ΔΟΜΗΜΕΝΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Μεταβλητές, Τελεστές, Είσοδος/Έξοδος

Μεταβλητές

- Οι **μεταβλητές** αντιστοιχούν σε θέσεις στη μνήμη του υπολογιστή στις οποίες γίνεται *προσωρινή αποθήκευση δεδομένων και αποτελεσμάτων*

`salary:`

`average:`

- Κάθε μεταβλητή έχει ένα **όνομα**
  - Διάκριση πεζών και κεφαλαίων (π.χ., `salary` διαφορετικό από `Salary`).
  - Προτιμούμε αντιπροσωπευτικά ονόματα (π.χ., `average` για μέσο όρο) .
  - <u>Προσοχή</u> στις δεσμευμένες λέξεις (`if`, `end`, `while`, `for`, `break`, …) και στα ονόματα συναρτήσεων (`disp`, `input`, `rem`, …).

Τύποι Μεταβλητών

- Ο **τύπος** μιας μεταβλητής στο Matlab/Octave μπορεί να είναι:
  - αριθμητική τιμή (ακέραιος αριθμός, δεκαδικός αριθμός, κλπ.)
  - αλφαριθμητικό
  - πίνακας (μονοδιάστατος, δισδιάστατος)

- Στο Matlab/Octave <u>δεν</u> δηλώνουμε τον τύπο των μεταβλητών
  - Οι μεταβλητές παίρνουν **δυναμικά** τον τύπο κατά την χρήση τους

- Μια μεταβλητή μπορεί να έχει **μόνο μία τιμή**
  - όταν η μεταβλητή πάρει νέα τιμή, η προηγούμενη τιμή χάνεται

- Οι μεταβλητές παίρνουν τιμές με τις *εντολές ανάθεσης*

```
μεταβλητή = έκφραση;
```

Τύποι Μεταβλητών – Παραδείγματα

- Αριθμός
```
x = 5;
```
Ο αριθμός 5 αποθηκεύεται στη μεταβλητή x

- Αλφαριθμητικό
```
name = 'Giorgos';
```
οι αλφαριθμητικές μεταβλητές έχουν σαν τιμές ακολουθίες χαρακτήρων οι οποίες εισάγονται ανάμεσα σε μονά εισαγωγικά

- Μονοδιάστατος Πίνακας
```
v = [1, 2, 3, 2, 1];
```
τα στοιχεία ενός πίνακα βρίσκονται ανάμεσα σε αγκύλες (όχι παρενθέσεις) και διαχωρίζονται με κόμματα ή με κενά

- Δισδιάστατος Πίνακας
```
A = [1, 2, 3; 4.1, 5, 2.3];
```
Οι δισδιάστατοι πίνακες ορίζονται με παρόμοιο τρόπο: δίνουμε τα στοιχεία κάθε γραμμής ενώ για να υποδείξουμε την αλλαγή γραμμής χρησιμοποιούμε το σύμβολο ';'

Ασθενές Σύστημα Τύπων

- Το Matlab/Octave υποστηρίζει ασθενές σύστημα τύπων:
  - Δεν ορίζουμε τον τύπο μιας μεταβλητής
  - Οι μετατροπές τύπων γίνονται αυτόματα

- Παράδειγμα:

`x = 2;`	Το x αποθηκεύει ένα ακέραιο αριθμό
`y = 3.2;`	Το y αποθηκεύει ένα δεκαδικό αριθμό
`z = x + y;`	Το x μετατρέπεται σε δεκαδικό αριθμό

Εντολές ανάθεσης

```
μεταβλητή = έκφραση;
```

- Η εκτέλεση μιας εντολής ανάθεσης υλοποιείται σε δυο διακριτά βήματα
  1. Υπολογισμός της *έκφρασης* στο δεξί μέλος
  2. Αποθήκευση της *τιμής* της έκφρασης (αποτέλεσμα) στη μεταβλητή που καθορίζεται αριστερά του τελεστή ανάθεσης =

- Επομένως η εντολή

```
x = x + 1;
```

είναι ανάθεση τιμής, και <u>όχι</u> μια αλγεβρική εξίσωση!

Παράδειγμα 1: Εντολές ανάθεσης

- **Ερώτηση:** Ποια θα είναι η τιμή της μεταβλητής y μετά την εκτέλεση του παρακάτω κώδικα?

```
x = 37;
y = x + 2;
x = 20;
```

- **Απάντηση:** Η μεταβλητή y θα έχει την τιμή 39, καθώς γίνεται μόνο μία ανάθεση τιμής (δεύτερη εντολή). Η ανάθεση x = 20 αλλάζει την τιμή της μεταβλητής x από 37 σε 20, ωστόσο αυτό δεν έχει καμία επιρροή στην τιμή της μεταβλητής y, καθώς οι εντολές εκτελούνται με τη σειρά.

