ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

ΠΛΗΡΟΦΟΡΙΚΗ Ι

Ενότητα 2: Έλεγχος συνθηκών

Μιχάλης Δρακόπουλος

Σχολή Θετικών επιστημών

Τμήμα Μαθηματικών

ΠΛΗΡΟΦΟΡΙΚΗ Ι (MATLAB)
Ενότητα 2

Σημειώσεις βασισμένες στο βιβλίο "Το MATLAB στην Υπολογιστική Επιστήμη και Τεχνολογία – Μια Εισαγωγή"

Έλεγχος συνθηκών - `if`

Ας μελετήσουμε το πρόβλημα του υπολογισμού του ελάχιστου της συνάρτησης $f(x)=x^2+bx+c$ στο διάστημα $[L,R]$. Έτσι, θα πάρουμε μια ιδέα για τους ελέγχους συνθηκών.

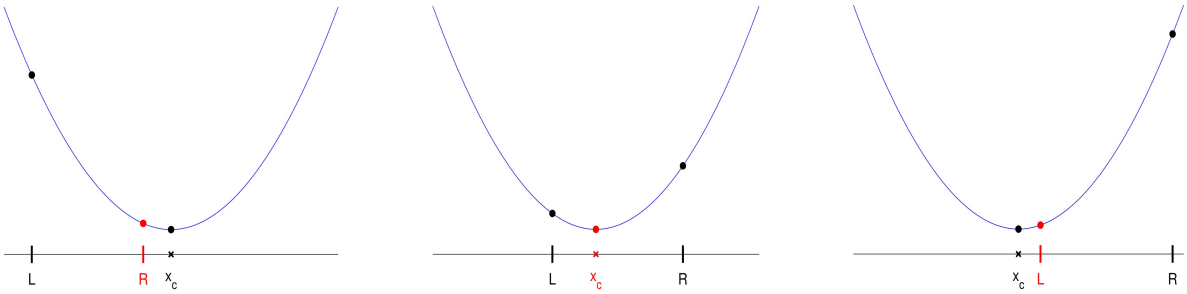
Γνωρίζουμε πως το ελάχιστο βρίσκεται στο κρίσιμο σημείο $x_c=-\frac{b}{2}$.

- Αν το $x_c \in [L,R]$, τότε το ελάχιστο είναι το $f(x_c)$.
- Αν το $x_c \notin [L,R]$, τότε το ελάχιστο είναι είτε το $f(L)$, είτε το $f(\mathrm{R})$.

Να γραφεί πρόγραμμα, το οποίο να ζητάει τους πραγματικούς αριθμούς *L*, *R*, *b* και *c* και να εμφανίζει το ελάχιστο της $f(x)$ στο $[L,R]$, καθώς και την τιμή του *x* στην οποία εμφανίζεται.

Υπάρχουν 3 περιπτώσεις:

(1) (2) (3)



(1)	(2)	(3)
$x_c > R$	$L \le x_c \le R$	$x_c < L$
min = f(R)	min = f(x_c)	min = f(L)

Ο ψευδοκώδικας του αλγορίθμου:

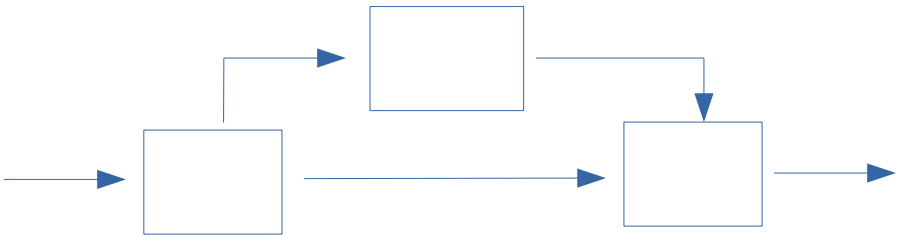
```
if xc < L
    εμφάνισε f(L) και L
else if L ≤ xc ≤ R
    εμφάνισε f(xc) και xc
else
    εμφάνισε f(R) και R
```

Για να μετατρέψουμε τον ψευδοκώδικα σε πρόγραμμα MATLAB χρειαζόμαστε:

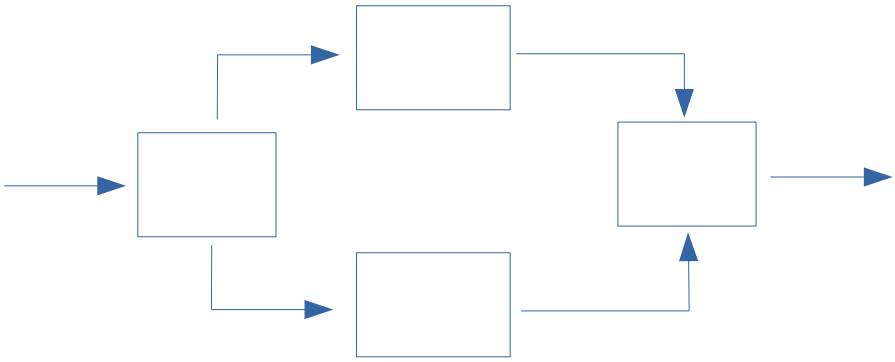
i. μια δομή γλώσσας που να επιτρέπει τη σύγκριση τιμών,
ii. μια δομή γλώσσας που να εκτελεί το κατάλληλο τμήμα κώδικα, ανάλογα με το αποτέλεσμα της σύγκρισης.

Άρα, υπάρχει η ανάγκη, ένα πρόγραμμα να μην εκτελείται πάντα σειριακά:

αλλά έτσι:



ή έτσι:



κτλ.

Δηλαδή, διάφορα τμήματα κώδικα εκτελούνται υπό συνθήκη.

Στο MATLAB, αυτό γίνεται με την εντολή if.

Η πιο βασική της μορφή είναι η δομή `if-else`:

Σύνταξη της `if-else`:

```
if λογική_έκφραση
   τμήμα κώδικα που εκτελείται αν η λογική έκφραση είναι αληθής
else
   τμήμα κώδικα που εκτελείται αν η λογική έκφραση είναι ψευδής
end
```

Λογικές (Boolean) εκφράσεις:

Όπως οι αριθμητικές εκφράσεις (πράξεις) όταν υπολογίζονται παράγουν αριθμητικές τιμές, οι λογικές εκφράσεις παράγουν την τιμή true (Αληθές) ή false (Ψευδές).

- Το false αντιστοιχεί στο 0 (και το 0 στο false).
- Το true αντιστοιχεί στο 1 (και οποιαδήποτε μη μηδενική τιμή αντιστοιχεί στο true).

Οι λογικές εκφράσεις υλοποιούνται με σχεσιακούς τελεστές (έναντι των αριθμητικών τελεστών των αριθμητικών εκφράσεων).

Μαθηματικά	MATLAB
<	<
≤	<=
>	>
≥	>=
=	==
≠	~=

Παράδειγμα της εντολής `if`:

Εκτελείται αν η συνθήκη είναι αληθής.

Εκτελείται αν η συνθήκη είναι ψευδής.

```
x=5;
y=2;
if (x>y)
    disp('Το x είναι μεγαλύτερο του y.');
else
    disp('Το x είναι μικρότερο ή ίσο του y.');
end
```

Λογικές Πράξεις:

Οι λογικές πράξεις υλοποιούνται με τους λογικούς τελεστές:

- Σύζευξη (ΚΑΙ – AND): `&&`
- Διάζευξη (Ή – OR): `||`
- Άρνηση (ΔΕΝ – NOT): `~`

} διμελείς τελεστές

Στο αρχικό παράδειγμα, ας υποθέσουμε ότι θέλουμε να εμφανίσουμε τα $f(x_c)$ και x_c αν είμαστε στην περίπτωση (2), (δηλαδή αν το $x_c \in [L,R]$) ή ένα πληροφοριακό μήνυμα σε αντίθετη περίπτωση.

Για να ελέγξουμε εάν το $x_c \in [L,R]$, απαιτούνται δύο συγκρίσεις: πρέπει το $x_c \geq L$ ΚΑΙ το $x_c \leq R$.

```
xc = -b/2;
if xc>=L && xc<=R
   fxc = c-(b/2)^2;
   fprintf('f(xc) = %6.3f xc = %6.3f\n', fxc,xc)
else
   disp('Είτε xc < L, είτε xc > R')
end
```

Πίνακας αληθείας:

p	q	p&&q	p||q
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

Αν το `p` είναι true, τότε το `~p` είναι false.

Αν το `p` είναι false, τότε το `~p` είναι true.

