ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΤΟΠΟΓΡΑΦΙΑΣ ΚΑΙ ΓΕΩΠΛΗΡΟΦΟΡΙΚΗΣ

Προγραμματισμός σε Octave/Matlab

Συμπληρωματικές σημειώσεις

Αναστάσιος Κεσίδης,

Αν. Καθηγητής

akesidis@uniwa.gr

ΠΕΡΙΕΧΟΜΕΝΑ

1 Βασική χρήση του Octave

1.1 Χρήση του Command Window ως αριθμομηχανή

Τι είναι το Command Window;

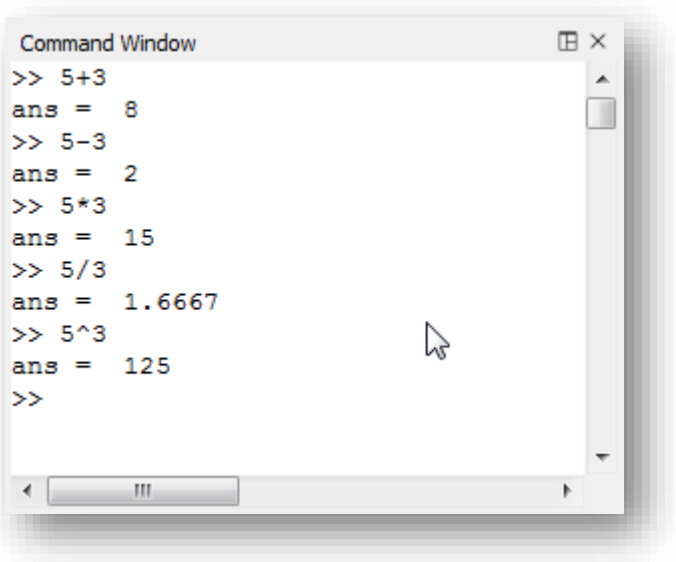
Είναι ο χώρος του Octave όπου μπορούν να πραγματοποιηθούν απευθείας υπολογισμοί ή να εκτελούνται κάποιες εντολές ή προγράμματα ή να εμφανίζονται τα αποτελέσματα από την εκτέλεση προγραμμάτων. Όταν το Command Window είναι σε κατάσταση αναμονής εμφανίζεται το σύμβολο >>. Στην περίπτωση αυτή μπορούμε να πραγματοποιήσουμε κάποιες πράξεις ή να τρέξουμε κάποιο πρόγραμμα.

Πώς μπορεί να χρησιμοποιηθεί το Command Window ως αριθμομηχανή;

Στο >> δίνουμε κατευθείαν την πράξη που μας ενδιαφέρει και έπειτα πατάμε το ENTER. Το αποτέλεσμα εμφανίζεται από κάτω.

Παράδειγμα 1-1 Βασικές πράξεις στο Command Window

Δώστε παραδείγματα εκτέλεσης των βασικών πράξεων πρόσθεσης, αφαίρεσης, πολλαπλασιασμού, διαίρεσης και εκθετικού στο Command Window

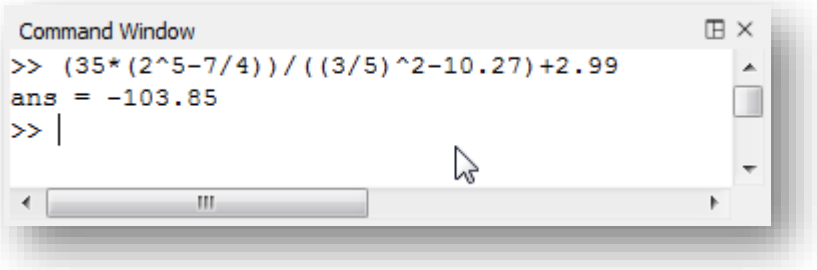


Μπορούν να πραγματοποιηθούν και πιο σύνθετοι αριθμητικοί υπολογισμοί;

Ναι, με σωστή χρήση των παρενθέσεων μπορεί να περιγραφεί οποιαδήποτε περίπλοκη αριθμητική έκφραση.

Παράδειγμα 1-2 Σύνθετοι αριθμητικοί υπολογισμοί

Υπολογίστε την τιμή του κλάσματος $\frac{35\cdot(2^5-\frac{7}{4})}{\left(\frac{3}{5}\right)^2-10.27}+2.99$



Το αποτέλεσμα είναι -103.85.

Στο Command Window εμφανίζονται συνεχώς διάφορα δεδομένα. Πως μπορούμε να το καθαρίσουμε;

Με την εντολή `clc`.

1.2 Δήλωση και διαχείριση μεταβλητών στο Command Window

Πως δηλώνουμε μεταβλητές στο Command Window;

Για να ορίσουμε την τιμή μιας μεταβλητής γράφουμε δίπλα από το >> το όνομά της και της αποδίδουμε τιμή μέσω της ισότητας. Εάν η μεταβλητή δεν έχει προηγουμένως οριστεί τότε δημιουργείται αυτόματα στην μνήμη του Octave και αμέσως μετά της αποδίδεται και η τιμή. Εάν η μεταβλητή υπάρχει ήδη, τότε η καινούργια τιμή που της δίνουμε αντικαθιστά την παλιά.

Πως μπορούμε να δούμε ποιες μεταβλητές υπάρχουν ανά πάσα στιγμή στην μνήμη του Octave;

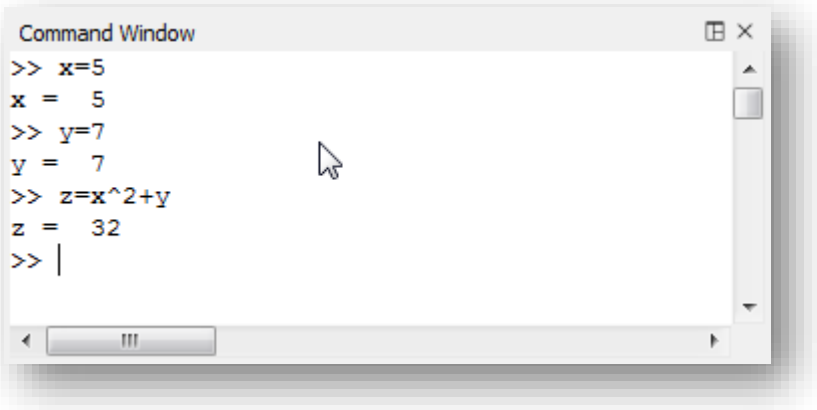
Χρησιμοποιώντας την εντολή `whos`.

Πως μπορούμε να διαγράψουμε κάποια μεταβλητή από την μνήμη του Octave;

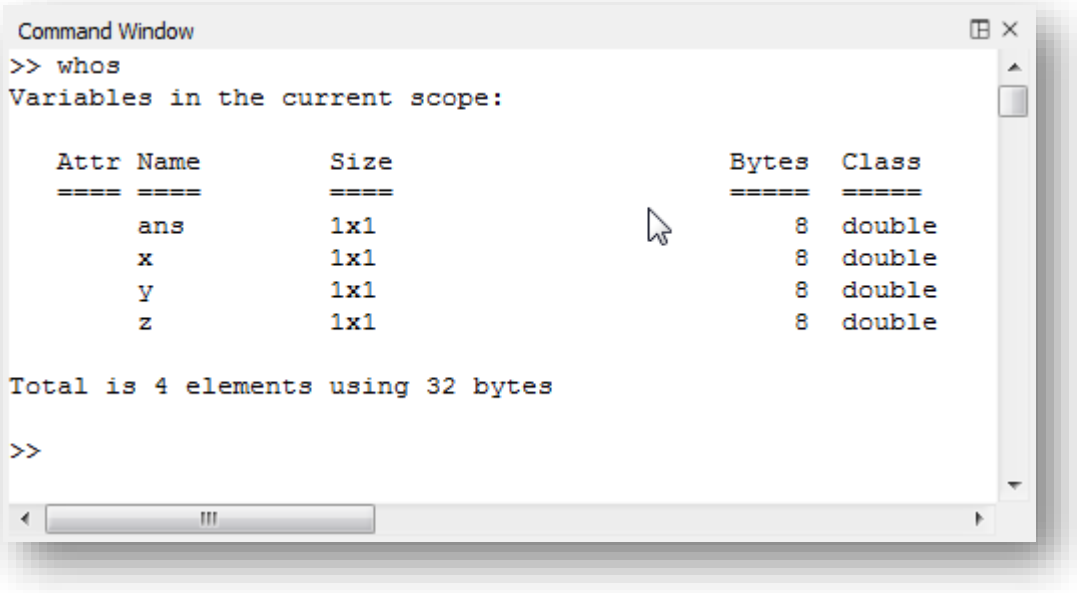
Με την εντολή `clear` ακολουθούμενη από την μεταβλητή που θέλουμε να σβήσουμε, π.χ. `clear x`. Εάν δώσουμε σκέτο `clear` τότε διαγράφονται όλες οι μεταβλητές από την μνήμη του Octave.

Παράδειγμα 1-3 Εμφάνιση και διαγραφή μεταβλητών

Για $x=5$ και $y=7$ υπολογίστε την παράσταση $z=x^2+y$

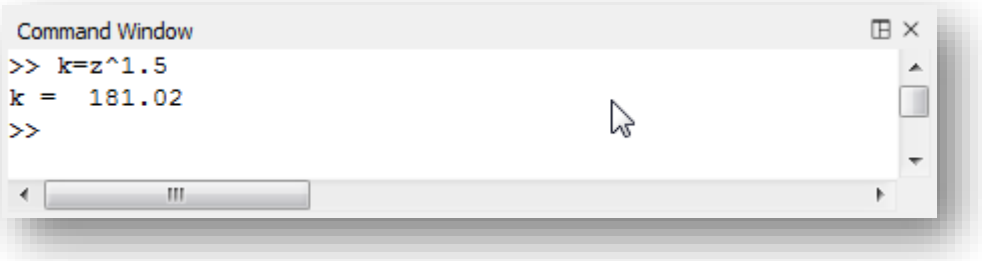


Καθαρίστε το Command Window (εντολή `clc`) και έπειτα εμφανίστε τις μεταβλητές που υπάρχουν στην μνήμη.

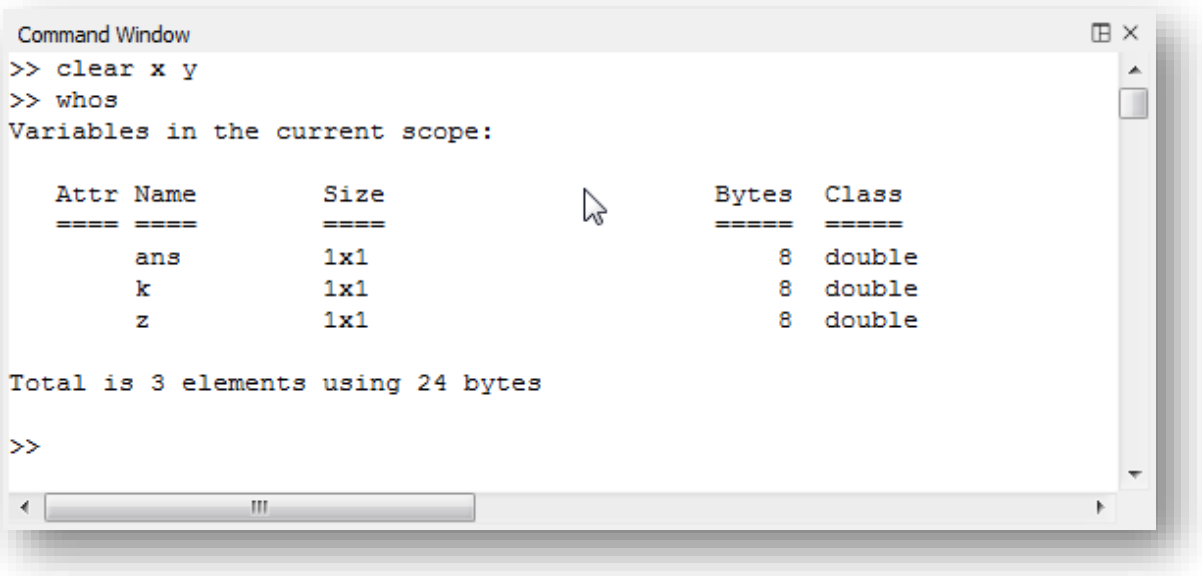


Η μεταβλητή ans δεν είναι ορισμένη από τον χρήστη αλλά είναι μια γενικής χρήσης μεταβλητή που χρησιμεύει για να εμφανίζονται αποτελέσματα πράξεων και υπολογισμών

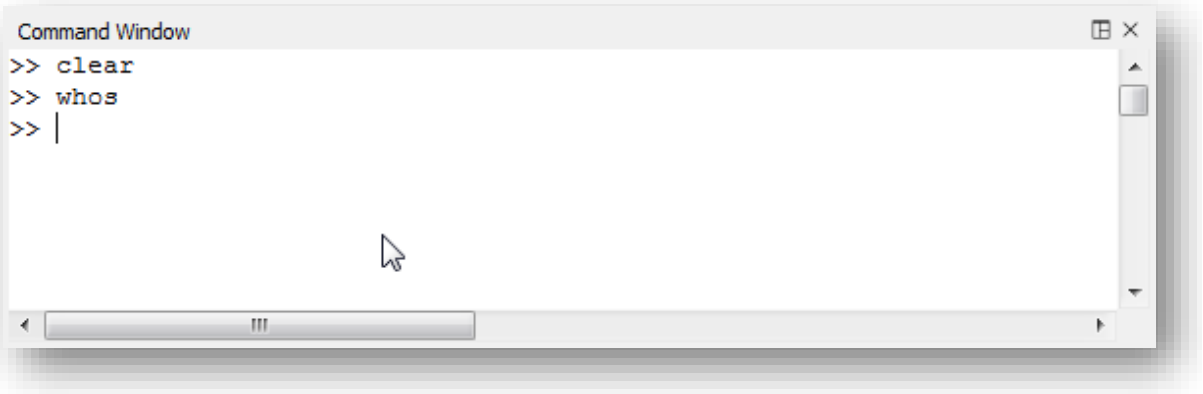
Υπολογίστε την τιμή $k = z^{1.5}$



Διαγράψτε τις μεταβλητές x και y και εμφανίστε έπειτα τις μεταβλητές που έχουν απομείνει.



Διαγράψτε όλες τις μεταβλητές και ελέγξτε την μνήμη.

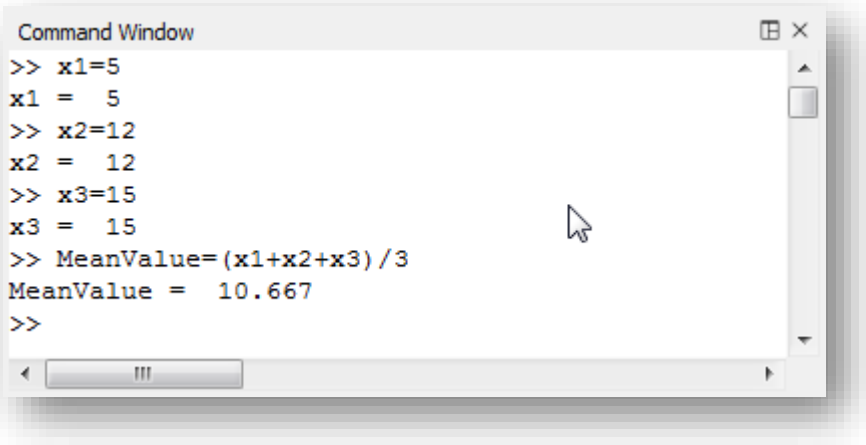


Έχουν σημασία τα πεζά ή κεφαλαία στις ονομασίες των μεταβλητών;

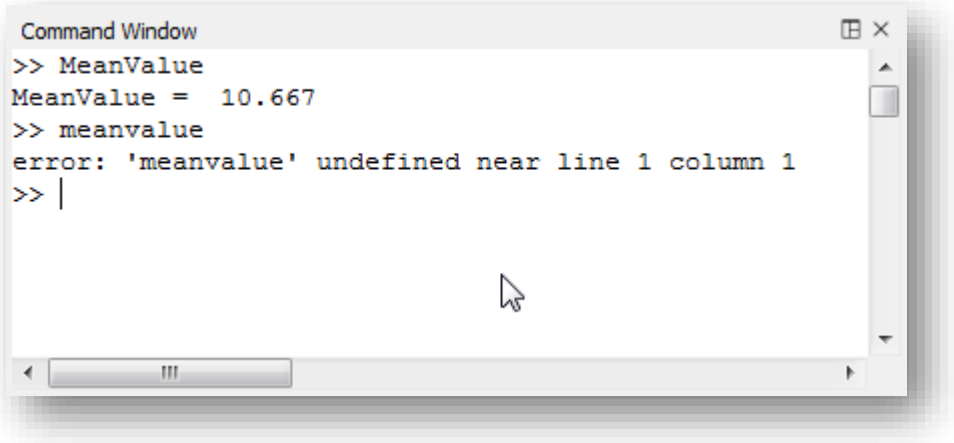
Ναι. Το Octave όπως και το Matlab είναι case-sensitive. Για παράδειγμα, το `MeanValue` και το `meanvalue` αντιπροσωπεύουν δύο διαφορετικές μεταβλητές στην μνήμη του Octave. Σε περίπτωση που ο χρήστης ζητήσει την τιμή μιας μεταβλητής που είναι γραμμένη με διαφορετικό τρόπο τότε εμφανίζεται μήνυμα λάθους στο Command Window.

Παράδειγμα 1-4 Χρήση πεζών/κεφαλαίων σε ονόματα μεταβλητών

Για $x_1 = 5, x_2 = 12$ και $x_3 = 15$ υπολογίστε την μέση τιμή $MeanValue = \frac{x_1+x_2+x_3}{3}$



Καθαρίστε το Command Window και ζητήστε να εμφανιστούν οι μεταβλητές `MeanValue` (που υπάρχει) και `meanvalue` (που δεν υπάρχει). Παρατηρήστε το μήνυμα λάθους που εμφανίζεται στην δεύτερη περίπτωση.



1.3 Δημιουργία προγραμμάτων στον Editor

Πως γράφουμε ένα πρόγραμμα στο περιβάλλον του Octave;

Στις προηγούμενες παραγράφους είδαμε το πώς μπορεί κανείς να χρησιμοποιήσει το Command Window για να κάνει κάποιες πράξεις ή να δηλώσει μεταβλητές και να κάνει κάποιους απλούς υπολογισμούς με αυτές τις μεταβλητές. Στην πραγματικότητα, όταν θέλουμε να γράψουμε ένα πρόγραμμα στο Octave που αποτελείται από ένα σύνολο εντολών χρησιμοποιούμε τον Editor. Το εργαλείο αυτό του Octave αποτελεί έναν εξελιγμένο κειμενογράφο (text editor) που είναι προσαρμοσμένος στις απαιτήσεις της γλώσσας προγραμματισμού του Octave. Έτσι, με την χρήση διάφορων χρωματισμών στο κείμενο ή αυτόματης μορφοποίησης οι εντολές του προγράμματος παρουσιάζονται με πιο εύληπτο τρόπο στον χρήστη του Octave.

Οι ρυθμίσεις για την μορφοποίηση του Editor υπάρχουν στην καρτέλα Edit -> Preferences… -> Editor styles.

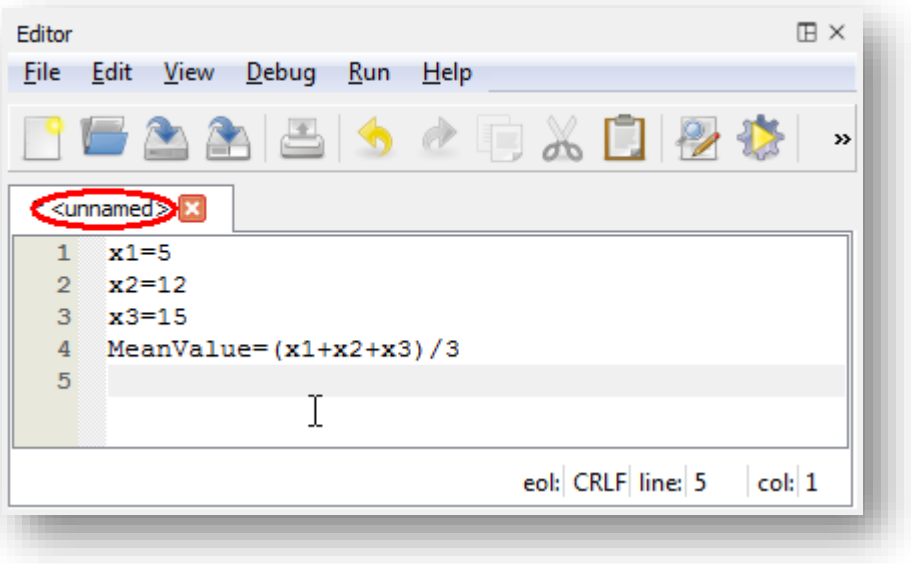
Πως εκτελούνται οι εντολές στον Editor;

Το πρόγραμμα εκτελείται πατώντας το F5 στο πληκτρολόγιο ή το εικονίδιο  πάνω από τον Editor. Την πρώτη φορά που θα εκτελεστεί το πρόγραμμα θα ζητηθεί να αποθηκευτεί στον δίσκο. Τα αρχεία του Octave, όπως και τα αρχεία του Matlab έχουν κατάληξη `.m`. Όταν το πρόγραμμα αποθηκευτεί τότε εκτελείται και τα αποτελέσματα εμφανίζονται είτε στο Command Window ή/και σε κάποιο γράφημα ή το όλο πρόγραμμα τρέχει σε παραθυρική μορφή (GUI – Graphical User Interface).

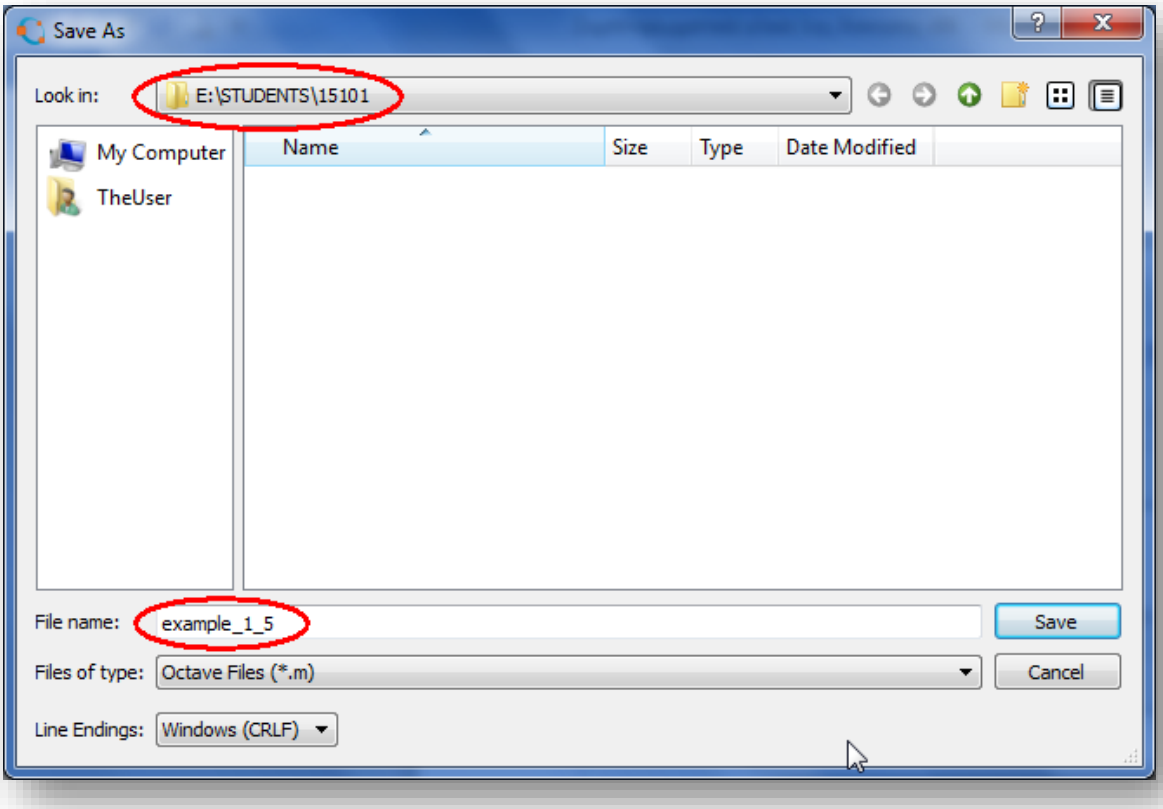
Κάθε φορά που ξανατρέχει το πρόγραμμα, αυτό πρώτα αποθηκεύεται (στο αρχείο που ορίσαμε την πρώτη φορά) και έπειτα εκτελείται. Με αυτό τον τρόπο είναι εξασφαλισμένο ότι ανά πάσα στιγμή στον δίσκο είναι αποθηκευμένη η τελευταία μορφή του προγράμματος και δεν χρειάζεται ο χρήστης να πατά συνεχώς Αποθήκευση του αρχείου.

