12 Strutture con MATLAB

Per inizializzare le strutture si può utillizare il costrutto:

```
S = struct('campo1',val1,'campo2',val2, ...);
```

oppure inizializzarne direttamente i campi con una serie di istruzioni:

```
S.campo1 = val1;
S.campo2 = val2;
...
```

12.1 Esercizi

Esercizio 12.1

Scrivere un programma per la gestione di un magazzino dove ogni prodotto nel magazzino è univocamente identificato da un codice a barre (un numero intero).

Il software di gestione associa ad ogni prodotto un carattere che indica la tipologia del prodotto e due numeri, il primo che indica il numero di pezzi in stock il secondo che indica il numero di pezzi ordinati.

Si ipotizzi che codice a barre, tipo, stock, ed ordine siano 4 vettori, già popolati, contenenti tutte le informazioni necessarie per la gestione del magazzino. (l'i-esimo elemento di stock e di ordine rappresentano le quantità relative al prodotto a cui è associato l'i-esimo elemento del vettore dei codici a barre).

Ad esempio un magazzino popolato sarà:

```
barcodes = [123 ; 1312 ; 12312 ; 1231 ; 99123];
tipo = ['A' ; 'A' ; 'X' ; 'W' ; 'W' ];
stock = [0 ; 300 ; 5 ; 6 ; 0 ];
ordine = [23 ; 100 ; 2 ; 100 ; 0 ];
```

Si strutturi la struttura magazzino e si scriva:

- la funzione `ricerca` che prende in ingresso un codice a barre ed il magazzino e restituisce un messaggio contenente il tipo di prodotto, il numero di pezzi in stock ed in ordine;
- un esempio di chiamata alla funzione `ricerca`;
- la funzione `ricercaMancanti` che, dato un parametro P ed il magazzino, restituisce al programma chiamante un vettore contenente i codici a barre dei prodotti:
  - se $P = 0$, non presenti in stock ma in ordine;
  - se $P = 1$, non presenti in stock che non sono nemmeno in ordine;
  - se $P = 2$, per cui ci sono più pezzi in ordine che attualmente in stock;
- Scrivere un esempio di chiamata alla funzione `ricercaMancanti`;
- Si scriva la funzione `aggiungiProdotto`, che permette di aggiungere al magazzino un nuovo prodotto (barcode, stock ed ordine);
- Scrivere un esempio di chiamata alla funzione `aggiungiProdotto`.

Esercizio 12.2

Scrivere un programma per simulare il gioco della roulette.

La roulette possiede 38 numeri (da 1 a 36, lo zero e il doppiozero). 0 (zero) e 00 (doppiozero) non sono ne pari ne dispari (vince il banco). Inizialmente, banco e giocatori possiedono 5000 euro ciascuno.

Implementare la simulazione di una serie di giocate di due giocatori Pippo e Pluto, che giocano seguendo le seguenti strategie:

- ad ogni giocata il giocatore Pippo punta 5 euro su pari o dispari con stessa probabilità. Se vince ottiene 2 volte la posta, se perde il banco incassa il valore giocato;
- ad ogni giocata il giocatore Pluto punta 1 euro sul 15 (se esce 15 vince 36 volte la posta).

Il gioco termina quando o il banco viene sbancato (arriva a 0 euro) o entrambi i giocatori non hanno più soldi per fare la propria puntata.

Si tenga traccia delle somme a disposizione di ogni giocatore e del banco ad ogni giocata dall'inizio del gioco fino alla sua fine. Grazie a queste informazioni, disegnare l'evoluzione della disponibilità monetaria dei due giocatori e del banco. Si disegnino i valori con delle linee di spessore 2, in rosso per Pippo, in blu per Pluto e in nero per il banco. Si disegni la legenda, il titolo e si forniscano le etichette per gli assi x e y.

