ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΪΚΑ
ΜΑΘΗΜΑΤΑ

Πληροφορική

Ενότητα 4: Α. Λογικές εκφράσεις (Παραστάσεις και Δείκτες).
Β. Δομές Προγραμματισμού

Κωνσταντίνος Καρατζάς
Τμήμα Μηχανολόγων Μηχανικών

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Δ.#7Α: ΠΛΗΡΟΦΟΡΙΚΗ
ΛΟΓΙΚΕΣ ΕΚΦΡΑΣΕΙΣ (ΠΑΡΑΣΤΑΣΕΙΣ & ΔΕΙΚΤΕΣ)

17 Μαρτίου 2015

Κώστας Καρατζάς
Τμήμα Μηχανολόγων Μηχανικών, ΑΠΘ

Λογικές εκφράσεις

Οι λογικές εκφράσεις είναι αναπόσπαστο μέρος των δομών ελέγχου.

Για την κατασκευή τους απαιτούνται

- **Τελεστές** (σχεσιακοί και λογικοί) και
- **Τελεστέοι** (μεταβλητές, σταθερές, κλπ)

Σχεσιακοί Τελεστές

Τελεστής	Συμβολσμός
Ίσο	==
Μικρότερο από	<
Μικρότερο ή ίσο από	<=
Μεγαλύτερο από	>
Μεγαλύτερο ή ίσο από	>=
Όχι ίσο	~=

Σχεσιακοί τελεστές σε «εκφράσεις»

Η χρήση των τελεστών οδηγεί πάντα σε έναν έλεγχο που έχει δύο δυνατά αποτελέσματα:

- **Αληθής**
- **Ψευδής**

Παράδειγμα

```
>> a=4; b=5;
>> a>=b
ans ??
```

Παράδειγμα

```
>> a=4; b=5;
>> a~=b
ans =
     ??
```

Παράδειγμα

```
>> x='mitsos'
>> x=='mitsos'
ans =
     1     1     1     1     1     1
```

Παράδειγμα

```
>> a=1:4;
>> b=4:-1:1;
>> a>b          % Τι κάνω εδώ;
ans =
     ??
```

Λογικοί Τελεστές

Τελεστής	Συνάρτηση	Συμβολισμός
Λογική σύζευξη	and(var1, var2)	&
Λογική διάζευξη	or(var1, var2)	|
Λογική σύζευξη ("βραχυκυκλωμένη") *		&& Π.χ. A=0, B=1 A&&B= f
Λογική διάζευξη ("βραχυκυκλωμένη")		||
Λογική άρνηση	not(var1)	~
Λογικό αποκλειστικό	xor(var1)	xor

* http://www.mathworks.com/help/matlab/ref/logicaloperatorsshortcircuit.html

Λογικοί Τελεστές (συνήθως χρησιμοποιούμενοι)

Τελεστής	Συνάρτηση	Συμβολισμός
Λογική σύζευξη	and(var1, var2)	&
Λογική διάζευξη	or(var1, var2)	|
Λογική σύζευξη ("βραχυκυκλωμένη")		&&
Λογική διάζευξη ("βραχυκυκλωμένη")		||
Λογική άρνηση	not	~

Πίνακας «αληθείας»

Inputs		and	or	xor	not
A	B	A & B and(A,B)	A | B or(A,B)	xor(A,B)	~A !A
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Πίνακας «αληθείας»

Inputs		and	or	xor	not
A	B	A & B and(A,B)	A | B or(A,B)	xor(A,B)	~A !A
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Λογικοί τελεστές

- Σύγκριση στοιχείο προς στοιχείο (οι τελεστέοι αποτελούν στοιχεία πινάκων)
  - **and**, **or (&, |)**
  - Κάθε στοιχείο κρίνεται ως προς το εάν είναι
    - Μηδέν -> **false**
    - Όχι μηδέν -> **true**
    - **Το αποτέλεσμα είναι ένας πίνακας λογικών τιμών (0,1) ίδιων διαστάσεων με τα συγκρινόμενα διανύσματα - πίνακες**
- Σύγκριση δύο λογικών τιμών (οι τελεστέοι αποτελούν «βαθμωτά» μεγέθη)
  - **βραχυκυκλωμένο and**, **or (&&, ||)**
  - Κάθε τιμή κρίνεται ως προς το εάν είναι
    - Μηδέν -> **false**
    - Όχι μηδέν -> **true**
    - **Το αποτέλεσμα είναι μία λογική τιμή (0,1)**

Λογικοί τελεστές

- Σύγκριση στοιχείο προς στοιχείο (οι τελεστέοι αποτελούν στοιχεία πινάκων)
  - **and, or (&, |)**
  - Κάθε στοιχείο κρίνεται ως προς το εάν είναι
    - Μηδέν -> **false**
    - Όχι μηδέν -> **true**
    - **Το αποτέλεσμα είναι ένας πίνακας λογικών τιμών (0,1) ίδιων διαστάσεων με τα συγκρινόμενα διανύσματα - πίνακες**
- Σύγκριση δύο λογικών τιμών (οι τελεστέοι αποτελούν «βαθμωτά» μεγέθη)
  - **βραχυκυκλωμένο and, or (&&, ||)**
  - Κάθε τιμή κρίνεται ως προς το εάν είναι
    - Μηδέν -> **false**
    - Όχι μηδέν -> **true**
    - **Το αποτέλεσμα είναι μία λογική τιμή (0,1)**

Παραδείγματα

```
>> a=1;
>> b=2;
>> a<0 | b>3
ans =
      ?
```

Παραδείγματα

```
>> a=1;
>> b=2;
>> a<0 | b>3
ans =
     0
```

Παραδείγματα

Έστω δυνάμεις F με συνιστώσες Fx και Fy:

```
>> Fx= [ -1   0   1   0   5]
>> Fy= [ pi   0   0   0   1]
>> Fx & Fy
ans =
      ????
```

Παραδείγματα

Έστω δυνάμεις F με συνιστώσες Fx και Fy:

```
>> Fx= [ -1   0   1   0   5]
>> Fy= [ pi   0   0   0   1]
>> Fx & Fy
ans =
            1   0   0   0   1
```

Παραδείγματα

```
>> a=-5:5;
>> b=5:-1:-5;
>> a>zeros(1,11)  & b<3*ones(1,11)
ans =
      ????
```

Παραδείγματα

```
>> a=-5:5;
>> b=5:-1:-5;
>> a>zeros(1,11) & b<3*ones(1,11)
ans =

     0     0     0     0     0     0     1     1     1     1     1
```

Παραδείγματα

```
B = [0    1    2    0]
>> ~B
ans = ????
        1    0      0    1

>> ~~B
ans = ????
        0    1      1    0

>> B~=0
ans = ????
        0    1      1    0
```

Παραδείγματα

```
A = [0   1   2   3]
B = [0   1   2   0]

>> xor(A,B)
ans = ????

   0   0   0   1
```

Παραδείγματα

```
A = [0   1   2   3]
B = [0   1   2   0]

>> or(A,B)
>> A|B
ans = ????

   0   1   1   1
```

Παραδείγματα

```
A = [0   1   2   3]
B = [0   1   2   0]

>> and(A,B)
>> A&B
ans = ????

   0   1   1   0
```