Παράδειγμα 1-5 Βασική χρήση του Editor

Γράψτε το Παράδειγμα σε μορφή προγράμματος στον Editor.

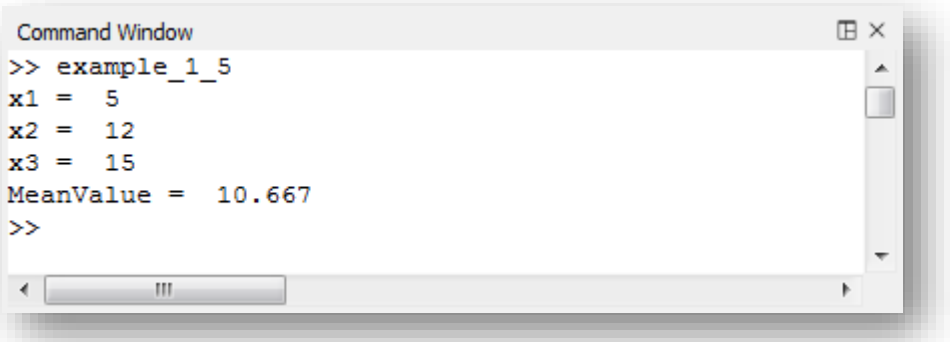


Παρατηρήστε ότι αρχικά δεν έχει καθοριστεί όνομα για το αρχείο. Πατήστε F5 στο πληκτρολόγιο ή το εικονίδιο  ώστε να τρέξει το πρόγραμμα. Την πρώτη φορά θα ζητήσει να αποθηκεύσει το αρχείο. Προσέξτε τον τρέχοντα φάκελο (που είναι ο ίδιος που υπάρχει στο άνω μέρος στο κεντρικό παράθυρο του Octave).
Δώστε όνομα αρχείου `example_1_5` (η κατάληξη `.m` μπαίνει αυτόματα από το Octave).



Τα ονόματα των αρχείων ΔΕΝ επιτρέπεται να έχουν κενό (space) χαρακτήρα. Αντί αυτού μπορεί να χρησιμοποιηθεί το _ ή το - .

Μόλις τρέξει το πρόγραμμα, στο Command Window εμφανίζονται οι αντίστοιχες εντολές και το τελικό αποτέλεσμα.

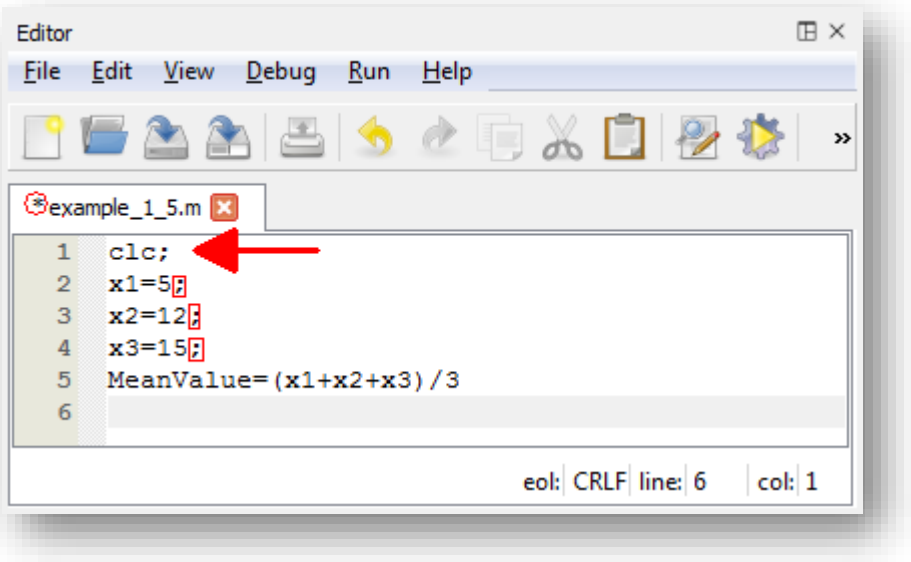


Γίνεται να εμφανίζεται μόνο το αποτέλεσμα στο Command Window και όχι όλες οι ενδιάμεσες εντολές;

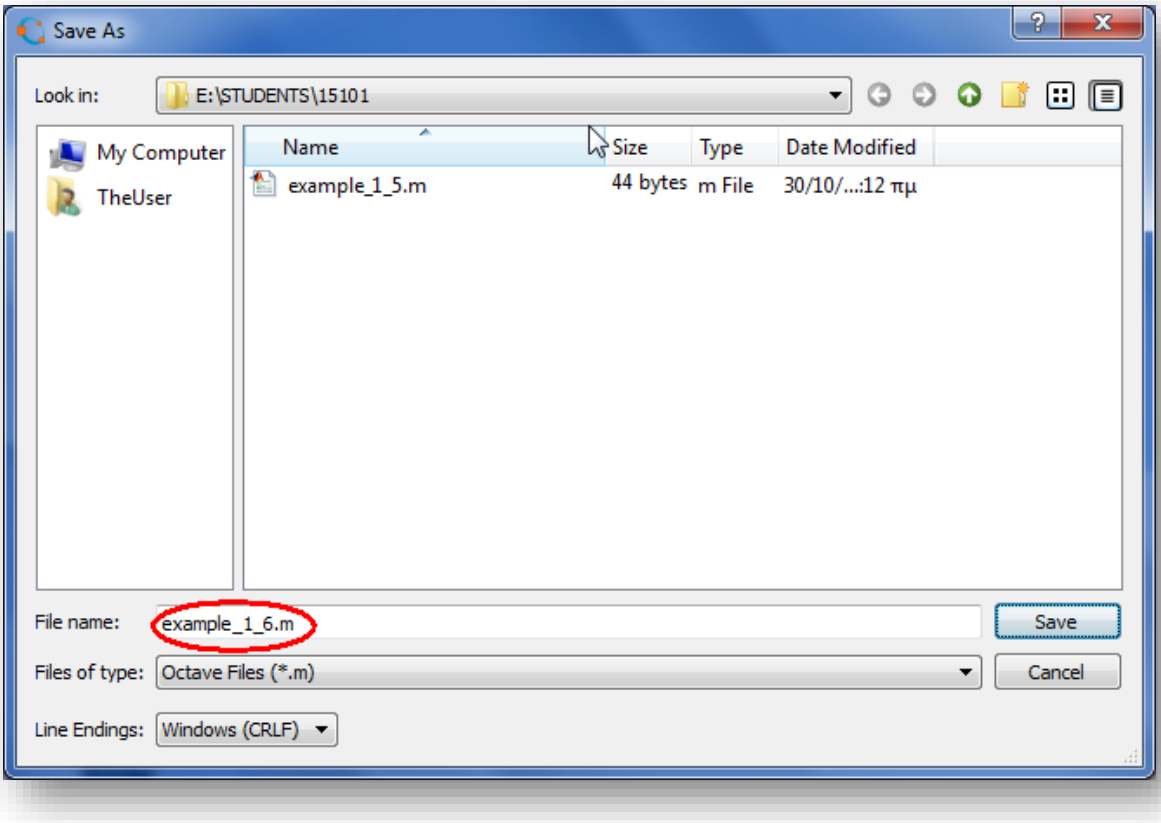
Ναι. Για να εκτελείται μια εντολή αλλά να ΜΗΝ εμφανίζεται στο Command Window θα πρέπει να τελειώνει με ερωτηματικό ; . Στην περίπτωση αυτή αφήνουμε χωρίς ερωτηματικό μόνο τις εντολές που πραγματικά επιθυμούμε να εμφανίζονται στο Command Window, για παράδειγμα το τελικό αποτέλεσμα κάποιων υπολογισμών, κλπ.

Παράδειγμα 1-6 Χρήση του τελικού ; στις εντολές

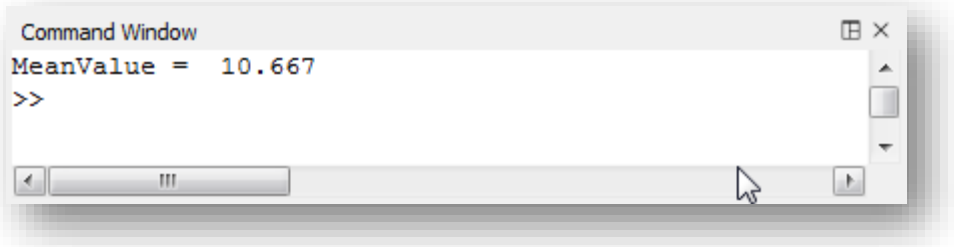
Τροποποιήστε τον κώδικα στο Παράδειγμα 1-5 ώστε να καθαρίζει το Command Window και να αποκρύπτει τις εντολές που ορίζουν τις τρεις μεταβλητές.



Ο αστερίσκος στο όνομα του αρχείου `example_1_5.m` υποδηλώνει ότι έχουν γίνει αλλαγές στον κώδικα από την τελευταία φορά που αποθηκεύτηκε το αρχείο.
Εάν τρέξουμε τώρα το πρόγραμμα με F5 τότε οι αλλαγές αυτές θα αποθηκευτούν στο αρχείο `example_1_5.m`. Έστω ότι θέλουμε να διατηρήσουμε το αρχείο αυτό ως έχει στο Παράδειγμα 1-5 και να αποθηκεύσουμε τις αλλαγές σε ένα καινούργιο αρχείο. Στην περίπτωση αυτή πατάμε στο εικονίδιο  (Save File As) και ορίζουμε το καινούργιο όνομα του αρχείου `example_1_6.m`.



Μόλις αποθηκευτεί το αρχείο `example_1_6.m` πατάμε F5 για να εκτελεστεί το πρόγραμμα. Το Command Window αδειάζει (λόγω της εντολής `clc`) και εμφανίζεται μόνο το αποτέλεσμα `MeanValue` (αφού μόνο η τελευταία γραμμή του κώδικα δεν είχε ερωτηματικό στο τέλος).



Στα επόμενα παραδείγματα η ονομασία του .m αρχείου θα συμβαδίζει με την αρίθμηση του εκάστοτε Παραδείγματος.

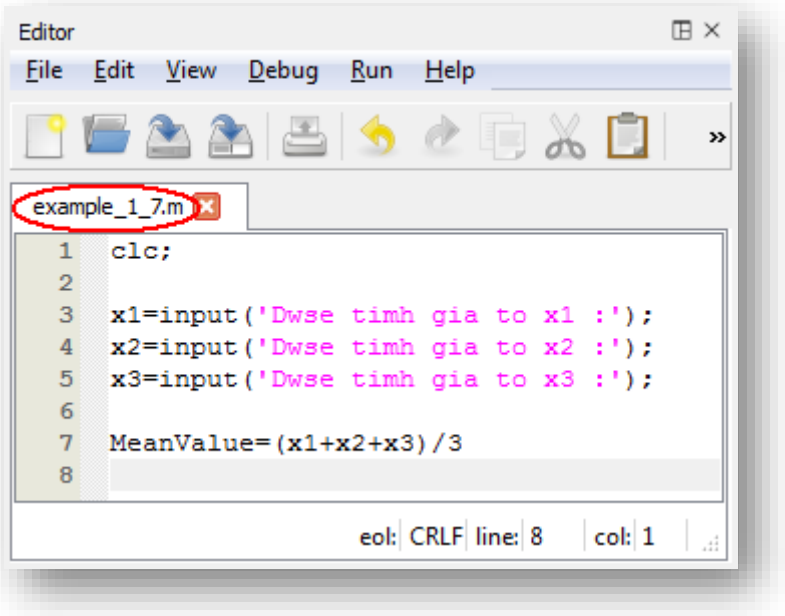
1.4 Εισαγωγή δεδομένων από τον χρήστη

Στα παραπάνω παραδείγματα οι τιμές των μεταβλητών είναι καθορισμένες στο πρόγραμμα. Υπάρχει τρόπος ώστε να ζητούνται αυτές οι τιμές από τον χρήστη κάθε φορά που εκτελείται το πρόγραμμα;

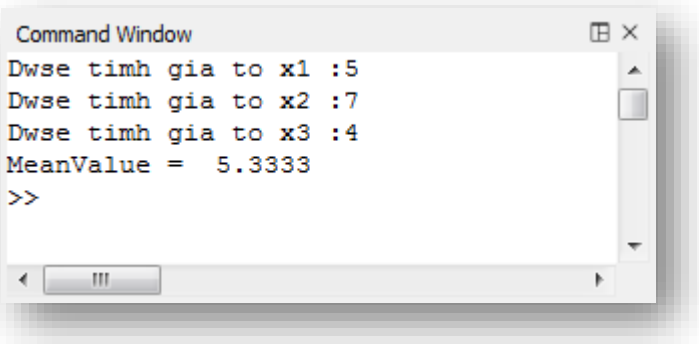
Πράγματι, η λειτουργικότητα του προγράμματος όπως φαίνεται στο Παράδειγμα 1-6 είναι πολύ περιορισμένη καθώς εκτελείται μόνο για τις συγκεκριμένες τιμές για τις μεταβλητές εισόδου. Θα ήταν πολύ πιο εύχρηστο εάν μπορούσε ο χρήστης στην αρχή του προγράμματος να δίνει τιμές για τις μεταβλητές που χρησιμοποιούνται. Αυτό επιτυγχάνεται με την χρήση της εντολής `input`. Η εντολή αυτή δέχεται σαν όρισμα ένα κείμενο (string) που συνήθως είναι ένα προτρεπτικό μήνυμα προς τον χρήστη σχετικά με την τιμή που πρέπει να δώσει. Όταν εκτελείται η `input` εμφανίζεται στο Command Window το σχετικό κείμενο και το Octave αναμένει από τον χρήστη να δώσει μια τιμή η οποία και αποδίδεται στην μεταβλητή που σχετίζεται με την συγκεκριμένη `input`.

Παράδειγμα 1-7 Εισαγωγή αριθμητικών δεδομένων από τον χρήστη με την εντολή `input`

Τροποποιήστε τον κώδικα στο Παράδειγμα 1-6 έτσι ώστε οι τιμές για τις μεταβλητές x_1, x_2 και x_3 να δίνονται μέσω της εντολής `input`.



Τρέξτε το πρόγραμμα. Για $x_1 = 5, x_2 = 7$ και $x_3 = 4$ δίνει μέση τιμή 5.3333



Η χρήση ελληνικών χαρακτήρων στα μηνύματα κειμένου δεν υποστηρίζεται από το Octave, ειδικά στο Command Window. Π.χ. στα παραπάνω παραδείγματα, στον Editor στις εντολές `input` τα ελληνικά μπορούν να χρησιμοποιηθούν κανονικά. Όμως κατά την εκτέλεση του προγράμματος στο Command Window τα ελληνικά δεν εμφανίζονται σωστά. Η λύση σε αυτές τις περιπτώσεις είναι να γράφεται το κείμενο στα λατινικά είτε σε greeklish είτε στα αγγλικά.

Στο Matlab, εν γένει, δεν εμφανίζονται παρόμοια προβλήματα.

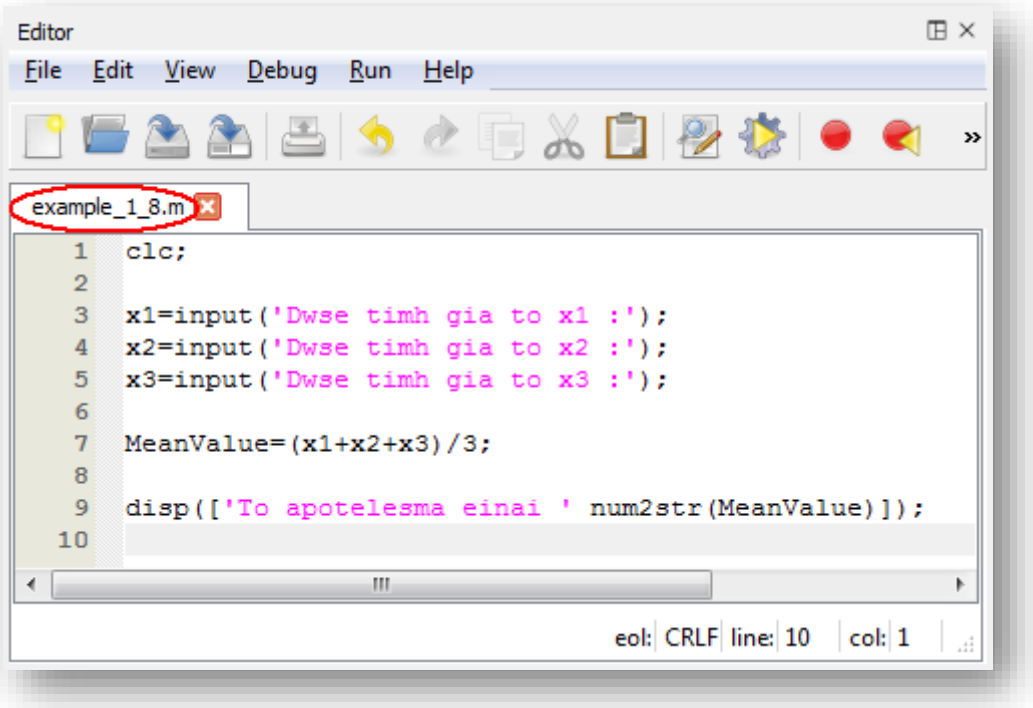
1.5 Μορφοποίηση της παρουσίασης των αποτελεσμάτων

Εκτός από το να μην βάλουμε το ερωτηματικό στο τέλος της εντολής, ποιοι άλλοι τρόποι υπάρχουν για να εμφανίσουμε τα αποτελέσματα στο Command Window;

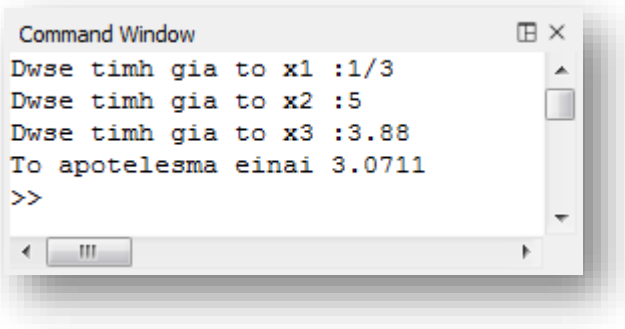
Η εντολή `disp` μπορεί να χρησιμοποιηθεί για την εμφάνιση αποτελεσμάτων στο Command Window. Σαν παραμέτρους μπορεί να λάβει τόσο κείμενο όσο και αριθμητικές μεταβλητές. Εναλλακτικά μπορεί να χρησιμοποιηθεί η εντολή `fprintf`.

Παράδειγμα 1-8 Εμφάνιση μορφοποιημένων αποτελεσμάτων με την εντολή `disp`

Τροποποιήστε τον κώδικα στο προηγούμενο Παράδειγμα έτσι ώστε το αποτέλεσμα να εμφανίζεται σε ένα επεξηγηματικό κείμενο χρησιμοποιώντας την εντολή `disp`.

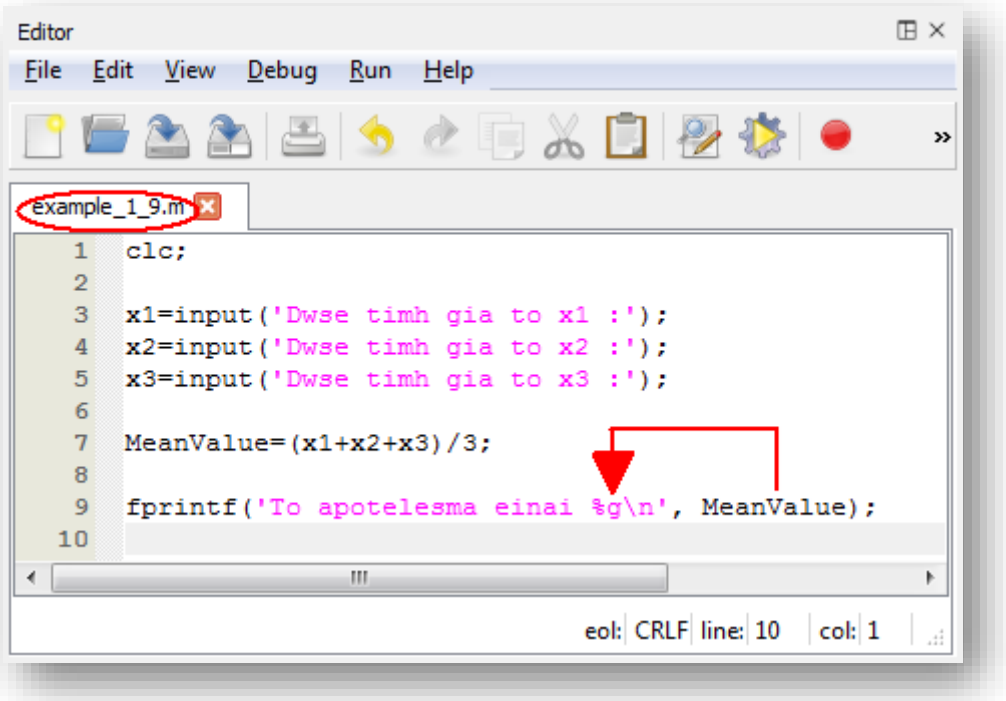


Η συνάρτηση `num2str` μετατρέπει μια αριθμητική μεταβλητή σε κείμενο. Στη συνέχεια, με την βοήθεια του άγκιστρου [ ] τα δύο κείμενα (το λεκτικό και το αποτέλεσμα της `num2str`) ενώνονται σε ένα ενιαίο κείμενο και εμφανίζονται με εντολή `disp`.
Τρέξτε το πρόγραμμα.
Παρατηρήστε ότι μπορούν να δοθούν τιμές για τις μεταβλητές x_1, x_2 και x_3 και σε κλασματική μορφή, π.χ. 1/3.
Παρατηρήστε ότι η συνάρτηση `num2str` επιστρέφει τους δεκαδικούς αριθμούς με ακρίβεια 4 δεκαδικών ψηφίων.