Esercizio 12.3

Scrivere in `MATLAB` una funzione per analizzare i codici IBAN dei conti correnti. Un codice IBAN è una sequenza di 27 caratteri alfanumerici così composta:

- 2 caratteri maiuscoli (sigla della nazione)
- 2 cifre (CIN Europeo)
- 1 carattere maiuscolo (CIN italiano)
- 5 cifre (ABI)
- 5 cifre (CAB)
- 12 cifre (numero di conto corrente)

Si scrivano prima le seguenti tre funzioni:

- `remove_spaces`, che prende in ingresso `str_in` e restituisce `str_out` conte-

nente tutti i caratteri di `str_in` tranne gli spazi.

- `all_upper`, che prende in ingresso una stringa e restituisce 1 solo se la stringa contiene soltanto caratteri maiuscoli, 0 altrimenti.
- `all_digit`, che prende in ingresso una stringa e restituisce 1 solo se la stringa contiene solo caratteri corrispondenti a cifre, 0 altrimenti.

Si usino poi tali funzioni per scrivere la funzione `check_iban` che richiede all'utente l'inserimento di un codice IBAN e restituisce 1 solo se, una volta tolti gli spazi dalla stringa IBAN, essa rispetta lo schema previsto.

Esercizio 12.4

Il sistema di messaggistica di Facebook permette di ricevere messaggi da qualsiasi mittente. Un messaggio è caratterizzato da un mittente e da un testo.

Vogliamo implementare un sistema di filtraggio per rilevare automaticamente messaggi indesiderati, basandoci sulle seguenti ipotesi semplificative:

- se il messaggio è ricevuto da una persona conosciuta, ossia da una persona nella lista degli amici, allora il messaggio non è da scartare
- se il messaggio è ricevuto da una persona sconosciuta, ossia non presente nella lista degli amici, allora è necessario esaminare la storia dei messaggi ricevuti in passato, per determinare un "valore atteso" che ci permetta di decidere se il messaggio appena ricevuto è "nella media".

Quindi servirà una funzione:

```
1 [buono, motivo] = filtra_messaggio(messaggio, messaggi, amici);
```

che riceve in ingresso:

- `messaggio`: una struttura dati a due campi: mittente (nome e cognome) e testo. Ad esempio:

```
1 messaggio.testo = 'Ciao come stai?';
2 messaggio.mittente.nome = 'Federico';
3 messaggio.mittente.cognome = 'Maggi';
```

- `messaggi`: un vettore di messaggi (definiti come `messaggio`) contenente i messaggi ricevuti in passato;
- `amici`: un vettore contenente gli amici. Utilizzeremo una struttura dati contenente i campi nome e cognome.

e restituisce:

- `buono`: di tipo logical ed è vero solo se il messaggio è buono;
- `motivo`: di tipo char, e vale:
  - `a` ad indicare che il messaggio è buono perchè inviato da un amico;
  - `m` ad indicare che il messaggio è buono perchè "nella media" dei messaggi passati;
  - `x` ad indicare che il messaggio è cattivo perchè non ha passato nessuno dei due criteri precedenti;

Per capire se un messaggio è "nella media" controlleremo se la sua lunghezza, senza spazi, il numero di vocali e il numero di consonanti sono simili a quelli medi dei messaggi passati. Implementare:

```
1 [l v c] = estrai_caratteristiche(testo)
```

- `testo` è il testo del messaggio da analizzare;
- `l` è la lunghezza del messaggio, esclusi gli spazi;
- `v` è il numero di vocali;
- `c` è il numero di consonanti.

```
1 [Mm, Dm] = valore_atteso(messaggi)
```

- Mm e' un vettore riga di 3 colonne, con il valor medio dei tre valori [l v c] calcolati su tutti i messaggi
- Dm e' un vettore riga di 3 colonne, con la deviazione standard dei tre valori [l v c] calcolati su tutti i messaggi

```
1 buono = controlla_contenuto(messaggio, messaggi)
```

la quale ritornerà un valore logical vero solo se il messaggio ha le caratteristiche `[l v c]` che soddisfano tutte le tre seguenti condizioni:

- media(l) - sqrt(2) * std(l) <= l <= media(l) + sqrt(2) * std(l)
- media(v) - sqrt(2) * std(v) <= v <= media(v) + sqrt(2) * std(v)
- media(c) - sqrt(2) * std(c) <= c <= media(c) + sqrt(2) * std(c)

Supporre che le strutture dati `amici` e `messaggi` siano già disponibili in un file `facebook.mat` e siano caricate all'inizio dello script.

Esercizio 12.5

Modellizzare il gioco del Giacomonero. Esso si svolge nel seguente modo: il dealer assegna ad un giocatore due carte ed una a sè stesso. A questo punto il giocatore ha due possibilità: chiedere una carta o stare. Se la somma delle carte del giocatore supera il 21, egli sballa e risulta perdente. Se non ha sballato può continuare a chidere carte finché non decide di stare. Se alla fine di questo processo il giocatore non ha sballato, il dealer deve estrarre delle carte finché il suo punteggio è inferiore o uguale a 16. Se oltrepassa il 21 il banco sballa e vince il giocatore. In caso contrario, se il punteggio del banco è strettamente inferiore a quello del giocatore (e il giocatore non ha sballato), la vittoria va al giocatore, altrimenti al dealer.

Il gioco del Giacomonero si gioca con un 6 mazzi da 52 carte (dal due al re, quattro semi). Il punteggio delle figure (Fante, Donna e Re) equivale a 10, l'Asso vale a discrezione del giocatore 11 oppure 1. Le altre carte valgono quanto il loro numero.

Scrivere uno script e delle funzioni in `MATLAB` che:

- crei un mazzo di carte completo (`crea_mazzo`);
- mischi il mazzo ordinato (`mescola_mazzo`);
- estragga una carta (`estrai_carta`);
- conti i punti di una mano (`somma_carte`);
- implementi la logica del gioco (`main_giacomonero`);

Opzionale: implementare le il gioco del Blackjack nella sua versione originale, in modo da considerare le opzioni di split, assicurazioni e la regola sul 21 a due carte.[1]

Esercizio 12.6

Scrivere un programma che chieda all'utente di inserire una serie di dati contenenti ognuno i seguenti attributi:

- città (stringa)
- giorno (intero positivo)
- mese (intero positivo)
- anno (intero positivo)

[1] `https://en.wikipedia.org/wiki/Blackjack`

- tipo di misurazione (char)
- valore (reale)

Ad esempio, l'utente potrà inserire:

```
Milano
04
12
2012
10.5
N
```

Dopo aver acquisito una certa quantità di dati, il programma dovrà chiedere all'utente il nome di una città e un tipo di misurazione. A questo punto il programma cercherà nell'archivio tutti i record riguardanti la città e il tipo di misurazione richiesti. Stamperà poi a video i dati selezionati ed il relativo valore minimo, massimo e medio dei valori.