Λογικοί τελεστές

- Σύγκριση στοιχείο προς στοιχείο (οι τελεστέοι αποτελούν στοιχεία πινάκων)
  - **and, or (&, |)**
  - Κάθε στοιχείο κρίνεται ως προς το εάν είναι
    - Μηδέν -> **false**
    - Όχι μηδέν -> **true**
    - **Το αποτέλεσμα είναι ένας πίνακας λογικών τιμών (0,1) ίδιων διαστάσεων με τα συγκρινόμενα διανύσματα - πίνακες**
- Σύγκριση δύο λογικών τιμών (οι τελεστέοι αποτελούν «βαθμωτά» μεγέθη)
  - **βραχυκυκλωμένο and, or (&&, ||)**
  - Κάθε τιμή κρίνεται ως προς το εάν είναι
    - Μηδέν -> **false**
    - Όχι μηδέν -> **true**
    - **Το αποτέλεσμα είναι μία λογική τιμή (0,1)**

Βραχυκυκλωμένοι λογ τελεστές

- Το 2ο μέρος λαμβάνεται υπόψη μόνο εάν το αποτέλεσμα δεν καθορίζεται πλήρως από το πρώτο μέρος, π.χ.
  - **A && B**: εάν το A είναι false, τότε το συνολικό αποτέλεσμα θα είναι σίγουρα false, ανεξάρτητα από τη λογική τιμή του B
  - **A&B**: η σύγκριση λαμβάνει υπόψη τις τιμές και του A αλλά και του B

Παραδείγματα

```
a = 1
b = 7

>> x = (a ~= 0) && (a/b > 18.5)
ans = ????

    0
```

Έστω ότι θέλω να υπολογίσω το x=-b-diakr/2*b μόνο στην περίπτωση που ο παρονομαστής δεν είναι μηδέν και diakr>0

Λογικές εκφράσεις

Εκτός των γνωστών τελεστών, το Matlab διαθέτει μία σειρά από **λογικές συναρτήσεις**, οι οποίες μπορούν να χρησιμοποιηθούν άμεσα σε διανύσματα, ***μειώνοντας έτσι σημαντικά τον κώδικα και επιταχύνοντας την ανάπτυξη ενός προγράμματος.***

Επιπρόσθετοι λογικοί τελεστές & σχετικές εντολές

`all`	Είναι όλα τα στοιχεία ενός πίνακα μη μηδενικά ή true?
`any`	Είναι κάποια από τα στοιχεία ενός πίνακα μη μηδενικά ή true?
`false`	Logical 0 (false)
`find`	Βρες θέσεις και τιμές μη μηδενικών στοιχείων
`islogical`	Είναι λογικός πίνακας?
`logical`	Μετατροπή αριθμητικών τιμών σε λογικές
`true`	Logical 1 (true)

Η εντολή find

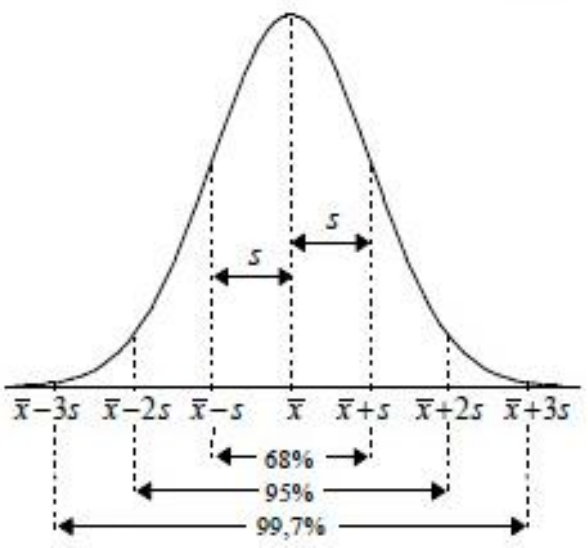
Μία ιδιαίτερα χρήσιμη εντολή του Matlab είναι η εντολή find, η οποία μας δίνει τη δυνατότητα να βρούμε στοιχεία ενός πίνακα που πληρούν μία λογική συνθήκη. Η σύνταξή της είναι:

find(λογική έκφραση)

και επιστρέφει μία λίστα από **θέσεις** του πίνακα που πληρούν την λογική έκφραση.

Παράδειγμα χρήσης της find

Έστω 100 πραγματικοί αριθμοί που προέκυψαν από μετρήσεις κατανάλωσης ηλεκτρικής ενέργειας. Με ενδιαφέρει να βρω τις καταναλώσεις που είναι μεαλύτερες κατά τουλάχιστον μία τυπική απόκλιση από τη μέση τιμή



Παράδειγμα χρήσης της find

Έστω 100 τυχαίοι αριθμοί σε ένα διάνυσμα. Σε ποιές θέσεις του βρίσκονται όσοι είναι > μ+σ ?

```
>> x=randn(1,100);
```

Και τώρα τι κάνω?

(daydreaming…..)

Έχω συναρτήσεις για m και σ???

mean(x), std(x)!!!!

Παράδειγμα χρήσης της find

Έστω 100 τυχαίοι αριθμοί σε ένα διάνυσμα. Σε ποιές θέσεις του βρίσκονται όσοι είναι > μ+σ ?

```
>> x=randn(1,100);
```

Και τώρα τι κάνω?

```
(daydreaming…..)
```

```
>> find(όσα x είναι > μ+σ))
>> find(x>mean(x)+std(x))
```

Παράδειγμα χρήσης της find

Και ποια είναι αυτά τα στοιχεία;

```
>> find(x>mean(x)+std(x))
```

Χρησιμοποιώ τους δείκτες θέσης στοιχείων!

```
x(δείκτες θέσης στοιχείων) %δηλαδή..

x(find(x>mean(x)+std(x)))
```