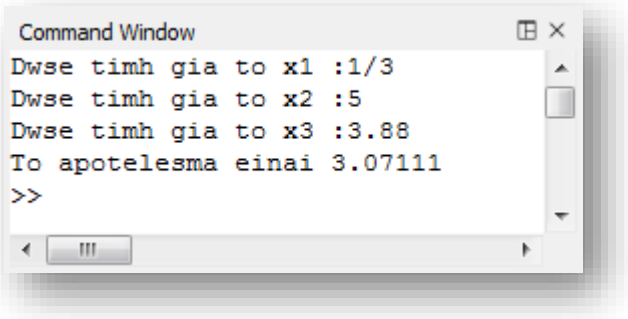


Παράδειγμα 1-9 Εμφάνιση μορφοποιημένων αποτελεσμάτων με την εντολή `fprintf`

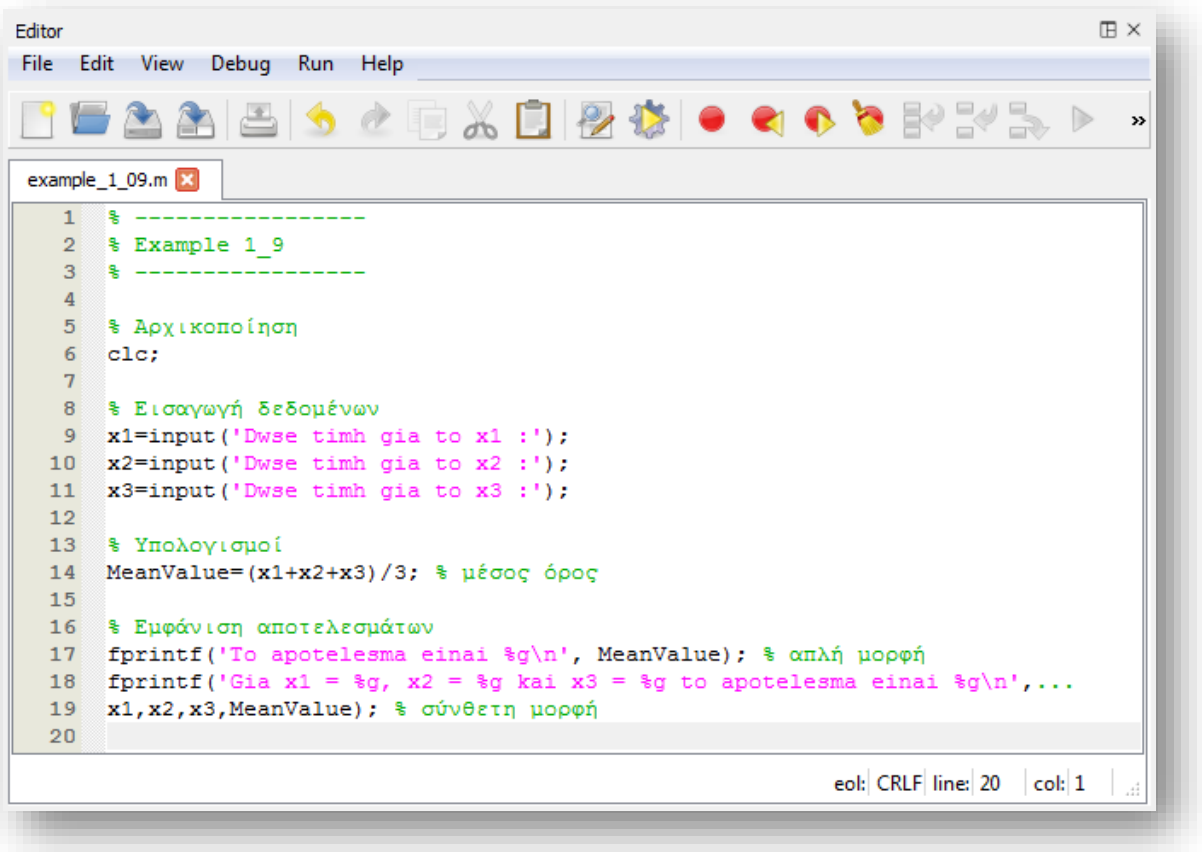
Τροποποιήστε τον κώδικα στο προηγούμενο Παράδειγμα έτσι ώστε το αποτέλεσμα να εμφανίζεται σε ένα επεξηγηματικό κείμενο χρησιμοποιώντας την εντολή `fprintf`.



Παρατηρήστε την σύνταξη της εντολής `fprintf`. Για κάθε αριθμητική μεταβλητή (που δίνεται στο τέλος της εντολής) υπάρχει μια υποδοχή στο κείμενο (που συμβολίζεται με το `%g`).
Ο χαρακτήρας `\n` στο τέλος του μηνύματος χρησιμεύει για αλλαγή γραμμής στο Command Window. Εάν δεν υπάρχει το `\n` τότε το επόμενο κείμενο στο Command Window θα εμφανιστεί στα δεξιά του μηνύματος.
Τρέξτε το πρόγραμμα.
Παρατηρήστε ότι η χρήση του `%g` στην συνάρτηση `fprintf` εμφανίζει τους δεκαδικούς αριθμούς με διαφορετική ακρίβεια ανάλογα με τον αριθμό. Στο συγκεκριμένο παράδειγμα το αποτέλεσμα εμφανίζεται με ακρίβεια 5 δεκαδικών ψηφίων.



Η εντολή `fprintf` μπορεί να χρησιμοποιηθεί για την παρουσίαση ακόμη πιο σύνθετων κειμένων με περισσότερες από μια αριθμητικές μεταβλητές. Για παράδειγμα, προσθέστε στον κώδικα μια ακόμη εντολή `fprintf` όπως εμφανίζεται παρακάτω στην γραμμή 18.

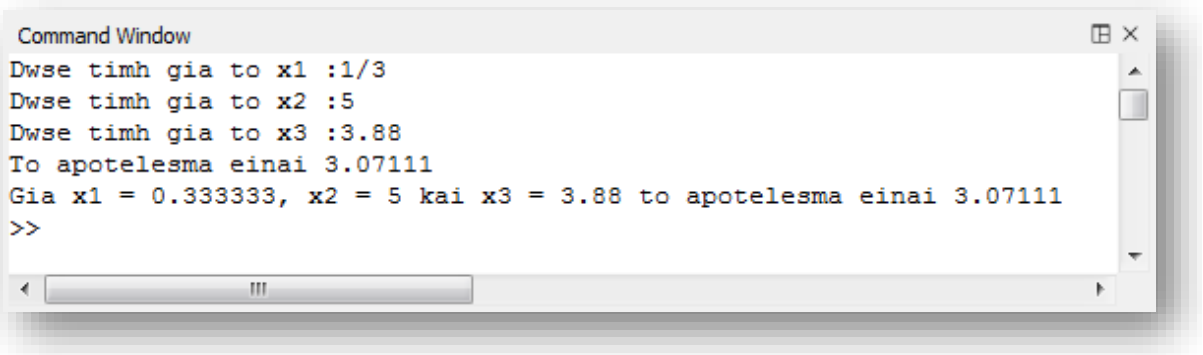


Όταν μια γραμμή στον Editor ξεκινά από το σύμβολο % τότε το κείμενο που ακολουθεί αγνοείται από τον Editor του Octave και χρωματίζεται με ξεχωριστό χρώμα, συνήθως πράσινο. Αυτός είναι ο τρόπος για να εισάγονται σχόλια στον κώδικα είτε ολόκληρης γραμμής (όπως οι πρώτες γραμμές του παραπάνω παραδείγματος) είτε στο τέλος κάποιας εντολής (π.χ. γραμμή 14 στο παραπάνω παράδειγμα).

Στα σχόλια που εισάγονται στον Editor μπορούν να χρησιμοποιηθούν ελληνικά.

Εάν το κείμενο σε μια γραμμή στον Editor είναι πολύ επίμηκες μπορεί να "σπάσει" σε περισσότερες γραμμές χρησιμοποιώντας τις τρεις τελείες . . . και συνεχίζοντας στην επόμενη γραμμή (όπως φαίνεται, για παράδειγμα, τις γραμμές 10 και 11 στον παραπάνω κώδικα).

Τρέξτε ξανά το πρόγραμμα του τελευταίου Παραδείγματος.
Παρατηρήστε ότι την πολλαπλή χρήση του `%g` στην συνάρτηση `fprintf`. Συγκεκριμένα, χρησιμοποιούνται τέσσερα `%g` μέσα στο κείμενο (γραμμή 10) με σκοπό την εμφάνιση των τεσσάρων αριθμητικών μεταβλητών που δίνονται στο τέλος της `fprintf` (γραμμή 11). Η αντιστοίχιση γίνεται με την σειρά εμφάνισης.
Παρατηρήστε επίσης, ότι όταν εκτελεστεί το πρόγραμμα οι αριθμοί στα `%g` παρουσιάζονται στο Command Window με διαφορετικό αριθμό δεκαδικών ψηφίων, ανάλογα με την περίπτωση. Π.χ. η μεταβλητή $x_1 = 1/3$ εμφανίζεται με 6 δεκαδικά ψηφία ενώ η $x_2 = 5$ εμφανίζεται ως ακέραιος.

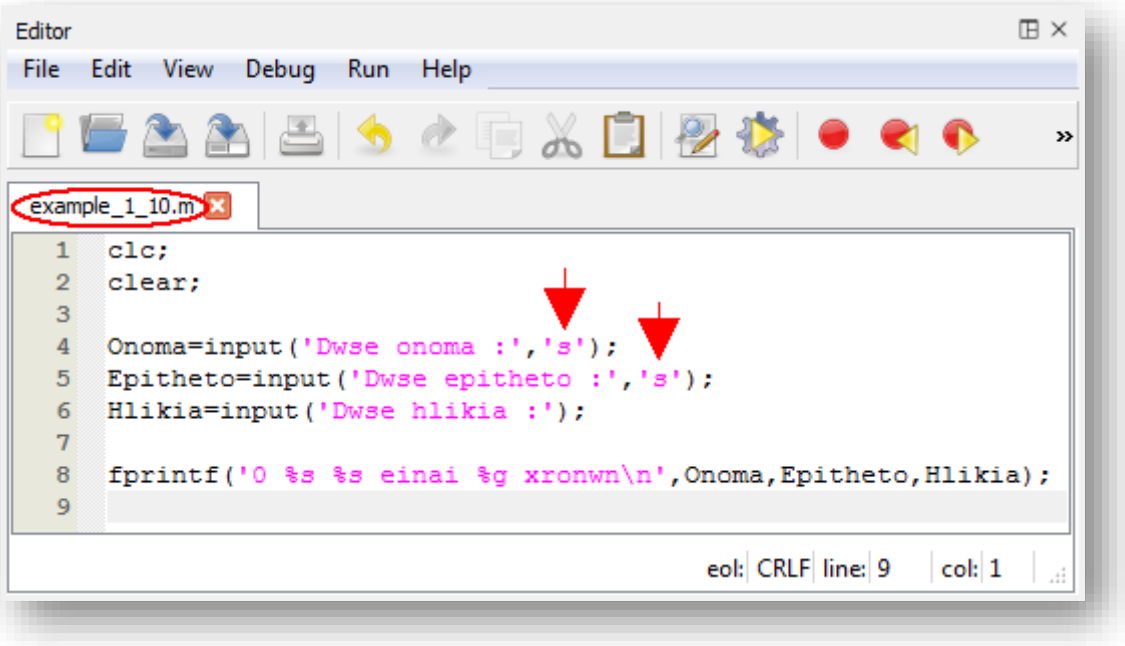


Πως μπορούμε να εισάγουμε και να μορφοποιήσουμε κείμενο (strings) από τον χρήστη;

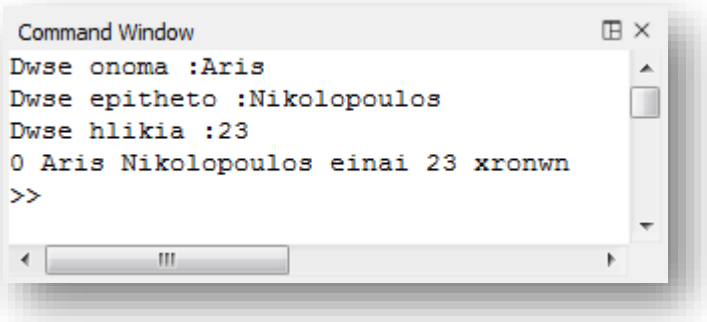
Η εντολή `input` δέχεται μια έξτρα παράμετρο `'s'` με την οποία δίνεται η δυνατότητα εισαγωγής αλφαριθμητικών (strings) από τον χρήστη.

Παράδειγμα 1-10 Εισαγωγή δεδομένων κειμένου από τον χρήστη με την εντολή `input`

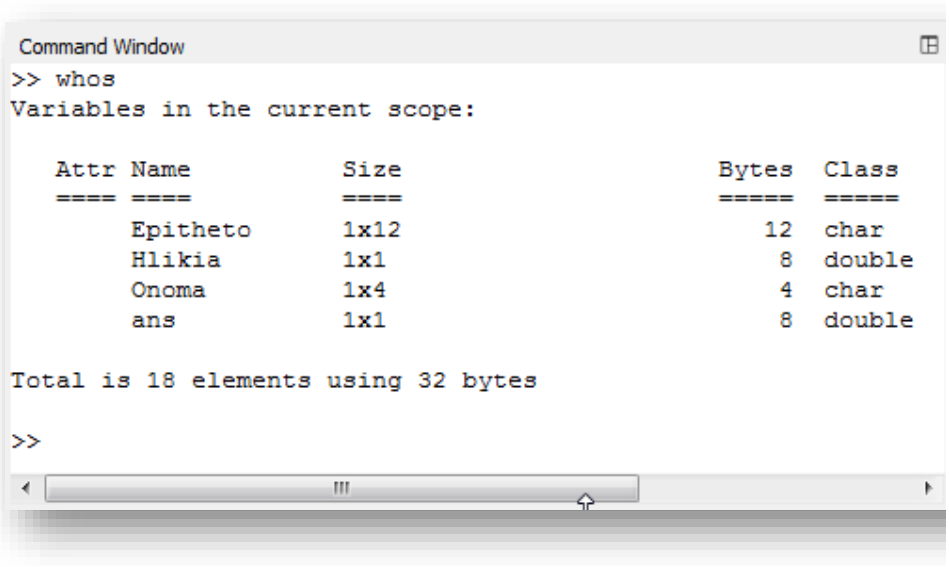
Γράψτε ένα πρόγραμμα που να ζητά το όνομα, το επίθετο και την ηλικία ενός ατόμου και έπειτα να εμφανίζει ένα σχετικό μορφοποιημένο κείμενο.



Τρέξτε το πρόγραμμα και δώστε κάποιες τιμές:



Το Octave όπως και στο Matlab υποστηρίζουν διάφορους τύπους μεταβλητών. Ο βασικός τύπος αριθμητικών μεταβλητών είναι ο τύπος `double`. Οι μεταβλητές κειμένου είναι τύπου `char`. Για παράδειγμα, δίνοντας την εντολή whos στο Command Window μετά το προηγούμενο παράδειγμα, εμφανίζει:



Παρατηρούμε ότι η μεταβλητή `Hlikia` είναι πραγματική μεταβλητή τύπου `double`, ενώ οι `Onoma` και `Epitheto` είναι μεταβλητές κειμένου τύπου `char`. Όπως φαίνεται στις δύο δεξιά στήλες ο τύπος δεδομένων `double` δεσμεύει 8 bytes στην μνήμη για κάθε μεταβλητή. Αντίθετα ο τύπος char είναι δυναμικός και το μήκος της μεταβλητής εξαρτάται από το πλήθος των χαρακτήρων του κειμένου που περιέχει. Έτσι η μεταβλητή `Epitheto` έχει μήκος 12 χαρακτήρες (Nikolopoulos) ενώ η μεταβλητή `Onoma` έχει μήκος 4 χαρακτήρες (Aris).

2 Έλεγχος ροής προγράμματος

2.1 Τελεστές σύγκρισης

Πως μπορούμε να συγκρίνουμε αριθμητικές μεταβλητές ως προς την τιμή τους;

Συχνά στα προγράμματα υπάρχει η ανάγκη να συγκριθεί η τιμή μιας αριθμητικής μεταβλητής με μια σταθερά ή οι τιμή δύο μεταβλητών μεταξύ τους. Για τον σκοπό αυτό χρησιμοποιούνται οι παρακάτω τελεστές σύγκρισης:

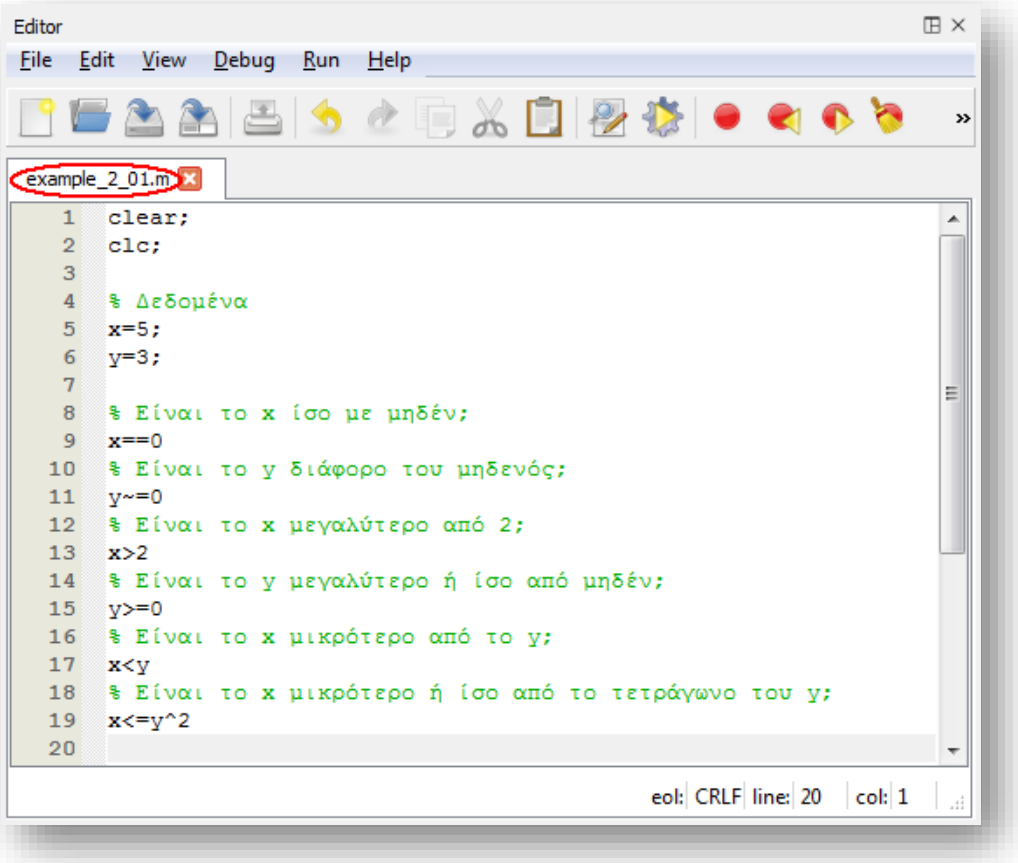
Τελεστές σύγκρισης

Πράξη	Τελεστής	Παράδειγμα	Αποτέλεσμα
Ισότητα	==	5==3	0 (ψευδές)
Ανισότητα	~=	5~=3	1 (αληθές)
Μεγαλύτερο	>	5>3	1 (αληθές)
Μεγαλύτερο ή ίσο	>=	5>=3	1 (αληθές)
Μικρότερο	<	5<3	0 (ψευδές)
Μικρότερο ή ίσο	<=	5<=3	0 (ψευδές)

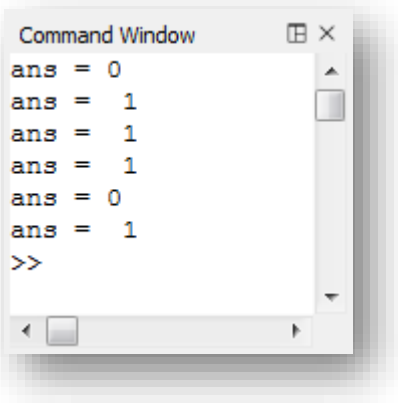
Το αποτέλεσμα ενός τελεστή σύγκρισης λαμβάνει τιμή είτε 1 (αληθές) είτε 0 (ψευδές).

Παράδειγμα 2-1 Σύγκριση αριθμητικών δεδομένων με την χρήση τελεστών σύγκρισης

Ορίστε τιμές για δύο πραγματικές μεταβλητές x και y και δοκιμάστε τους λογικούς τελεστές σε αυτές.



Τρέχοντας το πρόγραμμα δίνει τις αντίστοιχες έξι λογικές απαντήσεις:



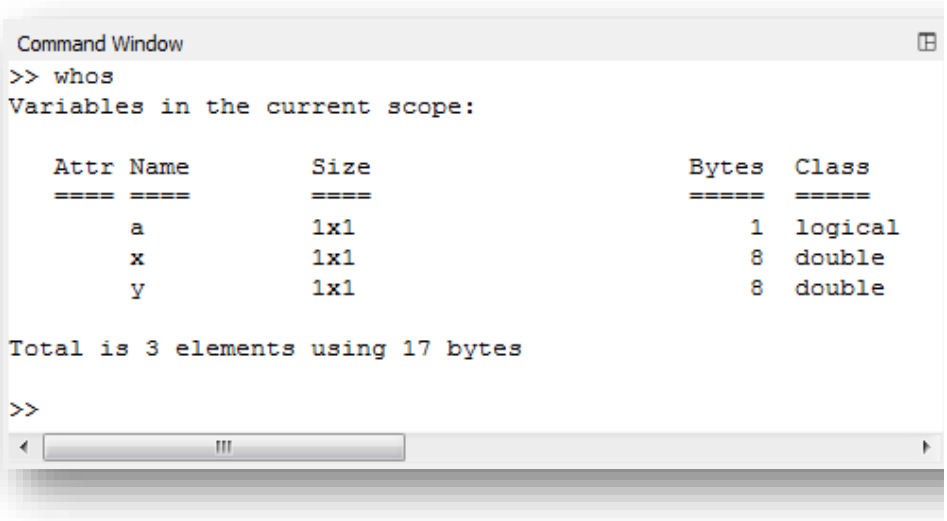
Το αποτέλεσμα μιας λογικής πράξης μπορεί να αποθηκευτεί και σε μια λογική μεταβλητή για περαιτέρω χρήση. Π.χ., σε συνέχεια του προηγούμενου παραδείγματος, γράφοντας στο Command Window την εντολή

```
>> a=(x>100)
```

δίνει

```
a=0
```

Με τον τρόπο αυτό η λογική μεταβλητή `a` περιέχει το λογικό αποτέλεσμα της σύγκρισης `x>100`, που είναι ψευδές. Δίνοντας `whos` στο Command Window επιστρέφει



Παρατηρούμε ότι ενώ οι μεταβλητές `x` και `y` είναι πραγματικές μεταβλητές τύπου `double`, η λογική μεταβλητή `a` είναι τύπου `logical`. Ο τύπος δεδομένων `logical` δεσμεύει 1 byte στην μνήμη για κάθε μεταβλητή, σε αντίθεση με τον τύπο `double` που δεσμεύει 8 bytes για κάθε μεταβλητή.