Soluzioni

Soluzione dell'esercizio 12.1

```
clear
clc
close all

%% Inizializzazione magazzino
magazzino.barcodes = [123 ; 1312 ; 12312 ; 1231 ; 99123];
magazzino.tipo = ['A' ; 'A' ; 'X' ; 'W' ; 'W' ];
magazzino.stock = [0 ; 300 ; 0 ; 6 ; 0 ];
magazzino.ordine = [23 ; 100 ; 2 ; 100 ; 0 ];

%% Chiamata a ricerca
messaggio = ricerca(magazzino,123);
disp(messaggio);

%% Chiamata a ricercaMancanti
prodotti_non_in_stock = ricercaMancanti(magazzino, 0);
disp(['Prodotti esauriti, ma in ordine: ' mat2str(
   prodotti_non_in_stock)]);
prodotti_non_ordinati = ricercaMancanti(magazzino, 1);
disp(['Prodotti esauriti e non in ordine: ' mat2str(
   prodotti_non_ordinati)]);
prodotti_esauriti = ricercaMancanti(magazzino, 2);
disp(['Prodotti con piu'' ordine che stock: ' mat2str(
   prodotti_esauriti)]);

%% Chiamata a aggiungiProdotti
barcode = 111;
tipo = 'X';
stock = 12;
ordine = 0;
magazzino = aggiungiProdotto(magazzino, barcode, tipo ,stock,
   ordine)
```

```
function msg = ricerca (magazzino, barcode)

bc_indici = find(magazzino.barcodes == barcode);
if isempty(bc_indici)
    msg = ['Il prodotto corrispondente al codice a barre ',
       num2str(barcode), ...
```

```
    ' non e'' in magazzino'];
else
    t = magazzino.tipo(bc_indici);
    s = magazzino.stock(bc_indici);
    o = magazzino.ordine(bc_indici);

    msg = ['Il prodotto corrispondente al codice a barre ',
       num2str(barcode), ...
    ' e'' di tipo ', num2str(t), '. Elementi in stock: ', ...
    num2str(s), ', in ordine: ', num2str(o) '.'];
end
```

```
function prodotti = ricercaMancanti(magazzino, P)

switch P
    case 0 % esauriti ma in ordine
        bc_indici = find(magazzino.stock == 0 & magazzino.
           ordine > 0);
    case 1 % esauriti e non in ordine
        bc_indici = find(magazzino.stock == 0 & magazzino.
           ordine == 0);
    case 2 % prodotti con piu' ordine che stock
        bc_indici = find(magazzino.ordine > magazzino.stock);
end

prodotti = magazzino.barcodes(bc_indici);
```

```
function magazzino = aggiungiProdotto(magazzino, barcode, tipo
   ,stock, ordine)

magazzino.barcodes = [magazzino.barcodes; barcode];
magazzino.tipo = [magazzino.tipo; tipo];
magazzino.stock = [magazzino.stock; stock];
magazzino.ordine = [magazzino.ordine; ordine];
```

Soluzione dell'esercizio 12.2

```
clear
close all
clc

cifra_iniziale = 50;