Λογικές Συναρτήσεις

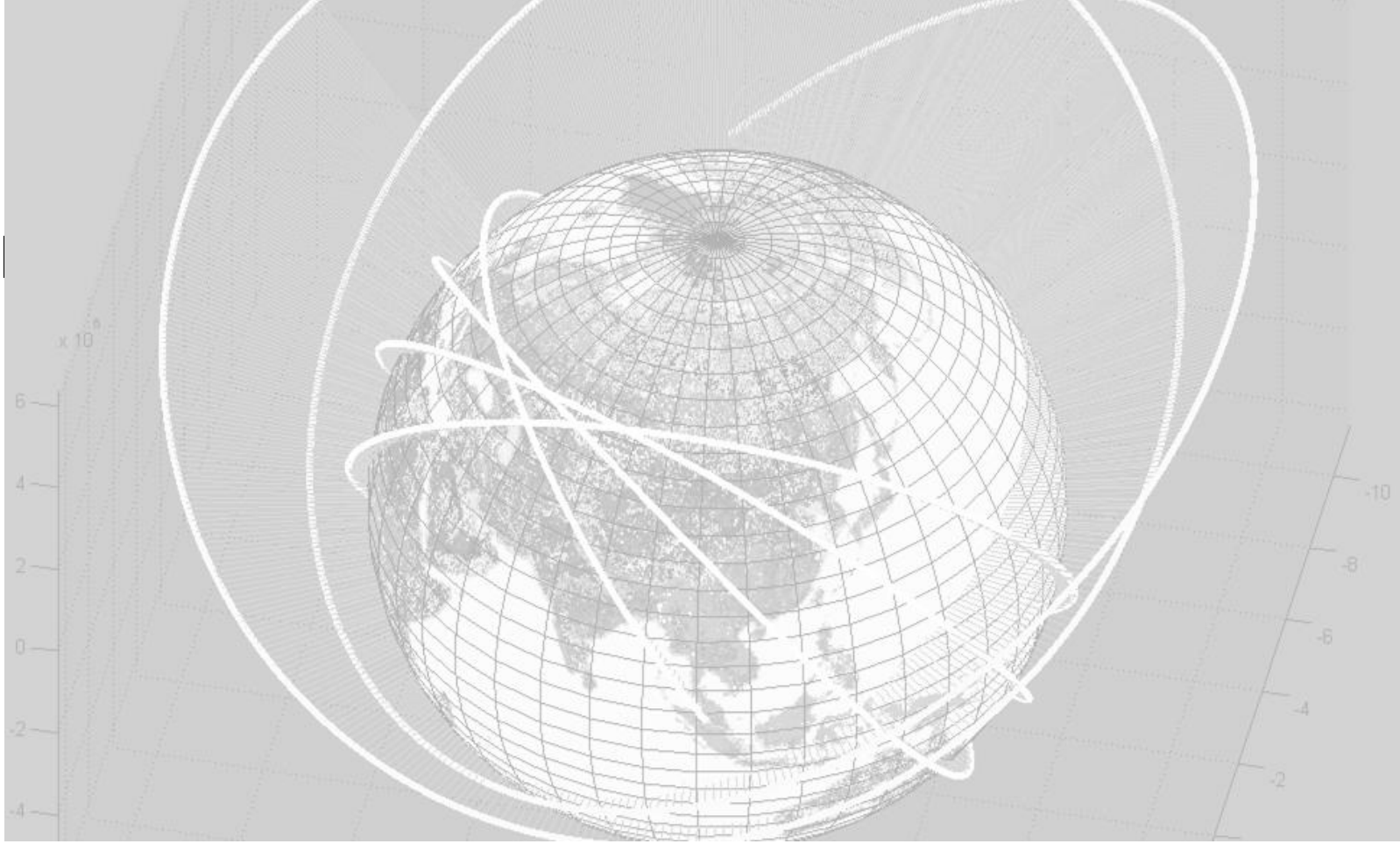
ischar	Έλεγξε αν είναι αλφαριθμητικό (string)
isempty	Έλεγξε αν πίνακας είναι άδειος
isequal	Έλεγξε αν οι πίνακες είναι ίσοι
isfinite	Βρες τα πεπερασμένα στοιχεία πίνακα
isinf	Βρες τα άπειρα στοιχεία πίνακα
isinteger	Έλεγξε αν ο πίνακας είναι ακέραιος
iskeyword	Βρες αν το όνομα είναι λέξη-κλειδί της MATLAB
isletter	Βρες τα στοιχεία που είναι αλφαβητικά γράμματα
islogical	Έλεγξε αν ο πίνακας είναι λογικός
isnan	Βρες τα στοιχεία πίνακα που είναι NaN
isreal	Έλεγξε αν ο πίνακας είναι πραγματικός
isscalar	Έλεγξε αν ο πίνακας είναι βαθμωτός
issorted	Έλεγξε αν το διάνυσμα είναι ταξινομημένο
isvarname	Έλεγξε αν το όνομα μεταβλητής είναι αποδεκτό
isvector	Έλεγξε αν είναι διάνυσμα

Παραδείγματα

```
>> a=[];
>> isempty(a)
ans =
     ?

>> a=[1 NaN; 3 4];
>> isnan(a)
ans =
     ??


>> a=[1 Inf 3 4];
>> isinf(a)
ans =
     ??
```

Συνοψίζοντας

- Σχεσιακοί τελεστές: συγκρίνουν δύο ποσότητες. Το αποτέλεσμα είναι true/false
- Λογικοί τελεστές:ελέγχουν μία παράσταση. Το αποτέλεσμα είναι true/false
  - Σύγκριση στοιχείο προς στοιχείο
  - Σύγκριση λογικών τιμών (βραχυκυκλ.)
- Λογικές συναρτήσεις: **`είναι_κάτι, βρές, κλπ`**



Κάθε τέλος είναι μία καινούργια αρχή!

Κώστας Καρατζάς

Τμήμα Μηχανολόγων Μηχανικών, ΑΠΘ

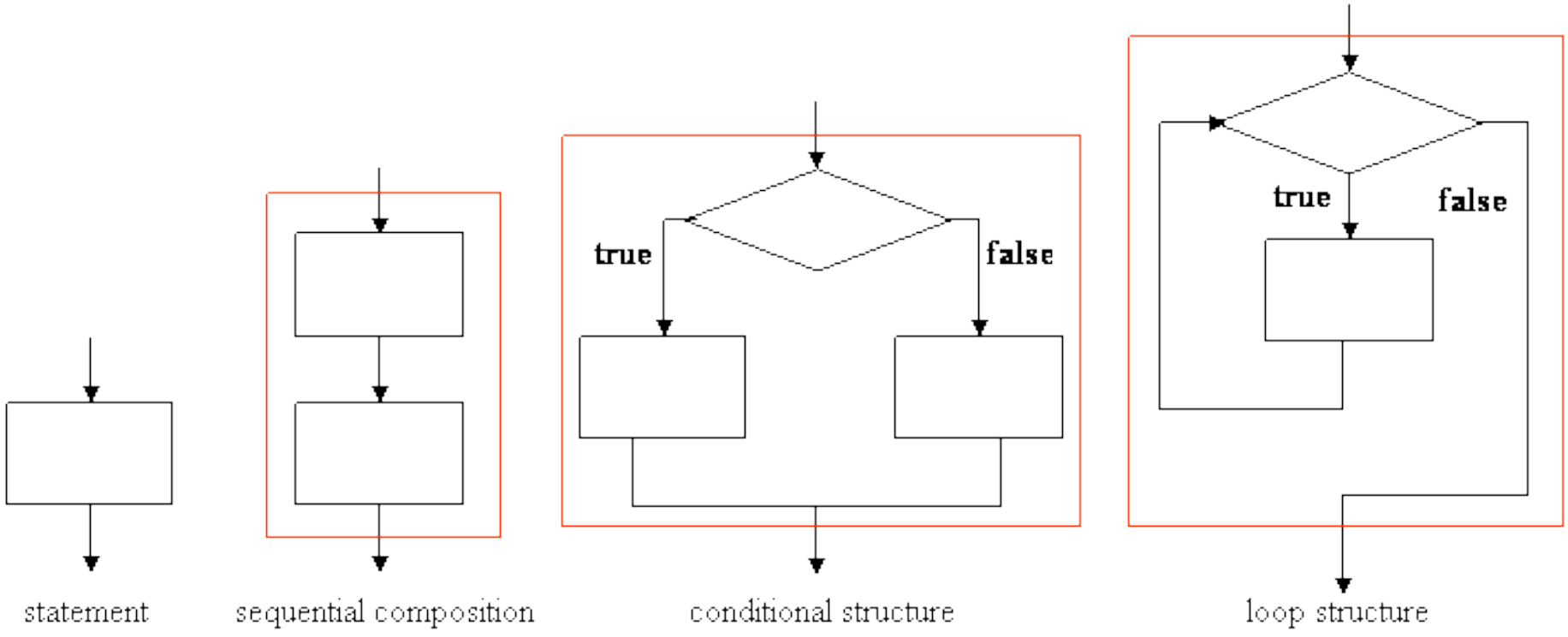
Δ.#7Β: ΠΛΗΡΟΦΟΡΙΚΗ
ΔΟΜΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