2.2 Δομές ελέγχου της ροής του προγράμματος

Πως μπορούμε να ορίσουμε τμήματα κώδικα τα οποία θα εκτελούνται μόνο εάν ισχύει (ή όχι) μια λογική συνθήκη;

Για τον σκοπό αυτό χρησιμοποιούνται οι δομές ελέγχου της ροής του προγράμματος που βασίζονται στην εντολή `if`. Οι βασικές τρεις βασικές δομές ελέγχου είναι οι εξής:

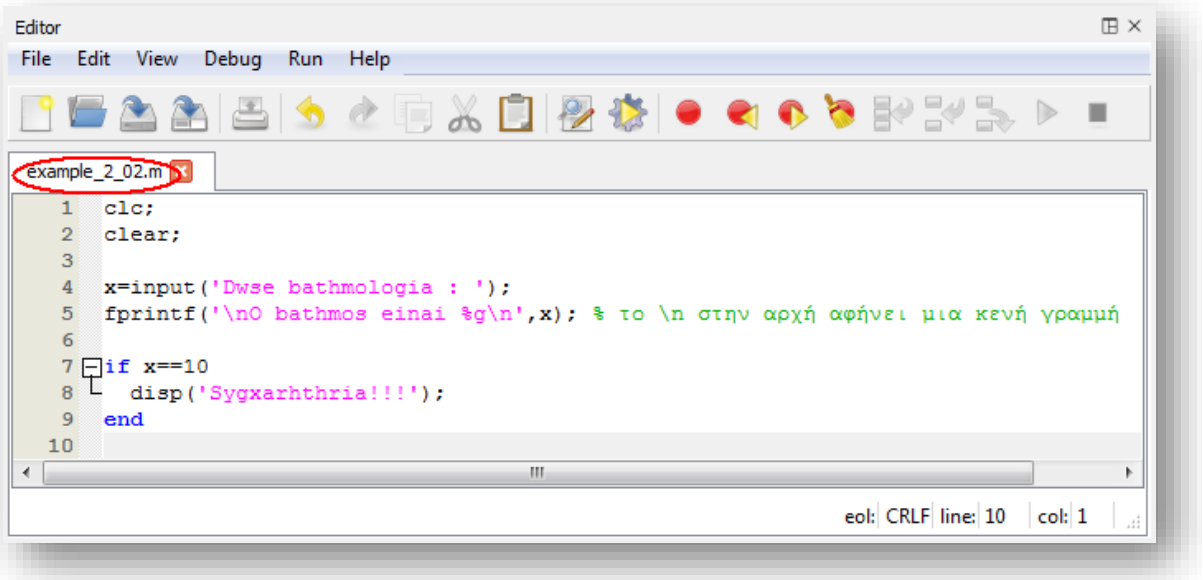
Δομή `if-end`

Σε αυτή την μορφή καθορίζεται η συνθήκη καθώς και οι εντολές που θα εκτελεστούν μόνο ισχύει η συνθήκη. Η δομή έχει την μορφή:

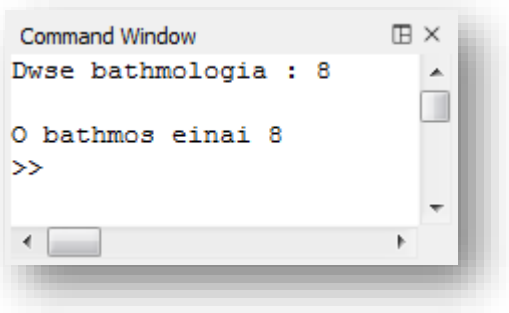
```
if <συνθήκη>
      <...>
      <εντολές που θα εκτελεστούν εάν η συνθήκη είναι αληθής>
      <...>
end
```

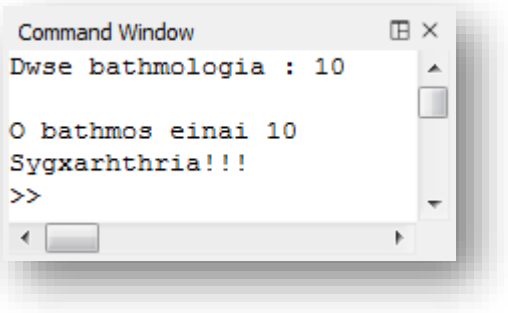
Παράδειγμα 2-2 Υπό συνθήκη εμφάνιση μηνύματος

Γράψτε ένα πρόγραμμα το οποίο να ζητά από τον χρήστη έναν βαθμό (θεωρούμε ότι είναι μεταξύ 0-10). Έπειτα, να εμφανίζει ένα μήνυμα σχετικά με τη βαθμολογία. Ειδικά εάν ο βαθμός είναι 10 τότε να εμφανίζει και το μήνυμα "Συγχαρητήρια!"



Τρέχοντας το πρόγραμμα για `x=8` και για `x=10` δίνει αντίστοιχα:





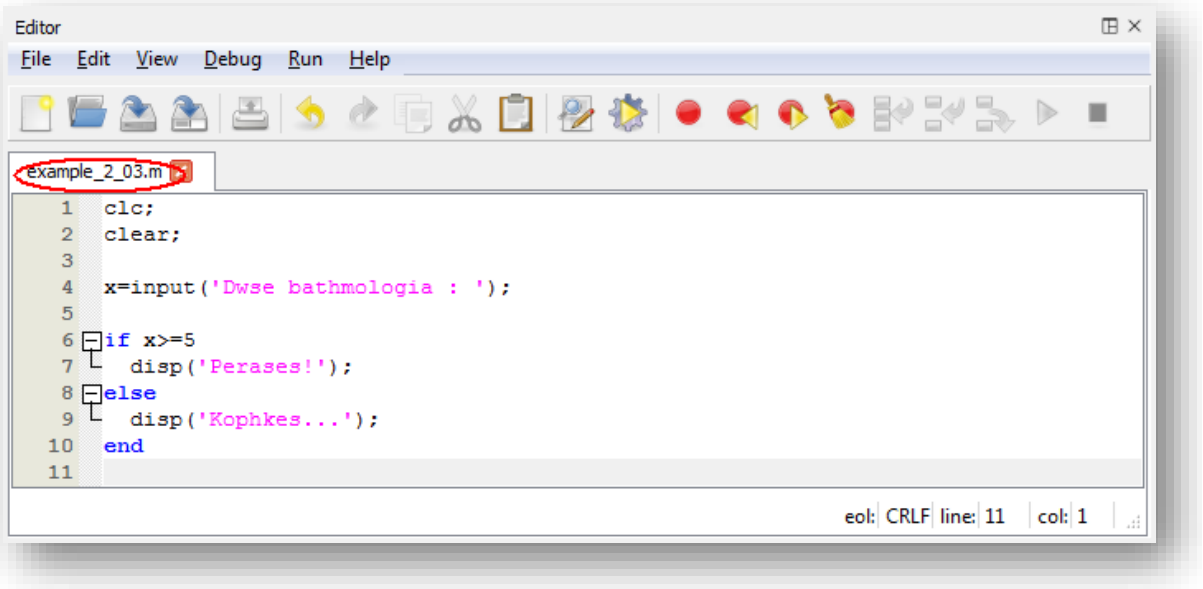
Δομή `if-else-end`

Σε αυτή την μορφή καθορίζεται η συνθήκη και οι εντολές που θα εκτελεστούν εάν η συνθήκη αυτή ισχύει καθώς και οι εντολές που θα εκτελεστούν εάν δεν ισχύει η συνθήκη. Η δομή έχει την μορφή:

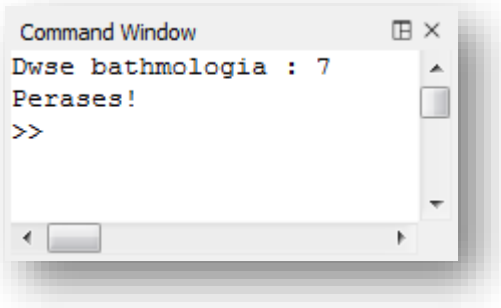
```
if <συνθήκη>
      <…>
      <εντολές που θα εκτελεστούν εάν η συνθήκη είναι αληθής>
      <…>
else
      <…>
      <εντολές που θα εκτελεστούν εάν η συνθήκη είναι ψευδής>
      <…>
end
```

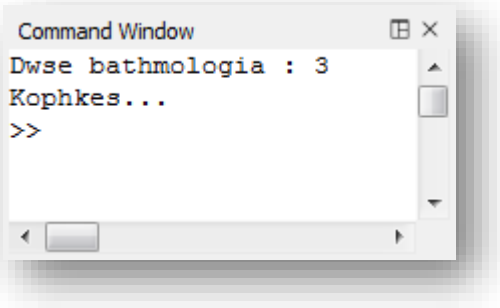
Παράδειγμα 2-3 Εμφάνιση διαφορετικών μηνυμάτων ανάλογα με το αποτέλεσμα της σύγκρισης

Γράψτε ένα πρόγραμμα το οποίο να ζητά από τον χρήστη έναν βαθμό (θεωρούμε ότι είναι μεταξύ 0-10). Εάν ο βαθμός είναι μεγαλύτερο ή ίσος από 5 να εμφανίζει το μήνυμα "Πέρασες!" αλλιώς να εμφανίζει το μήνυμα "Κόπηκες...".



Τρέχοντας το πρόγραμμα για `x=7` και για `x=3` δίνει αντίστοιχα:





Δομή `if-elseif-else-end`

Σε αυτή την μορφή καθορίζεται η συνθήκη και οι εντολές που θα εκτελεστούν εάν η συνθήκη αυτή ισχύει. Σε περίπτωση που δεν ισχύει η συνθήκη καθορίζονται επιπλέον μια ή περισσότερες συνθήκες καθώς και οι εντολές που θα εκτελεστούν σε περίπτωση που αυτές ισχύουν. Τέλος καθορίζονται οι εντολές που θα εκτελούνται εάν καμμία από τις προηγούμενες συνθήκες δεν ισχύει. Η δομή είναι:

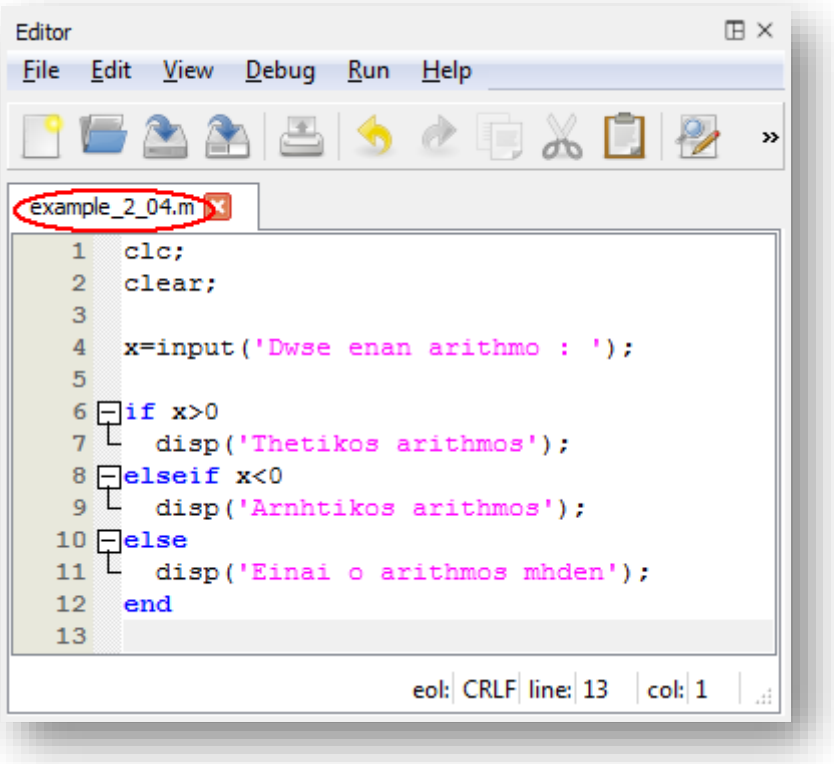
```
if <συνθήκη 1>
      <…>
      <εντολές 1>
      <…>
elseif <συνθήκη 2>
      <…>
      <εντολές 2>
      <…>
elseif <συνθήκη 3>
      <…>
      <εντολές 3>
      <…>
<…> (πιθανόν και άλλα elseif)
else
      <…>
      <εντολές που θα εκτελεστούν εάν δεν ισχύει καμία από τις προηγούμενες
      συνθήκες>
      <…>
end
```

Μια δομή `if-elseif-else-end` μπορεί να έχει ένα ή περισσότερα `elseif`. Επίσης, μπορεί να έχει ένα μόνο `Else` στο τέλος αλλά αυτό δεν είναι υποχρεωτικό.

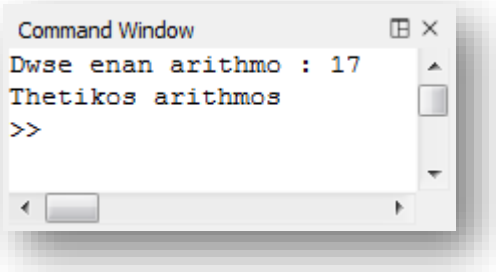
Η χρήση εσοχών σε όλες τις δομές `if` είναι ιδιαίτερα χρήσιμη και συνίσταται ανεπιφύλακτα. Βοηθά οπτικά στην καλύτερη κατανόηση του κώδικα καθώς και στον εντοπισμό πιθανών λαθών, όπως για παράδειγμα, ένα `if` χωρίς το αντίστοιχο `end` στο τέλος κλπ.

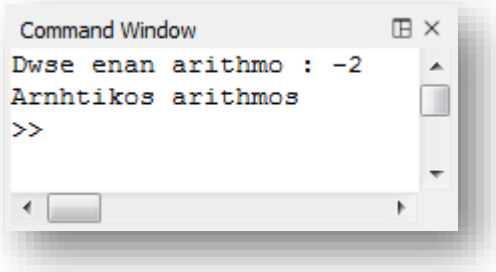
Παράδειγμα 2-4 Εμφάνιση διαφορετικών μηνυμάτων ανάλογα με πολλαπλά κριτήρια σύγκρισης

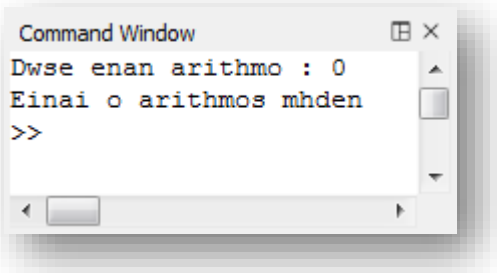
Γράψτε ένα πρόγραμμα το οποίο θα ζητά από τον χρήστη έναν αριθμό. Έπειτα, να εμφανίζει ένα σχετικό μήνυμα ανάλογα με το εάν ο αριθμός είναι θετικός, αρνητικός ή μηδέν.



Τρέχοντας το πρόγραμμα για `x=17`, `x=-2` και `x=0` δίνει, αντίστοιχα:







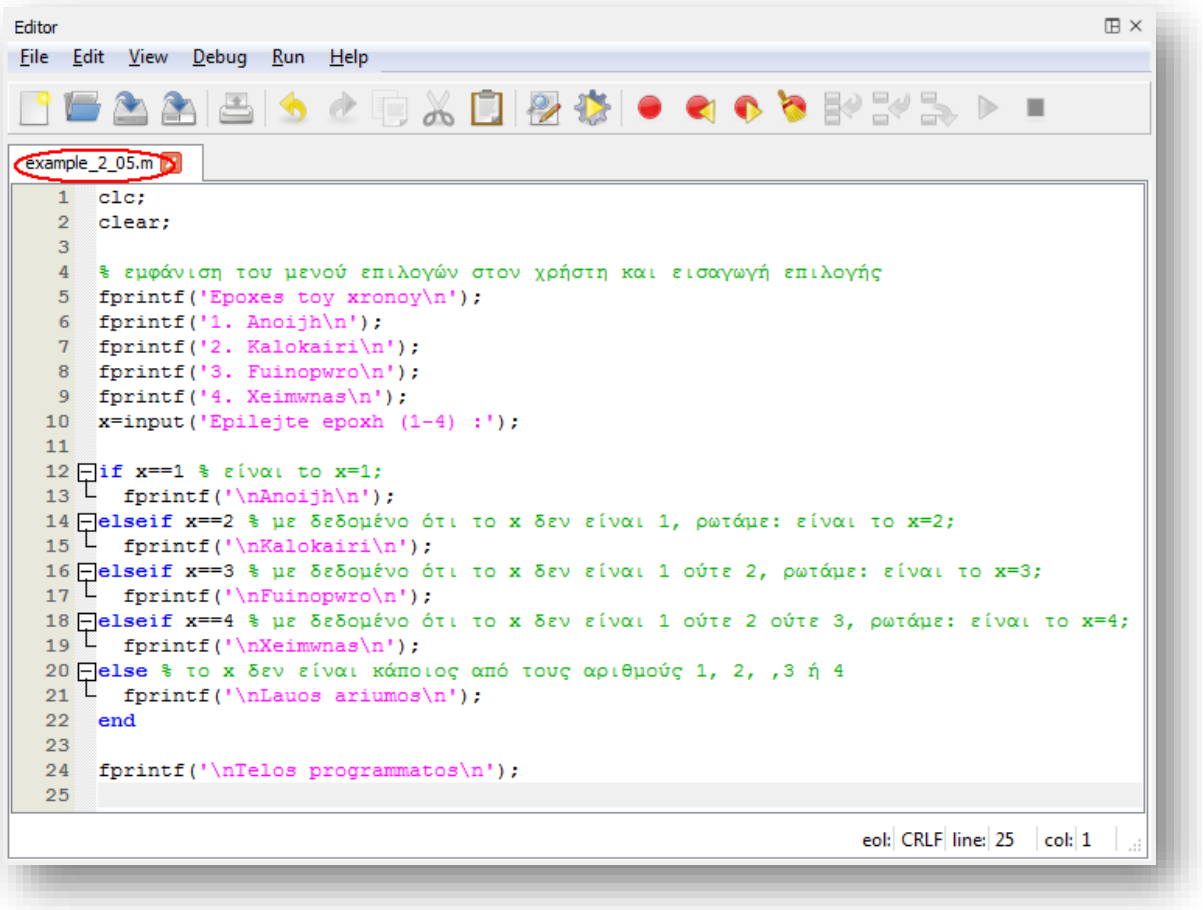
Πως ακριβώς ελέγχονται οι συνθήκες σε μια δομή `if-elseif-else-end` που αποτελείται από πολλαπλά `elseif`;

Οι συνθήκες ελέγχονται με την σειρά όπως είναι διατυπωμένες στην δομή `if-elseif-else-end`. Σε κάθε συνθήκη θεωρείται γνωστό ότι όλες οι προηγούμενες συνθήκες δεν ισχύουν, αφού εάν ίσχυε κάποια από τα

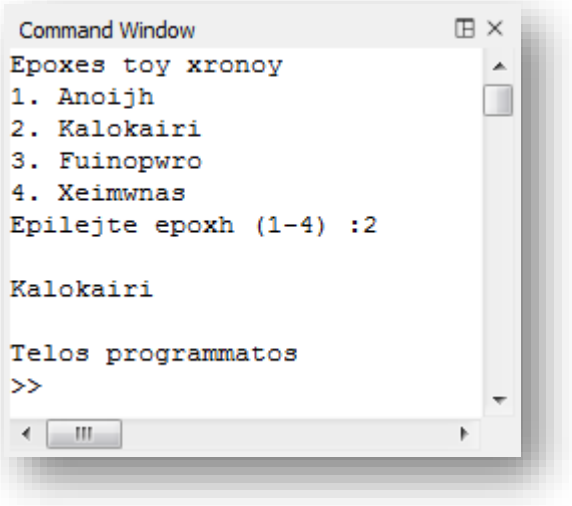
προηγούμενα `if` ή `elseif` τότε η ροή του προγράμματος θα είχε εκτελέσει τις αντίστοιχες εντολές και θα είχε βγει από την δομή.

Παράδειγμα 2-5 Χρήση της δομής `if-elseif-else-end` με πολλαπλά `elseif`

Γράψτε ένα πρόγραμμα το οποίο να εμφανίζει τις τέσσερις εποχές και να ζητά από τον χρήστη να επιλέξει τον αριθμό κάποιας εποχής. Έπειτα, να εμφανίζει την αντίστοιχη εποχή ανάλογα με τον αριθμό που επέλεξε ο χρήστης. Σε περίπτωση που επιλέξει λάθος αριθμό να εμφανίζεται σχετικό μήνυμα.



Στον παραπάνω κώδικα περιλαμβάνονται σχόλια που περιγράφουν το τι συμβαίνει καθώς εξετάζονται διαδοχικά οι διάφορες συνθήκες `if` και `elseif`.
Τρέχοντας το πρόγραμμα, π.χ. για `x=2` δίνει:



Δεν θα μπορούσαμε να γράψουμε το παραπάνω παράδειγμα με τέσσερα ξεχωριστά απλά `if`, ένα για κάθε περίπτωση;

Ναι. Όμως οι δομές `if-elseif-else-end` προσφέρουν δύο πλεονεκτήματα:

- ότι σε κάθε έλεγχο κάποιας συνθήκης γνωρίζουμε ότι οι προηγούμενες συνθήκες δεν ισχύουν.
- ο κώδικας είναι πιο λιτός

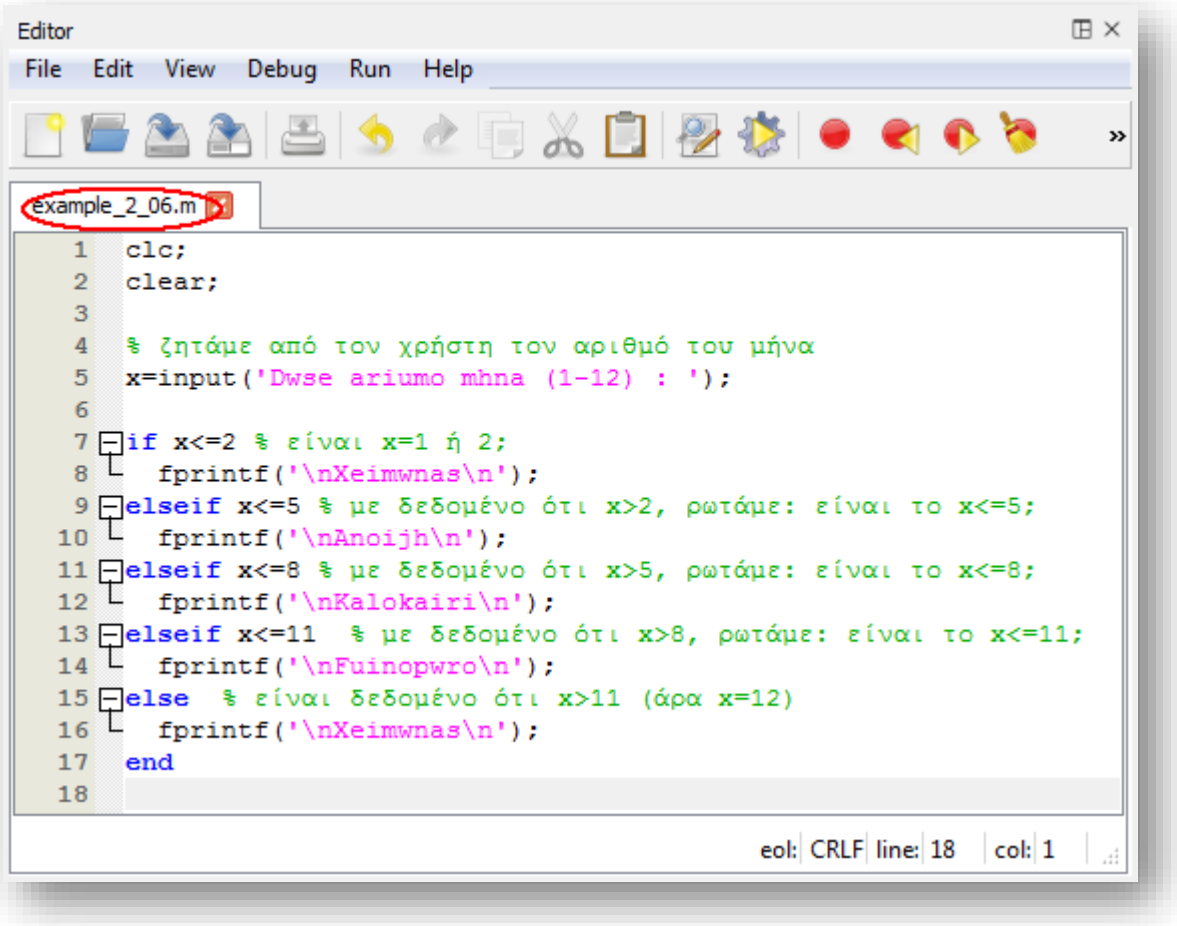
Έτσι, στο προηγούμενο παράδειγμα, εκτός από τα 4 ξεχωριστά `if` που θα χρειαζόμασταν (δηλαδή, ένα για κάθε εποχή) θα έπρεπε να γραφεί και ένα πέμπτο `if` που να αφορά την περίπτωση που ο αριθμός δεν είναι κάποιο από τα 1, 2, 3 ή 4 (δηλαδή την περίπτωση που καλύπτει το `else` στο παραπάνω παράδειγμα).

