8 MATLAB

8.1 Esercizi

Esercizio 8.1

Scrivere uno script che calcoli la sequenza di Fibonacci di lunghezza 20, e la stampi a schermo. Successivamente si richieda di inserire un numero $2 \leq n \leq 4180$ e valuti se il numero è di Fibonacci. Altrimenti restituisce il numero di Fibonacci più vicino. La successione di Fibonacci è definita così:

$$F(0) = 0$$
$$F(1) = 1$$
$$F(n) = F(n-1) + F(n-2), n > 1$$

Esercizio 8.2

Utilizzando il fatto che il quadrato di n è uguale alla somma dei primi n numeri dispari, calcolare il quadrato di un numero ($n < 100$) inserito dall'utente.

Esercizio 8.3

Chiedere all'utente due parole e stampare a video se una è anagramma dell'altra.

Esercizio 8.4

Creare una matrice `M` di dimensioni 6×7 contenente 0, 1, 2, matrice che rappresenta una situazione in una partita di forza 4 in corso.

Chiedere ai due giocatori, finché uno di questi non inserisce la lettera `'q'` (quit), di inserire la colonna (tra 1 e 7) dove intende inserire la propria pedina. Inserire la pedina nella colonna corretta e visualizzare la matrice `M` così ottenuta.

Bonus: scrivere una porzione dello script che controlli se un giocatore ha vinto, ovvero se ci sono 4 pedine adiacenti dello stesso giocatore in orizzontale, in verticale o **in diagonale**.

Esercizio 8.5

Verificare se una matrice quadrata di dimensione arbitraria è un quadrato magico. Una matrice è un quadrato magico se la somma degli elementi sulle righe, sulle colonne e sulla diagonale principale è la stessa.

Esercizio 8.6

Data una matrice 20×20 che rappresenta le partite di un campionato di calcio (con 0 per vittoria in casa, 1 per pareggio 2 per vittoria in trasferta come risultati possibili). Calcolare la classifica finale ordinata.

Esercizio 8.7

Data una matrice quadrata, leggerla a spirale e metterne il contenuto in un vettore. La lettura a spirale avviene andando a leggere la prima riga, poi l'ultima colonna, quindi l'ultima riga ed infine la prima colonna.

Esercizio 8.8

(TdE 2010 - modificato) Dopo una gara automobilistica si ha come risultato tre tabelle le cui colonne rappresentano gli n partecipanti (numerati da 1 a n) e le righe gli m giri di pista effettuati. Il valore di ogni generica cella (i, j) delle tabelle rappresenta il tempo impiegato (in minuti, secondi e millesimi) dal partecipante j per percorrere il giro i.

Si scrivano le istruzioni per:

- calcolare il tempo medio che è stato impiegato da ciascun partecipante per completare la gara;
- determinare il vincitore della gara (cioè il numero del partecipante il cui tempo di percorrenza totale è minore di quello degli altri partecipanti);

Supponiamo di avere un vettore che ci dica per ogni pilota quanti giri ha effettivamente percorso. Come cambia lo script?

Esercizio 8.9

Scrivere uno script che, ricevendo una matrice `M` di numeri interi, stampa a video una matrice `MR`, ottenuta da `M` nel seguente modo:

- calcola la media aritmetica dei valori di `M` e ne arrotonda il valore all'intero piú vicino;
- per i valori che in `M` sono minori della media, in `MR` si scriva nella posizione corrispondente il valore -1;

- per quelli superiori alla media si pone il valore 1;
- per gli altri (quelli uguali alla media) si pone lo stesso valore in M.

Esercizio 8.10

Si sviluppi uno script che riceve una matrice 8×8, che rappresenta la scacchiera su cui sono disposte 8 regine, e calcola se la configurazione delle 8 regine è corretta (nessuna regina mette in scacco un'altra regina), o meno. Si supponga che la matrice contenga il valore 0 in tutte le posizioni libere e il valore 1 nelle posizioni occupate dalle regine.