17 Μαρτίου 2015

Κώστας Καρατζάς
Τμήμα Μηχανολόγων Μηχανικών, ΑΠΘ

Δομές Προγραμματισμού

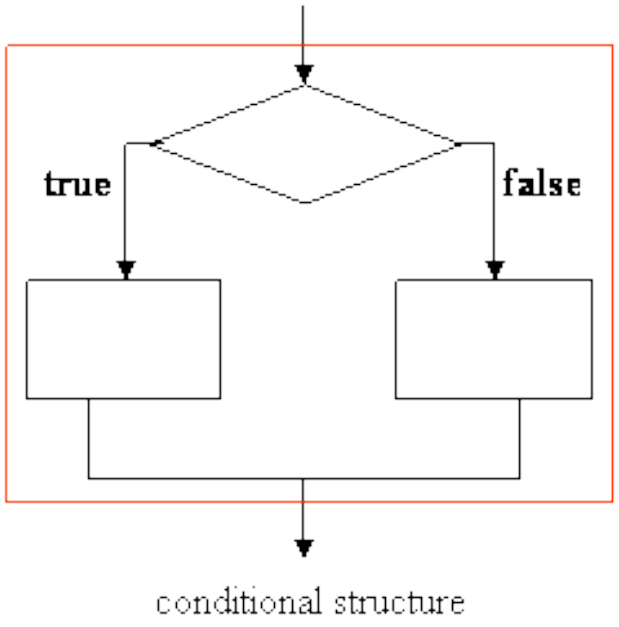
Όπως σε όλες τις γλώσσες προγραμματισμού, έτσι και στο Matlab υπάρχουν οι βασικές δομές προγραμματισμού (ελέγχου και επανάληψης).

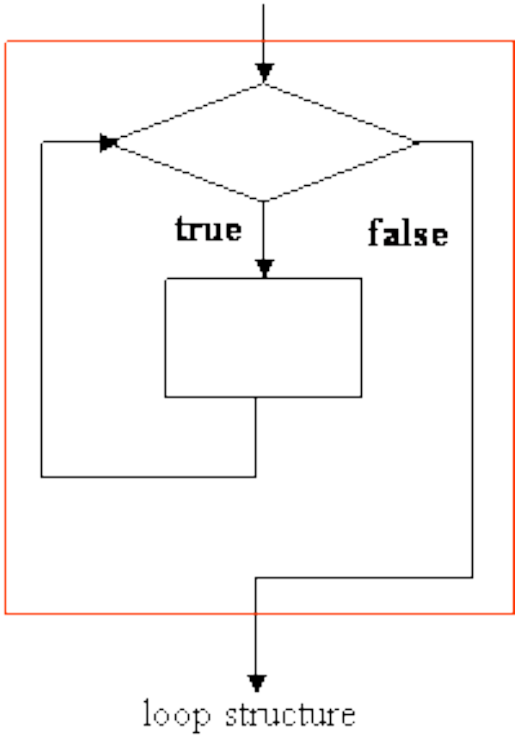


Δομές Προγραμματισμού

Δομή ελέγχου: **if** και **select case**

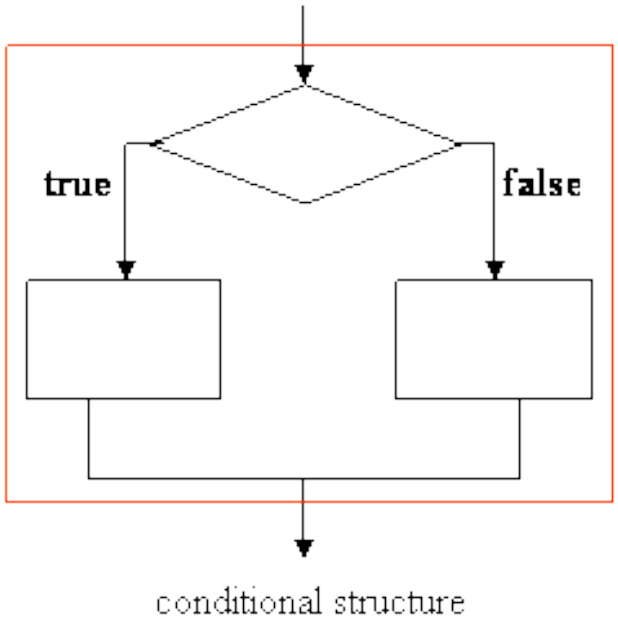
Δομή επανάληψης: **for** και **while**



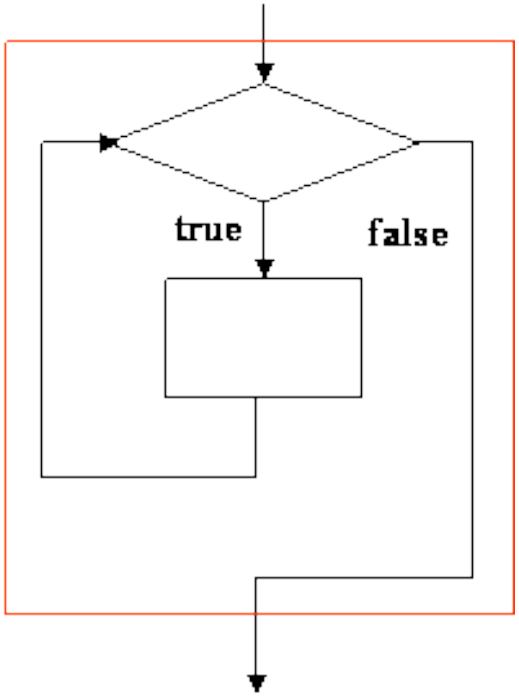


Δομές Προγραμματισμού

Επιπλέον, το Matlab παρέχει μία σειρά από εντολές για εύκολους ελέγχους χωρίς να απαιτείται η χρήση βρόχων και έλεγχος για κάθε στοιχείο.