```

```
7  banco = cifra_iniziale;
8  storicoBanco = cifra_iniziale;
9
10 giocatore.nome = 'Pippo';
11 giocatore.budget = cifra_iniziale;
12 giocatore.posta = 5;
13 giocatore.fattoreVittoria = 1;
14 giocatore.storicoBudget = cifra_iniziale;
15
16 giocatore(2).nome = 'Pluto';
17 giocatore(2).budget = cifra_iniziale;
18 giocatore(2).posta = 1;
19 giocatore(2).fattoreVittoria = 36;
20 giocatore(2).storicoBudget = cifra_iniziale;
21
22 % iterazioni del gioco
23 while (siContinuaAGiocare(giocatore, banco))
24
25     % scegliere giocata del giocatore1
26     % se dispari == 0 giocatore 1 sceglie pari
27     % se dispari == 1 giocatore 1 sceglie dispari
28     dispari = round(rand(1));
29
30     % giro la roulette, numero random tra 0 - 37
31     % 37 equivale a 00
32     numero = giraLaRoulette();
33
34     %Calcolo del vettore della vittoria dei giocatori
35     if(numero == 37 || numero == 0)
36         vince([1, 2]) = 0;
37     else
38         if(mod(numero,2) == dispari)
39             vince(1) = 0;
40         else
41             vince(1) = 1;
42         end
43
44         if numero == 15
45             vince(2) = 1;
46         else
47             vince(2) = 0;
48         end
49     end
50
```

```
51      %Calcolo ricompense giocatori e banco
52      for ii = 1 : numel(giocatore)
53          if giocatore(ii).budget >= giocatore(ii).posta
54              if vince(ii) == 0
55                  %Sconfitta giocatore
56                  giocatore(ii).budget =  giocatore(ii).budget -
                        giocatore(ii).posta;
57                  banco =  banco + giocatore(ii).posta;
58              elseif vince(ii) == 1
59                  %Vittoria giocatore
60                  giocatore(ii).budget =  giocatore(ii).budget +
                        giocatore(ii).fattoreVittoria * giocatore(ii
                        ).posta;
61                  banco =  banco - giocatore(ii).fattoreVittoria
                        * giocatore(ii).posta;
62              end
63          end
64      end
65
66      %Aggiorno storico giocatori e banco
67      for ii = 1 : numel(giocatore)
68          giocatore(ii).storicoBudget(end + 1) = giocatore(ii).
                budget;
69      end
70      storicoBanco(end + 1) = banco;
71
72  end
73
74  plotRoulette(giocatore, storicoBanco);
```

```
1   function numero = giraLaRoulette()
2
3   numero = randi(38)-1;
```

```
1   function res = siContinuaAGiocare(giocatore, banco)
2
3   budgetCorrenti = [giocatore.budget];
4   posta = [giocatore.posta];
5   res = (any(budgetCorrenti >= posta) && banco > 0);
```

```
1   function plotRoulette(giocatore, storicoBanco)
2
3   spessore = 2;
```

```
4
5  figure();
6  plot(giocatore(1).storicoBudget , 'r' , 'LineWidth' , spessore)
7  hold on;
8  plot(giocatore(2).storicoBudget, 'b' , 'LineWidth' , spessore)
9  plot(storicoBanco , 'k' , 'LineWidth' , spessore)
10 title('Evoluzione della Roulette nel tempo');
11 xlabel('Numero giocata');
12 ylabel('Euro');
13 legend(giocatore(1).nome , giocatore(2).nome, 'Banco','Location
      ','northwest');
```

Soluzione dell'esercizio 12.3

```
1 clear
2 clc
3 close all
4 %IT 02 L 12345 12345 123456789012
5 check_iban()
```

```
1 function str_out = remove_spaces(str_in)
2     str_out = str_in(str_in ~= ' ');
3 end
```

```
1 function str_out = all_alpha(str_in)
2     str_out = all(str_in >= 'A' & str_in <= 'Z');
3 end
```

```
1 function r = all_digit(str_in)
2     r = all(str_in >= '0' & str_in <= '9');
3 end
```

```
1 function is_valid = check_iban()
2     % Inserimento IBAN
3     iban = input('Inserire IBAN: ', 's');
4
5     % Rimuovo spazi
6     iban = remove_spaces(iban);
7
8     % Controllo validita'
9     is_valid = all_alpha(iban(1:2)) & all_digit(iban(3:4)) &
         all_alpha(iban(5)) & ...
```

```
        all_digit(iban(6:end)) & length(iban) == 27;

    % Oppure:
    %is_valid = all_alpha(iban([1, 2, 5])) & all_digit(iban([3, 4, 6:end])) & ...
    %    length(iban) == 27;
end
```