Παράδειγμα 2: Εντολές ανάθεσης

- **Ερώτηση:** Μπορούμε να αλλάξουμε τη σειρά των πρώτων δύο εντολών, χωρίς να αλλάξει το τελικό αποτέλεσμα ;

```
x = 37;
y = x + 2;
x = 20;
```

➡

```
y = x + 2;
x = 37;
x = 20;
```

?

- **Απάντηση:** Όχι. Μια μεταβλητή πρέπει να έχει οριστεί προτού τη χρησιμοποιήσουμε σε κάποια έκφραση. Στο συγκεκριμένο παράδειγμα, αν αλλάξουμε τη σειρά των εντολών, τότε η εκτέλεση της εντολής `y = x + 2` δεν θα μπορεί να εκτελεστεί καθώς η μεταβλητή x δεν έχει αρχικοποιηθεί με κάποια τιμή.
- **Προσοχή:** Οι εντολές ενός προγράμματος εκτελούνται πάντα με τη σειρά!

Παράδειγμα 3: Εντολές ανάθεσης

- **Ερώτηση:** Έστω ότι έχουμε δύο μεταβλητές:

```
x = 3;
y = 5;
```

Πως θα αντιμεταθέσουμε τις τιμές τους?

- **Απάντηση:** Θα πρέπει να χρησιμοποιήσουμε μία τρίτη μεταβλητή ώστε να αποθηκεύσουμε (προσωρινά) την τιμή της μίας από τις δύο μεταβλητές. Ο κώδικας φαίνεται παρακάτω:

```
tmp = x;
x = y;
y = tmp;
```

Τελεστές (πράξεις)

- Ένας **τελεστής (operator)** είναι ένα σύμβολο που αναπαριστά μια συγκεκριμένη πράξη σε ένα ή περισσότερα δεδομένα.

- Τα δεδομένα καλούνται **τελεστέοι (operands)** και μπορούν να είναι *μεταβλητές*, *αριθμοί*, ή/και *κλήσεις συναρτήσεων*.

- Οι τελεστές χρησιμοποιούνται σε **εκφράσεις (expressions)**.
  - Παράδειγμα: `a = b + c;`

- Οι πιο συχνοί τελεστές είναι:
  - **Αριθμητικοί** τελεστές (+ , - , * , / , ...)
  - **Συγκριτικοί** τελεστές (== , > , < , ...)
  - **Λογικοί** τελεστές (&& , || , ~ , ...)

Αριθμητικοί Τελεστές

- **Πρόσθεση, Αφαίρεση:** `+ -`

```
x = 1 + 2;
y = x - 3;
```

- **Πολλαπλασιασμός, Διαίρεση:** `* /`

```
x = 5 / 2;
y = 2 * x;
```

- **Ύψωση σε δύναμη:** `^`

```
x = 5 ^ 3;
```

- **Υπόλοιπο ακέραιας διαίρεσης:** `rem`

```
x = rem(5, 3);
```

Παράδειγμα: Αριθμητικές Πράξεις

- **Ερώτηση:** Ποια είναι η τιμή των μεταβλητών `Z` και `W` μετά την εκτέλεση του παρακάτω κώδικα?

```
X = 5;
Y = 10;

Z = X * Y;
W = X / Y;
```

- **Απάντηση:** Η τιμή της μεταβλητής `Z` είναι `50` και της `W` είναι `0.5`

Συγκριτικοί Τελεστές

- **Ίσο:** `==`
- **Διαφορετικό:** `~=`
- **Ανισότητα:** `>` `<` `>=` `<=`
- Παρατηρούμε ότι για την ισότητα στις λογικές προτάσεις χρησιμοποιείται το διπλό ίσο (==) το οποίο είναι ο τελεστής σύγκρισης
  - Προσοχή, το μονό ίσο (=) χρησιμοποιείται για εκχώρηση τιμής
- Το αποτέλεσμα της σύγκριση μεταξύ δύο αριθμών είναι:
  - η **λογική μονάδα (1)** αν είναι αληθής η σύγκριση
  - το **λογικό μηδέν (0)** αν είναι ψευδής

Παράδειγμα 1: Πράξεις με συγκρίσεις

- Έστω ότι δηλώνουμε τις ακόλουθες μεταβλητές:

```
x = 4;
y = 3;
```

- Ποιο είναι το αποτέλεσμα των παρακάτω συγκρίσεων?