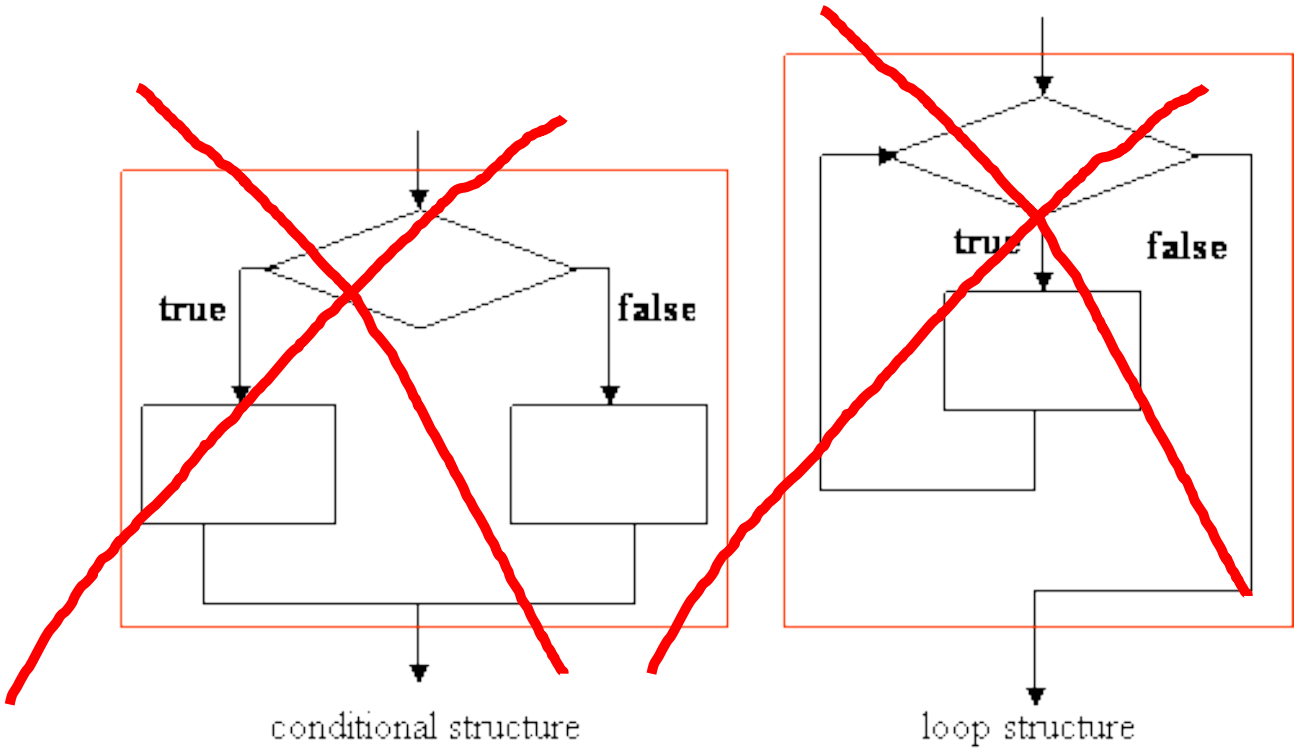
conditional structure

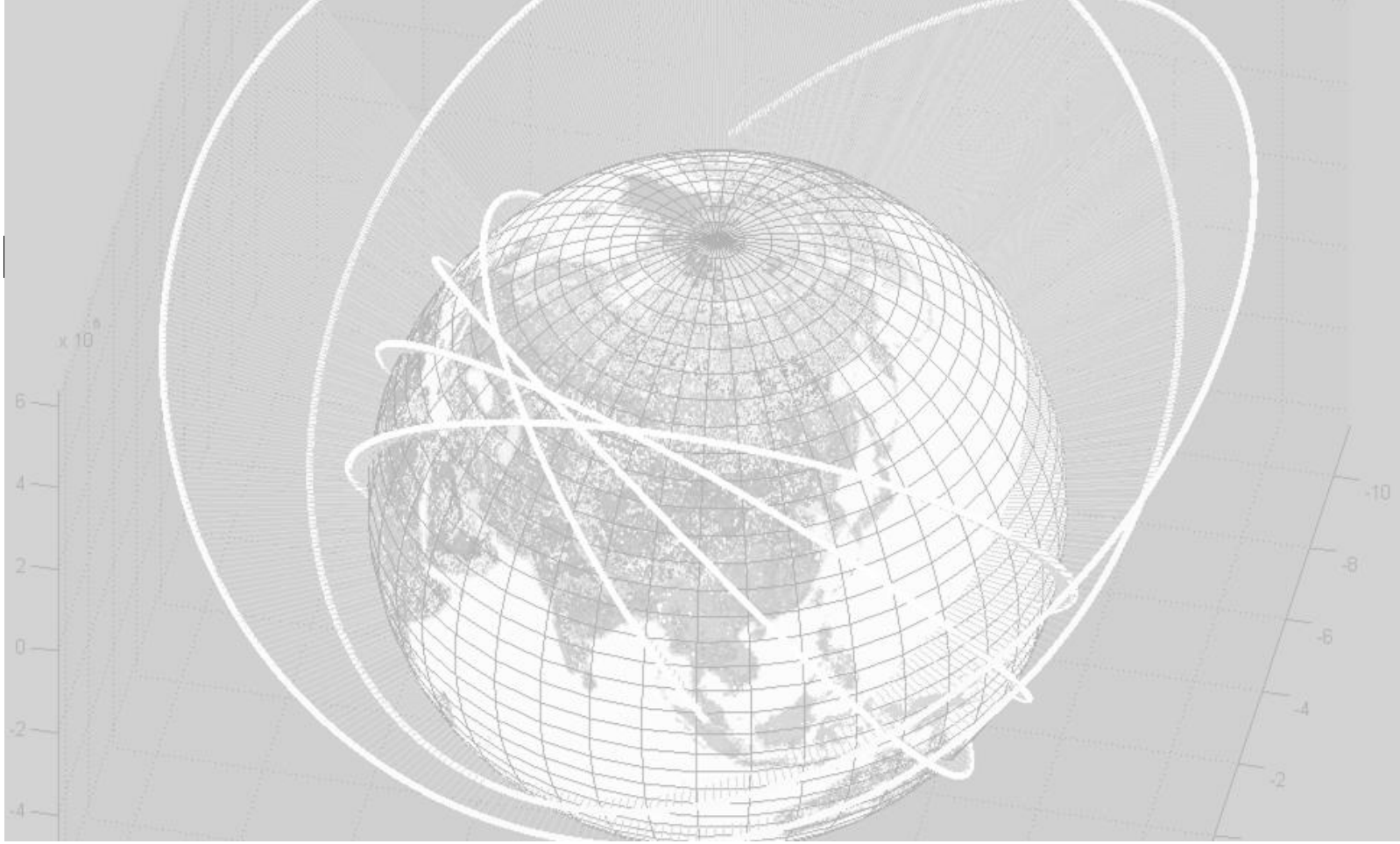


loop structure

Δομές Προγραμματισμού

Επιπλέον, το Matlab παρέχει μία σειρά από εντολές για εύκολους ελέγχους **χωρίς να απαιτείται η χρήση βρόχων και έλεγχος για κάθε στοιχείο**.





Δομές Ελέγχου

Δομές Ελέγχου

Η βασική εντολή ελέγχου είναι η **`if...else... end`**

Μπορούμε να έχουμε εμφωλευμένες δομές ελέγχου.

```
if (συνθήκη 1)
   …
   ομάδα εντολών 1
   …
else        …
   ομάδα εντολών 2
   …
end
```

Δομές Πολλαπλού Ελέγχου

Η βασική εντολή ελέγχου είναι η if..else.. end. Μπορούμε να έχουμε εμφωλευμένες δομές ελέγχου.

```
if (συνθήκη 1)
      ομάδα εντολών 1
elseif (συνθήκη 2)
      ομάδα εντολών 2
elseif (συνθήκη 3)
      ομάδα εντολών 3
else
      ομάδα εντολών 4
end
```

Δομές Πολλαπλού Ελέγχου

```
if (συνθήκη 1)
   ομάδα εντολών 1
else
      if (συνθήκη 2)
      ομάδα εντολών 2
      else
            if (συνθήκη 3)
              ομάδα εντολών 3
            else
              ομάδα εντολών 4
            end
      end
end
```

```
if (συνθήκη 1)
      ομάδα εντολών 1

elseif (συνθήκη 2)
      ομάδα εντολών 2

elseif (συνθήκη 3)
      ομάδα εντολών 3
else
      ομάδα εντολών 4


end
```