Soluzioni

Soluzione dell'esercizio 8.1

```
clear
clc
close all

% Inizializza sequenza
fibo = zeros(1,20);

% Calcolo i primi 20 numeri di fibonacci
fibo(1) = 0;
fibo(2) = 1;
for ii = 3:20
    fibo(ii) = fibo(ii-1) + fibo(ii-2);
end

fibo

a = input('Inserire un numero (tra 2 e 4180): ');

if sum(a == fibo) > 0
    disp([num2str(a) 'e'' un numero di Fibonacci']);
else
    % Cerco i numeri di fibonacci più piccoli di a e scelgo l'
       ultimo
    inferiori = fibo(fibo < a);
    inferiori = inferiori(end);

    % Cerco i numeri di Fibonacci più grandi di a e scelgo il
       primo
    superiori = fibo(fibo > a);
    superiori = superiori(1);

    % Cerco il più vicino tra il più grande numero di fibonacci
        più piccolo
    % di a e il più piccolo numero di Fibonacci più grandi di a
    if superiori - a < a - inferiori
        vicino = superiori;
    else
        vicino = inferiori;
    end
```

```
37
38      disp(['Il numero di Fibonacci piu'' vicino a ' num2str(a) '
            e'' ' num2str(vicino)]);
39
40  end
```

Soluzione dell'esercizio 8.2

```
1   clc
2   clear
3
4   N = input('Inserire il numero da elevare al quadrato (n < 100):
        ');
5
6   numeri = 2 * [0 : N - 1] + 1;
7
8   % Soluzione alla C
9   c = 1;
10  somma_while = 0;
11  while c <= N
12      somma_while = somma_while + numeri(c);
13      c = c + 1;
14  end
15
16  % Soluzione ibrida
17  somma_ibrida = 0;
18  for c = numeri
19      somma_ibrida = somma_ibrida + c;
20  end
21
22  % Soluzione alla MATLAB
23  somma_matlab = sum(numeri);
24
25  fprintf('Soluzione alla C: %d^2 = %d\n', N, somma_while);
26  fprintf('Soluzione ibrida: %d^2 = %d\n', N, somma_ibrida);
27  fprintf('Soluzione alla MATLAB: %d^2 = %d\n', N, somma_matlab);
```

Soluzione dell'esercizio 8.3

```
1   clear
2   clc
3
4   parola1 = input('Inserire la prima parola: ','s');
```

```
parola2 = input('Inserire la seconda parola: ','s');

% soluzione 1: ISTOGRAMMI
isto1 = zeros(1,255);
isto2 = zeros(1,255);

for ii = parola1
    isto1(ii) = isto1(ii) + 1;
end

for ii = parola2
    isto2(ii) = isto2(ii) + 1;
end

fprintf('Le due parole ');
if any(isto1 ~= isto2)
    fprintf('non ');
end
fprintf('sono una l''anagramma dell''altra\n');

% soluzione 2: ORDINA E CONFRONTA
if length(parola1) == length(parola2)
    ordinata1 = sort(parola1);
    ordinata2 = sort(parola2);
    is_anagramma = all(ordinata1 == ordinata2);
else
    is_anagramma = false;
end

fprintf('Le due parole ');
if ~is_anagramma
    fprintf('non ');
end
fprintf('sono una l''anagramma dell''altra\n');
```

Soluzione dell'esercizio 8.4

```
clear;
clc;
close all;

M = zeros(6,7);