Προτεραιότητα τελεστών:

- πρώτα οι **αριθμητικοί** τελεστές
- μετά οι **σχεσιακοί** τελεστές και η άρνηση
- μετά οι διμελείς **λογικοί** τελεστές

Άλλες δομές του if:

- Σκέτο `if` :

```
if  λογική έκφραση
   κώδικας
end
```

π.χ.

```
if L>R
   temp = L;
   L = R;
   R = temp;
end
```

- `if – elseif` :

```
if λογική έκφραση 1
   κώδικας
elseif λογική έκφραση 2
   κώδικας
elseif λογική έκφραση 3
   κώδικας
else
   κώδικας
...
end
```

Εκτελείται, αν η λογική έκφραση 1 είναι true.

Εκτελείται, αν η λογική έκφραση 1 είναι false και η λογική έκφραση 2 είναι true.

Εκτελείται, αν η λογική έκφραση 1 είναι false και η λογική έκφραση 2 είναι false, αλλά η λογική έκφραση 3 είναι true.

Εκτελείται, αν όλες οι παραπάνω λογικές εκφράσεις είναι false.

→ Ας δούμε τώρα το πρόγραμμα για το αρχικό μας πρόβλημα, δηλαδή της εύρεσης του ελάχιστου της συνάρτησης $f(x)=x^2+bx+c$ στο διάστημα $[L,R]$.

```
% Script Eg1_2
%
% min της x^2+bx+c στο [L,R]

% Είσοδος δεδομένων
b=input('Δώσε το b: ');
c=input('Δώσε το c: ');
L=input('Δώσε το L: ');
R=input('Δώσε το R, με L<R: ');
fprintf('Δευτεροβάθμια: x^2+bx+c, b = %5.2f , c = %5.2f \n', b, c);
fprintf('Διάστημα: [L,R], L = %5.2f , R = %5.2f \n \n', L, R);

%Υπολογισμός κρίσιμου σημείου
xc= - b/2;
if xc < L
    % Το min είναι στο αριστερό όριο
    fL = L^2+b*L+c;
    fprintf('x ελαχιστοποίησης = %5.2f \n', L)
    fprintf('Ελάχιστη τιμή της f = %5.2f \n', fL)
elseif L <= xc && xc <=R
    % Το min στο κρίσιμο σημείο
    fxc = c - (b/2)^2;
    fprintf('x ελαχιστοποίησης = %5.2f \n', xc)
    fprintf('Ελάχιστη τιμή της f = %5.2f \n', fxc)
else
    % Το min στο δεξί όριο
    fR = R^2 + b*R +c
    fprintf('x ελαχιστοποίησης = %5.2f \n', R)
    fprintf('Ελάχιστη τιμή της f = %5.2f \n', fR)
end
```

→ Μια βελτίωση της δομής του προγράμματος:
Διαχωρισμός της εξόδου από τους υπολογισμούς

\+ : Λιγότερη πληκτρολόγηση (των εντολών εξόδου)
– : Ίσως ανάγκη δημιουργίας επιπλέον μεταβλητών

```
. . .
xc= - b/2;
if xc < L
    xmin = L;
elseif L <= xc && xc <=R
    xmin = xc;
else
    xmin = R;
end
fmin = xmin^2 + b*xmin +c;
fprintf('x ελαχιστοποίησης = %5.2f \n', xmin)
fprintf('Ελάχιστη τιμή της f = %5.2f \n', fmin)
```

} 1 φορά!

Παρατηρούμε εδώ, πως χρησιμοποιήσαμε ως επιπλέον μεταβλητή το xmin, αλλά επειδή κάναμε τον υπολογισμό του min της f έξω από το if, αντικαταστήσαμε τις τρεις μεταβλητές fL, fxc και fR με μία (fmin).

Κάποια επιπλέον στοιχεία:

► `min` και `max`:

Εύρεση ελάχιστου / μέγιστου μεταξύ δύο τιμών (μεταβλητών ή αριθμητικών εκφράσεων)

Σύνταξη: `min(αριθμητική έκφραση 1, αριθμητική έκφραση 2)`

► Αν *x* πραγματικός αριθμός:

`floor(x)` → στρογγυλοποίηση προς το $-\infty$.
`ceil(x)` → στρογγυλοποίηση προς το $+\infty$.
`round(x)` → στρογγυλοποίηση προς το πλησιέστερο
`fix(x)` → στρογγυλοποίηση προς το 0

► Αν *x*, *y* θετικοί ακέραιοι:

`rem(x,y)` → υπόλοιπο της διαίρεσης του *x* με το *y*.