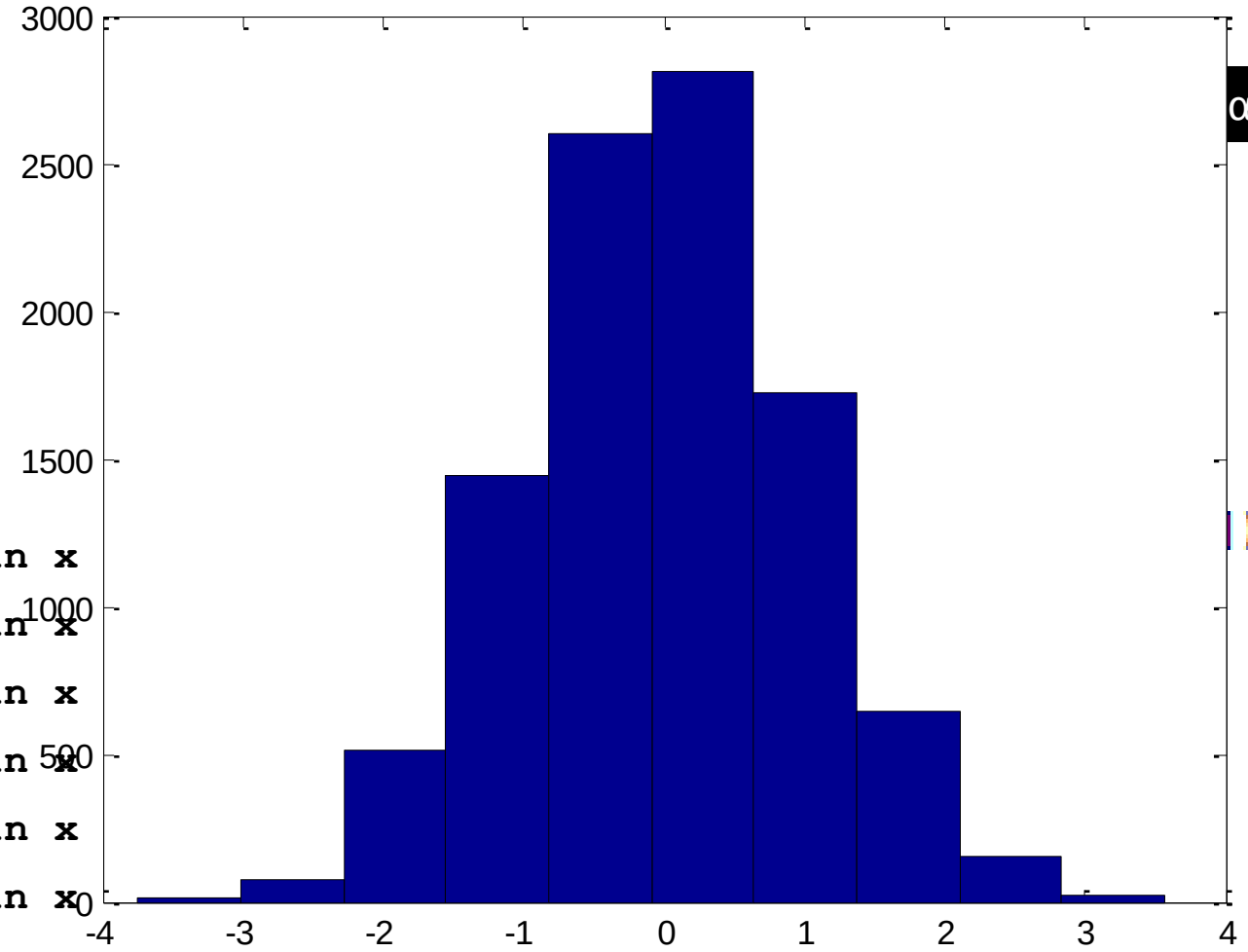
Παράδειγμα δομής ελέγχου

```
% count ones in x
x = round(randn(1,10));
ids = find(x==1);

if (isempty(ids))
    disp('There are no ones in x')
else
    disp(['There are ',num2str(length(ids)),' in x'])
end
```

% round: στον εγγύτερο ακέραιο

```
There are 3 in x
There are 2 in x
There are 2 in x
There are 4 in x
There are 2 in x
There are 2 in x
There are 1 in x
There are no ones in x
```

Παράδειγμα δομής ελέγχου



αιο

```
There are 3 in x
There are 2 in x
There are 2 in x
There are 4 in x
There are 2 in x
There are 2 in x
There are 1 in x
There are no ones in x
```

Η δομή switch

Χρησιμοποιείται όταν έχουμε έναν προκαθορισμένο αριθμό περιπτώσεων.
Η σύνταξη είναι:

switch case *έκφραση*

    case *περίπτωση 1*

        ομάδα εντολών 1

    case *περίπτωση 2*

        ομάδα εντολών 2

    …

    otherwise

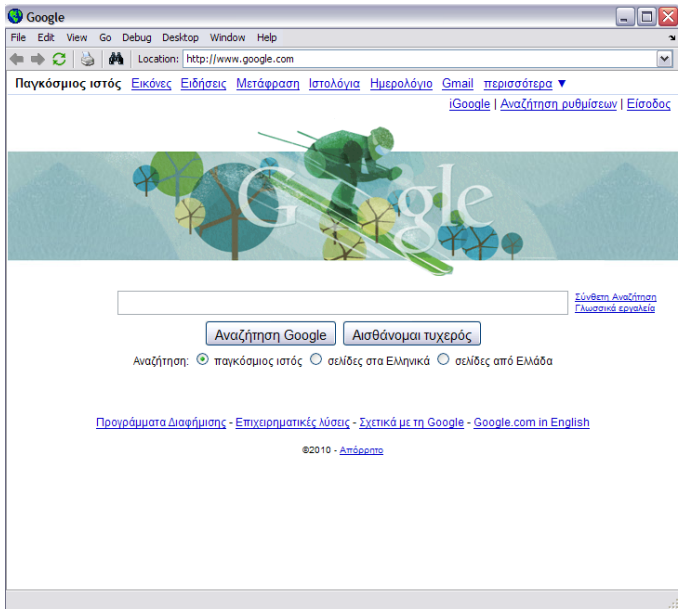
        ομάδα εντολών N

end

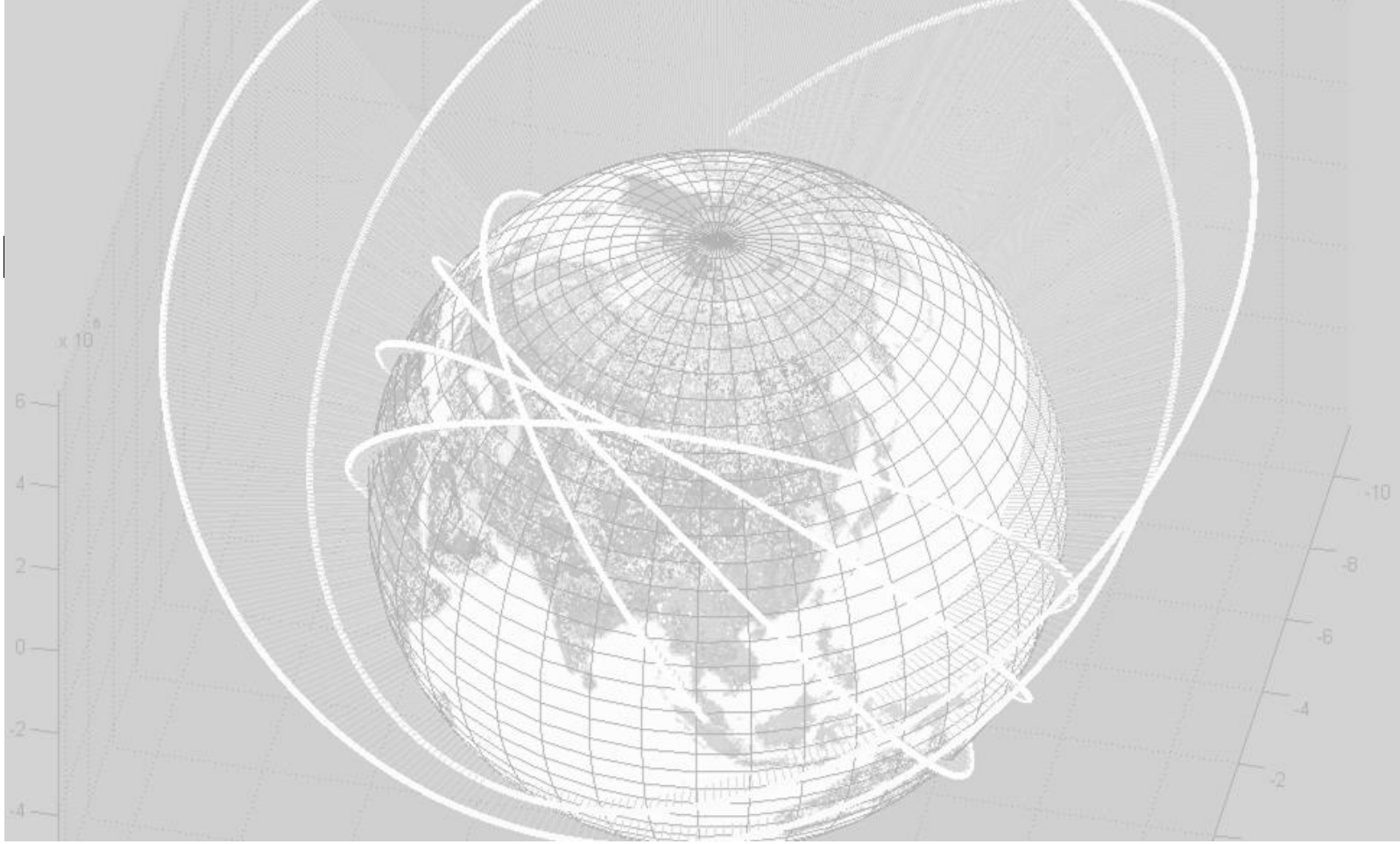
Παράδειγμα χρήσης της εντολής switch

```
site=input('Please, enter site:','s'); % site is a string ('s')

% open the URL in Matlab browser
switch site
    case 'google'
        web http://www.google.com

    case 'yahoo'
        web http://www.yahoo.com

    case 'eclass'
        web https://eclass.meng.auth.gr

    otherwise
        disp('No such URL!')
end
```