`x > y`	Το αποτέλεσμα είναι 1 (αληθές), καθώς η τιμή της μεταβλητής x είναι μεγαλύτερη από την τιμή της μεταβλητής y
`x == 2`	Το αποτέλεσμα είναι 0 (ψευδές), καθώς η τιμή της μεταβλητής x δεν ισούται με 2. Το == ελέγχει ισότητα, ενώ το = χρησιμοποιείται για ανάθεση τιμής.
`x ~= 10`	Το αποτέλεσμα είναι 1 (αληθές), καθώς η τιμή της μεταβλητής x δεν έχει τιμή10

Παράδειγμα 2: Πράξεις με συγκρίσεις

- Ποιο είναι το αποτέλεσμα των παρακάτω συγκρίσεων?

`100 >= 100` Το αποτέλεσμα είναι 1 (αληθές)

`80 <= 50` Το αποτέλεσμα είναι 0 (ψευδές)

`100 ~= 50` Το αποτέλεσμα είναι 1 (αληθές)

Παράδειγμα 3: Πράξεις με συγκρίσεις

- **Ερώτηση:** Ποια είναι η τιμή του p μετά την εκτέλεση του παρακάτω κώδικα?

```
p = (1 == 1)
```

- **Απάντηση:** Η μεταβλητή p θα έχει σαν τιμή το 1 (λογικό Αληθές). Η έκφραση 1==1 έχει ως αποτέλεσμα το 1 το οποίο αποθηκεύεται στη μεταβλητή p

Παράδειγμα 4: Πράξεις με συγκρίσεις

- **Ερώτηση:** Ποια είναι η τιμή του `q` μετά την εκτέλεση του παρακάτω κώδικα?

```
q = (2 < 1)
```

- **Απάντηση:** Η μεταβλητή `q` θα έχει σαν τιμή το `0` (λογικό Ψευδές). Η έκφραση `2 < 1` έχει ως αποτέλεσμα το `0` το οποίο αποθηκεύεται στη μεταβλητή `q`.

Βασικές λογικές πράξεις (τελεστές)

- **NOT (Άρνηση):**
  - Αντιστρέφει την τιμή (0 → 1, 1 → 0)
  - Στα μαθηματικά συμβολίζεται ως A′
  - Στο MATLAB/Octave συμβολίζεται με `~A`
- **AND (ΚΑΙ):**
  - Το αποτέλεσμα είναι 1 μόνο αν και οι δύο τελεστέοι είναι 1
  - Στα μαθηματικά συμβολίζεται ως A · B
  - Στο MATLAB/Octave συμβολίζεται με `A && B`
- **OR (Ή):**
  - Το αποτέλεσμα είναι 1 αν τουλάχ. ένας από τους τελεστέους είναι 1
  - Στα μαθηματικά συμβολίζεται ως A + B
  - Στο MATLAB/Octave συμβολίζεται με `A || B`

- *Οι **λογικοί τελεστές** συνδυάζονται με συγκρίσεις για τη λήψη αποφάσεων σε προγράμματα.*

Πίνακας αλήθειας λογικών πράξεων

- Ο **πίνακας αλήθειας** δείχνει το αποτέλεσμα μιας λογικής έκφρασης για κάθε πιθανό συνδυασμό των τελεστέων

- Ο πίνακας αληθείας των βασικών λογικών πράξεων είναι:

A	B	NOT A	A AND B	A OR B
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

Χρήση λογικών πράξεων σε συνθήκες

- Η Άλγεβρα Boole χρησιμοποιείται ευρέως στον προγραμματισμό για λήψη αποφάσεων

- **Παράδειγμα:**

```
can_drive = (age > 18) && (has_license == 1);
```

- *Θα αναφέρουμε περισσότερα σχετικά με τους λογικούς τελεστές στην διάλεξη για τις «εντολές απόφασης»*

Προτεραιότητα Πράξεων

- Η *προτεραιότητα* υπολογισμών καθορίζει τη σειρά εκτέλεσης των τελεστών σε μια έκφραση

- Η έκφραση `a/b*c` ισούται με `a/(b*c)` ή με `(a/b)*c` ?
  - Π.χ., το `3/2*6` ισούται με `3/(2*6)` ή με `(3/2)*6` ?

