Politecnico di Milano
Informatica B, AA 2021/2022
Laboratorio 4

Luca Frittoli (luca.frittoli@polimi.it)
Mirko Salaris (mirko.salaris@polimi.it)

1 Dicembre 2021

1. **Riscaldamento** Costruisci le seguenti variabili senza fare utilizzo di cicli (e senza scriverle "manualmente").

```
a =

     1     2     3     4     5

b =

    50    40    30    20    10

c =

    51    42    33    24    15

mat1 =

     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0    57

mat2 =
     1     2     3     4     5
     2     4     6     8    10
     3     6     9    12    15
     4     8    12    16    20
     5    10    15    20    25

mat3 =

     1           2           3           4           5
     1           4           9          16          25
     1           8          27          64         125
     1          16          81         256         625
     1          32         243        1024        3125
```

Soluzione

```
a = 1:5;
b = 50:-10:10;
c = a+b;

mat1(5,5) = 57;
mat2 = a.*a';
mat3 = a.^(a')
```

2. **Acquisizione da tastiera** Si scriva un programma che prenda in input una sequenza di al più 10 numeri pari, e ne stampi la media. L'acquisizione dovrà terminare se viene inserito un numero dispari.

Soluzione

```
k = 0;
pari = true;
media = 0;
while (pari && (k < 10))
    a = input('Inserire un numero, dispari per terminare: ');
    pari = (mod(a, 2) == 0);
    if pari
        media = (media * k + a) / (k+1);
    end
    k = k + 1;
end
disp(media);
```

3. **Unità imperiali** Si scriva un programma che prenda in input una capacità espressa in galloni (gal) e pinte (pt) e la converta in litri (l). Conversione: 1 gal = 8 pt, 1 pt = 0.568 l.

Soluzione

```
% A differenza del C, in Matlab è possibile acquisire direttamente degli array.
% La soluzione con due richieste separate e' comunque perfettamente valida.
A = input('Inserire una misura in galloni e pinte: ');
M = (A(1) * 8 + A(2)) * 0.568
```

4. **Polinomi** Scrivi un programma per calcolare il valore di un polinomio in un dato punto. Il programma chiede all'utente i coefficienti del polinomio e successivamente il valore della variabile x, e calcola la y corrispondente.

   Bonus: stampa il grafico del polinomio nell'intervallo $[-5, 5]$, evidenziando il punto (x, y) calcolato in precedenza.

   Suggerimento: per i coefficienti del polinomio è possibile utilizzare la funzione `input()` una sola volta; l'utente dovrà inserire i coefficienti tra parentesi quadre e separati da spazi.

Esempio: a input dell'utente "`[1 0 3 4]`" e successivamente 2 come valore per la x, corrisponderà il polinomio $x^0 + 3x^2 + 4x^3$ e quindi il risultato 45.

Soluzione

```
coeff = input("Inserisci i coefficienti del polinomio," + ...
    "tra parentesi quadre e separati da spazi");

x = input("Inserisci il valore di x");

gradi(1:length(coeff)) = (1:length(coeff)) - 1;
y = coeff*(x.^gradi)';
display(y)

% bonus
ascisse = -5:0.1:5;
ordinate = zeros(size(ascisse));
for i=1:length(coeff)
    ordinate = ordinate + coeff(i) * ascisse .^ gradi(i);
end

plot(ascisse,ordinate)
hold on
plot(x,y,'o')
```

5. **Shift a sinistra** Si scriva un programma che, dato un array di cifre binarie, effettui lo shift a sinistra fino a quando non si ottiene un 1 nella prima posizione.

   Esempio: `[0 0 1 1 0 1 0]` $\rightarrow$ `[1 1 0 1 0 0 0]`.

Soluzione

```
a = input('Inserire un vettore riga contenente solo i valori 1 e 0: ');
ii = 1;
% cerchiamo il primo indice il cui elemento sia diverso da zero
while a(ii) == 0
    ii = ii+1;
end
% Creiamo un vettore formato dalle due parti di a:
    % gli elementi dal primo indice non zero fino alla fine
    % gli elementi dal primo al primo indice non zero (escluso)
risultato = [a(ii:end) a(1:ii-1)];

% --- oppure ---

% Usiamo find per trovare gli indici in cui a non e` zero
indicinonzero = find(a);
% Salviamo solo il primo indice non zero
primoindicenonzero = indicinonzero(1);
```

```
% Come sopra, creiamo un vettore formato dalle due parti di a
risultato = [a(primoindicenonzero:end) a(1:primoindicenonzero-1)]
```

6. **Matrici simmetriche** Si scriva un programma che prenda in input una matrice quadrata e verifichi se è simmetrica o antisimmetrica.