web: εντολή που λείπει στο Octave



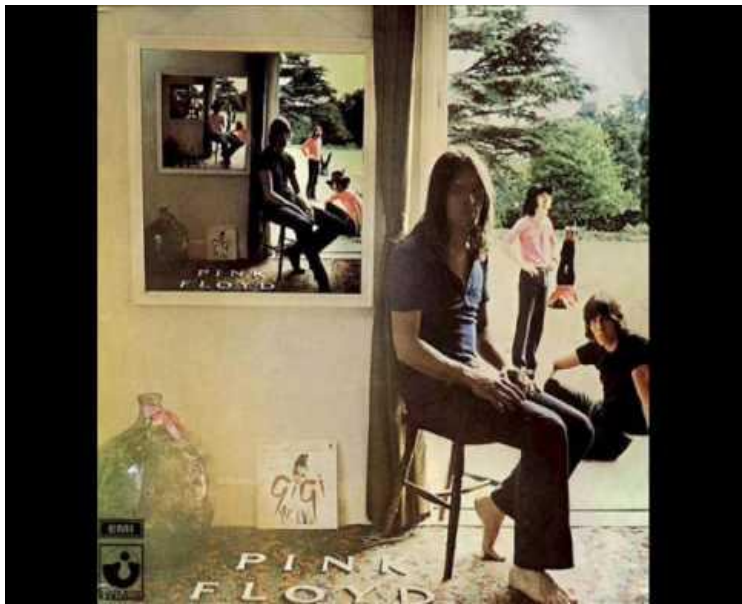

```
>> visit_site
Please, enter site:google
```



Δομές Επανάληψης

Και τι γίνεται όταν θέλουμε επαναλήψεις?

- Τι κοινό έχουν:
  - Ένας σκύλος που κυνηγά την ουρά του
  - Ένας μηχανισμός που κινεί προϊόντα σε αυτόματο πωλητή αναψυκτικών
  - Το σύστημα ελέγχου πρόσφυσης σε αυτοκίνητο
  - Κλπ
  - Επαναλήψεις!



Πρόβλημα: Ακολουθία Fibonacci

1 1 2 3 5 8 13 21 34, …

- F1 = 1
- F2 = 1
- F (n) = F(n-1)+F(n-2)

Τι συμβαίνει όταν το n δεν είναι γνωστό εκ των προτέρων, αλλά εισάγεται κατά την εκτέλεση του προγράμματος;

Δομές Επανάληψης

Μία διαδικασία που επαναλαμβάνεται αρκετές φορές έως ότου ο λόγος για τον οποίο γίνεται η επανάληψη αυτή εκλείψει, περιγράφεται με ....

....

....

μία **δομή επανάληψης!!!**

Δομές Επανάληψης

Οι βασικές δομές επανάληψης στο Matlab είναι οι **for** και **while**. Η for χρησιμοποιείται όταν είναι γνωστός εκ των προτέρων ο αριθμός των επαναλήψεων, ενώ η while όταν δεν είναι γνωστός (χρήση συνθήκης). Η σύνταξη των εντολών είναι:

```
for   μετρητής=αρχή : βήμα : τέλος
          …
          εντολές
          …
end
```

Εκτελείται όσες φορές καθορίζει ο μετρητής.

```
while   συνθήκη
          …
          εντολές
          …
end
```

Εκτελείται όσο η συνθήκη είναι αληθής

Για τις εντολές for και while πρώτα γίνεται ο έλεγχος μετά η εκτέλεση!

Παράδειγμα δομής επανάληψης με **for**

```
for i=1:2:10
    disp(i^2)
end
```

```
     1
     9
    25
    49
    81
```

```
for i=[2 4 1 6]
  disp(i^2)
end
```

```
     4
    16
     1
    36
```

Δομές Επανάληψης

Η βασικές δομές επανάληψης στο Matlab είναι οι for και **while**. Η for χρησιμοποιείται όταν είναι γνωστός εκ των προτέρων ο αριθμός των επαναλήψεων, ενώ η while όταν δεν είναι γνωστός (χρήση συνθήκης). Η σύνταξη των εντολών είναι:

for *μετρητής=αρχή : βήμα : τέλος*

…

εντολές

…

end

Εκτελείται όσες φορές καθορίζει ο μετρητής.

while *συνθήκη*

…

εντολές

…

end

Εκτελείται όσο η συνθήκη είναι αληθής

Παράδειγμα χρήσης της εντολής **while**

```
dt = 0;
counter=0;
tic
t0 = cputime; % έναρξη χρονομέτρησης

while (dt<5)  % 5 second
    counter = counter + 1;
    dt = cputime - t0;
end
toc

-> counter: 6900767
```

Ποια η τιμή του μετρητή

- Στο for
- Στο while

Με τη βοήθεια παραδείγματος..

Fibonacci:

- F1 = 1
- F2 = 1
- F (n) = F(n-1)+F(n-2)

```
function apotelesma = fiboko(n)
%upologismos Fibonacci
apotelesma=zeros(1,n); %giati???

apotelesma(1)=1;
apotelesma(2)=1;

for i=3:n
   apotelesma(i)=apotelesma(i-1)+apotelesma(i-2);
   fprintf('timi deikti i: %.0f \n', i)
end

   fprintf('TELIKI timi deikti i: %.0f \n', i)

end
```

```
fiboko(20)


timi deikti i: 20
TELIKI timi deikti i: 20
```

```
function apotelesma = fiboko(n)
%upologismos FIbonacci
apotelesma=zeros(1,n); %giati???

apotelesma(1)=1;
apotelesma(2)=1;

i=3
while i<=n
    apotelesma(i)=apotelesma(i-1)+apotelesma(i-2);
    fprintf('timi deikti i: %.0f \n', i)
    i=i+1;
end

fprintf('TELIKI timi deikti i: %.0f \n', i)

end
```

```
fiboko(20)


timi deikti i: 20
TELIKI timi deikti i: 21
```

Why is this happening?

