4.3. Εντολές Ελέγχου και Επανάληψης

Οι εντολές ελέγχου ροής (flow control) μας επιτρέπουν να δημιουργήσουμε προγράμματα με επαναληπτικές διαδικασίες & αλγορίθμους, να κάνουμε συγκρίσεις και να παίρνουμε αποφάσεις. Για να γίνει αυτό, όλες οι γλώσσες προγραμματισμού διαθέτουν:

α) **τελεστές για συγκρίσεις και λογικές πράξεις** (<, or, and),
β) **εντολές έλεγχου** (if, switch-case), και,
γ) **εντολές επαναλήψεων** (for, while, do).

4.3.1. Λογικές Εκφράσεις και Τελεστές Σύγκρισης

Το MatLab διαθέτει 6 τελεστές σύγκρισης των οποίων τα αποτελέσματα δίνονται πάντοτε σαν λογικές τιμές Αληθές-Ψευδές (True-False) μέσω μητρώων που περιέχουν, αντίστοιχα: 1 ή 0.

Μαθημ. Σύμβολο	Τελεστής MatLab	Περιγραφή
<	**<**	μικρότερο
≤	**<=**	μικρότερο ή ίσο
>	**>**	μεγαλύτερο
≥	**>=**	μεγαλύτερο ή ίσο
=	**==**	ίσο
≠	**~=**	διάφορο

Παραδείγματα Συγκρίσεων:

```
>> A = [6 4 2 0];  ↵
>> B = [1 3 5 0];  ↵

>> A > B  ↵
ans =
     1     1     0     0
>> A < B  ↵
ans =
     0     0     1     0
>> A <= B  ↵
ans =
     0     0     1     1
>> A ~= B  ↵
ans =
     1     1     1     0
```

Επιπλέον των τελεστών σύγκρισης, το MatLab όπως και όλες οι γλώσσες προγραμματισμού, διαθέτει τρεις (3) ακόμη τελεστές για τη σύνδεση των λογικών συγκρίσεων και τη δημιουργία πιο σύνθετων λογικών εκφράσεων, τους:

Μαθ. Σύμβολο	Τελεστής MatLab	Περιγραφή
NOT	**~**	όχι
AND	**&**	και
OR	**|**	ή

Όπως και στις αριθμητικές πράξεις, υπάρχει μια ιεραρχία στην εκτέλεση των λογικών πράξεων που ακολουθεί τη παρακάτω σειρά:

1. Αν υπάρχουν παρενθέσεις, υπολογίζονται αρχίζοντας από την εσωτερική.
2. Αν υπάρχουν αριθμητικές πράξεις εκτελούνται πριν από κάθε σύγκριση.
3. Υπολογίζονται όλοι οι τελεστές σύγκρισης
4. εκτελούνται τα NOT (~)
5. εκτελούνται τα AND (&)
6. εκτελούνται τα OR (|)

Ο πίνακας αληθείας των τριών τελεστών (~, &, |) είναι ο παρακάτω:

A	B	~A	A&B	A|B
Α - 1	Α - 1	Ψ - 0	Α - 1	Α - 1
Α - 1	Ψ - 0	Ψ - 0	Ψ - 0	Α - 1
Ψ - 0	Α - 1	Α - 1	Ψ - 0	Α - 1
Ψ - 0	Ψ - 0	Α - 1	Ψ - 0	Ψ - 0

Παράδειγμα Σύνθετης Λογικής Έκφρασης:

```
>> A = [6 4 2 0];
>> B = [1 3 5 0];

>> A < 5
ans =
     0     1     1     1
>> B < 5
ans =
     1     1     0     1
>> (A < 5) & ~(B < 5)
ans =
     0     0     1     0
```

4.3.2. Λογικές Συναρτήσεις

Εκτός των λογικών και συγκριτικών τελεστών, το MatLab διαθέτει και δεκάδες λογικών συναρτήσεων. Οι συναρτήσεις αυτές υλοποιούν σύνθετες λογικές πρέξεις ή ελέγχους που συναντάμε πολύ συχνά στο προγραμματισμό και διευκολύνουν τον χρήστη στην ανάπτυξη πολύπλοκων προγραμμάτων.

