MATLAB come linguaggio di programmazione

- Anche gli algoritmi più semplici richiedono l'esecuzione ripetuta di istruzioni e l'esecuzione condizionata di alcune parti;
- La costruzione ripetuta di blocchi di codice in MATLAB viene eseguita tramite cicli;
- Esistono due diversi modi per realizzare cicli:
- il ciclo incondizionato o ciclo a contatore
  - for ... end
- ed il ciclo condizionato
  - while ... end

Costrutti sintattici - Ciclo for

- Sintassi semplificata:

```
for  variabile = inizio : incremento : fine
        blocco istruzioni
end
```

L'incremento può essere omesso, in tal caso il valore di default è 1.

- **Esempio 1:** per il calcolo della somma dei primi 100 numeri interi; S=0;

```
for i = 1:100
    S = S + i;
end
```

- La sintassi più generale è:

```
for variabile = espressione
        blocco istruzioni
end
```

- dove espressione è in generale un array;
- L'effetto del costrutto è di assegnare, ad ogni ripetizione del ciclo, a variabile una delle colonne di espressione e di ripetere quindi il blocco di istruzioni tante volte quante sono le colonne di espressione.

- **Esempio 2**.
- Ricordando che l'istruzione 0:2:m genera un vettore riga contenente valori compresi fra 0 e m intervallati di due unità, il seguente ciclo ripeterà il blocco di istruzioni per j = 0, 2, 4, ..., m:

```
>> m = 10;
>> for j = 0:2:m
       x(j+1) = j + 1;
   end
>> x =
    1 0 3 0 5 0 7 0 9 0 11
```

- Un uso meno familiare è il seguente.

```
> > S = 0;
    for i = [1 15 100]
        S = S + i;
    end
```

- Essendo [1 15 100] un vettore riga, la variabile i assumerà successivamente i valori 1, 15, 100 e quindi il blocco di istruzioni sarà ripetuto tre volte;
- al termine del ciclo la variabile S varrà 116.

- Se invece consideriamo il ciclo seguente

```
> > S = 0;
    for i = [ 1 15 100]'
    S = S + i;
end
```

- questa volta l'espressione [1 15 100]' è costituita da un vettore colonna;
- alla variabile i viene assegnato quindi una sola volta il vettore colonna [1 15 100]' e il blocco di istruzioni verrà eseguito una volta;
- la variabile S al termine del ciclo conterrà il vettore [1 15 100]'.

- È bene notare che il ciclo for dovrebbe essere evitato ogniqualvolta sia possibile risolvere il problema utilizzando comandi appositi;
- È più veloce ed efficiente definire la matrice diagonale D utilizzando il comando diag:

```
» D = diag(1:4)
D =
1 0 0 0
0 2 0 0
0 0 3 0
0 0 0 4
```

- Piuttosto che ricorrere ad un ciclo for:

```
» for j = 1 : 4
D(j,j) = j ;
end
```

Costrutti sintattici - Ciclo while

- Gli esempi precedenti hanno la caratteristica di ripetere un blocco di istruzioni un numero prefissato di volte.
- Spesso si ha la necessità di ripetere un certo numero di operazioni diverse volte a seconda che una certa condizione sia verificata oppure no.
- In questo caso si utilizza il costrutto

```
while Condizione
    blocco di istruzioni
end
```

- Dove Condizione è un'espressione che MATLAB valuta numericamente e che viene interpretata come vera se diversa da zero.

- **Esempio 3:**

```
x = 0 ; k = 1 ;
while k <= 10
   x(k) = k^2 ;
   k = k + 1 ;
end
» k
k =
11
» x
x =
1 4 9 16 25 36 49 64 81 100
```

- Naturalmente sarebbe stato più veloce ed elegante definire il vettore x nella maniera seguente:

```
» x = [1:10] .^ 2
```

- **Esempio 4:** (Calcolo di successioni di numeri interi)
- Consideriamo il problema del calcolo della seguente successione di numeri interi generata a partire da a_1 numero intero: $a_{n+1} = 3a_n - 1;\ n \geqslant 1$
- Ad esempio per $a_1 = 9$ la successione fornisce i termini 9; 26; 77; 230; 689; ... e continua a crescere indefinitamente.
- Chiaramente per generare i primi 20 elementi della successione potremmo utilizzare un ciclo for come in precedenza.
- Se si dovesse invece arrestare il calcolo della successione quando $a(n) > a^*$ assegnato