```

```
7  turno_giocatore = 1;
8  a = 6;
9  while (a ~= 'q')
10     disp(['E'' il turno del giocatore ' num2str(turno_giocatore
           )]);
11     a = input('Inserire una giocata (numero di colonna 1-7) o
           uscire (''q''): ');
12
13     if a ~= 'q'
14         if (M(1, a) ~= 0)
15             disp('Giocata illegale');
16         else
17             indici = M(:,a) == 0;
18             pos_libera = sum(indici);
19             M(pos_libera, a) = turno_giocatore;
20             imagesc(M);
21             if (turno_giocatore == 1)
22                 turno_giocatore = 2;
23             else
24                 turno_giocatore = 1;
25             end
26         end
27
28         % ---------------------------------- %
29         % BONUS: controllo vittoria
30
31         % direzioni lungo cui mi posso spostare
32         direzioni = [1 0; -1 0; 0 1; 0 -1; 1 1; 1 -1; -1 1; -1
               -1];
33         [n_direzioni, ~] = size(direzioni);
34
35         [n, m] = size(M);
36
37         vincitore = 0;
38
39         % scorro su tutti gli elementi della matrice, finché
               non ho trovato
40         % un vincitore
41         ii = 1;
42         while ii <= n && vincitore == 0
43             jj = 1;
44             while jj <= m && vincitore == 0
45                 % inizia check solo se siamo in una casella che
                       ha una
```

```
46                  % pedina
47                  if M(ii, jj) ~= 0
48                      % per ogni direzione, mi sposto finché non
                           usciamo dal
49                      % dominio o non troviamo una casella di
                           colore diverso
50                      % o arriviamo a 4 caselle uguali in fila
51                      giocatore = M(ii, jj);
52                      dd = 1;
53
54                      % provo tutte le possibili direzioni a
                           partire dalla
55                      % casella corrente
56                      while dd <= n_direzioni && vincitore == 0
57                          contatore = 1;
58                          direzione = direzioni(dd, :);
59
60                          % riparto sempre dalla posizione
                               corrente
61                          posizione = [ii, jj];
62
63                          % mi sposto lungo la direzione
                               selezionata e
64                          % controllo di essere ancora dentro la
                               matrice
65                          posizione = posizione + direzione;
66
67                          posizione_ok = posizione(1) >= 1 && ...
68                              posizione(1) <= 6 && ...
69                              posizione(2) >= 1 && ...
70                              posizione(2) <= 7;
71
72                          while posizione_ok
73                              % se la posizione è ok, controllo
                                   che la nuova
74                              % casella appartenga ancora al
                                   giocatore che
75                              % sto considerando; se è vero,
                                   aumento il
76                              % contatore
77                              if M(posizione(1), posizione(2)) ~=
                                    giocatore
78                                  posizione_ok = 0;
79                              else
```

```
80                                  contatore = contatore + 1;
81
82                                  % se il contatore è arrivato a
                                        4, abbiamo un
83                                  % vincitore; settando vincitore
                                        a un numero
84                                  % diverso da 0 e posizione_ok a
                                        0, sia il
85                                  % ciclo interno che quello
                                       esterno terminano
86                                  % subito
87                                  if contatore == 4
88                                      vincitore = giocatore;
89                                      posizione_ok = 0;
90                                  else
91                                      posizione = posizione +
                                           direzione;
92                                      posizione_ok = posizione(1)
                                            >= 1 && ...
93                                          posizione(1) <= 6 &&
                                                ...
94                                          posizione(2) >= 1 &&
                                                ...
95                                          posizione(2) <= 7;
96                                  end
97                              end
98                          end
99                          dd = dd + 1;
100                     end
101                 end
102                 jj = jj + 1;
103             end
104             ii = ii + 1;
105         end
106
107         if vincitore ~= 0
108             disp(['Il vincitore e'' il giocatore ' num2str(
                    vincitore)])
109             a = 'q';
110         end
111         % FINE BONUS
112         % ---------------------------------------- %
113     end
114 end
```

Soluzione dell'esercizio 8.5

```
clear
clc
close all

M = magic(4);
%M = randi(4,3);
[r, c] = size(M);

% Controllo matrice quadrata
assert(r == c);

% Calcolo somme su righe
somme = zeros(1,2 * r + 1);
for ii = 1:r
    somme(ii) = sum(M(ii,:));
end

% Calcolo somme su colonne
for ii = (r+1):2*r
    somme(ii) = sum(M(:,ii-r));
end

% Calcolo somma su diagonale
somme(2*r+1) = sum(diag(M));

somme

if sum(somme == somme(1)) == 2*r+1
    disp('La matrice e'' un quadrato magico');
else
    disp('La matrice non e'' un quadrato magico');
end
```

Soluzione dell'esercizio 8.6

```
clear
clc

```

```
4  squadre = { 'Atalanta' 'Bologna' 'Cagliari' 'Chievo' 'Empoli' '
       Fiorentina' ...
5      'Frosinone', 'Genoa' 'Inter' 'Juventus' 'Lazio'     'Milan'
           'Napoli' 'Parma' ...
6      'Roma' 'Sampdoria' 'Sassuolo' 'SPAL' 'Torino' 'Udinese'};
7  squadre_alt = squadre;
8
9  risultati = randi(3, 20) - 1;
10 for ii = 1:20
11     risultati(ii,ii) = -1;
12 end
13
14 % Versione alla C
15 punti = zeros(20,1);
16 for ii = 1:20
17     punti(ii) = sum(risultati(ii,:) == 0) * 3 + sum(risultati(
           ii,:) == 1) + ...
18         sum(risultati(:,ii) == 2) * 3 + sum(risultati(:,ii) ==
               1);
19 end
20
21 %Versione alla Matlab
22 punti_alt = sum(risultati == 0,2) * 3 + sum(risultati == 1,2) +
       ...
23         sum(risultati' == 2,2) * 3 + sum(risultati' == 1,2);
24
25 assert(sum(punti == punti_alt) == 20)
26
27 %Ordiniamo le squadre
28 while (~isempty(punti))
29     maxi = max(punti);
30     trovato = 0;
31     posizione = 1;
32     while trovato == 0
33         if punti(posizione) == maxi
34             disp([squadre{posizione} ' : ' num2str(punti(
                   posizione)) ' punti.']);
35             punti(posizione) = [];
36             squadre(posizione) = [];
37             trovato = 1;
38         else
39             posizione = posizione + 1;
40         end
41     end
```

```
42 end
43 disp('-----------------------------------');
44
45 %Oppure chiediamo a MATLAB
46 [punti_alt, indici] = sort(punti_alt,'descend');
47 squadre_alt = squadre_alt(indici);
48 for ii = 1:20
49     disp([squadre_alt{ii} ' : ' num2str(punti_alt(ii)) ' punti.
          ']);
50 end
```