Τέτοιες συναρτήσεις είναι οι:

- **xor(A,B)** η οποία εκτελεί τη λογική πράξη A XOR B (Exclusive OR).
- **any(X)** η οποία επιστρέφει 1 αν έστω και ένα στοιχείο του X δεν είναι 0. Αν όλα είναι 0 επιστρέφει 0.
- **all(X)** η οποία επιστρέφει 1 αν όλα τα στοιχεία του X δεν είναι 0. Αν έστω και ένα είναι 0 επιστρέφει 0.
- **find(X)** η οποία επιστρέφει τους δείκτες των στοιχείων που δεν είναι μηδέν.
- **isempty(X)** η οποία επιστρέφει 1 αν το X είναι κενός πίνακας.
- **isnan(X)** η οποία επιστρέφει 1 για τα στοιχεία του X που είναι NaN και 0 για τα άλλα.
- **ischar(X)** η οποία επιστρέφει 1 αν το X είναι Character.
- **strcmp(A,B)** η οποία επιστρέφει 1 αν οι σειρές χαρακτήρων A & B είναι ίδιοι.
- *κ.ά.*

Παραδείγματα:

```
>> X = eye(3)
X =
     1     0     0
     0     1     0
     0     0     1
>> any(X)
ans =
     1     1     1
>> all(X)
ans =
     0     0     0
>> find(X)
ans =
     1
     5
     9
>> isempty(X)
ans =
     0
>> isnan(X)
ans =
     0     0     0
     0     0     0
     0     0     0
>> ischar(X)
ans =
     0
```

4.3.3. Η Εντολή Ελέγχου IF-ELSE

Η βασική εντολή ελέγχου της ροής ενός προγράμματος είναι η IF. Η εντολή IF έχει διάφορες μορφές σύνταξης. Σε οποιαδήποτε μορφή και αν χρησιμοποιηθεί η IF, πάντα ελέγχει τη συνθήκη που ακολουθεί τη λέξη κλειδί IF. Αν η συνθήκη είναι αληθής (1) τότε εκτελείται το μπλοκ των εντολών που ακολουθεί τη συνθήκη μετά το κόμμα. Αν η συνθήκη είναι ψευδής τότε εκτελείται το μπλοκ των εντολών που ακολουθεί τη λέξη κλειδί ELSE, ή, αν αυτό δεν υπάρχει, η εντολή τελειώνει στην END.

Οι κύριοι τρόποι σύνταξης είναι οι παρακάτω τρεις. Σημειώνουμε ότι, το κόμμα που ακολουθεί τη συνθήκη παίζει το ρόλο του THEN που συναντάμε σε άλλες γλώσσες και δεν πρέπει να παραλείπεται.

1) Η σύνταξη της **IF-END** δεν έχει μπλοκ ELSE και έχει τη μορφή:

```
if  <συνθήκη>,
    <μπλοκ εντολών>
end
```

Στη μορφή αυτή η IF ελέγχει τη συνθήκη και αν είναι αληθής (=1) εκτελεί το μπλοκ των εντολών. Αν είναι ψευδής (=0) η εντολή τελειώνει και το πρόγραμμα συνεχίζει με την εντολή που ακολουθεί την END.

2) Η σύνταξη της **IF-ELSE-END** είναι η πλήρης σύνταξη και έχει τη μορφή:

```
if  <συνθήκη>,
    <μπλοκ εντολών 1>
else
    <μπλοκ εντολών 2>
end
```

Στη μορφή αυτή η IF ελέγχει τη συνθήκη και, αν είναι αληθής (=1) εκτελεί το μπλοκ των εντολών 1, ενώ αν είναι ψευδής (=0) εκτελεί το μπλοκ των εντολών 2 (ELSE). Όποιο μπλοκ εντολών και αν εκτελεστεί, η εντολή IF τελειώνει και το πρόγραμμα συνεχίζει με την εντολή που ακολουθεί την END.