Τι άλλα πλεονεκτήματα έχει η χρήση της δομής `if-elseif-else-end`;

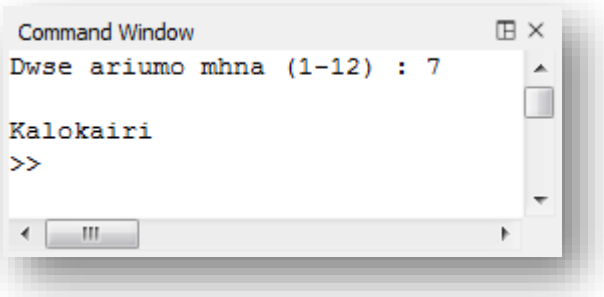
Η δομή είναι ιδιαίτερα χρήσιμη στην περίπτωση όπου δίνονται κάποια διαστήματα τιμών και θέλουμε να ελέγξουμε σε ποιο διάστημα εμπίπτει ο αριθμός που δίνει ο χρήστης. Στην περίπτωση αυτή εκμεταλλευόμαστε το γεγονός ότι σε κάθε έλεγχο συνθήκης `elseif` γνωρίζουμε ότι οι προηγούμενες συνθήκες είναι ψευδείς. Με αυτό τον τρόπο μπορούμε να καθορίσουμε διαδοχικά όρια των διαστημάτων όπως φαίνεται στο παρακάτω παράδειγμα.

Παράδειγμα 2-6 Έλεγχος εάν κάποιος αριθμός ανήκει σε κάποιο διάστημα τιμών.

Γράψτε ένα πρόγραμμα το οποίο να ζητά από τον χρήστη τον αριθμό ενός μήνα και έπειτα, να εμφανίζει την αντίστοιχη εποχή. Θεωρούμε ότι ο αριθμός που δίνει ο χρήστης είναι μεταξύ 1 και 12. Αποθηκεύστε το πρόγραμμα ως `example_2_06.m`



Τρέχοντας το πρόγραμμα, π.χ. για `x=7` (Ιούλιος) δίνει:



Παράδειγμα 2-7 Έλεγχος εάν κάποιος αριθμός ανήκει σε κάποιο διάστημα τιμών (έλεγχος με αντίστροφη σειρά)

Γράψτε ένα πρόγραμμα το οποίο να ζητά από τον χρήστη μια θερμοκρασία. Έπειτα να εμφανίζει ένα μήνυμα ανάλογα με τα παρακάτω διαστήματα τιμών:

Διάστημα τιμών	Μήνυμα
$x \geq 40$	Καύσωνας
$25 \leq x < 40$	Ζέστη
$15 \leq x < 25$	Καλός καιρός
$0 \leq x < 15$	Κρύο
$x < 0$	Παγετός

Editor

File Edit View Debug Run Help

example_2_07.m

```
1  clc;
2  clear;
3
4  % Ζητείται η θερμοκρασία από τον χρήστη
5  x=input('Dwse thermokrasia : ');
6
7  if x>=40 % είναι x>=40;
8    disp('Kayswnas');
9  elseif x>=25 % με δεδομένο ότι x<40, ρωτάμε: είναι το x>=25;
10   disp('Zesth');
11 elseif x>=15 % με δεδομένο ότι x<25, ρωτάμε: είναι το x>=15;
12   disp('Kalos kairos');
13 elseif x>=0 % με δεδομένο ότι x<15, ρωτάμε: είναι το x>=0;
14   disp('Kryo');
15 else % είναι δεδομένο ότι x<0
16   disp('Pagetos');
17 end
18
19 fprintf('\nTelos programmatos\n');
20
```

eol: CRLF line: 20 col: 1

Τρέχοντας το πρόγραμμα, π.χ. για `x=32` και `−6` δίνει, αντίστοιχα:

Command Window

```
Dwse thermokrasia : 32
Zesth

Telos programmatos
>>
```

Command Window

```
Dwse thermokrasia : -6
Pagetos

Telos programmatos
>>
```

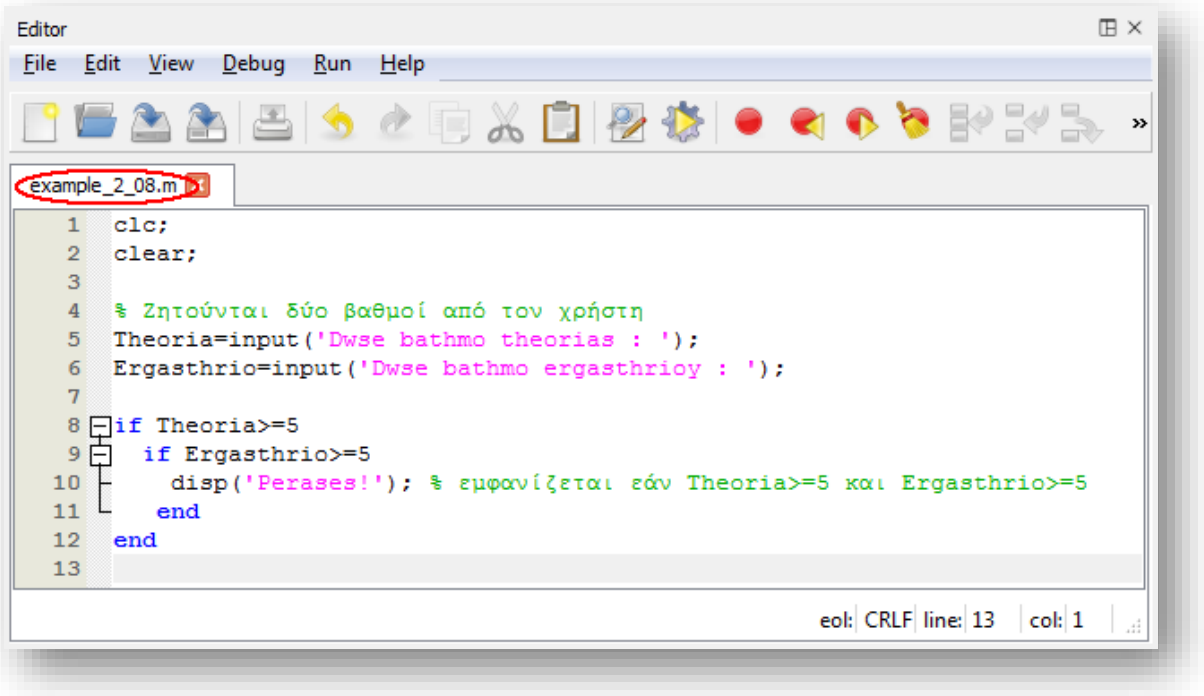
2.3 Εμφωλευμένες δομές ελέγχου

Μπορούν να χρησιμοποιηθούν δομές ελέγχου στο εσωτερικό άλλων δομών ελέγχου;

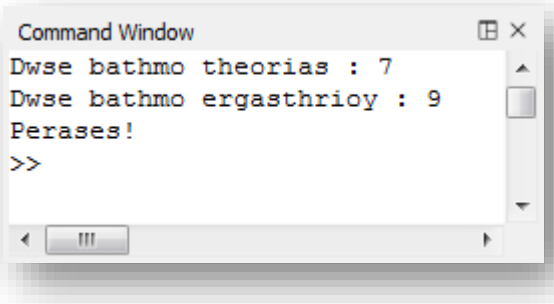
Ναι. Οι δομές αυτές ονομάζονται εμφωλευμένες (nested) και παρέχουν την δυνατότητα να ορίζονται σύνθετες δομές αναφορικά με την ροή του προγράμματος. Για παράδειγμα, μια δομή if-end εμφωλευμένη σε μια άλλη μπορεί να χρησιμοποιηθεί ώστε να οριστεί κώδικας που θα εκτελείται όταν ισχύουν ταυτόχρονα δύο λογικές συνθήκες (λογική πράξη γινομένου, βλέπε και παράγραφο 2.4).

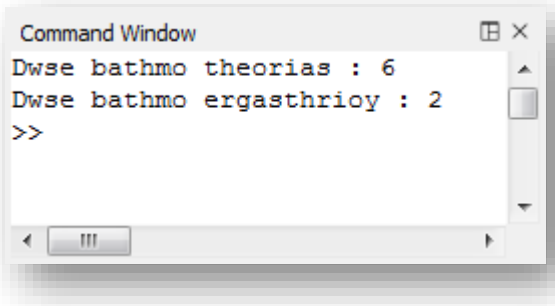
Παράδειγμα 2-8 Έλεγχος εάν ισχύουν δύο συνθήκες ταυτόχρονα

Γράψτε ένα πρόγραμμα το οποίο να ζητά από τον χρήστη την βαθμολογία της θεωρίας και του εργαστηρίου ενός μαθήματος. Έπειτα να εμφανίζει ένα μήνυμα μόνο εάν ο βαθμός είναι πάνω από 5 και στα δύο.



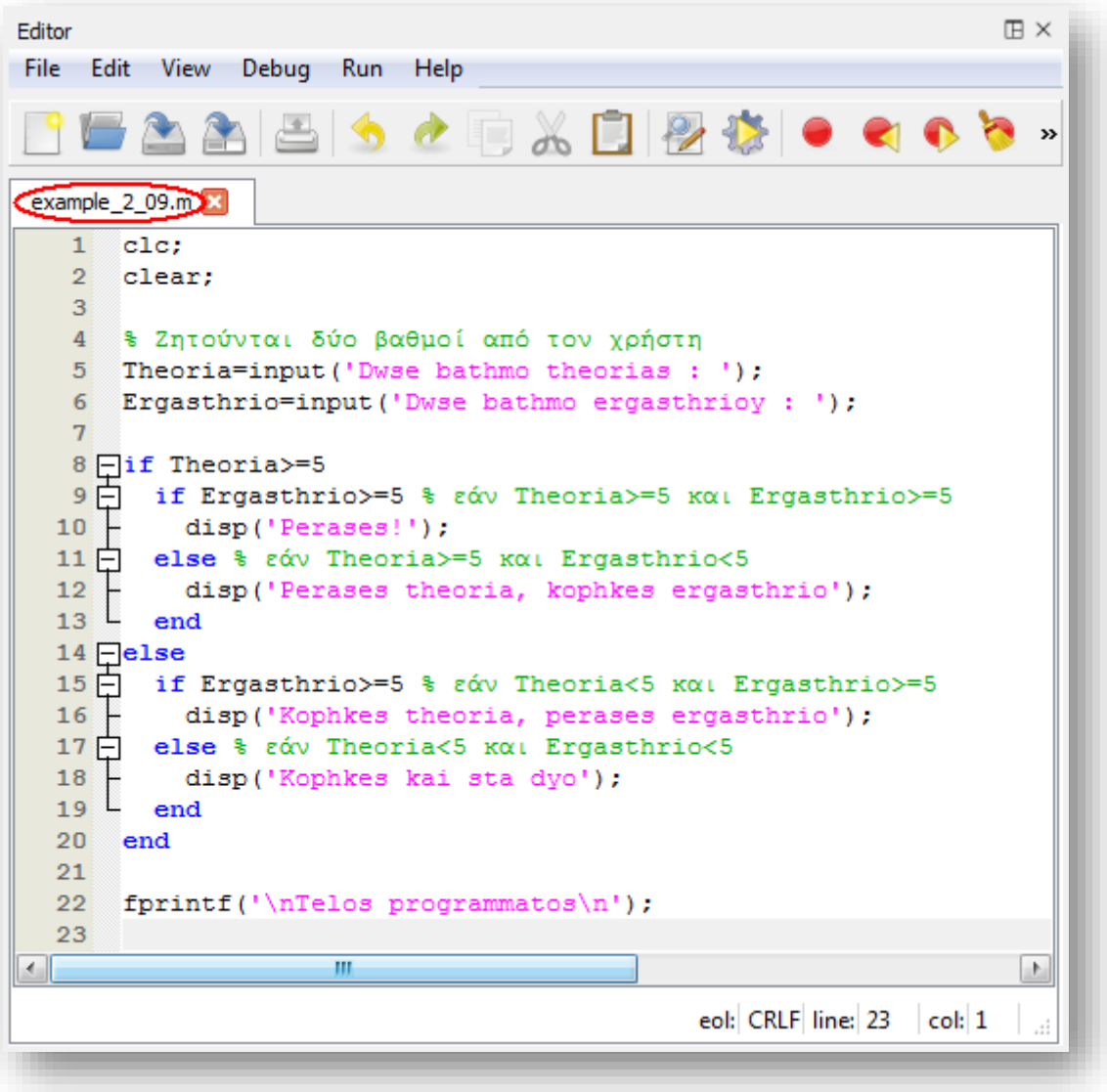
Τρέχοντας το πρόγραμμα, π.χ. για θεωρία 7 και εργαστήριο 9 εμφανίζει το μήνυμα ενώ για θεωρία 6 και εργαστήριο 2 δεν εμφανίζει τίποτα.



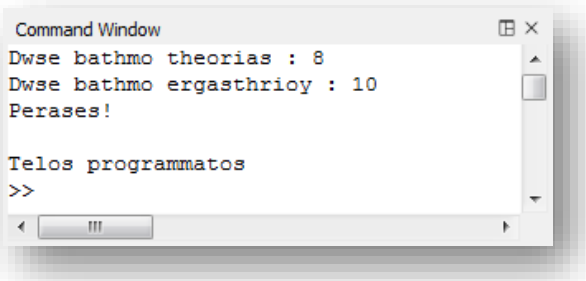


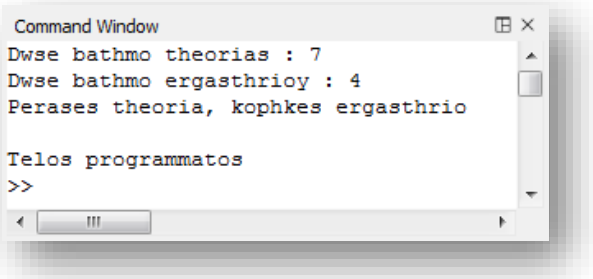
Παράδειγμα 2-9 Εξέταση όλων των περιπτώσεων ανάμεσα σε δύο λογικές συνθήκες

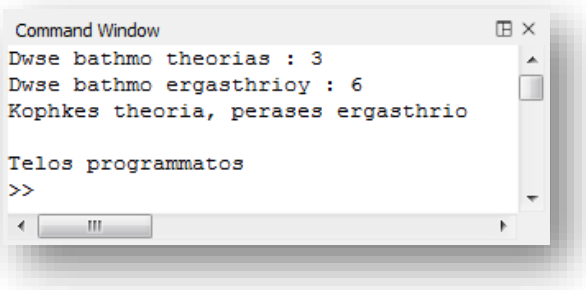
Τροποποιείστε τον κώδικα στο προηγούμενο Παράδειγμα έτσι ώστε να καλύπτει και τις τέσσερις διαφορετικές περιπτώσεις που μπορεί να τύχουν σε σχέση με τις δύο βαθμολογίες.

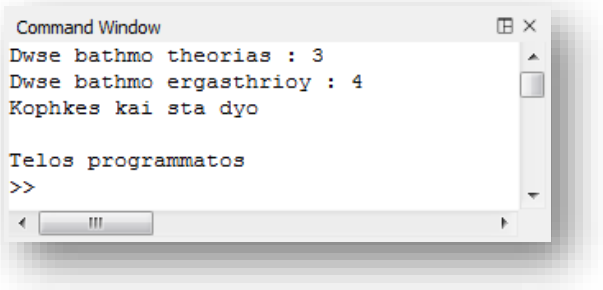


Τρέχοντας το πρόγραμμα για τέσσερις διαφορετικές περιπτώσεις δίνει:









2.4 Λογικές πράξεις

Μπορούμε να κάνουμε πράξεις μεταξύ λογικών παραστάσεων;

Ναι. Μπορούμε να συνδυάσουμε δύο ή περισσότερες λογικές παραστάσεις έτσι ώστε να παράγουμε ένα καινούργιο λογικό συμπέρασμα. Οι τρεις βασικές λογικές πράξεις είναι το λογικό γινόμενο (AND), το λογικό άθροισμα (OR) και η λογική άρνηση (NOT).

Λογικό γινόμενο - πράξη AND

Το λογικό γινόμενο είναι 1 (αληθές) όταν όλα τα ορίσματά του είναι 1 (αληθή). Ο πίνακας αληθείας της πράξης AND είναι:

AND - Λογικό γινόμενο – Σύμβολο &&

Όρισμα A	Όρισμα B	A && B	Παράδειγμα	Ερμηνεία
0 (ψευδές)	0 (ψευδές)	0 (ψευδές)	`>> (5>9) && (-3>10)` `ans = 0`	Είναι το 5 μεγαλύτερο από το 9 **και** το -3 μεγαλύτερο από 10; Όχι
0 (ψευδές)	1 (αληθές)	0 (ψευδές)	`>> (5>9) && (-3<10)` `ans = 0`	Είναι το 5 μεγαλύτερο από το 9 **και** το -3 μικρότερο από 10; Όχι
1 (αληθές)	0 (ψευδές)	0 (ψευδές)	`>> (5<9) && (-3>10)` `ans = 0`	Είναι το 5 μικρότερο από το 9 **και** το -3 μεγαλύτερο από 10; Όχι
1 (αληθές)	1 (αληθές)	1 (αληθές)	`>> (5<9) && (-3<10)` `ans = 1`	Είναι το 5 μικρότερο από το 9 **και** το -3 μικρότερο από 10; Ναι

Λογικό άθροισμα - πράξη OR

Το λογικό άθροισμα είναι 1 (αληθές) όταν τουλάχιστον ένα από τα ορίσματά του είναι 1 (αληθές). Ο πίνακας αληθείας της πράξης OR είναι:

OR - Λογικό άθροισμα – Σύμβολο ||

Όρισμα A	Όρισμα B	A || B	Παράδειγμα	Ερμηνεία
0 (ψευδές)	0 (ψευδές)	0 (ψευδές)	`>> (5>9) || (-3>10)` `ans = 0`	Είναι το 5 μεγαλύτερο από το 9 **ή** το -3 μεγαλύτερο από 10; Όχι
0 (ψευδές)	1 (αληθές)	1 (αληθές)	`>> (5>9) || (-3<10)` `ans = 1`	Είναι το 5 μεγαλύτερο από το 9 **ή** το -3 μικρότερο από 10; Ναι
1 (αληθές)	0 (ψευδές)	1 (αληθές)	`>> (5<9) || (-3>10)` `ans = 1`	Είναι το 5 μικρότερο από το 9 **ή** το -3 μεγαλύτερο από 10; Ναι
1 (αληθές)	1 (αληθές)	1 (αληθές)	`>> (5<9) || (-3<10)` `ans = 1`	Είναι το 5 μικρότερο από το 9 **ή** το -3 μικρότερο από 10; Ναι

Λογική άρνηση - πράξη NOT

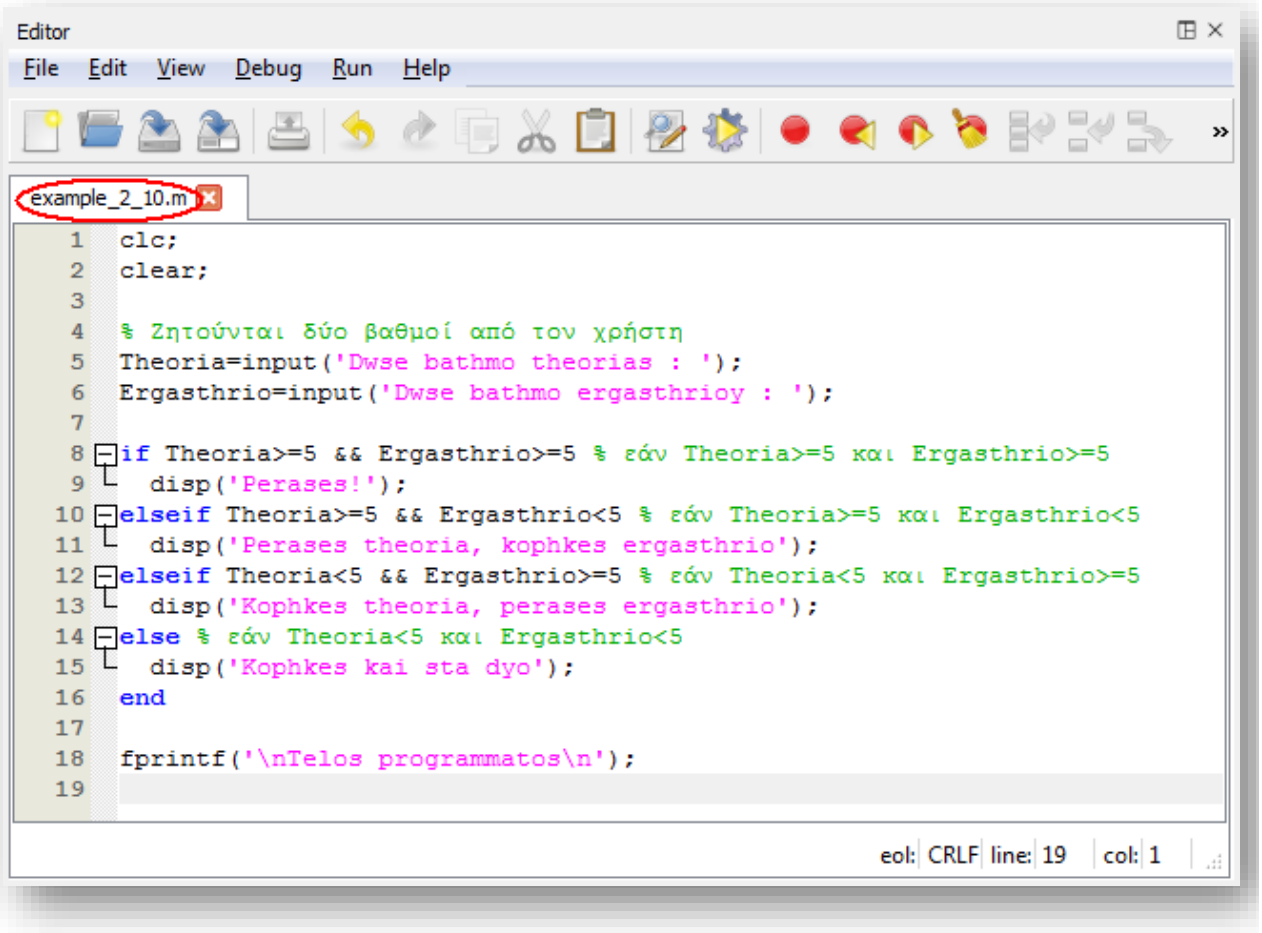
Η λογική άρνηση είναι 1 (αληθής) όταν το όρισμα είναι 0 (ψευδές) και αντιστρόφως. Ο πίνακας αληθείας της πράξης NOT είναι:

NOT – Λογική άρνηση – Σύμβολο ~

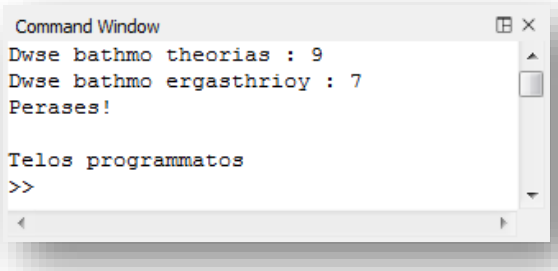
Όρισμα A	~A	Παράδειγμα	Ερμηνεία
0 (ψευδές)	1 (αληθές)	`>> ~(5>9)` `ans = 1`	**Δεν** είναι το 5 μεγαλύτερο από το 9 ; Ναι
1 (αληθές)	0 (ψευδές)	`>> ~(5<9)` `ans = 0`	**Δεν** είναι το 5 μικρότερο από το 9 ; Όχι

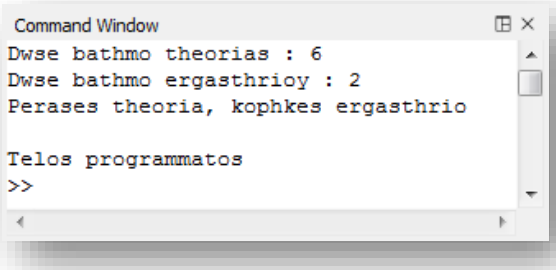
Παράδειγμα 2-10 Χρήση του λογικού γινομένου AND

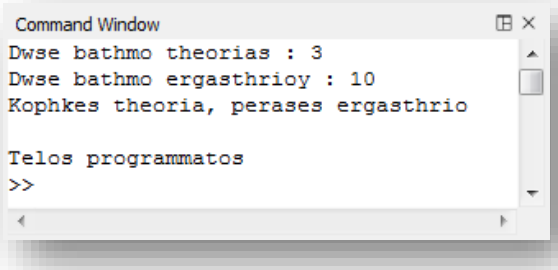
Τροποποιείστε τον κώδικα στο Παράδειγμα 2-9 χρησιμοποιώντας λογικό γινόμενο όπου χρειάζεται.