   Suggerimento: una matrice quadrata `A` è simmetrica se `A(k,l) = A(l,k)` per ogni coppia di indici `k,l`, mentre è antisimmetrica se `A(k,l) = -A(l,k)` per ogni coppia di indici `k,l`. Per controllare se due matrici sono uguali è possibile utilizzare la funzione `isequal`.

   Bonus: si risolva l'esercizio senza utilizzare cicli.

Soluzione

```
A = input('Inserire una matrice quadrata: ');
if isequal(A, A')
    disp('Matrice simmetrica')
elseif isequal(A, -A')
    disp('Matrice antisimmetrica')
else
    disp('Matrice ne` simmetrica ne` antisimmetrica')
end
```

7. **Approssimazione di π** Si scriva un programma che calcoli un'approssimazione di π usando la serie di Gregory-Leibniz:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \ldots + \frac{(-1)^n}{2n+1} + \ldots$$

   Il programma dovrà prendere in input l'indice N a cui troncare la somma e stampare l'errore assoluto commesso nell'approssimazione.

   Suggerimento: Matlab dispone di tantissime costanti matematiche come π (`pi`).

Soluzione

```
N = input('Inserire N: ');
% calcoliamo 4 volte la somma della serie di Gregory-Leibniz fino a N
s = 4 * sum((-1).^(0:N) ./ (2*(0:N)+1))
% calcoliamo l'errore assoluto rispetto al valore esatto di pi
err = abs(pi - s)
```

8. **Crivello di Eratostene** Implementare il Crivello di Eratostene, un semplice algoritmo per trovare tutti i numeri primi entro un certo N inserito dall'utente. L'algoritmo prende in input un array, detto "setaccio", contenente i numeri interi da 2 a N (1 non è un numero primo!). L'algoritmo cancella dall'array tutti i multipli del primo numero del setaccio (escluso il primo multiplo), poi tutti i multipli del secondo numero (escluso il primo multiplo), e così via, finché non rimangono solo i numeri primi.

   Suggerimento 1: un modo per cancellare certi elementi di un array è cambiare il loro valore in `[]`.
   Suggerimento 2: non è necessario scorrere tutti i numeri del setaccio, bastano quelli il cui quadrato

non supera N. Infatti, se $p < N$ è un numero primo tale che $p^2 > N$, i suoi multipli $p \cdot r$ con $r < p$ sono già stati cancellati dall'algoritmo, quindi il minimo multiplo non ancora cancellato sarebbe p^2, che però è già fuori dal setaccio in quanto $p^2 > N$.

Soluzione

```
N = input('Inserire il massimo numero del setaccio: ');
setaccio = 2:N;
ii = 2;
while ii^2 < N
    setaccio(mod(setaccio,ii) == 0 & fix(setaccio/ii) > 1) = [];
    ii = ii + 1;
end

primi = setaccio
```

9. **Sottomatrici** Si scriva un programma che prenda in input due matrici `A` e `B` e controlli se `B` è una sottomatrice di `A`.

Soluzione

```
A = input('Inserisci una matrice: ');
B = input('Inserisci una seconda matrice: ');
% Flag
sottomatrice = false;
% Calcoliamo le dimensioni di A e B
[N_A M_A] = size(A);
[N_B M_B] = size(B);
% Proviamo tutte le possibili posizioni in cui B puo` essere sottomatrice di A
for k = 1 : N_A-N_B+1
    for l = 1 : M_A-M_B+1
        % Controlliamo se B e` sottomatrice partendo da (k,l)
        if isequal(A(k : k+N_B-1, l : l+M_B-1), B)
            sottomatrice = true;
            break;
        end
    end
end
if sottomatrice
    disp('B e` sottomatrice di A');
else
    disp('B non e` sottomatrice di A');
end
```

10. **Numeri di Fibonacci** Si scriva una funzione `fibonacci` che prenda in input un intero positivo n e calcoli il corrispondente elemento nella successione di Fibonacci $0, 1, 1, 2, 3, 5, 8, 13, \ldots$. Si testi la funzione utilizzando il seguente script:

```
n = randi([1,20], 10);
passed_test = 0;
for i=1:10
    if fibonacci(n(i)) + fibonacci(n(i)+1) == fibonacci(n(i)+2)
        passed_test = passed_test + 1;
    else
        disp(['did not pass test ', num2str(i)])
    end
end

disp(['Passed ', num2str(passed_test), ' out of 10'])
```

Suggerimento: si ricordi che ogni elemento della successione di Fibonacci (eccetto i primi due, che sono per definizione $0, 1$) è la somma dei due precedenti.

Soluzione

```
function fn = fibonacci(n)
    if n <= 0 || floor(n) ~= ceil(n)
        error('n deve essere un intero positivo');
    elseif n == 1
        fn = 0;
    elseif n == 2
        fn = 1;
    else
        fn = 1;
        f1 = 0;
        for k=3:n
            fn = fn + f1;
            f1 = fn - f1;
        end
    end
end
```