3) Η σύνταξη της **IF-ELSELF-ELSEIF-…-ELSE-END** είναι ο κομψότερος τρόπος γραφής πολλών IF, τη μια μέσα στην άλλη (nested), και έχει τη μορφή:

```
if  <συνθήκη 1>,
    <μπλοκ εντολών 1>
elseif <συνθήκη 2>,
       <μπλοκ εντολών 2>
else
    <μπλοκ εντολών 3>
end
```

Στη μορφή αυτή η IF ελέγχει τη συνθήκη 1 και, αν είναι αληθής (=1) εκτελεί το μπλοκ των εντολών 1, ενώ αν είναι ψευδής (=0) ελέγχει τη συνθήκη 2 και, αν είναι αληθής (=1) εκτελεί το μπλοκ των εντολών 2, ενώ αν είναι ψευδής (=0) εκτελεί το μπλοκ των εντολών 3 (ELSE). Όποιο μπλοκ εντολών και αν εκτελεστεί, η εντολή IF τελειώνει και το πρόγραμμα θα συνεχίσει μετά την END.

Τις εντολές ελέγχου τις γράφουμε πάντα με εσοχές (indentation). Η εσοχή αυξάνει όταν έχουμε τη μια εντολή μέσα στην άλλη. Η γραφή αυτή κάνει τον κώδικα πιο ευανάγνωστο και τον εντοπισμό λαθών πιο εύκολο.
Π.χ.:

```
>> A = 2;  ↵
>> if A<0, disp('A arnitiko'), else disp('A thetiko'), end  ↵
A thetiko

>> A = -3;  ↵
>> if A<0, disp('A arnitiko'), else disp('A thetiko'), end  ↵
A arnitiko

>> A = 0;  ↵
>> if A<0, disp('A arnitiko'), else disp('A thetiko'), end  ↵
A thetiko
```

Στη τελευταία περίπτωση η απάντηση δεν είναι ακριβής. Για να γίνει σωστό το παραπάνω IF πρέπει να χρησιμοποιήσουμε τη δομή **elseif** ώστε να ελέγξουμε και τις τρεις δυνατές περιπτώσεις.

```
>> A = 0;  ↵
>> if A<0,                  ↵
      disp('A arnitiko')    ↵
   elseif A>0               ↵
      disp('A thetiko')     ↵
   else                     ↵
      disp('A = miden')     ↵
   end                      ↵
A = miden
```

4.3.4. Η Εντολή Ελέγχου SWITCH-CASE

Η εντολή ελέγχου SWITCH είναι η δεύτερη εντολή ελέγχου που διαθέτει το MatLab. Η SWITCH είναι μια τροποποιημένη IF η οποία, αντί της λογικής έκφρασης ελέγχει τη τιμή μιας μεταβλητής, και, αντί να έχει μόνο 2 μπλοκ εντολών (1-0, THEN-ELSE) έχει απεριόριστες περιπτώσεις (CASE) με μπλοκ εντολών, όσες είναι και οι αναμενόμενες τιμές της μεταβλητής.

Η σύνταξη της SWITCH έχει τη μορφή:

```
switch <βαθμωτή έκφραση>
   case <τιμή_1>
      <μπλοκ εντολών 1>
   case <τιμή_2>
      <μπλοκ εντολών 2>
   … … …
   case <τιμή_N>
      <μπλοκ εντολών N>
   otherwise
      <μπλοκ εντολών >
end
```

Στις εντολές IF οι λογικές εκφράσεις μπορεί να πάρουν μόνο δυο τιμές 1 ή 0, και καλύπτονται από τα δυο μπλοκ εντολών (then-else). Αντίθετα, στην εντολή SWITCH όσες CASE και να χρησιμοποιήσουμε δεν μπορούμε να καλύψουμε όλες τις πιθανές τιμές της μεταβλητής. Για το λόγο αυτό προσθέτουμε πάντα μετά τις CASE και την εντολή OTHERWISE η οποία καλύπτει όλες τις υπόλοιπες, μη αναμενόμενες, τιμές και εμφανίζει κάποιο προειδοποιητικό μήνυμα.