- Προτεραιότητα τελεστών :
  1. `( )`
  2. `^`
  3. `~`
  4. `*` `/` (από αριστερά προς τα δεξιά)
  5. `+` `-` (από αριστερά προς τα δεξιά)
  6. `<` `<=` `>` `>=` `==` `~=`
  7. `&&`
  8. `||`

Παράδειγμα 1: Προτεραιότητα Πράξεων

- **Ερώτηση:** Ποια είναι η τιμή της μεταβλητής `Z` και της μεταβλητής `W` μετά την εκτέλεση των παρακάτω πράξεων?

```
X = 5;
Y = 10;

Z = X * Y + X;
W = X / Y ^ 2;
```

- **Απάντηση:** Η τιμή της μεταβλητής `Z` είναι `55` και της `W` είναι `0.05`.

Παράδειγμα 2: Προτεραιότητα Πράξεων

- **Ερώτηση:** Ποια είναι η τιμή της μεταβλητής x μετά την εκτέλεση των παρακάτω πράξεων?

```
x = 5 + 3 * 4 / 2;
```

- **Απάντηση:** Η τιμή της μεταβλητής x είναι 11. Ο πολλαπλασιασμός και η διαίρεση έχουν την ίδια προτεραιότητα, οπότε η εκτέλεση τους θα γίνει από αριστερά προς τα δεξιά: η έκφραση `3 * 4 / 2` ισούται με 6, το οποίο προστίθεται με το 5.

Σχόλια

- Το σύμβολο % δηλώνει σχόλια

- Τα σχόλια είναι ελεύθερο κείμενο και χρησιμοποιούνται για να προσθέσουμε μία **επεξήγηση** ή **σημείωση** σε κάποιο σημείο του προγράμματός μας
  - Δεν λαμβάνονται υπόψιν κατά την εκτέλεση του προγράμματος

- Παράδειγμα:

```
% store radius of the circle
r = 5;

% calculate area of the circle
a = 3.14 * r ^ 2;
```

Διάβασμα τιμών από το πληκτρολόγιο

- Συνάρτηση `input()`
  - `result = input(prompt);`

- Κατά την εκτέλεση της, η `input`:
  - Εμφανίζει στην οθόνη το μήνυμα που είναι μέσα στην παρένθεση και περιμένει από το χρήστη να πληκτρολογήσει ένα αριθμό ή αλφαριθμητικό
  - Όταν ο χρήστης πατήσει το πλήκτρο Enter (↵), τότε αυτό που έγραψε θα αποθηκευθεί στην μεταβλητή που είναι αριστερά του =

- Παράδειγμα:

```
result = input('Enter salary: ');
```

Εκτύπωση στη γραμμή εντολών

- Αν δεν τερματίσουμε μια εντολή με το ελληνικό ερωτηματικό (;), τότε το αποτέλεσμα της εντολής θα τυπώνεται στην οθόνη.

- Παράδειγμα:

```
f = 451             % Εκτυπώνεται ο αριθμός 451

c = (f-32)*5/9      % Εκτυπώνεται η τιμή της (f-32)*5/9

c                   % Εκτυπώνεται η τιμή της c

f , c               % Εκτυπώνεται η τιμή των f και
                    % c σε 2 γραμμές
```

Εκτύπωση στη γραμμή εντολών (συνέχ)

- Με την συνάρτηση `disp()`
  - τυπώνει την τιμή μόνο μίας μεταβλητής, αλφαριθμητικού ή έκφρασης

- Με την συνάρτηση `fprintf()`
  - πλήρης έλεγχος της μορφής της εκτύπωσης

Η συνάρτηση `disp()`

- Τυπώνει την τιμή μόνο μίας μεταβλητής, αλφαριθμητικού ή έκφρασης

- Παραδείγματα:
  - εκτύπωση της μεταβλητής x: `disp(x);`
  - εκτύπωση της έκφρασης 1 + 1: `disp(1+1);`
  - εκτύπωση του «Code is cool!»: `disp('Code is cool!');`
  - εκτύπωση της έκφρασης '1 + 1': `disp('1+1');`
  - εκτύπωση του αριθμού 13: `disp(13);`

Η συνάρτηση `fprintf()`

- Η εντολή `fprintf` χρησιμοποιείται αντί της `disp` στην περίπτωση που θέλουμε να εκτύπωση τα δεδομένα σε συγκεκριμένη **μορφή**

- Η σύνταξη της `fprintf` στη γενική περίπτωση έχει ως εξής:

```
fprintf(format, a, b, ....)
```

- όπου το όρισμα **`format`** είναι ένα αλφαριθμητικό το οποίο μπορεί να περιέχει *προσδιοριστές εμφάνισης*
  - Ένας <u>*προσδιοριστής εμφάνισης*</u> ελέγχει τον συμβολισμό, τη στοίχιση, το πλήθος των ψηφίων, και άλλα χαρακτηριστικά της μορφής της εκτύπωσης

- Οι παράμετροι **a**, **b**, …. τυπώνονται με τη μορφή που καθορίζεται στο αλφαριθμητικό `format`