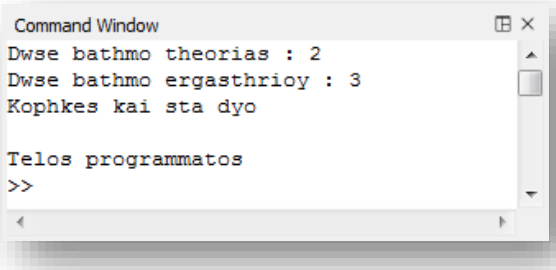


Τρέχοντας το πρόγραμμα για τέσσερις διαφορετικές περιπτώσεις δίνει:



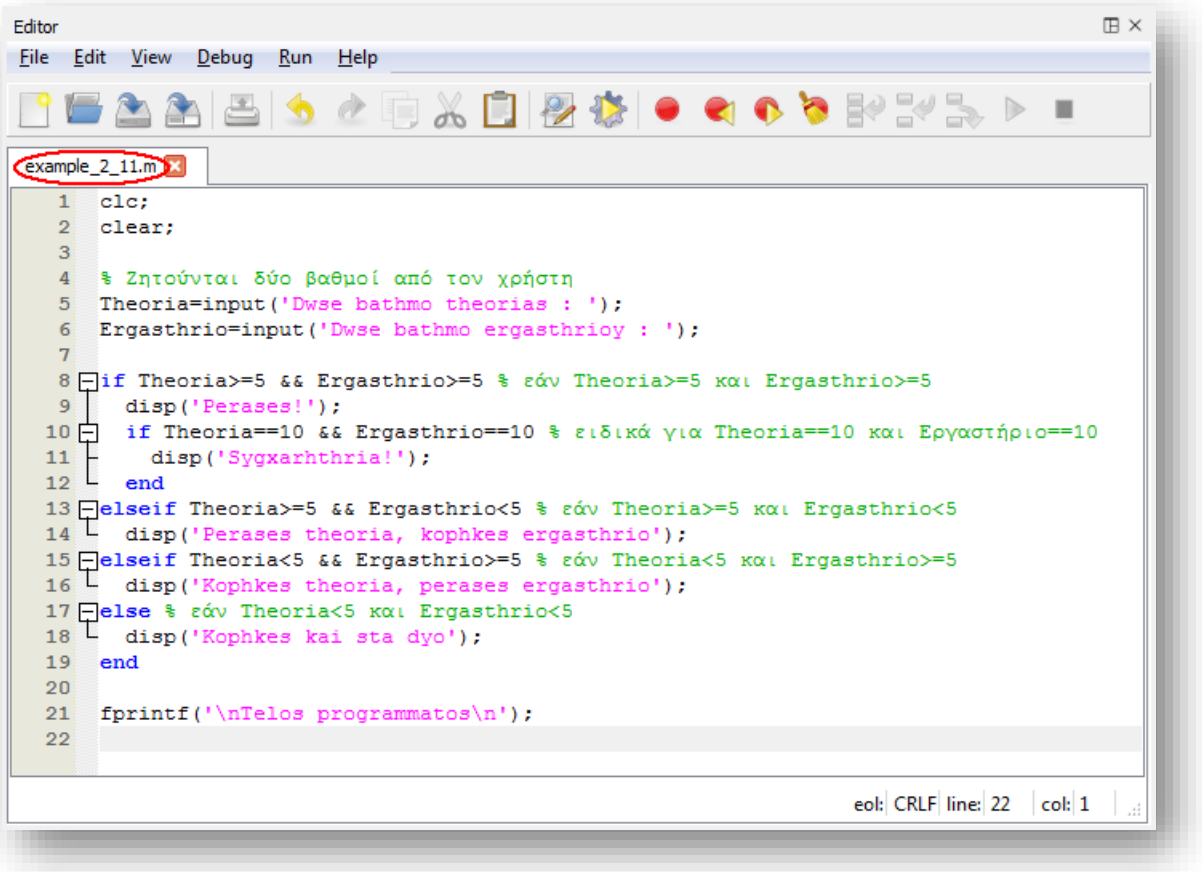




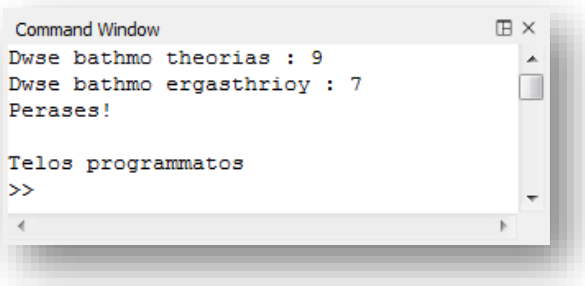


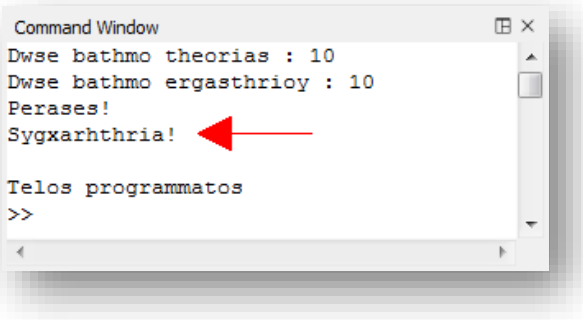
Παράδειγμα 2-11 Χρήση του λογικού γινομένου AND και σε εμφωλευμένο IF

Τροποποιείστε τον κώδικα στο προηγούμενο Παράδειγμα έτσι ώστε να εμφανίζει ένα μήνυμα "Συγχαρητήρια!" εάν και οι δύο βαθμοί είναι 10.



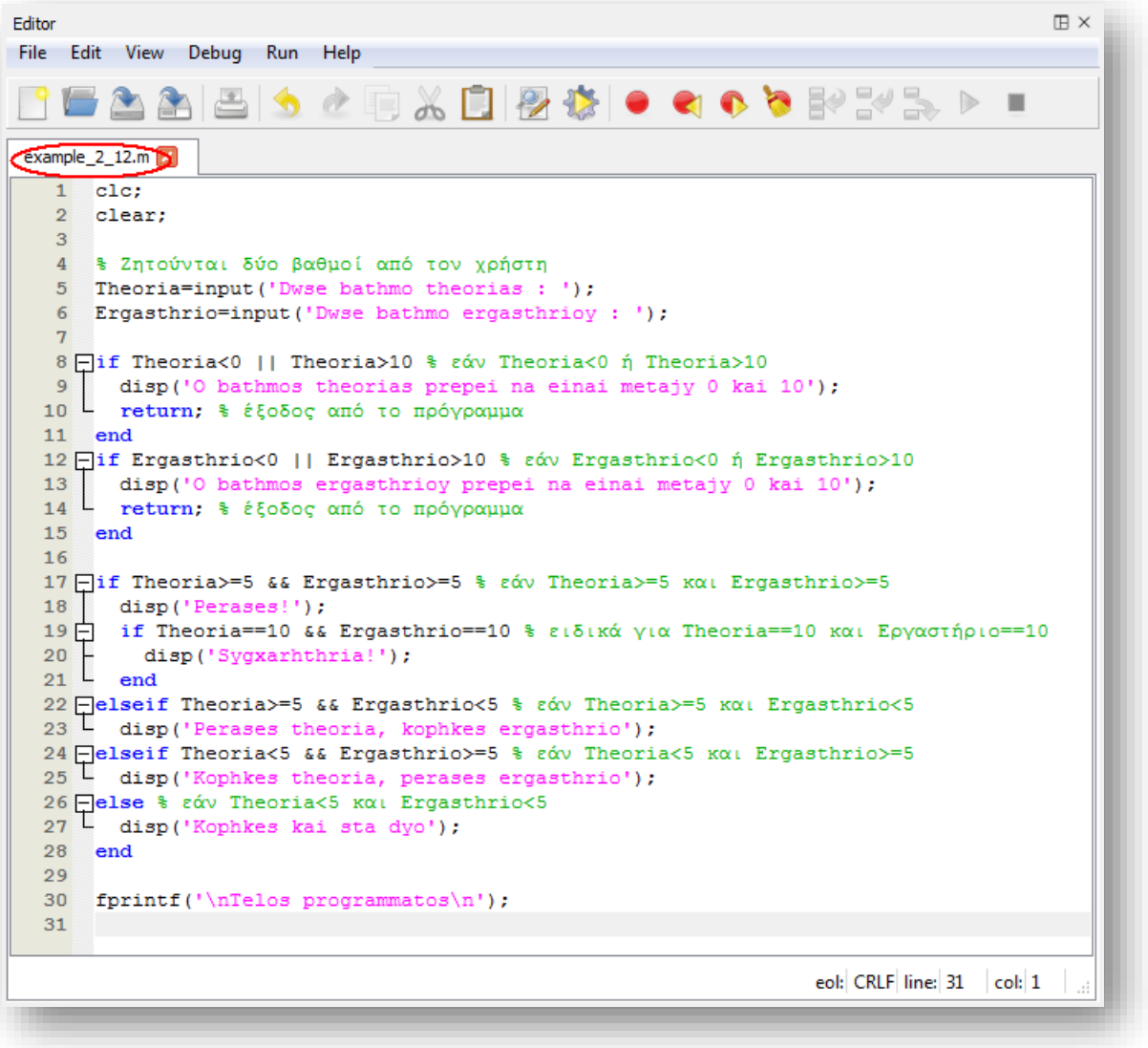
Στο πρόγραμμα έχει προστεθεί το if στις γραμμές 10-12 όπου εξετάζεται εάν και οι δύο βαθμολογίες είναι 10 και εμφανίζει το επιπλέον μήνυμα. Τρέχοντας το πρόγραμμα για δύο διαφορετικές περιπτώσεις δίνει:



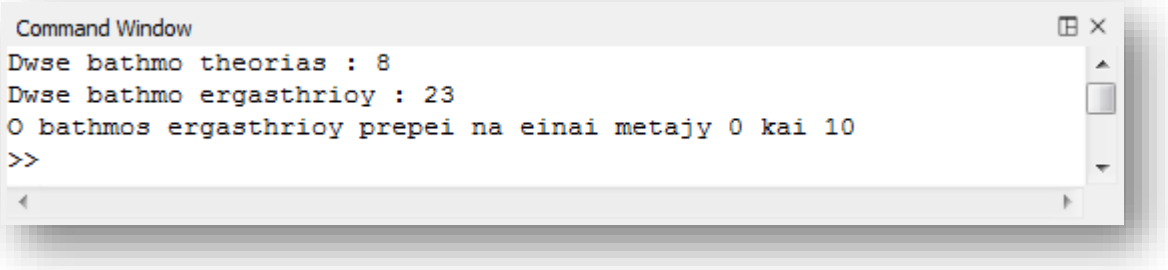


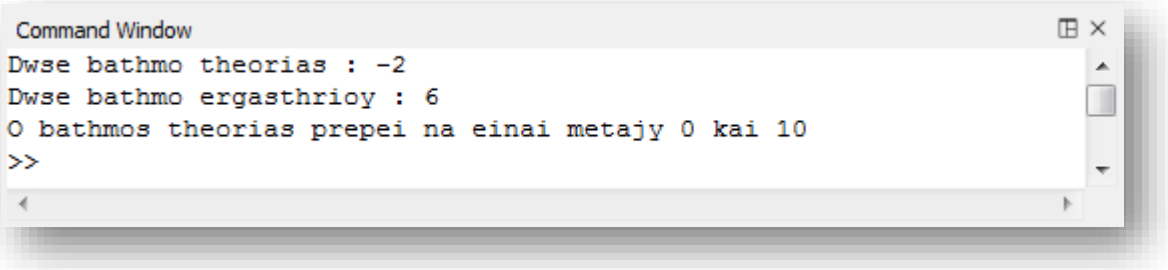
Παράδειγμα 2-12 Χρήση του λογικού αθροίσματος OR

Τροποποιείστε τον κώδικα στο Παράδειγμα 2-11 έτσι ώστε να ελέγχει εάν οι αριθμοί που έχει δώσει ο χρήστης είναι μεταξύ 0 και 10. Αν δεν συμβαίνει αυτό να εμφανίζει σχετικό μήνυμα και να τερματίζει το πρόγραμμα.



Στο πρόγραμμα έχουν προστεθεί οι γραμμές 8 έως 15 όπου γίνεται ο έλεγχος με δύο λογικά OR (σύμβολο || στις γραμμές 8 και 12, αντίστοιχα) για τις τιμές της θεωρίας και του εργαστηρίου, αντίστοιχα. Τρέχοντας το πρόγραμμα για λάθος δεδομένα δίνει:





Η εντολή `return` τερματίζει άμεσα την εκτέλεση του προγράμματος.

3 Βρόχοι επανάληψης

3.1 Γενικά

Ποια είναι η σκοπιμότητα χρήσης βρόχων επανάληψης;

Υπάρχουν περιπτώσεις όπου θέλουμε πραγματοποιηθούν κάποιοι υπολογισμοί για όλες τις τιμές ενός συνόλου δεδομένων ή όσο ισχύει κάποια συνθήκη. Σε αυτή την περίπτωση μπορεί να χρησιμοποιηθεί ένας βρόχος επανάληψης που θα εκτελείται είτε ένα προκαθορισμένο πλήθος φορών (βρόχοι τύπου `for…end`) ή όσο ισχύει κάποια συνθήκη (βρόχοι τύπου `while…end`).

3.2 Βρόχος `for…end`

Σε ποιες περιπτώσεις χρησιμοποιείται ο βρόχος for…end;

Ο βρόχος `for…end` χρησιμοποιείται όταν το πλήθος των επαναλήψεων είναι γνωστό εξ αρχής. Για παράδειγμα, εάν υπάρχουν N αριθμοί για τους οποίους θέλουμε να υπολογίσουμε π.χ. το άθροισμα ή την μέση τιμή ή την μικρότερη τιμή, τότε μπορεί να χρησιμοποιηθεί ένας βρόχος `for…end`.

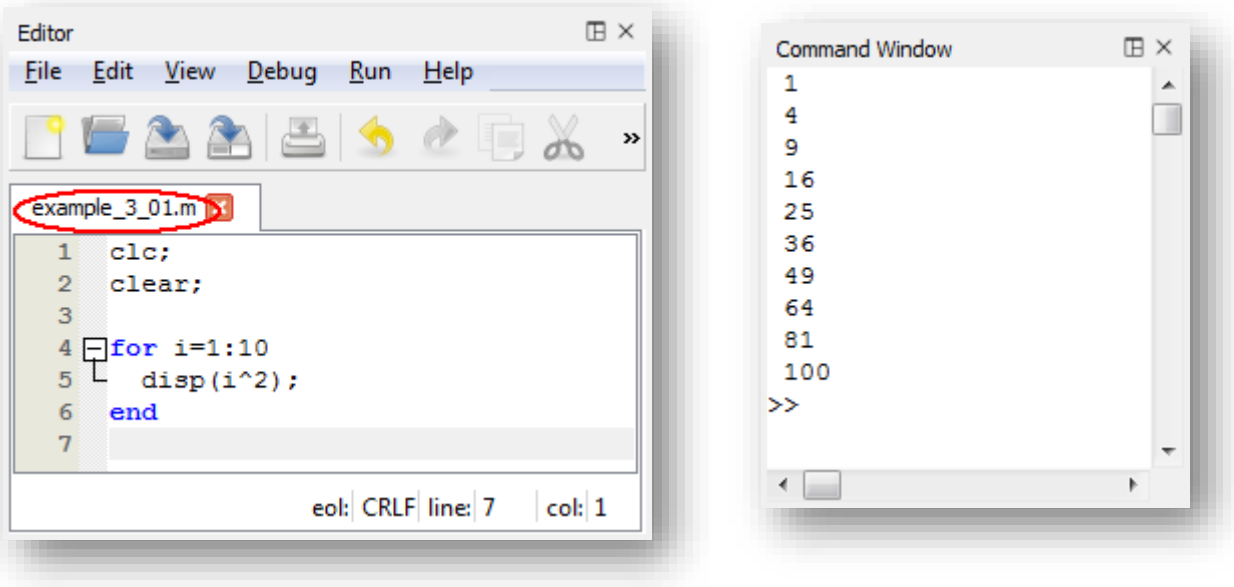
Οι εντολές που περιέχονται μέσα σε έναν βρόχο `for…end` ή `while…end` σκόπιμο είναι να έχουν εσοχή, δηλαδή να είναι γραμμές ένα tab πιο μέσα. Αυτό γίνεται ακόμη πιο αναγκαίο όταν υπάρχουν εμφωλευμένοι βρόχοι επανάληψης ο ένας μέσα στον άλλο.

Πώς δομείται ο βρόχος for…end;

Ο βρόχος περιλαμβάνει έναν μετρητή δίπλα από την εντολή `for` συνοδευόμενο από την αρχική και την τελική τιμή του και πιθανόν το βήμα αύξησης. Ο βρόχος ολοκληρώνεται με την εντολή `end`. Όλες οι εντολές που βρίσκονται μεταξύ του `for` και του `end` επαναλαμβάνονται όσες φορές καθορίζεται από την αρχική, την τελική τιμή και το βήμα του μετρητή. Για παράδειγμα, ο παρακάτω βρόχος έχει ως μετρητή την μεταβλητή `i` που παίρνει τιμές από `1` έως `10` και εμφανίζει τα τετράγωνα των αντίστοιχων αριθμών. Να σημειωθεί ότι ο μετρητής είναι συνήθως το `i` αλλά αυτό δεν είναι υποχρεωτικό. Οποιοδήποτε όνομα μεταβλητής μπορεί να χρησιμοποιηθεί ως μετρητής, όπως `i`, `j`, `itter`, κ.α.

Παράδειγμα 3-1 Βασική μορφή βρόχου for…end

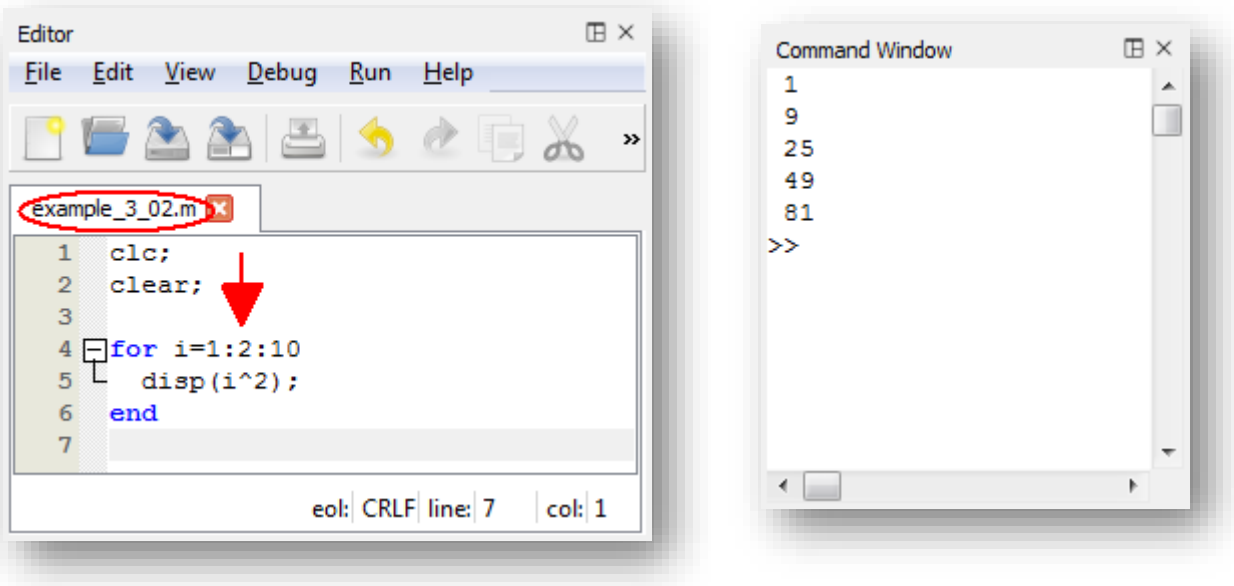
Γράψτε ένα πρόγραμμα που θα εμφανίζει τα τετράγωνα των αριθμών από 1 έως 10.