Για παράδειγμα, η παρακάτω συνάρτηση χρησιμοποιεί μια CASE για να μετατρέπει το βαθμό ενός μαθητή στην αντίστοιχη λεκτική φράση:

M-file: **axiologisi.m**

```
function S = axiologisi(vathmos)
switch floor(vathmos)
    case (0),  S = 'Aporriptetai';
    case (1),  S = 'Aporriptetai';
    case (2),  S = 'Aporriptetai';
    case (3),  S = 'Aporriptetai';
    case (4),  S = 'Aporriptetai';
    case (5),  S = 'Kala';
    case (6),  S = 'Kala';
    case (7),  S = 'Poly Kala';
    case (8),  S = 'Poly Kala';
    case (9),  S = 'Arista';
    case (10), S = 'Arista';
    otherwise, S = 'Error';
end
```

```
>> axiologisi(0)
ans =
Aporriptetai
>> axiologisi(1)
ans =
Aporriptetai
>> axiologisi(4)
ans =
Aporriptetai
>> axiologisi(5)
ans =
Kala
>> axiologisi(7)
ans =
Poly Kala
>> axiologisi(10)
ans =
Arista
>> axiologisi(100)
ans =
Error
>> axiologisi(-1)
ans =
Error
```

4.3.5. Η Εντολή Επανάληψης FOR

Η εντολή επανάληψης FOR είναι η πρώτη από τις δυο εντολές επανάληψης που διαθέτει το MatLab. Με τη FOR πραγματοποιούμε συγκεκριμένο αριθμό επαναλήψεων, μετρημένο εκ των προτέρων, σύμφωνα με τη λίστα του δείκτη της FOR. Η βασική σύνταξη της FOR είναι:

```
for <δείκτης> = <λίστα>,
    <μπλοκ εντολών>
end
```

Συχνά έχουμε περισσότερες FOR εμφωλευμένες (nested) τη μία μέσα στην άλλη με αποτέλεσμα τη γεωμετρική αύξηση των συνολικών επαναλήψεων για το εσωτερικό μπλοκ εντολών:

```
for <δείκτης 1> = <λίστα 1>,
   for <δείκτης 2> = <λίστα 2>,
      for <δείκτης 3> = <λίστα 3>,
          <μπλοκ εντολών>
      end
   end
end
```

Η σύνταξη της λίστας του δείκτη της FOR γίνεται με τον τελεστή (**:**) και τους εξής τρόπους:

1. δείκτης = από : βήμα : έως (π.χ.: η for i = 1:2:9 θα κάνει 5 επαναλήψεις)
2. δείκτης = από : έως (π.χ.: η for i = 1:9 θα κάνει 9 επαναλήψεις)
3. δείκτης = [<διάνυσμα>] (π.χ.: η for i = [1 3 7 9] θα κάνει 4 επαναλήψεις)

Π.χ.:

```
>> loop = 0; for i=1:2:9, loop=loop+1; end, loop
loop =
     5
>> loop = 0; for i=1:9, loop=loop+1; end, loop
loop =
     9
>> loop = 0; for i=[1 3 7 9], loop=loop+1; end, loop
loop =
     4
>> for i=1:10,
      for j=1:10,
         A(i,j) = i/j;
      end
   end
>> A
A =
  Columns 1 through 7
    1.0000   0.5000   0.3333   0.2500   0.2000   0.1667   0.1429
    2.0000   1.0000   0.6667   0.5000   0.4000   0.3333   0.2857
    3.0000   1.5000   1.0000   0.7500   0.6000   0.5000   0.4286
    4.0000   2.0000   1.3333   1.0000   0.8000   0.6667   0.5714
    5.0000   2.5000   1.6667   1.2500   1.0000   0.8333   0.7143
    6.0000   3.0000   2.0000   1.5000   1.2000   1.0000   0.8571
    7.0000   3.5000   2.3333   1.7500   1.4000   1.1667   1.0000
    8.0000   4.0000   2.6667   2.0000   1.6000   1.3333   1.1429
    9.0000   4.5000   3.0000   2.2500   1.8000   1.5000   1.2857
   10.0000   5.0000   3.3333   2.5000   2.0000   1.6667   1.4286
  Columns 8 through 10
    0.1250   0.1111   0.1000
    0.2500   0.2222   0.2000
    0.3750   0.3333   0.3000
    0.5000   0.4444   0.4000
    0.6250   0.5556   0.5000
    0.7500   0.6667   0.6000
    0.8750   0.7778   0.7000
    1.0000   0.8889   0.8000
    1.1250   1.0000   0.9000
    1.2500   1.1111   1.0000
```