Το αλφαριθμητικό `format`

- Το αλφαριθμητικό **`format`** μπορεί να περιέχει:
  - Χαρακτήρες, πχ. `'T'`, `'o'`, `' '`
  - Ειδικούς χαρακτήρες, πχ. `\n`
  - Προσδιοριστές εμφάνισης (βλέπε επόμενη διαφάνεια), πχ. `%d` , `%f`

- Παράδειγμα:

```
x = 3;
y = 10.3;
fprintf('Το x είναι %d και το y είναι %f \n', x, y);
```

Προσδιοριστές εμφάνισης

- Οι **προσδιοριστές εμφάνισης** καθορίζουν τη μορφή της εκτύπωσης

- Οι πιο συχνοί προσδιοριστές εμφάνισης είναι οι ακόλουθοι:

Χαρακτήρας	Περιγραφή
%d	Ακέραιος αριθμός
%f	Δεκαδικός αριθμός
%s	Αλφαριθμητικό
%c	Χαρακτήρας

Παράδειγμα: Προσδιοριστές εμφάνισης

- Έστω ότι θέλουμε να τυπώσουμε τα περιεχόμενα δύο μεταβλητών στο ίδιο μήνυμα:

```
x = 3;
y = 10;

fprintf('To x είναι %d και το y είναι %d', x, y);
```

Ειδικοί Χαρακτήρες

- Τα αλφαριθμητικά μπορούν να περιέχουν **ειδικούς χαρακτήρες** οι οποίοι αναπαραστούνται με **χαρακτήρες απόδρασης** (escape characters)
  - καθορίζουν μη εκτυπώσιμους χαρακτήρες

- Οι πιο συχνοί ειδικοί χαρακτήρες είναι οι ακόλουθοι:

Χαρακτήρας	Περιγραφή
`\n`	Νέα γραμμή
`\t`	Οριζόντια στοίχιση (tab)
`\\`	\
`%%`	% (το σύμβολο)

Παράδειγμα: Ειδικοί Χαρακτήρες

```
fprintf('line1\nline2\nline3\n');
```

Έξοδος:

```
line1
line2
line3
```

```
fprintf('Το ΦΠΑ είναι %d%%\n', 24);
```

Έξοδος:

```
Το ΦΠΑ είναι 24%
```

Άσκηση 1: Υπολογισμός ηλικίας

- Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει το έτος γέννησης ενός ατόμου
- Στη συνέχεια, ο υπολογιστής θα υπολογίζει την ηλικία και θα την εμφανίζει στην οθόνη του υπολογιστή

- Λύση:

```
etos = input('Δώσε έτος γέννησης:');

ilikia = 2026 - etos;

fprintf('Η ηλικία σας είναι %d', ilikia);
```

Άσκηση 2: Εμβαδό και περίμετρος κύκλου

- Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει την ακτίνα ενός κύκλου
- Στη συνέχεια, θα υπολογίζει την περίμετρο και το εμβαδό και θα το εμφανίζει στην οθόνη του υπολογιστή

- Λύση:

```
aktina = input('Δώσε ακτίνα κύκλου:');

perimetros = 2 * 3.14 * aktina;
emvado     = 3.14 * (aktina ^ 2);

fprintf('Η περίμετρος είναι %f \n', perimetros);
fprintf('Το εμβαδό είναι %f \n', emvado);
```

Συχνά λάθη

- Μη αρχικοποιημένη μεταβλητή

```
y = x + 1;          ERROR
disp(z);            ERROR
```

- Πράξεις με μεταβλητές διαφορετικού τύπου

```
a = 'hello';
b = 42;
c = a + b;          ERROR
```

- Λάθος χρήση παρενθέσεων

```
y = (3 + 4) * 5;    35
y = 3 + 4 * 5;      23
```

- Χρήση δεσμευμένης λέξης

```
for = 5;            ERROR
```

Ανακεφαλαίωση

- Οι βασικές αλγοριθμικές ενέργειες και οι αντίστοιχες εντολές στο MATLAB/Octave:

είσοδος δεδομένων:	`input()`
έξοδος αποτελεσμάτων:	`disp()` `fprintf()`
πράξεις και αναθέσεις τιμών σε μεταβλητές:	`+ - * / ^ =`
πράξεις σύγκρισης:	`< <= >= > == ~=`
Πράξεις λογικής άλγεβρας:	`&& || ~`

- Μπορείτε πλέον να γράψετε προγράμματα που εκτελούνται σειριακά (από πάνω προς τα κάτω) και εκτελούν αριθμητικές και λογικές πράξεις