- Una soluzione possibile è rappresentata dalla seguente funzione:

```
function x = successione(a1,as)
 % Sintassi x = successione(a1,as)
 % Calcola la successione
 % a(n+1)= 3*a(n)-1
 % partendo da a(1)=a1, fino a quando a(n) < as
 a(1)=a1;
 n=1;
 while a(n) < as
 a (n+1) = 3*a(n)-1;
 n = n+1;
 end
```

STRUTTURE
if – elseif – else – end

- Spesso si pone la necessità di dover eseguire un comando solo qualora sia verificata una data condizione.
- In questo caso si fa ricorso alla struttura if ... end che assume la forma:

```
if condizione
    corpo di istruzioni
end
```

- In questo caso il corpo di istruzioni viene eseguito solo se l'espressione condizione risulta vera.

Esempio 5:

```
» a = -5 ;
if a < 0
   a = -1
end
» a =
-1
```

- Per i casi in cui c'è una alternativa si utilizza il seguente costrutto:

```
if condizione
   corpo di istruzioni 1
else
   corpo di istruzioni 2
end
```

- Se l'espressione condizione risulta vera viene eseguito il corpo di istruzioni 1;
- altrimenti viene eseguito il corpo di istruzioni 2.
- Notare che uno dei due corpi di istruzioni viene sempre eseguito.
- **Esempio 6:**

```
» a = 5 ;
if a < 0
   a = -1
else
   a = 1
end
» a =
1
```

- Quando ci sono più di due alternative il costrutto da usare è il seguente:

```
if condizione1
        corpo di istruzioni 1
elseif condizione2
        corpo di istruzioni2
elseif condizione3
        corpo di istruzioni3
elseif ....
        ....
end
```

- **Esempio 7:**

```
» a = -5 ;
if a < 0
   a = -1
elseif a > 0
   a = 1
elseif a == 0
   a
end
» a =
-1
```

- Notare che utilizzando questo costrutto può darsi che nessuno dei *corpi di istruzioni* venga eseguito; Per evitare che ciò accada si può inserire un else come ultimo *statement* prima della chiusura del ciclo.

- Il comando break consente di uscire in maniera forzata da un ciclo ed evita in questo caso che siano calcolati più di N termini della successione.
- Quando il comando break viene eseguito MATLAB salta automaticamente all'istruzione end che termina il ciclo.
- Un comando che svolge una funzione analoga in MATLAB è return.
- La differenza è che return interrompe l'esecuzione della funzione e ritorna al programma da cui tale funzione era stata chiamata.

La struttura **switch**

- La struttura switch è un'alternativa all'impiego delle istruzioni condizionali if, else, elseif.
- La sintassi è la seguente:

```
switch   Espressione  (scalare o stringa)
  case  Valore1
        blocco di istruzioni 1
  case  Valore 2
        blocco di istruzioni  2
        …
  otherwise
        blocco di istruzioni n
end
```

- Tale struttura risulta particolarmente utile qualora si debbano eseguire numerose scelte di tipo esclusivo (se una è verificata le altre sono false).
- I blocchi di istruzioni sono eseguiti solo se Espressione assume il corrispondente Valore.
- L'ultimo blocco di istruzioni sarà eseguito solo nel caso in cui Espressione non abbia assunto nessuno dei precedenti valori.
- Tutto ciò che può essere programmato con switch può essere programmato anche con le strutture if.

- Ad esempio il blocco di istruzioni della function successione2b che corrisponde alla scelta tra x(n) pari e dispari diventerà:

```
z = rem(x(n),2) ;
switch z
   case 0
    x(n+1) = x(n)/2+1;
   case 1
    x(n+1) = 3*x(n)-1;
   otherwise
    disp('Non hai inserito un numero intero');
end
```

- L'istruzione disp consente di visualizzare una stringa di testo sullo schermo del calcolatore.

Osservazioni

- Le strutture condizionali non necessitano dell'apposizione then dopo il comando if, a differenza di quanto accade in altri linguaggi;
- Ad ogni comando if, for, while deve necessariamente corrispondere un comando end; questo è di particolare importanza in presenza di cicli annidati.
- Tutte le istruzioni sopra introdotte si possono digitare su linee diverse oppure sulla stessa linea di comando, separate da virgole.
- La prima modalità di scrittura favorisce la leggibilità di un codice, mediante l'uso dell'indentazione.