[Μπορεί να εφαρμοστεί και σε πραγματικές τιμές]

► Άλλες χρήσιμες συναρτήσεις:

`sqrt(x)` → $\sqrt{x}$
`abs(x)` → $|x|$
`exp(x)` → e^x
`log(x)` → $\ln(x)$
`log10(x)` → $\log(x)$
`sin(x)` → ημ(*x*)
`cos(x)` → συν(*x*)
`tan(x)` → εφ(*x*)
`rand( )` → τυχαίος αριθμός στο (0,1)

►Shortcircuiting

Σε μια σύνθετη λογική έκφραση, το 2ο μέρος υπολογίζεται μόνο όταν δε μπορεί να βγει συμπέρασμα (να υπολογιστεί το αποτέλεσμα) μόνο από το 1ο.

Άρα στη λογική έκφραση (ΛΕ): `ΛΕ1 && ΛΕ2` η ΛΕ2 υπολογίζεται μόνο εάν η ΛΕ1 είναι true. Αυτό συμβαίνει διότι αν η ΛΕ1 ήταν false, το αποτέλεσμα θα ήταν ούτως ή άλλως false.

Ομοίως, στη λογική έκφραση: `ΛΕ1 || ΛΕ2` η ΛΕ2 υπολογίζεται μόνο εάν η ΛΕ1 είναι false.

Καταλήγουμε επομένως στο συμπέρασμα πως είναι προτιμότερο να συντάσσεται πρώτα η έκφραση που δε δημιουργεί προβλήματα, π.χ., `(x~=0) && (y/x==0)` και η απλούστερη από τις δύο εκφράσεις, για παράδειγμα: `(x>1) || (3*x^2 - 2*x < y/(x^2 + 2))`.

Π.χ., η έκφραση `(3==7) && (2==(3/0))` θα επέστρεφε ως αποτέλεσμα false (0) και όχι σφάλμα, που κάποιος θα περίμενε.

► Ένα παράδειγμα σύνθετης λογικής έκφρασης είναι ο έλεγχος για το εάν ένα έτος είναι δίσεκτο.

Ένα έτος θεωρείται δίσεκτο εάν διαιρείται με το 4 αλλά όχι με το 100 ή αν διαιρείται με το 400.

Ο τρόπος που θα εκφράζαμε τη λογική αυτή έκφραση είναι:

```
(rem(year,4) == 0 && rem(year,100) ~= 0) || rem(year,400) == 0
```

ή καλύτερα:

```
rem(year,400) == 0 || (rem(year,4) == 0 && rem(year,100) ~= 0)
```

► Σύγκριση λογικών μεταβλητών με τις τιμές true / false:

Παρακάτω, βλέπουμε δύο παραδείγματα, στα οποία χρησιμοποιούνται δύο λογικές μεταβλητές (`isLeapYear` και `isEven`) και μία ακέραια μεταβλητή (`n`).

Παράδειγμα 1:

Αντί της: `if isLeapYear == true` προτιμάται η: `if isLeapYear` .

Παράδειγμα 2:

Αντί της :

```
if  rem(n,2) == 0
    isEven = true;
else
    isEven = false;
end
```

προτιμάται η:

```
isEven = (rem(n,2) == 0);
```

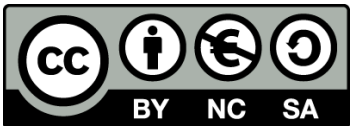
.

Σημειώματα

Σημείωμα Αναφοράς

Μιχάλης Δρακόπουλος. «Πληροφορική Ι. Ενότητα 2: Έλεγχος συνθηκών». Έκδοση: 1.0. Αθήνα 2014. Διαθέσιμο από τη δικτυακή διεύθυνση: http://opencourses.uoa.gr/modules/document/?course=MATH105.

Σημείωμα Αδειοδότησης



[1] http://creativecommons.org/licenses/by-nc-sa/4.0/

Ως **Μη Εμπορική** ορίζεται η χρήση:

- που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
- που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
- που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

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.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο τη 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.