- For: εκτελείται έως ότου ο μετρητής φτάσει στη μέγιστη (καθορισμένη από το χρήστη) τιμή. Αμέσως μετά παύει η εκτέλεση της επανάληψης και το πρόγγραμμα συνεχίζεται…
- While: εκτελείται όσο ισχύει η συνθήκη. Άρα για να περατωθεί η επανάληψη, πρέπει η συνθήκη να γίνει ψευδής.

Από for σε while

Μπορώ να μετατρέψω ένα for loop σε ένα

while loop μετατρέποντας την επανάληψη σε λογική συνθήκη

for -- > while

```
for i=1:2:10
    disp(i^2)

end
```

```
i=1;
while i<=10
  disp(i^2)
  i+i+2;
end
```

for -- > while

```
for i=[2 4 1 6]
  disp(i^2)
end
```

```
i=[2 4 1 6]
while i<=noe
    disp(i^2)
    noe=noe+1;
end
```

Συνοψίζοντας

`for`: όταν γνωρίζω αριθμό επαναλήψεων

`while` : όταν δεν γνωρίζω αριθμό επαναλήψεων αλλά γνωρίζω συνθήκη ελέγχου

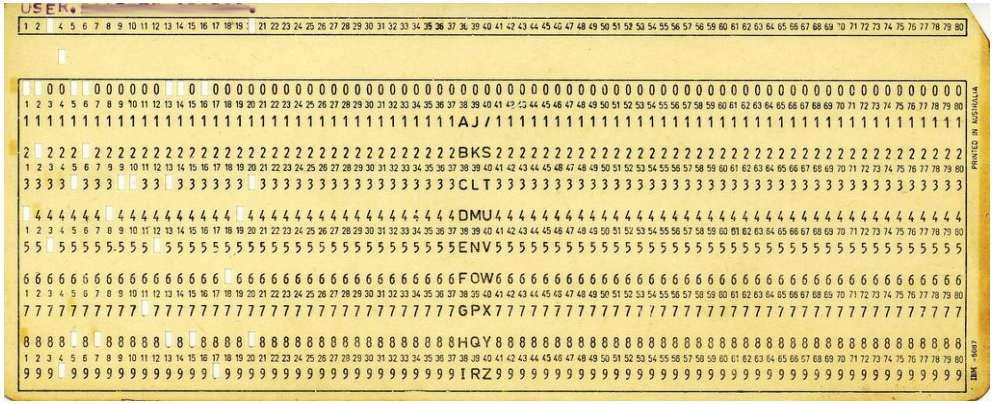
Τα **`while`** χρησιμοποιούν συνθήκες ελέγχου: συνεχίσουν έως ότου αυτές γίνουν ψευδείς

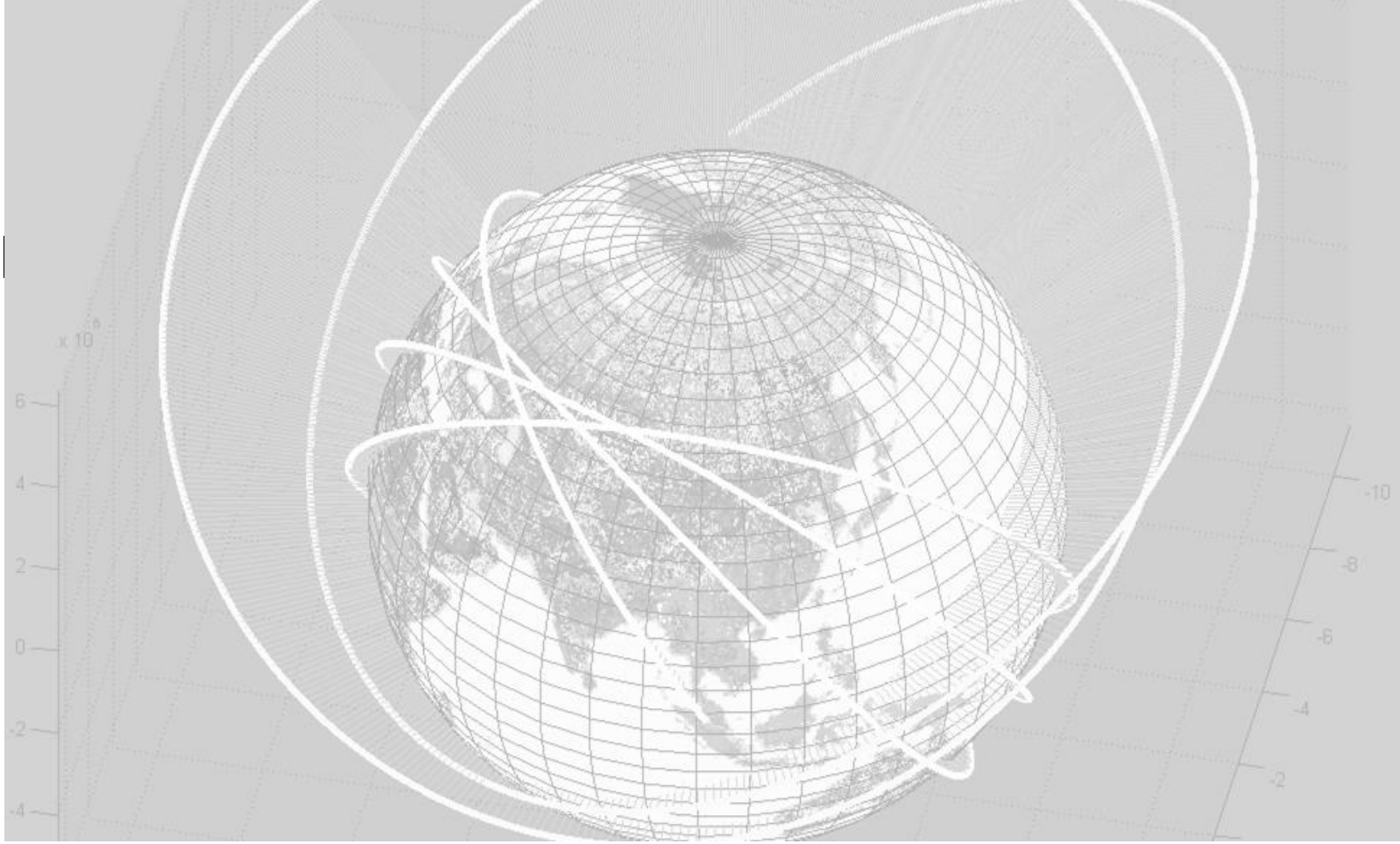
Τα **`for loops`** μπορούν να μετατραπούν σε **`while loops`**

Και κάτι τελευταίο…

Μόλις 30 χρόνια πριν!

Ένα πρόγραμμα (Fortran) αποτελούμενο από 278 διάτρητες κάρτες. Οι συνδυασμοί στις κάρτες αντιστοιχούσαν σε εντολές Fortran.





Τέλος

Κώστας Καρατζάς

Τμήμα Μηχανολόγων Μηχανικών, ΑΠΘ

Σημείωμα Αναφοράς

- Copyright Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, Κωνσταντίνος Καρατζάς. «Πληροφορική. Ενότητα 4: Α. Λογικές εκφράσεις (Παραστάσεις και Δείκτες). Β. Δομές Προγραμματισμού». Έκδοση: 1.0. Θεσσαλονίκη 2014. Διαθέσιμο από τη δικτυακή διεύθυνση: https://opencourses.auth.gr/courses/OCRS328/

Σημείωμα Αδειοδότησης

Τέλος ενότητας

Θεσσαλονίκη, Εαρινό Εξάμηνο 2014-2015

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σημειώματα

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.