Η αργή ταχύτητα εκτέλεσης της FOR αποτελεί ένα μειονέκτημα που μας οδηγεί στο να μην τη χρησιμοποιούμε όταν έχουμε άλλη καλύτερη εναλλακτική τεχνική. Τέτοιες τεχνικές είναι η χρήση πράξεων με μητρώα με διαστάσεις ανάλογες των τιμών του δείκτη της FOR, οι οποίες εκτελούνται σε ένα κλάσμα του χρόνου που θα απαιτούσε η αντίστοιχη FOR.
Στο επόμενο παράδειγμα βλέπουμε τη σύγκριση της ταχύτητας ενός βρόγχου FOR με ένα διανυσματικό βρόγχο, για τον ορισμό τιμών σε ένα διάνυσμα 900,000 στοιχείων. Ο χρόνος μετριέται με τη βοήθεια της clock. Βλέπουμε ότι ο βρόγχος FOR χρειάζεται 50-πλάσιο χρόνο για το ίδιο αποτέλεσμα.

```
>> c1 = clock; A = 1:900000; vect = clock-c1
vect =
         0         0         0         0         0    0.0200

>> c1 = clock; for i = 1:900000, A(i) = i; end, loop = clock-c1
loop =
         0         0         0         0         0    1.0620
```

4.3.6. Η Εντολή Επανάληψης WHILE

Η εντολή WHILE είναι η δεύτερη από τις δυο εντολές επανάληψης που διαθέτει το MatLab. Η WHILE δεν πραγματοποιεί γνωστό εκ των προτέρων αριθμό επαναλήψεων, αλλά τερματίζει τις επαναλήψεις όταν εκπληρωθεί η συνθήκη της WHILE.

Η ιδιότητα αυτή της WHILE την κάνει ευάλωτη, είτε σε σφάλματα λογικής μέσα στη συνθήκη, ή, σε αριθμητικά σφάλματα ακριβείας, και έτσι μπορεί εύκολα να μπεί σε άπειρο βρόγχο χωρίς να μπορεί να τερματίσει. Για το λόγο αυτό η WHILE πρέπει να σχεδιάζεται προσεκτικά και να περιέχει πάντα δικλείδες ασφαλείας όπως είναι: ο μέγιστος επιτρεπόμενος αριθμός επαναλήψεων, η ελάχιστη διαφορά του αποτελέσματος μεταξύ επαναλήψεων, κ.ά., σχετικά με το εκάστοτε πρόβλημα όρια.

Η σύνταξη της WHILE είναι:

```
while <συνθήκη>,
    <μπλοκ εντολών>
end
```

Η WHILE μπορεί να αντικαταστήσει μια FOR αν της προσθέσουμε, με δύο (2) επιπλέον εντολές, τη λειτουργία του δείκτη της FOR. Για παράδειγμα, οι παρακάτω εντολές FOR και WHILE είναι ισοδύναμες:

```
for i = 1:2:9,
   <μπλοκ εντολών>
end
```

```
i = 1
while i <= 9,
   <μπλοκ εντολών>
   i = i + 2
end
```

Το επόμενο παράδειγμα χρησιμοποιεί την WHILE για να βρεί (μετά από 53 επαναλήψεις) την τιμή του **eps**, του μικρότερου αριθμού που μπορεί διακρίνει το MatLab:

M-file: **whiletest.m**

```
e = 1;
k = 0;
while (1+e)>1,
    e = e/2;
    k = k+1;
end
e_min = e*2
loops = k
```

```
>> eps  ↵
eps =
  2.2204e-016


>> whiletest  ↵
e_min =
  2.2204e-016
loops =
    53
```

4.3.7. Οι Εντολές CONTINUE & BREAK

Και οι δύο εντολές σταματούν την εκτέλεση του τρέχοντος βρόγχου και μεταφέρουν τον έλεγχο έξω από αυτόν. Η διαφορά τους είναι ότι, αν υπάρχουν πολλαπλοί βρόγχοι ο ένας μέσα στον άλλο, η **BREAK** τερματίζει όλους τους βρόγχους και συνεχίζει με την επόμενη εντολή, ενώ η **CONTINUE** τερματίζει μόνο τον τρέχοντα βρόγχο και συνεχίζει την εκτέλεση στον αμέσως πιο έξω βρόγχο που τον περιβάλλει.