3 ----
2 4
6 8

**Somma e Differenza fra matrici**

Date 2 matrici M= $\begin{matrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{matrix}$ e N= $\begin{matrix} N_{11} & N_{12} & N_{13} \\ N_{21} & N_{22} & N_{23} \\ N_{31} & N_{32} & N_{33} \end{matrix}$ calcolare la somma M+N e la differenza M-N delle due matrici.

$$\text{sum\_mat} = \begin{matrix} M_{11} + N_{11} & M_{12} + N_{12} & M_{13} + N_{13} \\ M_{21} + N_{21} & M_{22} + N_{22} & M_{23} + N_{23} \\ M_{31} + N_{31} & M_{32} + N_{33} & M_{33} + N_{33} \end{matrix}$$

$$\text{diff\_mat} = \begin{matrix} M_{11} - N_{11} & M_{12} - N_{12} & M_{13} - N_{13} \\ M_{21} - N_{21} & M_{22} - N_{22} & M_{23} - N_{23} \\ M_{31} - N_{31} & M_{32} - N_{33} & M_{33} - N_{33} \end{matrix}$$

Analisi Dati Input

M( r, c ) – matrice valori reali con 0 < r <= MAXR, 0 < c <= MAXC
N( r1, c1) – matrice valori reali con 0 < r1 <= MAXR, 0 < c1 <= MAXC
r1 = r, c = c1 – M ed N devono avere le stesse dimensioni

Analisi Dati di Output

Con la funzione di somma ottengo sum_mat ( r, c ) – matrice somma di M ed N
Con la funzione di differenza ottengo diff_mat ( r, c ) – matrice differenza di M ed N

Verifica Dati Input

L'utente ha la possibilità di selezionare le due matrici sulle quali operare, avendo la facoltà di selezionare anche la medesima matrice effettuando così l'operazione adoperando matrici identiche. Viene inizializzata la matrice risultante richiamando la funzione creaMatrice() che provvede ad settare i campi di tale matrice con le righe e le colonne di una delle due matrici( Asserzione – le righe e le colonne delle due matrici scelte sono uguali, nel caso non dovesse essere verificata tale condizione indispensabile, verrà notificato all'utente un messaggio di errore specifico).
I valori delle matrici risultato( sia per la somma che per la differenza ) verranno appositamente settate dalle rispettive funzioni somma() e differenza() che riceveranno, oltre alla matrice che conterrà la matrice risultante, anche le due matrici scelte preventivamente dall'utente sulle quali intendeva effettuare l'operazione.

Implementazione

```
---------- somma tra matrici ----------
per i che varia da 1 a r
        per j che varia da 1 a c
                sum_mat[i][j] <- m[i][j] + n[i][j]
---------- differenza tra matrici ----------
Per i che varia da 1 a r
        Per j che varia da 1 a c
                diff_mat[i][j] <- m[i][j] – n[i][j]
```

Interfaccia

```
----------OPERAZIONI MATRICI----------
Powered by A. Suglia & N. Chekalin
----------MENU'----------
1 - Inserire Matrice
2 - Stampa Matrice
3 - Sovrascrivi
4 - Matrice Trasposta
```

5 - Somma Matrici
6 - Differenza Matrici
7 - Prodotto Scalare Matrici
8 - Prodotto Vettoriale Matrici
0 - Esci
Scelta : 5
Inserire l'ID della matrice sulla quale operare
MATRICI INIZIALIZZATE: 2
Scelta ID -> 0
Inserire l'ID della seconda matrice sulla quale operare
MATRICI INIZIALIZZATE: 2
Scelta ID -> 1
----Matrice 3 ----
2 4
6 8
----------OPERAZIONI MATRICI----------
Powered by A. Suglia & N. Chekalin
-----------------MENU'----------------
1 - Inserire Matrice
2 - Stampa Matrice
3 - Sovrascrivi
4 - Matrice Trasposta
5 - Somma Matrici
6 - Differenza Matrici
7 - Prodotto Scalare Matrici
8 - Prodotto Vettoriale Matrici
0 - Esci
Scelta : 6
Inserire l'ID della matrice sulla quale operare
MATRICI INIZIALIZZATE: 2
Scelta ID -> 0
Inserire l'ID della seconda matrice sulla quale operare
MATRICI INIZIALIZZATE: 2
Scelta ID -> 1
----Matrice 4 ----
0 0
0 0

**Prodotto Vettoriale**

Date 2 matrici $M=\begin{matrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{matrix}$ e $N=\begin{matrix} N_{11} & N_{12} & N_{13} \\ N_{21} & N_{22} & N_{23} \\ N_{31} & N_{32} & N_{33} \end{matrix}$, calcolare il prodotto MN.