Soluzione dell'esercizio 8.7

```
1  clear
2  clc
3
4  M = randi(10,7);
5  M_old = M;
6
7  vec = [];
8
9  while(~isempty(M))
10
11     vec = [vec M(1,:)];
12     M(1,:) = [];
13     if (~isempty(M))
14         vec = [vec M(:,end)'];
15         M(:,end) = [];
16     end
17     if (~isempty(M))
18         vec = [vec M(end,end:-1:1)];
19         M(end,:) = [];
20     end
21     if (~isempty(M))
22         vec = [vec M(end:-1:1,1)'];
23         M(:,1) = [];
24     end
25 end
26
27 assert(sum(vec) == sum(M_old(:)));
28
29 M_old
30 vec
```

Soluzione dell'esercizio 8.8

```
clear
clc

n_piloti = 10;
n_giri = 30;

minuti = randi(2,n_piloti,n_giri);
secondi = 60 * rand(n_piloti,n_giri);
millesimi = 1000 * rand(n_piloti,n_giri);

tempo_medio = mean(minuti * 60 + secondi + millesimi / 1000, 2)
   ;
tempo_vinc = min(tempo_medio);

vinc = find(tempo_medio == tempo_vinc);

disp(['Il vincitore e '' ' num2str(vinc)]);
```

Soluzione dell'esercizio 8.9

```
clear
clc

M = randi(20,5);


MR = zeros(size(M));

media_M = round(mean(M(:)));

MR(M < media_M) = -1;
MR(M > media_M) = 1;
MR(M == media_M) = media_M;
```

Soluzione dell'esercizio 8.10

```
clear
clc
close all

% creo scacchiera vuota
```

```
6  scacchiera = zeros(8);
7
8  % posizioni casuali per le regine. NB: non ci possono essere
       ripetizioni!
9  posizioni = zeros(8, 1);
10 for ii = 1:8
11     nuova_posizione = randi(64);
12     while sum(posizioni == nuova_posizione) > 0
13         nuova_posizione = randi(64);
14     end
15     posizioni(ii) = nuova_posizione;
16 end
17
18 % setta le posizioni nella scacchiera
19 scacchiera(posizioni) = 1;
20
21 % controlla che ci siano esattamente 8 regine
22 assert(sum(scacchiera, 'all') == 8)
23
24 % stampa la scacchiera
25 scacchiera
26
27
28 %Controllo righe
29 righe_ok = all(sum(scacchiera,2) <= 1);
30
31 %Controllo colonne
32 colonne_ok = all(sum(scacchiera) <= 1);
33
34 diag_ok = 1;
35 anti_diag_ok = 1;
36
37 %Controllo diagonali principali (alla C)
38 if righe_ok && colonne_ok && diag_ok
39     for ii = 1:7
40         somma = 0;
41         count_col = 1;
42         count_row = ii;
43         while (count_row <= 8)
44             somma = somma + scacchiera(count_row,count_col);
45             count_col = count_col + 1;
46             count_row = count_row + 1;
47         end
48         if somma > 1
```

```
            diag_ok = 0;
        end
    end
end

if righe_ok && colonne_ok && diag_ok
    for ii = 2:7
        somma = 0;
        count_col = ii;
        count_row = 1;
        while (count_col <= 8)
            somma = somma + scacchiera(count_row,count_col);
            count_col = count_col + 1;
            count_row = count_row + 1;
        end
        if somma > 1
            diag_ok = 0;
        end
    end
end

%Controllo diagonali principali (alla Matlab)
if righe_ok && colonne_ok
    sum_diag = zeros(13,1);
    for ii = -6:6
        sum_diag(ii+7) = sum(diag(scacchiera,ii));
    end
    diag_ok = all(sum_diag <= 1);
end

%Controllo antidiagonali
if righe_ok && colonne_ok && diag_ok
    antiscacchiera = flip(scacchiera);
    sum_anti_diag = zeros(13,1);
    for ii = -6:6
        sum_anti_diag(ii+7) = sum(diag(antiscacchiera,ii));
    end
    anti_diag_ok = all(sum_anti_diag <= 1);
end

if righe_ok && colonne_ok && diag_ok && anti_diag_ok
    disp('Le regine sono ben disposte');
else
    disp('Almeno una coppia di regine e'' mal disposta');
```

```
93 end
```