Soluzione dell'esercizio 12.4

```
clear
clc
close all

load('facebook.mat', 'amici', 'messaggi');

% Nuovo messaggio
% messaggio.testo = 'Ciao come stai?';
% messaggio.mittente.nome = 'Federico';
% messaggio.mittente.cognome = 'Maggi';
%
% % Controllo del contenuto di un messaggio, dati i messaggi precedenti
% [buono, motivo] = filtra_messaggio(messaggio, messaggi, amici);

% Nuovo messaggio
messaggio.testo = 'Cras aliquam massa ullamcorper sapien';
messaggio.mittente.nome = 'Federico';
messaggio.mittente.cognome = 'Maggi';

% Controllo del contenuto di un messaggio, dati i messaggi precedenti
[buono, motivo] = filtra_messaggio(messaggio, messaggi, amici);
```

```
function [buono, motivo] = filtra_messaggio(messaggio, messaggi, amici)

buono = 0;
for ii = 1:numel(amici)
    if strcmp(messaggio.mittente.nome,amici(ii).nome) && ...
            strcmp(messaggio.mittente.cognome,amici(ii).cognome)
```

```
        buono = 1;
        motivo = 'a';
        return;
    end
end


if controlla_contenuto(messaggio, messaggi)
    buono = 1;
    motivo = 'm';
else
    motivo = 'x';
end
```

```
function [l, v, c] = estrai_caratteristiche(testo)

spazi = testo == ' ';
vocali = testo == 'a' | testo == 'e' | testo == 'i' | testo ==
   'o' | testo == 'u';
consonanti = ~spazi & ~vocali;

l = length(testo(~spazi));
v = length(testo(vocali));
c = length(testo(consonanti));
```

```
function [Mm, Dm] = valore_atteso(messaggi)

N = length(messaggi);
M = zeros([N 3]);

% per ogni messaggio
for ii = 1:N
    msg = messaggi(ii).testo;
    [l, v, c] = estrai_caratteristiche(msg);
    M(ii, :) = [l v c];
end

Mm = mean(M);
Dm = std(M);
```

```
function buono = controlla_contenuto(messaggio, messaggi)

% calcolo valore atteso su tutti i messaggi
```

```
[Mm, Dm] = valore_atteso(messaggi);

% caratteristiche del testo da esaminare
msg = messaggio.testo;
[l, v, c] = estrai_caratteristiche(msg);
F = [l v c];

% estremi inferiori
int_inf = Mm - sqrt(2) * Dm;

% estremi superiori
int_sup = Mm + sqrt(2) * Dm;

% confronto tutte le caratteristiche con l'intervallo cosi`
   costruito
buono = all(F >= int_inf) && all(F <= int_sup);
```

Soluzione dell'esercizio 12.5

```
clear
clc
close all

mazzo = crea_mazzo();
mazzo = mescola_mazzo(mazzo);

[mazzo, carte_banco] = estrai_carta(mazzo);
disp(['Il banco ha ' carte_banco.numero ' di ' carte_banco.seme
   ]);
[mazzo, carte_giocatore] = estrai_carta(mazzo);
disp(['Il giocatore ha ' carte_giocatore.numero ' di '
   carte_giocatore.seme]);
[mazzo, carte_giocatore(2)] = estrai_carta(mazzo);
disp(['Il giocatore ha ' carte_giocatore(2).numero ' di '
   carte_giocatore(2).seme]);

play = 'S';
while (play == 'S')
    %Giocata giocatore
    play = input('Vuoi una carta? (S,N) ','s');
    if play == 'S'
        [mazzo, carte_giocatore(end+1)] = estrai_carta(mazzo);
```

```
21          disp(['Il giocatore ha pescato ' carte_giocatore(end).
               numero ' di ' carte_giocatore(end).seme]);
22      end
23
24      if somma_carte(carte_giocatore) > 21
25          disp('Hai sballato!!!');
26          play = 'N';
27      end
28
29  end
30
31  %% Giocata banco
32  flag_banco = 0;
33  while somma_carte(carte_giocatore) <= 21 && flag_banco == 0
34      [mazzo, carte_banco(end+1)] = estrai_carta(mazzo);
35      disp(['Il banco ha pescato ' carte_banco(end).numero ' di '
            carte_banco(end).seme]);
36
37      if somma_carte(carte_banco) > 16
38          flag_banco = 1;
39      end
40  end
41
42  %% Controllo vittoria
43  if somma_carte(carte_giocatore) > somma_carte(carte_banco) &&
       somma_carte(carte_giocatore) < 22
44      disp('Hai vinto');
45  else
46      disp('Vince il banco');
47  end
```

```
1   function mazzo = crea_mazzo()
2
3   numeri = '1234567689JQK';
4   semi = 'CQFP';
5   count = 1;
6   singolo_mazzo = struct('seme',[],'numero',[]);
7   for ii = 1:length(semi)
8       for jj = 1:length(numeri)
9           singolo_mazzo(count).seme = semi(ii);
10          singolo_mazzo(count).numero = numeri(jj);
11          count = count + 1;
12      end
13  end
```

```
14
15 mazzo = [];
16 for ii = 1:6
17     mazzo = [mazzo singolo_mazzo];
18 end
```

```
1  function mazzo = mescola_mazzo(mazzo)
2
3  n_carte = length(mazzo);
4  ind = randperm(n_carte);
5  mazzo = mazzo(ind);
```

```
1  function [mazzo, carta] = estrai_carta(mazzo)
2
3  carta = mazzo(1);
4  mazzo(1) = [];
```

```
1  function somma = somma_carte(carte)
2
3  somma = 0;
4  flag_asso = 0;
5  for ii = 1:length(carte)
6      if carte(ii).numero == 'J' || carte(ii).numero == 'Q' ||
          carte(ii).numero == 'K'
7          somma = somma + 10;
8      elseif str2double(carte(ii).numero) > 1
9          somma = somma + str2double(carte(ii).numero);
10     elseif flag_asso == 0
11         somma = somma + 11;
12         flag_asso = 1;
13     else
14         somma = somma + 1;
15     end
16 end
17
18 if somma > 21 && flag_asso == 1
19     somma = somma - 10;
20 end
```