Στο προηγούμενο παράδειγμα υπονοείται ότι το βήμα αύξησης του μετρητή είναι 1. Μπορούμε όμως να ορίσουμε και διαφορετικό βήμα προσθέτοντάς το ενδιάμεσα από την αρχική και τελική τιμή του μετρητή. Στο παρακάτω παράδειγμα, ο μετρητής `i` παίρνει τιμές `1`, `3`, `5`, `7`, `9` και συνεπώς εμφανίζονται τα τετράγωνα μόνο αυτών των αριθμών:

Παράδειγμα 3-2 Βήμα μετρητή σε βρόχο for…end

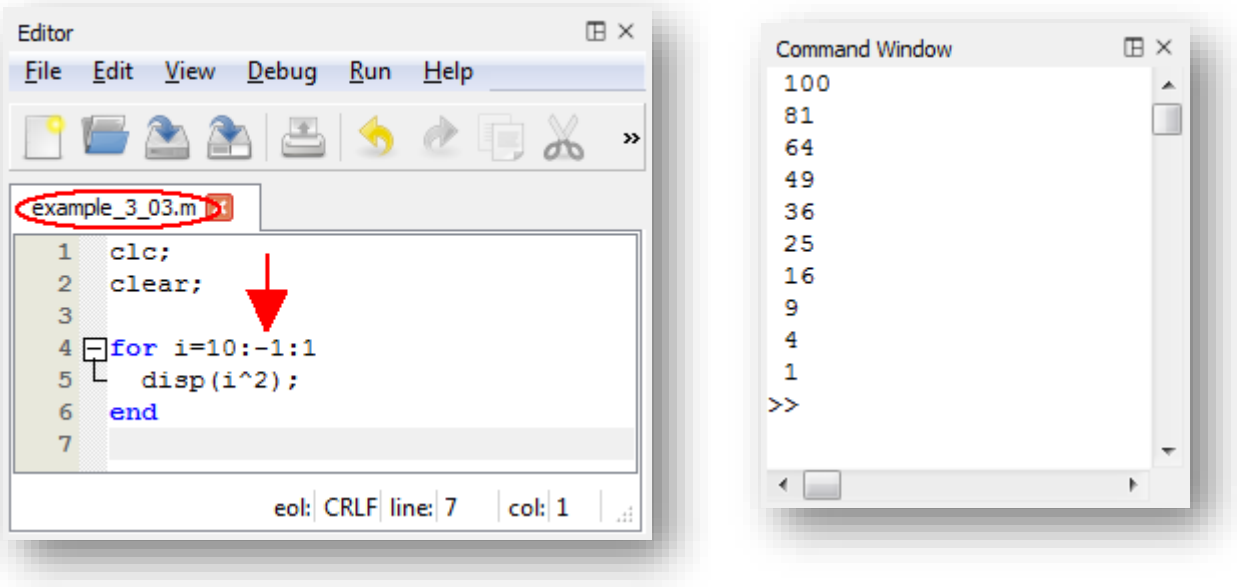
Γράψτε ένα πρόγραμμα που θα εμφανίζει τα τετράγωνα μόνο των περιττών των αριθμών από 1 έως 10.



Το βήμα μπορεί να χρησιμοποιηθεί και για να έχουμε φθίνουσα αρίθμηση του μετρητή. Στο παρακάτω παράδειγμα ο μετρητής `i` παίρνει τιμές από `10` έως `1`:

Παράδειγμα 3-3 Φθίνουσα αρίθμηση μετρητή σε βρόχο for…end

Γράψτε ένα πρόγραμμα που θα εμφανίζει τα τετράγωνα των αριθμών από 10 έως 1.

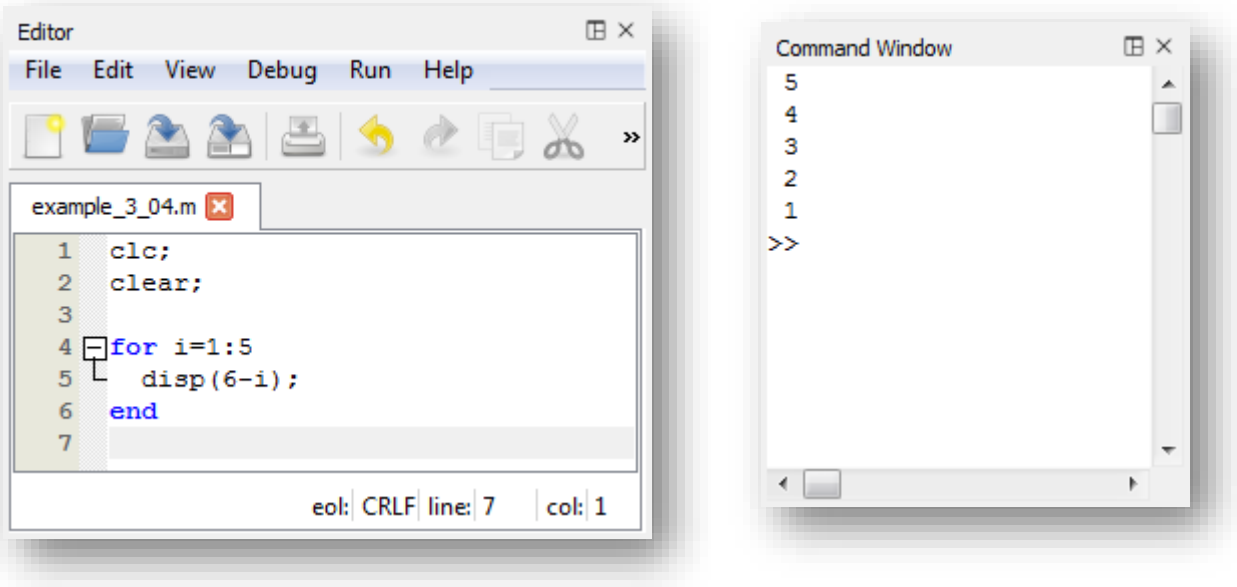


Πώς μπορεί να χρησιμοποιηθεί ο μετρητής μέσα στον βρόχο;

Συνήθως ο μετρητής `i` χρησιμεύει σε συνδυασμό με πίνακες ή διανύσματα για τον προσδιορισμό του `i`-οστού του στοιχείου. Σε άλλες περιπτώσεις, εντός του βρόχου χρησιμοποιείται μια συνάρτηση ως προς το `i`, όπως π.χ. η ύψωση σε δύναμη στα προηγούμενα τρία παραδείγματα. Είναι σημαντικό να τονιστεί ότι η συνάρτηση αυτή μπορεί να έχει οποιαδήποτε μορφή. Για παράδειγμα, μπορεί ο μετρητής `i` να αυξάνεται αλλά να εμφανίζονται αριθμοί οι αριθμοί σε φθίνουσα σειρά, όπως στο παρακάτω παράδειγμα όπου ενώ ο μετρητή λαμβάνει τιμές από 1 έως 5 στο Command Window εμφανίζονται οι τιμές με ανάστροφη σειρά.

Παράδειγμα 3-4 χρήση του μετρητή εντός του βρόχου for…end

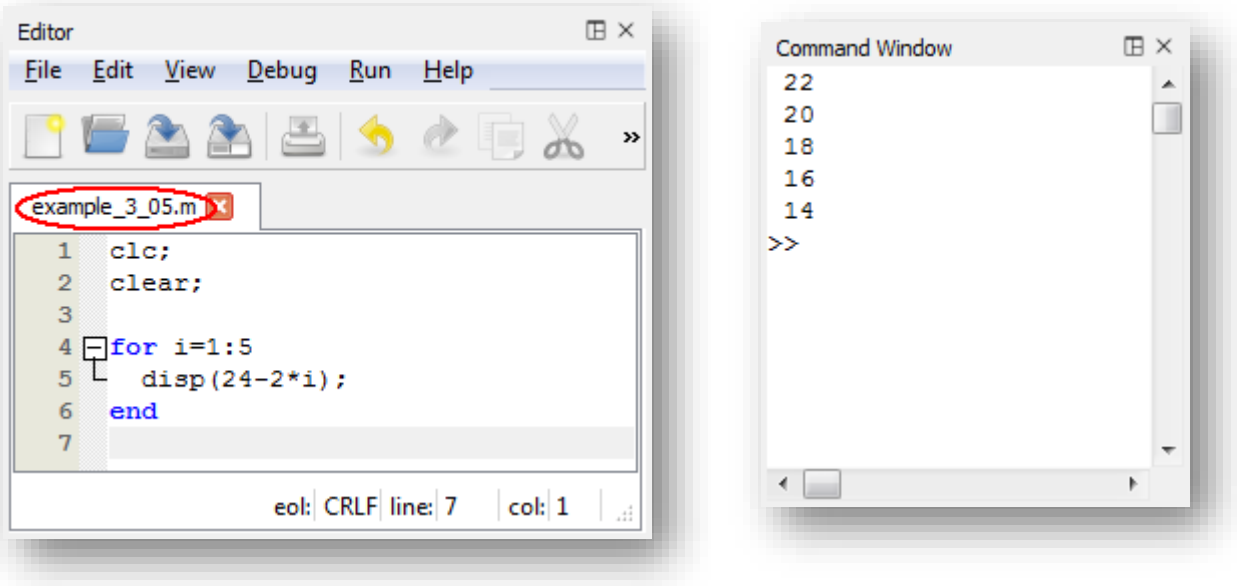
Γράψτε ένα πρόγραμμα στο οποίο ενώ ο μετρητής θα αυξάνει από 1 έως 5 στην οθόνη θα εμφανίζει τους αριθμούς από 5 έως 1.



Η συνάρτηση μπορεί να έχει και πιο σύνθετη μορφή όπως φαίνεται στο επόμενο παράδειγμα όπου χωρίς να πειράξουμε την γραμμή 4 εμφανίζουμε τις τιμές `22, 20, 18, 16, 14`.

Παράδειγμα 3-5 Συνάρτηση με βάση τον μετρητή εντός του βρόχου for…end

Τροποποιήστε τον κώδικα στο Παράδειγμα 3-4 έτσι ώστε να εμφανίζονται οι αριθμοί `22, 20, 18, 16, 14` χωρίς να αλλάξετε την γραμμή 4.



3.3 Βρόχος `while…end`

Σε ποιες περιπτώσεις χρησιμοποιείται ο βρόχος while…end;

Ο βρόχος `while…end` χρησιμοποιείται όταν το πλήθος των επαναλήψεων δεν είναι γνωστό εξ αρχής ή δεν μας ενδιαφέρει. Αντίθετα, αυτό που καθορίζει το πόσες επαναλήψεις θα γίνουν είναι το εάν ισχύει κάποια λογική

συνθήκη που αναγράφεται στα δεξιά του `while`. Όσο η συνθήκη αυτή είναι αληθής οι επαναλήψεις επαναλαμβάνονται. Μόλις η συνθήκη πάψει να ισχύει η ροή του προγράμματος συνεχίζει κάτω από το `end` του βρόχου με τις επόμενες εντολές που πιθανόν υπάρχουν.

Έχει ο βρόχος while…end κάποιες ιδιαιτερότητες σχετικά με τον for…end;

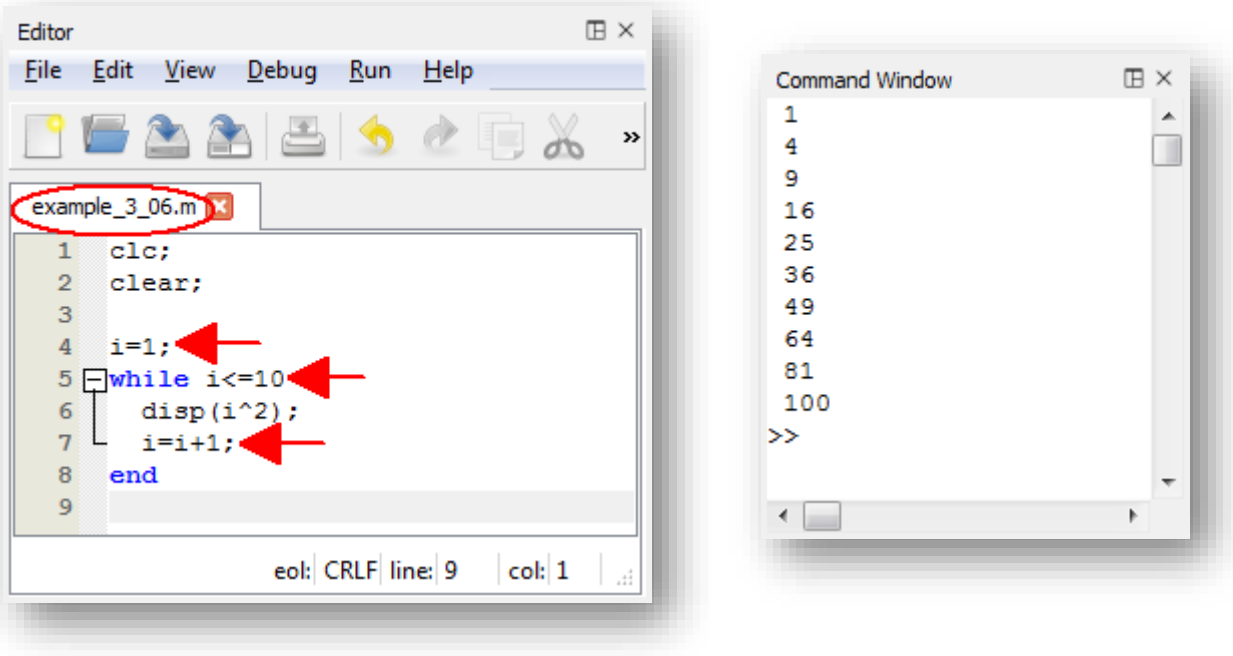
Εφόσον ο βρόχος while…end εκτελείται όσο ισχύει η συνθήκη θα πρέπει να υπάρχει η δυνατότητα σε κάποια από τις επαναλήψεις η συνθήκη να γίνει ψευδής ώστε να σταματήσει ο βρόχος και η ροή του προγράμματος να συνεχιστεί με τις επόμενες εντολές. Σε αντίθετη περίπτωση θα έχουμε έναν ατέρμονο βρόχο που θα έχει σαν αποτέλεσμα το πρόγραμμα να κολλήσει.

Πώς μπορούμε να γράψουμε τον μετρητή του Παραδείγματος 3-1 χρησιμοποιώντας βρόχο while…end;

Ενώ ο βρόχος for…end φροντίζει μόνος του να αυξάνει τον μετρητή κατά το επιθυμητό βήμα, στην περίπτωση του βρόχου while…end αυτό είναι ευθύνη του προγραμματιστή και δίνονται με διαφορετικό τρόπο. Θα πρέπει να φροντίσουμε να δώσουμε ξεχωριστή εντολή για την αρχική τιμή του μετρητή, να ορίσουμε την συνθήκη τερματισμού με βάση τον μετρητή και να δώσουμε ξεχωριστή εντολή για την αύξηση του μετρητή, όπως φαίνεται στο παρακάτω Παράδειγμα.

Παράδειγμα 3-6 Βασική μορφή βρόχου while…end

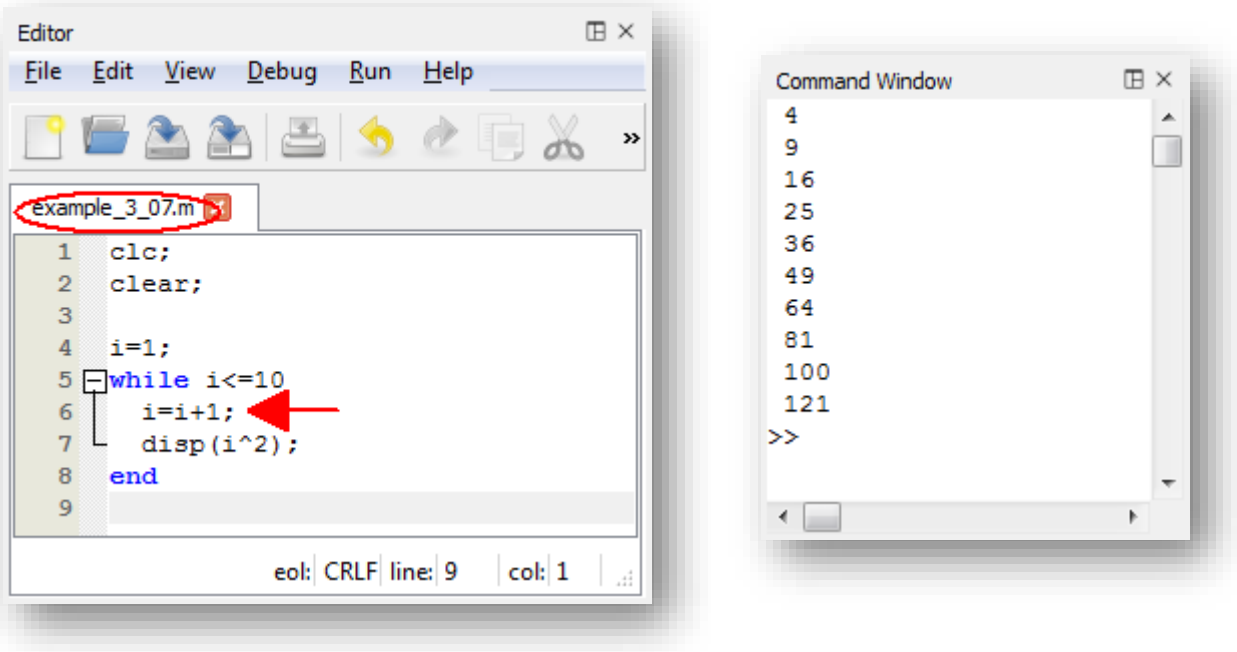
Γράψτε ένα πρόγραμμα που θα εμφανίζει τα τετράγωνα των αριθμών από 1 έως 10.



Είναι σημαντικό να τονιστεί ότι έχει σημασία το πού βάζουμε την εντολή αύξησης του μετρητή. Για παράδειγμα, εάν στο προηγούμενο παράδειγμα εναλλάξουμε τις εντολές στις γραμμές 6 και 7, δηλαδή πρώτα να αυξάνει τον μετρητή και έπειτα να εμφανίζει το τετράγωνό του τότε το αποτέλεσμα που λαμβάνουμε είναι διαφορετικό.

Παράδειγμα 3-7 Σημασία της σειράς εντολών στον βρόχο while…end

Τροποποιείστε τον κώδικα στο προηγούμενο Παράδειγμα εναλλάσσοντας τις γραμμές 6 και 7.



Παρατηρούμε ότι, καθώς πρώτα αυξάνεται ο μετρητής και μετά χρησιμοποιείται για την εμφάνιση του τετραγώνου του, αυτό έχει ως αποτέλεσμα να εμφανίζονται τα τετράγωνα των αριθμών από 2 έως 11.

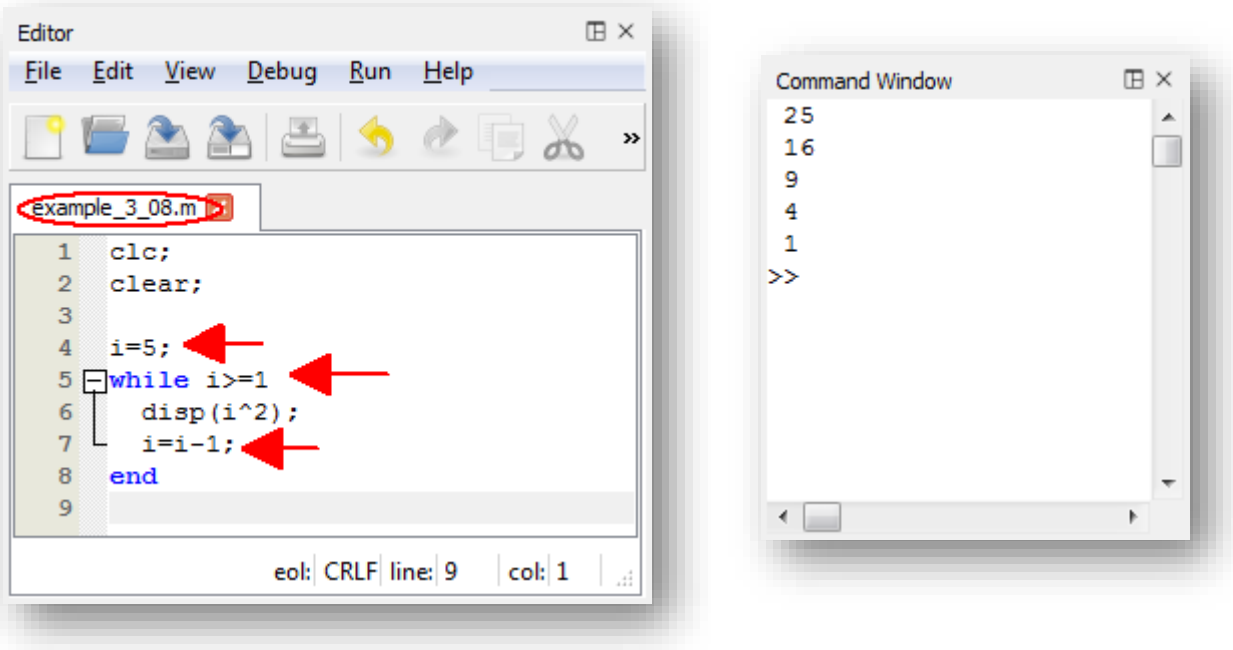
Στα παραπάνω δύο παραδείγματα εάν έλειπε η εντολή `i=i+1` που αυξάνει τον μετρητή τότε θα είχαμε έναν ατέρμονο βρόχο και το πρόγραμμα θα "κολλούσε". Πράγματι, στην περίπτωση αυτή, το `i` διατηρεί συνέχεια την αρχική τιμή 1 (που του δώσαμε στην γραμμή 4) η οποία όμως δεν μεταβάλλεται πουθενά παρακάτω, με αποτέλεσμα η συνθήκη i<=10 να ισχύει συνεχώς, χωρίς να υπάρχει κάποια προοπτική να γίνει ψευδής και έτσι η ροή του προγράμματος κυκλοφορεί συνεχώς μέσα στον βρόχο `while…end` χωρίς να μπορεί το πρόγραμμα να συνεχίσει παρακάτω.

Ένα "κολλήσει" το πρόγραμμα σε έναν ατέρμονο βρόχο τότε κάνουμε κλικ στο Command Window και πατάμε επανειλημμένα CTRL+C (όπως το Copy) ώστε να τερματιστεί άμεσα η εκτέλεση του προγράμματος.

Είναι προφανές, ότι στους βρόχους while…end το βήμα καθορίζεται στην εντολή αύξησης του μετρητή. Για παράδειγμα, εάν θέλουμε να αυξάνεται ο μετρητής κατά 2 τότε η εντολή πρέπει να γίνει `i=i+2`. Η ίδια εντολή μπορεί να χρησιμοποιηθεί και για να έχουμε φθίνουσα αρίθμηση του μετρητή. Στο παρακάτω παράδειγμα ο μετρητής `i` παίρνει τιμές από `5` έως `1` και εμφανίζει τα αντίστοιχα τετράγωνα. Προσέξτε ότι η αρχική τιμή του i είναι 5, ότι η ανισότητα έχει αλλάξει φορά και ότι ο μετρητής μειώνεται κατά 1.

Παράδειγμα 3-8 Φθίνουσα αρίθμηση μετρητή σε βρόχο while…end

Γράψτε ένα πρόγραμμα που θα εμφανίζει τα τετράγωνα των αριθμών από 5 έως 1.