Analisi Dati Input

M( r, c ) – matrice valori reali con 0 < r <= MAXR, 0 < c <= MAXC

N( r1, c1) – matrice valori reali con 0 < c1 <= MAXC
N.B. Il numero di colonne c di M deve essere uguale al numero di righe r di N
Analisi Dati di Output
prod_mat(r,c1) – matrice valori reali

Verifica Dati Input
L'utente ha la possibilità di selezionare le due matrici sulle quali operare, avendo la facoltà di selezionare anche la medesima matrice effettuando così l'operazione adoperando matrici identiche. Ovviamente tale operazione, prima di poter essere effettivamente svolta deve necessariamente soddisfare determinati vincoli per poter portare a termine in modo corretto l'intero processo di elaborazione e pertanto viene dapprima verificata la correttezza delle dimensioni delle matrici scelte. Tali matrici scelte, contribuiscono con le loro dimensioni a settare, con l'apposita funzione creaMatrice(), la matrice risultante che per l'esattezza risulterà avere le righe della prima matrice scelta e le colonne della seconda matrice scelta.
Dopo aver opportunamente settato tali dimensioni viene richiamata la specifica funzione prodotto() che riceve come parametro oltre che la matrice che conterrà i risultati, anche le due matrici scelte per fare l'operazione.

Implementazione
```
Per i che varia da 1 a r
        Per j che varia da 1 a c1
                Per k che varia da 1 a c
                        Prod_mat[i][j] <- prod_mat[i][j] + m[i][k] * n[k][j]
```

## Interfaccia
```
----------OPERAZIONI MATRICI----------
Powered by A. Suglia & N. Chekalin
----------------MENU'---------------
1 - Inserire Matrice
2 - Stampa Matrice
3 - Sovrascrivi
4 - Matrice Trasposta
5 - Somma Matrici
6 - Differenza Matrici
7 - Prodotto Scalare Matrici
8 - Prodotto Vettoriale Matrici
0 - Esci
Scelta : 8
Inserire l'ID della matrice sulla quale operare
MATRICI INIZIALIZZATE: 2
Scelta ID -> 0
Inserire l'ID della seconda matrice sulla quale operare
MATRICI INIZIALIZZATE: 2
Scelta ID -> 1
----Matrice 3 ----
30 36 42
66 79 96
92 125 150
```

# Operazioni Matrici

*Modifiche apportate per la gestione di una matrice*

Per poter gestire una matrice mediante un array monodimensionale è necessario innanzitutto modificare il campo relativo alla struttura dati adoperata per rappresentare la matrice.

Prima della modifica:

```
struct matrice
{
        float mat [righe][colonne];
        int righe;
        int colonne;
        int id;
}
```

Post modifica:

```
struct matrice
{
        float mat [righe*colonne];
        int righe;
        int colonne;
        int id;
}
```

Accesso ai dati mediante formula con offset i * num_colonne_totali + j

Oppure mediante puntatore singolo

```
struct matrice
{
        float *mat;
        int righe;
        int colonne;
        int id;
}
```

Accesso ai dati mediante l'indirizzo della matrice alla quale viene sommato un offset determinato dalla formula i*num_colonne_totali.

O mediante doppio puntatore

```
Struct matrice
{
    Float **mat;
    int righe
    int colonne
    int id

}
```

Accesso ai dati mediante risoluzione del riferimento dell'indirizzo della matrice alla quale va sommato il numero di righe e una seguente risoluzione del riferimento di tale indirizzo con offset pari al numero di colonne.

Nell'ultimo caso considerato prima di poter adoperare nel proprio programma tale struttura dati è necessario procedere nell'allocazione dinamica della memoria che provvederà a metter a disposizione un

numero di celle di memoria pari al prodotto delle righe e delle colonne della matrice che si intende rappresentare.
Tale allocazione verrà effettuata dopo aver letto le righe e le colonne effettive della matrice, operazione che viene fatta congiuntamente all'interno della funzione creaMatrice().

Pseudocodice della funzione creaMatrice() per l'implementazione mediante singolo puntatore

```
creaMatrice( matrice *, righe, colonne )
{
        // leggi le righe e le scrive nel campo righe della struttura
        // leggi le colonne e le scrive nel campo colonne della struttura
        // alloca una spazio di memoria per un totale di elementi pari al prodotto delle righe e delle
        colonne ed assegna tale spazio al puntatore campo della matrice m
        // verifica che l'effettiva operazione di allocazione di memoria sia andata a buon fine
        Altrimenti segnala un errore all'utente ed esce dal programma.
}
```

Mentre per l'implementazione della struttura mediante doppio puntatore sarà necessario adoperare una funzione creaMatrice() si fatta

```
creaMatrice ( matrice *, righe, colonne )
{
        // leggi le righe e le scrive nel campo righe della struttura
        // leggi le colonne e le scrive nel campo colonne della struttura
        // alloca una spazio di memoria per un totale di elementi pari al numero delle righe assegnando
        tale spazio di memoria al puntatore alla matrice, campo della struttura.
        // verifica che l'effettiva operazione di allocazione di memoria sia andata a buon fine
        Altrimenti segnala un errore all'utente ed esce dal programma.
        // per ogni riga allocata per la matrice viene allocato una spazio di memoria pari al numero di
        colonne effettivo della matrice
        // verifica che l'effettiva allocazione di memoria sia andata a buon fine, altrimenti segnala un
        errore che compromette il proseguimento dell'esecuzione del programma.
}
```

Infine, dopo le modifiche sopracitate qualsiasi sia la struttura dati che si adopererà per rappresentare la matrice, il codice e le funzionalità resteranno completamente invariate con la sola differenza che si dovrà omettere l'uso della funzione relativa all'allocazione dinamica nel caso in cui la struttura adoperata sia del tipo: float matrice [righe] [colonne] oppure float matrice [righe * colonne].