Soluzione dell'esercizio 12.6

```
1  clear
2  clc
```

```
close all

% acquisizione dati
dati = acquisizione_dati_meteo();

% richiesta dato da visualizzare
[city, tipo] = interrogazione_archivio_meteo();

% ricerca dati e restituzione min, media, max
[dati_selezionati, minimo, medio, massimo] = ...
    calcolo_statistiche_meteo(dati, city, tipo);

% stampa a video delle statistiche
stampa_statistiche(dati_selezionati, city, tipo, minimo, medio,
    massimo);
```

```
function dati = acquisizione_dati_meteo()
    next = 1;
    dati = [];
    ii = 0;

    while next == 1
        ii = ii + 1;

        dati(ii).city = input('Citta'': ','s');
        dati(ii).giorno = input('Giorno: ');
        dati(ii).mese = input('Mese: ');
        dati(ii).anno = input('Anno: ');
        dati(ii).tipo = input('Tipo: ','s');
        dati(ii).valore = input('Valore: ');

        next = input('Per inserire un nuovo record premere 1,
           altrimenti 0: ');
    end

    fprintf('%d dati inseriti.\n', ii);
end
```

```
function [city, tipo] = interrogazione_archivio_meteo()
    city = input('Citta` di interesse: ', 's');
    tipo = input('Tipo misura da selezionare: ', 's');
end
```

```
function [dati_selezionati, minimo, medio, massimo] = ...
    calcolo_statistiche_meteo(dati, city, tipo)

    for ii = 1:numel(dati)
        res(ii) = strcmp(dati(ii).city,city);
    end

    indici = res & [dati.tipo] == tipo;

    dati_selezionati = dati(indici);

    minimo = min([dati_selezionati.valore]);
    massimo = max([dati_selezionati.valore]);
    medio = mean([dati_selezionati.valore]);
end
```

```
function stampa_statistiche(dati_selezionati, city, tipo,
   minimo, medio, massimo)
    fprintf('Statistiche della misura %c in citta'' %s\n', tipo
       , city);

    for r = dati_selezionati
        fprintf('%d/%d/%d %f\n', r.giorno, r.mese, r.anno, r.
           valore);
    end

    fprintf('\nMin: %3.2f, med: %3.2f, max: %3.2f\n', minimo,
       medio, massimo);
end
```