3.4 Εντολή `break`

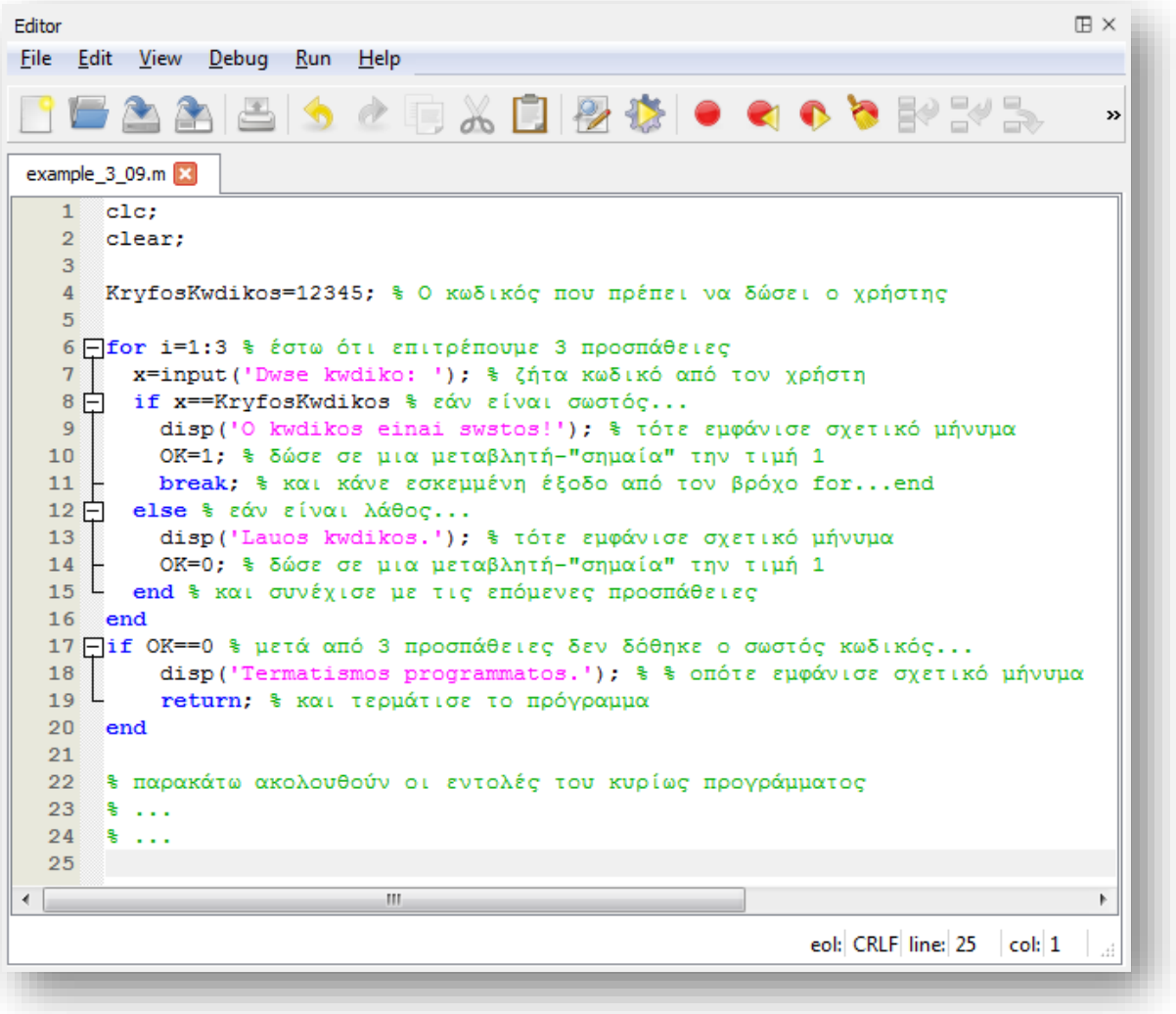
Τι είναι η εντολή break και πώς σχετίζεται με τους βρόχους επανάληψης;

Υπάρχουν περιπτώσεις όπου θέλουμε να έχουμε εσκεμμένη έξοδο από ένα βρόχο επανάληψης πριν ολοκληρωθούν όλες οι επαναλήψεις του (σε βρόχους `for…end`) ή γίνει η συνθήκη ψευδής (σε βρόχους `while…end`). Σε αυτή την περίπτωση χρησιμοποιείται η εντολή `break` συνήθως σε συνδυασμό με κάποιο if το οποίο καθορίζει και το πότε θα γίνει η εσκεμμένη έξοδος.

Ένα παράδειγμα χρήσης της εντολής `break` σε βρόχους `for…end` είναι για όταν το πρόγραμμα ζητά από τον χρήστη μια συγκεκριμένη τιμή (π.χ. κωδικό εισόδου) για να καθορισμένο πλήθος επαναλήψεων (π.χ. τρεις προσπάθειες).

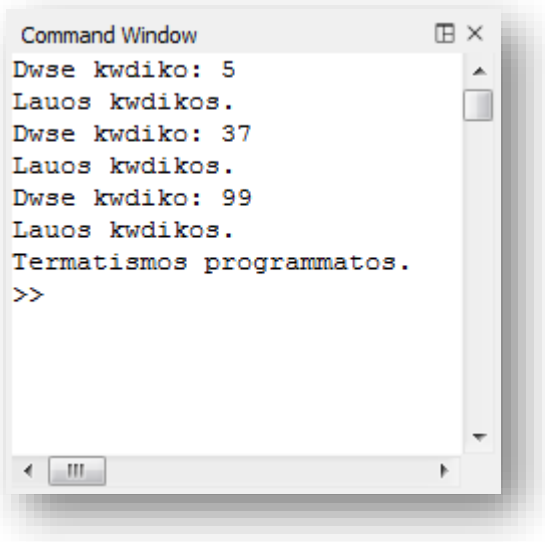
Παράδειγμα 3-9 Χρήση της εντολής break σε βρόχο for…end

Γράψτε ένα πρόγραμμα το οποίο θα ζητά από τον χρήστη έναν κωδικό εισόδου. Θα επιτρέπει τρεις προσπάθειες και εάν δεν είναι επιτυχείς τότε το πρόγραμμα θα τερματίζεται αλλιώς θα συνεχίζει παρακάτω.

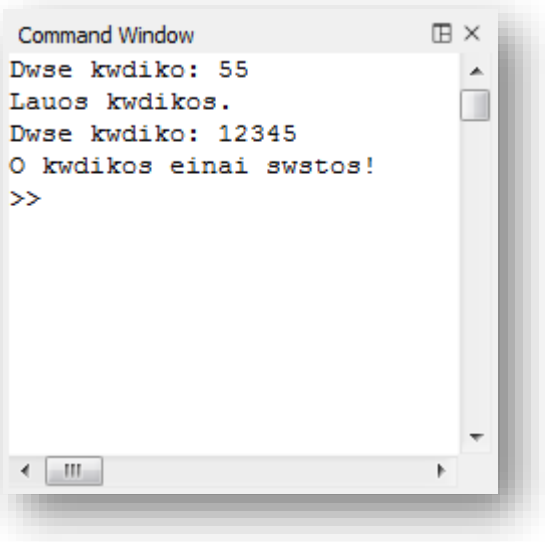


Η λογική του προγράμματος είναι η εξής:
Ο βρόχος `for…end` πρόκειται να εκτελεστεί 3 φορές. Κάθε φορά ζητάει έναν κωδικό. Εάν ο κωδικός είναι ο σωστός, τότε η "σημαία" `OK` παίρνει την τιμή `1` και με την εντολή `break` βγαίνει εσκεμμένα από τον βρόχο (πριν ολοκληρωθούν όλες οι επαναλήψεις). Στην γραμμή 17, αφού έχει τελειώσει ο βρόχος (είτε εσκεμμένα με σωστό κωδικό, είτε αφού περάσουν τρεις αποτυχημένες προσπάθειες) το `if` ελέγχει εάν η "σημαία" `OK` έχει την τιμή `1`. Εάν ναι, τότε σημαίνει ότι ο κωδικός δόθηκε σωστά (κάποια από τις τρεις φορές). Αλλιώς, το `OK` είναι `0` που σημαίνει ότι και οι τρεις προσπάθειες ήταν αποτυχημένες. Εάν λοιπόν στην γραμμή 17 το `OK==0` σημαίνει ότι πρέπει να τερματιστεί το πρόγραμμα καθώς η ταυτοποίηση απέτυχε. Αλλιώς, δηλαδή εάν το OK είναι ΜΗ μηδενικό, τότε ο χρήστης έχει ταυτοποιηθεί και το πρόγραμμα θα συνεχίσει κανονικά στις γραμμές 21 και κάτω.

Τρέχοντας το πρόγραμμα, για δύο διαφορετικές περιπτώσεις δίνει:



Τρεις αποτυχημένες προσπάθειες



Σωστός κωδικός

Η εντολή `break` μπορεί να χρησιμοποιηθεί και σε βρόχους `while…end` σε περιπτώσεις όπου το πρόγραμμα ζητά από τον χρήστη να διαλέξει μέσα από ένα σύνολο επιτρεπτών επιλογών. Εάν ο χρήστης δεν επιλέξει κάποια από τις διαθέσιμες επιλογές τότε πρόγραμμα ξαναρωτά τον χρήστη, αλλιώς συνεχίζει παρακάτω. Σε αυτή την περίπτωση η συνθήκη για το `while` ορίζεται `1` δηλαδή να είναι πάντοτε αληθής . Αυτό μοιάζει να αντιβαίνει στην λογική του βρόχου `while…end` όμως το σκεπτικό είναι: κάνε συνεχώς επαναλήψεις στον βρόχο και βγες μέσω της `break` όταν ικανοποιηθεί κάποιο `if` μέσα στον βρόχο.

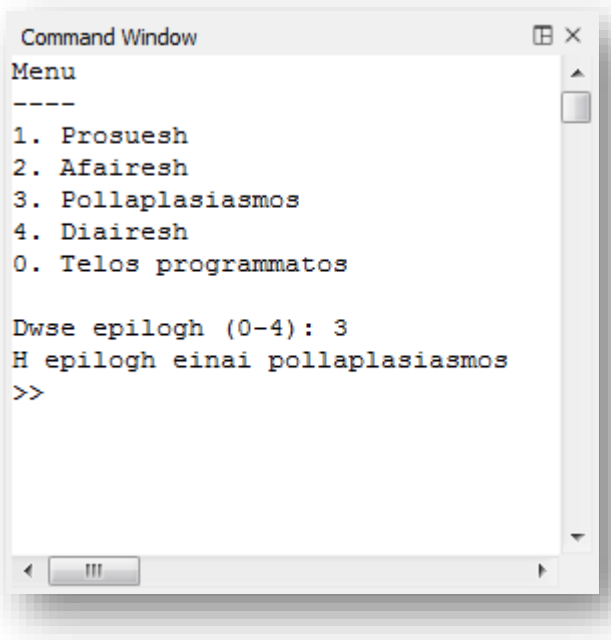
Παράδειγμα 3-10 Χρήση της εντολής break σε βρόχο while…end

Γράψτε ένα πρόγραμμα το οποίο θα εμφανίζει στον χρήστη ένα μενού όπου θα πρέπει να επιλέξει μια από τις τέσσερις αριθμητικές πράξεις. Σε περίπτωση που ο χρήστης δώσει λάθος επιλογή το πρόγραμμα θα τον ξαναρωτά, αλλιώς θα συνεχίζει παρακάτω.

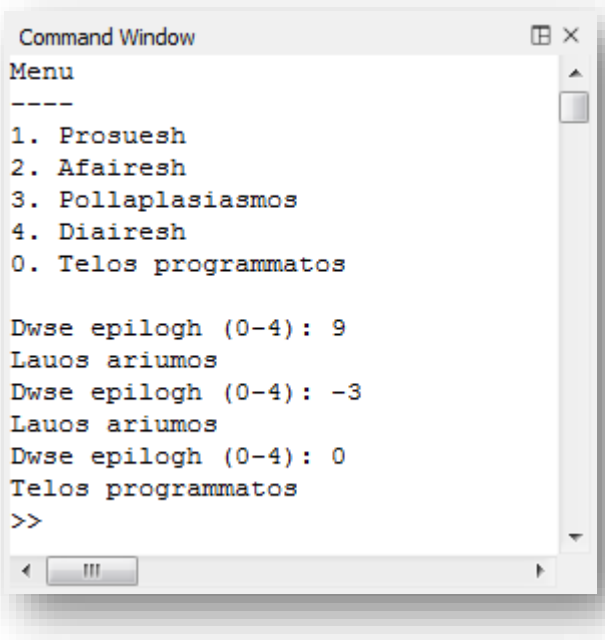
```
1  clc;
2  clear;
3
4  disp('Menu'); % εμφάνισε ένα μενού για τις 4 πράξεις και για έξοδο
5  disp('----');
6  disp('1. Prosuesh');
7  disp('2. Afairesh');
8  disp('3. Pollaplasiasmos');
9  disp('4. Diairesh');
10 disp('0. Telos programmatos');
11 disp(' ');
12 while 1 % ζήτα συνεχώς από τον χρήστη μια επιλογή
13   fprintf('Dwse epilogh (0-4): ');
14   x=input('');
15   if x==1 || x==2 || x==3 || x==4 ||x==0 % εάν είναι μια από τις
επιτρεπτές...
16     break; % τότε βγες από τον βρόχο
17   else
18     disp('Lauos ariumos'); % αλλιώς εμφάνισε σχετικό μήνυμα...
19   end
20 end % και συνέχισε να ζητάς από τον χρήστη μια επιλογή
21
22 if x==0 % ο χρήστης επέλεξε 0 οπότε
23   disp('Telos programmatos');
24   return; % τερματίζεται το πρόγραμμα
25 elseif x==1
26   disp('H epilogh einai prosuesh');
```

```
27    % εντολές για την πρόσθεση
28    % ...
29  elseif x==2
30    disp('H epilogh einai afairesh');
31    % εντολές για την αφαίρεση
32    % ...
33  elseif x==3
34    disp('H epilogh einai pollaplasiasmos');
35    % εντολές για τον πολλαπλασιασμό
36    % ...
37  elseif x==4
38    disp('H epilogh einai diairesh');
39    % εντολές για την διαίρεση
40    % ...
41  end
```

Τρέχοντας το πρόγραμμα, για δύο διαφορετικές περιπτώσεις δίνει:



Επιλογή 3



Λανθασμένες επιλογές και έξοδος

? *Στην γραμμή 15 του κώδικα γιατί να μην γράψουμε απλά* `if x>=0 && x<=4;`

! Διότι σε αυτή την περίπτωση το `if` θα είναι αληθές και για απαντήσεις που δεν είναι σωστές όπως π.χ. εάν ο χρήστης έδινε 3.5 ή 1.99. Συνεπώς πρέπει στο `if` να δηλώσουμε επακριβώς ποιες είναι οι αποδεκτές επιλογές.

3.5 Παραδείγματα χρήσης των βρόχων επανάληψης

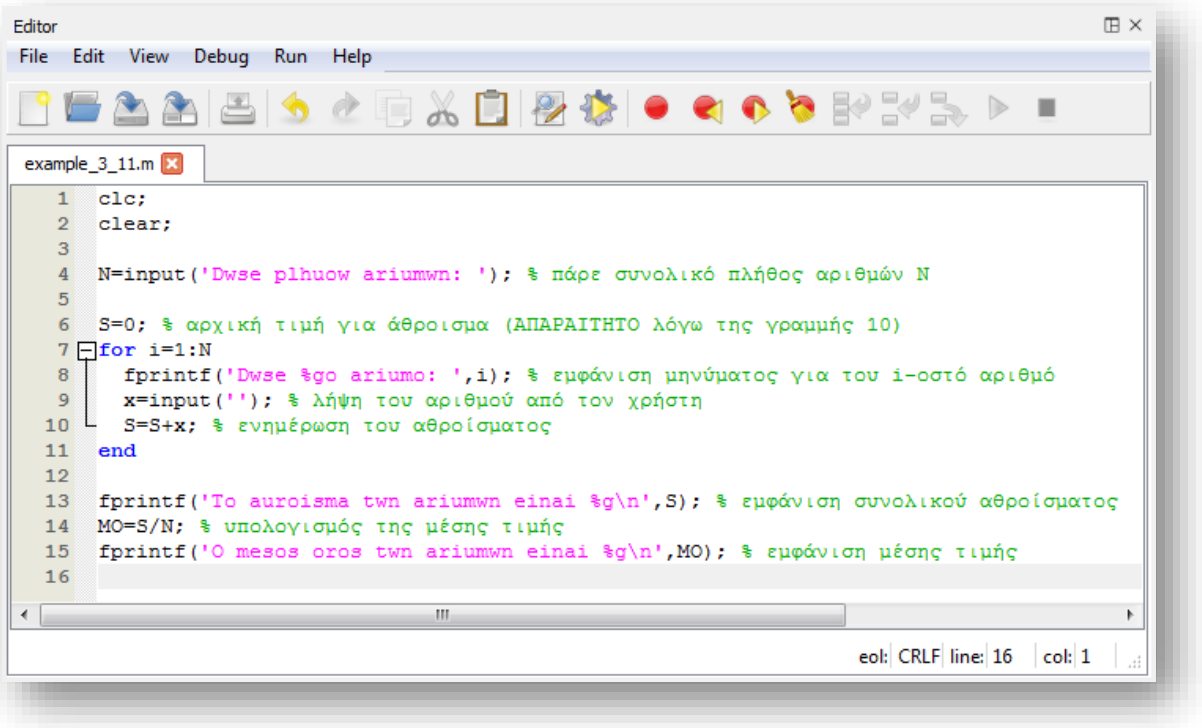
Πώς χρησιμοποιούνται οι βρόχοι επανάληψης στην πράξη;

Τα επόμενα παραδείγματα παρουσιάζουν ορισμένες τυπικές εφαρμογές όπου χρησιμοποιούνται βρόχοι επανάληψης τόσο με την μορφή `for…end` όσο και με την `while…end`.

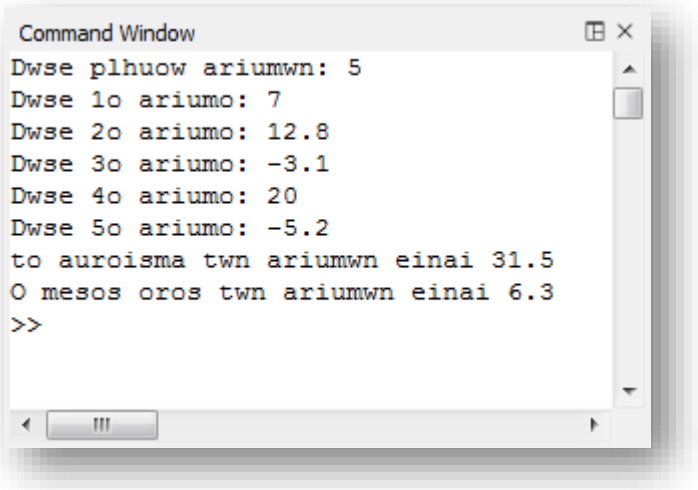
Δύο διαδικασίες που είναι συνήθεις σε διάφορα προγράμματα είναι ο υπολογισμός του αθροίσματος ενός συνόλου τιμών καθώς και της μέσης τιμής τους.

Παράδειγμα 3-11 Εύρεση αθροίσματος και μέσου όρου σε ένα σύνολο τιμών

Γράψτε ένα πρόγραμμα που να ζητά από τον χρήστη ένα συγκεκριμένο πλήθος αριθμών και έπειτα να εμφανίζει το άθροισμα και την μέση τιμή τους.



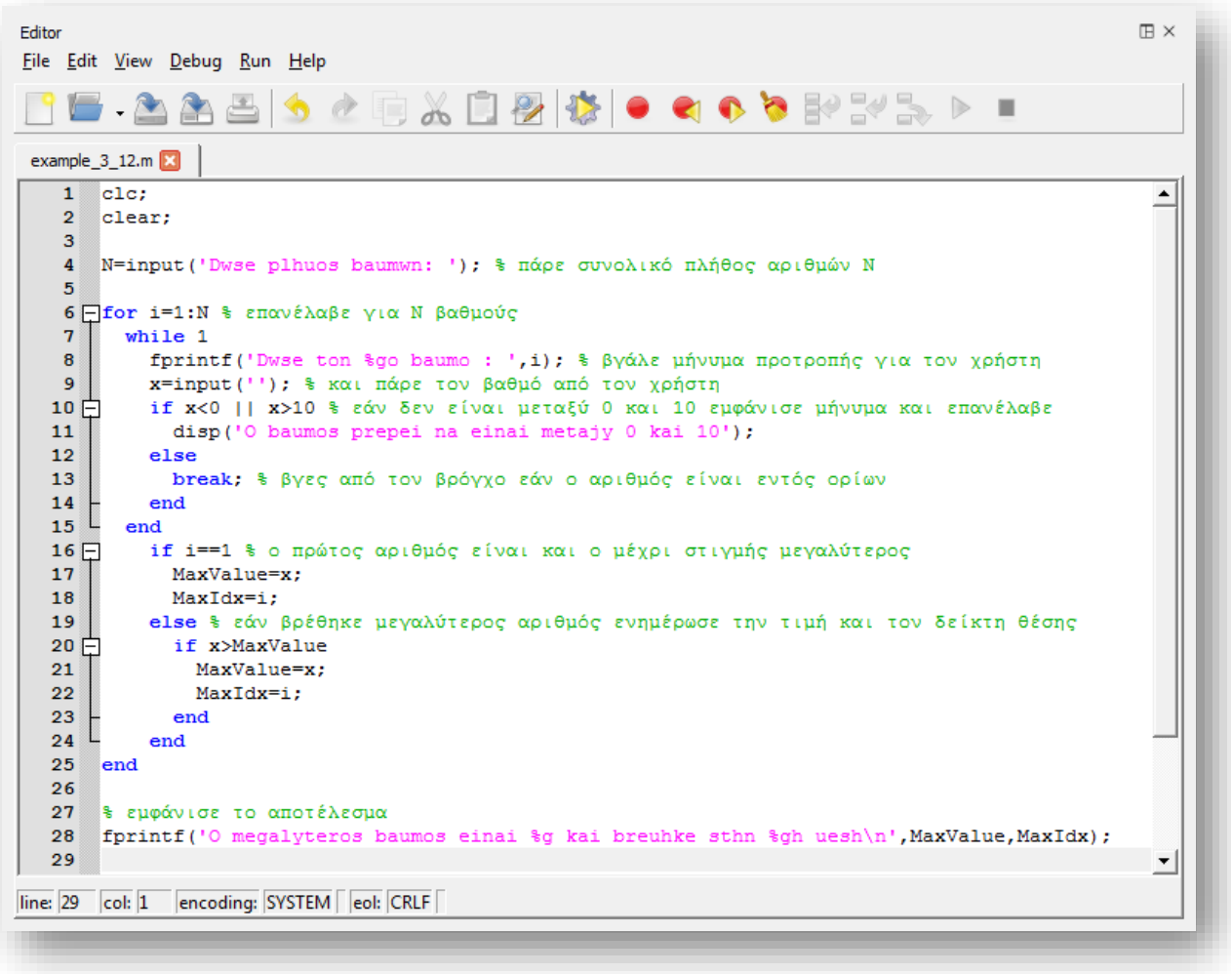
Τρέχοντας το πρόγραμμα για πέντε αριθμούς δίνει:



Μια ακόμη συνηθισμένη διαδικασία είναι ο υπολογισμός του μέγιστου (ή αντίστοιχα, του ελάχιστου) από ένα σύνολο τιμών. Συχνά, εκτός από το μέγιστο (ή το ελάχιστο) μας ενδιαφέρει να βρούμε σε ποια θέση (μέσα στο σύνολο των τιμών) βρέθηκε. Επιπλέον, όταν εισάγονται τιμές στο πρόγραμμα οι οποίες πρέπει να βρίσκονται εντός κάποιων ορίων τότε πρέπει να ενσωματώσουμε τους κατάλληλους ελέγχους ώστε να ελέγχεται η ορθότητα της τιμής εισόδου.

Παράδειγμα 3-12 Εύρεση αθροίσματος και μέσου όρου σε ένα σύνολο τιμών

Γράψτε ένα πρόγραμμα που να ζητά από τον χρήστη ένα συγκεκριμένο πλήθος N βαθμολογιών. Έπειτα να εισάγει τις βαθμολογίες ελέγχοντας κάθε φορά ώστε η βαθμός να είναι μεταξύ 0 και 10. Έπειτα, να εμφανίζει μήνυμα με τον μεγαλύτερο βαθμό και σε ποια θέση βρέθηκε.



Τρέχοντας το πρόγραμμα για πέντε αριθμούς δίνει: