ΔΟΜΗΜΕΝΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Συναρτήσεις

Συναρτήσεις

- Υποθέστε ότι θέλουμε να υπολογίσουμε την καθαρή τιμή ενός προϊόντος χωρίς ΦΠΑ

- Γράφουμε τον παρακάτω κώδικα:

```
net = 8 / 1.24;
```

Συναρτήσεις

- Αν θέλαμε να υπολογίσουμε την καθαρή τιμή πολλών προϊόντων:

```
net = 124.0 / 1.24;
net = 59.9 / 1.24;
net = 12.5 / 1.24;
...
```

- Ωστόσο αυτό δεν βολεύει και πολύ:
  - Ο ίδιος υπολογισμός *επαναλαμβάνεται* σε πολλά διαφορετικά σημεία.
  - Αν αλλάξει το ποσοστό του ΦΠΑ, θα πρέπει να διορθωθεί παντού.
  - Αυξάνονται οι πιθανότητες να γίνει κάποιο λάθος (π.χ. λάθος συντελεστής, παράλειψη της διαίρεσης κ.λπ.).

Συναρτήσεις

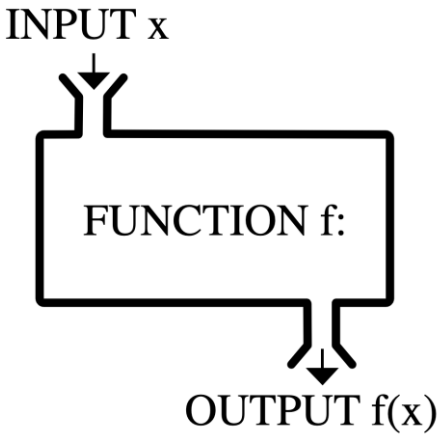
- Είναι προτιμότερο να κάνουμε:

```
a = priceWithoutVAT(4);
```

αντί

```
a = 4 / 1.24;
```

- Γιατί?
  - Ο κώδικας είναι πιο καθαρός και ευανάγνωστος.
  - Αν αλλάξει ο ΦΠΑ, το αλλάζουμε σε ένα σημείο.
  - Ελαχιστοποιείται η πιθανότητα λαθών.



Συναρτήσεις

- Για να μπορούμε να γράφουμε απλά `priceWithoutVAT(4)` πρέπει να ορίσουμε πρώτα μια συνάρτηση με όνομα `priceWithoutVAT`

- Πως γίνεται αυτό?
  - Δημιουργούμε ένα αρχείο με όνομα `priceWithoutVAT.m`
  - Γράφουμε τον κώδικα που χρειάζεται:

```
function net = priceWithoutVAT (price)

    net = price / 1.24;

end
```

Αρχεία Συναρτήσεων (m-files)

- Ένα αρχείο συνάρτησης (m-file) περιέχει την υλοποίηση μιας συνάρτησης.

- Το *όνομα του αρχείου* πρέπει να είναι το *ίδιο* με το *όνομα της συνάρτησης* που περιέχει.

- Κάθε συνάρτηση στο Octave/Matlab γράφεται σε δικό της αρχείο
  - Η συνάρτηση `priceWithoutVAT` ΠΡΕΠΕΙ να γραφτεί σε ένα αρχείο με όνομα `priceWithoutVAT.m`
  - Η συνάρτηση `priceWithDiscount` ΠΡΕΠΕΙ να γραφτεί σε ένα αρχείο με όνομα `priceWithDiscount.m`

- Το Octave/Matlab εντοπίζει τον κώδικα μιας συνάρτησης ΜΟΝΟ σε αρχεία με το ίδιο όνομα και κατάληξη .m

Σύνταξη

```
function net = priceWithoutVAT (price)
    net = price / 1.24;
end
```

Σύνταξη

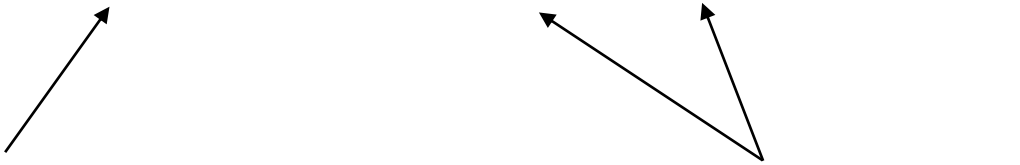
```
function net = priceWithoutVAT (price)
    net = price / 1.24;
end
```

Σημασιολογία των Συναρτήσεων

- Μία συνάρτηση ορίζεται σε ένα ξεχωριστό αρχείο.

- Το αρχείο πρέπει πάντα να αρχίζει με μια γραμμή σαν αυτή:

```
function res = add (arg1, arg2)
```



όνομα μεταβλητής που θα αποθηκευτεί το αποτέλεσμα

ονόματα παραμέτρων

Σημασιολογία των Συναρτήσεων

- Το υπόλοιπο αρχείο περιέχει εντολές οι οποίες:
  - Εκτελούν υπολογισμούς πάνω στις *παραμέτρους*,
  - Αναθέτουν μια τιμή στη *μεταβλητή αποτελέσματος*.

```
function res = add (arg1, arg2)
    res = arg1 + arg2;
end
```

Γιατί χρησιμοποιούμε συναρτήσεις?

- Το πρόβλημα διασπάται σε μικρότερα κομμάτια και μπορεί να επιλυθεί πιο εύκολα («*Διαίρει και βασίλευε*»).
- Με την χρήση των συναρτήσεων ένα πρόγραμμα μπορεί να χωριστεί σε άλλα μικρότερα κομμάτια (δομές).
- Η συνάρτηση είναι ένα *σύνολο εντολών* που καλείται και χρησιμοποιείται από ένα σημείο ενός προγράμματος.
- Ο δομημένος προγραμματισμός βασίζεται στην αρχή της επίλυσης μίας εφαρμογής προγραμματισμού κάνοντας χρήση μικρότερων δομικών στοιχείων.

Πλεονεκτήματα Συναρτήσεων

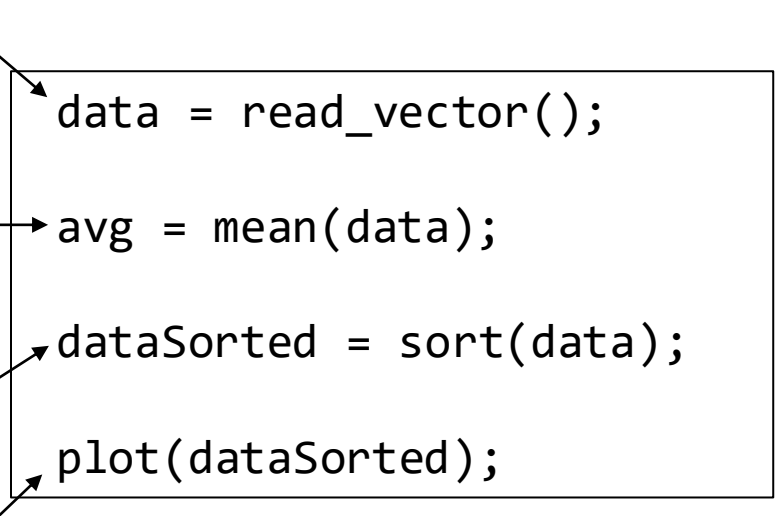
- Λογική σχεδίαση προγράμματος.
- «Μαύρα κουτιά».
- Αποφυγή επανάληψης κώδικα στο ίδιο πρόγραμμα.
- Επαναχρησιμοποίηση κώδικα σε άλλα προγράμματα.

Πως χρησιμοποιούνται οι συναρτήσεις?

- Εντοπίζουμε ένα «πρόβλημα» που χρειάζεται να επιλυθεί σαν μέρος του προγράμματος.
- Επιλύουμε αλγοριθμικά το πρόβλημα και γράφουμε τον κώδικα που χρειάζεται.
- Δίνουμε στον κώδικα του προβλήματος ένα όνομα: αυτό τον μετατρέπει σε συνάρτηση.
- Όταν συναντήσουμε ξανά το ίδιο πρόβλημα, χρησιμοποιούμε απλώς το όνομα της συνάρτησης για να ζητήσουμε να εκτελεστεί *εμβόλιμα* ο κώδικας της συνάρτησης προτού ο έλεγχος επιστρέψει πίσω.

Παράδειγμα

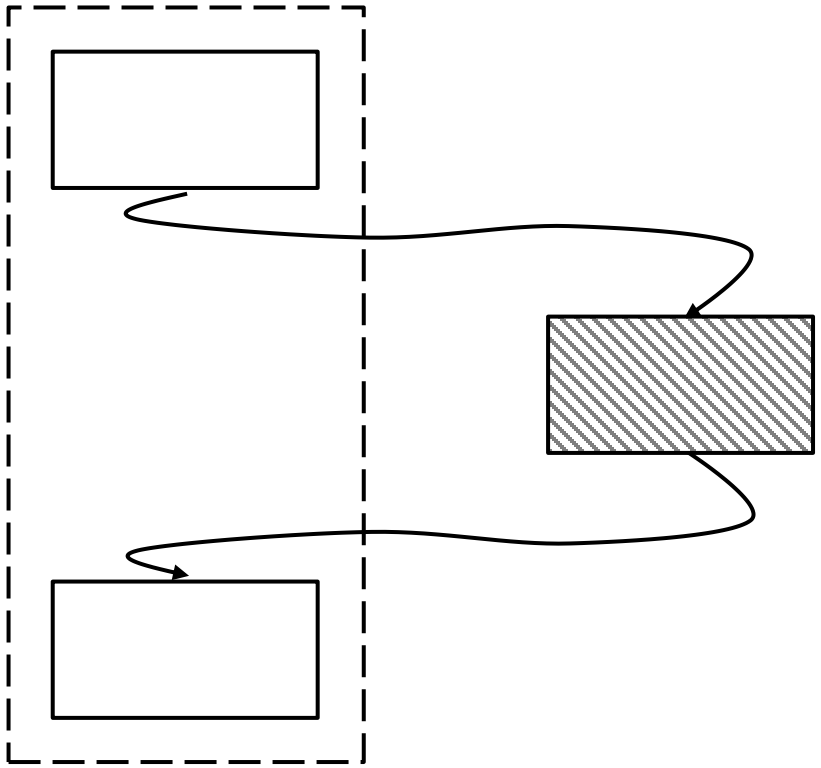
- Έστω ότι θέλουμε να φτιάξουμε ένα πρόγραμμα το οποίο:
  - διαβάζει έναν πίνακα με αριθμούς
    πχ., *τον [5, 6, 1, 7, 2, 4, 3];*
  - Υπολογίζει τον μέσο όρο τους
    πχ., *4*
  - τους ταξινομεί σε αύξουσα σειρά
    πχ., *[1, 2, 3, 4, 5, 6, 7];*
  - τους απεικονίζει σαν γράφημα



Μία συνάρτηση μπορεί να θεωρηθεί σαν ένα «μαύρο κουτί» το οποίο δέχεται κάποιες πληροφορίες (δεδομένα) και παράγει κάποιο αποτέλεσμα (π.χ., τιμές)

Μια άλλη μορφή ροής ελέγχου

- Όταν ένα πρόγραμμα *καλεί* μια *συνάρτηση*, η εκτέλεση μεταφέρεται στις εντολές της συνάρτησης, και αφού εκτελεστούν *επιστρέφει* στον αρχικό κώδικα.



Είδη Συναρτήσεων

- Συναρτήσεις χρήστη - Ορισμένες από τον προγραμματιστή.
- Συναρτήσεις βιβλιοθήκης - Ορισμένες από το Octave/Matlab.

Είδη Συναρτήσεων

- Συναρτήσεις χρήστη - Ορισμένες από τον προγραμματιστή.
- Συναρτήσεις βιβλιοθήκης - Ορισμένες από το Octave/Matlab.

Συναρτήσεις χρήστη - Ορισμός συναρτήσεων

- Ορισμός συνάρτησης:

```
function [παράμετροι εξόδου ] = όνομα (παράμετροι εισόδου )
    εντολές
end
```

- Κλήση συνάρτησης:

```
[ορίσματα εξόδου ] = όνομα (ορίσματα εισόδου );
```

Ονομασία Συναρτήσεων

- Το όνομα αρχίζει με γράμμα (του αγγλικού αλφαβήτου).
- Το όνομα μπορεί να περιέχει γράμματα, αριθμούς ή κάτω παύλες (_).
- Δεν χρησιμοποιούνται ονόματα που έχουν δεσμευτεί από το Octave/Matlab (πχ., εντολές, συναρτήσεις βιβλιοθήκης, κλπ.).
- Προτιμώνται μικρά ονόματα για πρακτικούς λόγους αν και δεν υπάρχει περιορισμός στο μήκος των ονομάτων.
- Η επιλογή του ονόματος πρέπει να γίνεται με τρόπο ώστε να είναι περιγραφικό και σχετικό με την λειτουργιά της συνάρτησης.

Παραδείγματα

- Η παρακάτω συνάρτηση υπολογίζει το άθροισμα δυο αριθμών και επιστρέφει το αποτέλεσμα:

```
function s = addition(a, b)
    s = a + b;
end
```

- Για να χρησιμοποιηθεί η συνάρτηση πρέπει να γίνει κλήση της με κάποιον από τους παρακάτω τρόπους:

```
z = addition (15, 8);
z = addition (k, l);
z = addition (18, x);
z = addition (y, 2);
...
```

Παρατηρήσεις

- Υπάρχει αντιστοιχία μεταξύ του τύπου και του αριθμού των παραμέτρων που χρησιμοποιούνται στον ορισμό και στην κλήση της συνάρτησης:

Ορισμός Συνάρτησης	**Κλήση Συνάρτησης**

`function s = addition(a, b)`
`  s = a + b;`
`end`
	`x = addition (15, 8);`
	`y = addition (k, l);`
	`z = addition (18.2, -2);`

- Οι μεταβλητές που χρησιμοποιούνται για να ορίσουν την συνάρτηση λέγονται ***παράμετροι*** *(parameters)*.

- Οι εκφράσεις που χρησιμοποιούνται στην κλήση μιας συνάρτησης λέγονται ***ορίσματα*** *(arguments)*.

- Μόλις γίνει κλήση της συνάρτησης, *οι τιμές των ορισμάτων μεταφέρονται στις αντίστοιχες παραμέτρους*.

Παρατηρήσεις

- Μια συνάρτηση μπορεί να μην επιστρέφει κάποια τιμή:
  - Τέτοιες συναρτήσεις συνήθως εκτελούν κάποια ενέργεια (π.χ., τυπώνουν κείμενο, γράφημα, κλπ.) είτε στην οθόνη ή σε κάποιο αρχείο.

- Παράδειγμα:

```
function prompt()
    disp('Program for managing course grades');
    disp('Each grade should be a number between 0-10');
end
```

Η εντολή `return`

- Η εντολή `return` επιστρέφει τον έλεγχο στο πρόγραμμα κλήσης πριν φτάσει στο τέλος της συνάρτησης.
- Μια συνάρτηση μπορεί να έχει περισσότερα από ένα `return`.
- Σε κάθε περίπτωση, όταν η εκτέλεση της συνάρτησης φτάσει στην εντολή `return`, τότε η συνάρτηση τερματίζεται άμεσα και επιστρέφει.

- Το παρακάτω παράδειγμα βρίσκει τον μεγαλύτερο από δυο ακέραιους:

```
function m = max_of_two(a, b)
    if a > b
        m = a;
        return
    end

    m = b;
end
```

Παρατηρήσεις

- Μία συνάρτηση μπορεί να επιστρέφει περισσότερες από μία τιμές.
- Η παρακάτω συνάρτηση παίρνει σαν παράμετρο τρεις αριθμούς και επιστρέφει τον μικρότερο και τον μεγαλύτερο:

```
function [minimum, maximum] = minmax(a, b, c)
    if a > b & a > c
        maximum = a;
    elseif b > c
        maximum = b;
    else
        maximum = c;
    end
    if a < b & a < c
        minimum = a;
    elseif b < c
        minimum = b;
    else
        minimum = c;
    end
end
```

Σχεδιασμός συναρτήσεων, απόκρυψη υλοποίησης

- Μια συνάρτηση:
  - υλοποιεί κάποιον αλγόριθμο ή διαδικασία,
  - παίρνει δεδομένα εισόδου κατά την κλήση, και
  - επιστρέφει πίσω δεδομένα εξόδου.

- Μια συνάρτηση πρέπει να έχει:
  - μοναδικό όνομα,
  - καλά σχεδιασμένη διεπαφή (interface):
    - η διεπαφή ορίζει πως επικοινωνεί με το περιβάλλον της → τι είδους *παραμέτρους* χρειάζεται για να λειτουργήσει

Μια καλά σχεδιασμένη συνάρτηση λειτουργεί σαν «μαύρο κουτί»

Παράδειγμα – Σχεδιασμός Συναρτήσεων

- Χαρακτηρισμός βαθμού:
  - Πως θα μετατρέψουμε τον ακόλουθο κώδικα σε συνάρτηση;

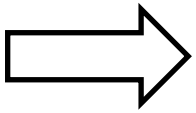
```
vathmos = input('Vathmos: ');

if vathmos >= 8.5
    disp('Excellent');
elseif vathmos >= 6.5
    disp('Very good');
elseif vathmos >= 5.0
    disp('Good');
else
    disp('Fail');
end
```

Βασικός στόχος στη σχεδίαση συναρτήσεων είναι να είναι αρκετά γενικές ώστε να μπορούν να επαναχρησιμοποιηθούν εύκολα

Παράδειγμα – Σχεδιασμός Συναρτήσεων

- Τι παράμετρους χρειάζεται η συνάρτηση;

```
vathmos = input('Vathmos:');

if vathmos >= 8.5
    disp('Excellent');
elseif vathmos >= 6.5
    disp('Very good');
elseif vathmos >= 5.0
    disp('Good');
else
    disp('Fail');
end
```



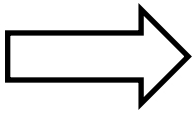
```
function getRank(vathmos)

    if vathmos >= 8.5
        disp('Excellent');
    elseif vathmos >= 6.5
        disp('Very good');
    elseif vathmos >= 5.0
        disp('Good');
    else
        disp('Fail');
    end
end
```

Ένας σωστός σχεδιασμός επιτρέπει σε μια συνάρτηση να μπορεί να χρησιμοποιηθεί σε πολλά διαφορετικά σενάρια.

Παράδειγμα – Σχεδιασμός Συναρτήσεων

- Τι επιστρέφει η συνάρτηση;

```
function getRank(vathmos)
 if vathmos >= 8.5
        disp('Excellent');
    elseif vathmos >= 6.5
        disp('Very good');
    elseif vathmos >= 5.0
        disp('Good');
    else
        disp('Fail');
    end
end
```



```
function m = getRank(vathmos)
 if vathmos >= 8.5
        m = 'Excellent';
    elseif vathmos >= 6.5
        m = 'Very good';
    elseif vathmos >= 5.0
        m = 'Good';
    else
        m = 'Fail';
    end
end
```

Μία συνάρτηση συνήθως υλοποιείται ανεξάρτητα από τον τρόπο που διαβάζει ή γράφει τα δεδομένα. Το πρόγραμμα που την καλεί είναι υπεύθυνο να χειρίζεται τον τρόπο που θα διαβάζονται ή θα γράφονται.

Παράδειγμα – Υπολογισμός παραγοντικού

- **Υπολογισμός παραγοντικού** $N! = 1 \times 2 \times 3 \times \cdots \times N$

```
function p = Factorial (n)
    p = 1;
    for i = 1:n
        p = p * i;
    end
end
```

- Κλήση από αρχείο προγράμματος ή τη γραμμή εντολών:

```
a = Factorial (5);
```

- Κλήση από άλλη συνάρτηση:

```
function c = Combinations (n, k)
  c = Factorial(n) / ( Factorial(k) * Factorial(n-k) );
end
```

Παράδειγμα – Υπολογισμός Εμβαδού

- Ορισμός συνάρτησης:

```
function e = circleArea (r)
    e = 3.14 * r * r;
end
```

- Κλήση από άλλη συνάρτηση:

```
function e = ringArea (inner, outer)
    inner_area = circleArea(inner);
    outer_area = circleArea (outer);
    e = outer_area – inner_area;
end
```

Κλήση Συνάρτησης

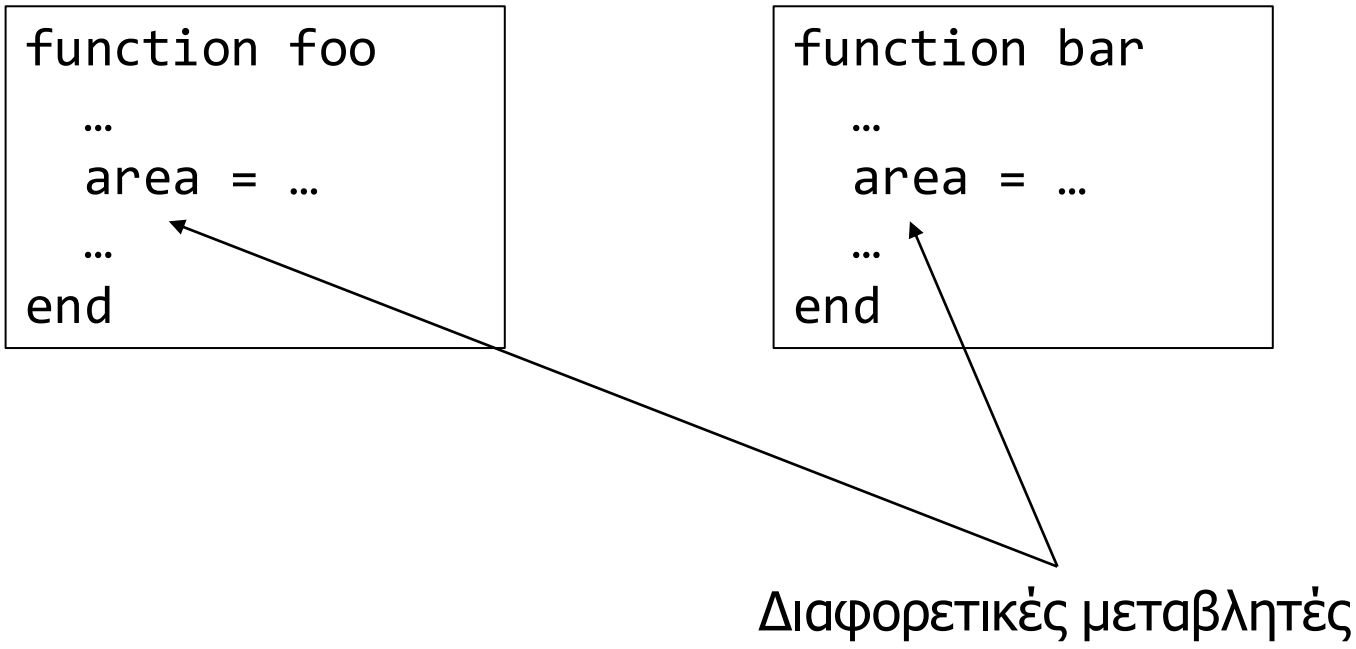
- Όταν μια συνάρτηση καλείται, τότε δεσμεύεται μνήμη για να αποθηκευτούν:
  - οι τιμές των παραμέτρων της συνάρτησης,
  - οι τιμές των μεταβλητών που χρησιμοποιεί η συνάρτηση (τοπικές μεταβλητές).

Τοπικές Μεταβλητές

- Μια συνάρτηση μπορεί να ορίσει δικές της *τοπικές μεταβλητές*.
- Οι τοπικές μεταβλητές έχουν νόημα ΜΟΝΟ μέσα στη συνάρτηση:
  - *Δημιουργούνται* όταν καλείται η συνάρτηση.
  - *Παύουν να υπάρχουν* όταν η συνάρτηση επιστρέψει.
- Οι παράμετροι μιας συνάρτησης είναι επίσης τοπικές μεταβλητές.
- Οι τιμές των παραμέτρων *αρχικοποιούνται αντιγράφοντας* την τιμή του ορίσματος.
- **Προσοχή:** Οι μεταβλητές που ορίζονται μέσα σε μία συνάρτηση δεν επηρεάζουν ούτε επηρεάζονται από μεταβλητές που ορίζονται σε άλλες συναρτήσεις.
  - Ακόμα και αν έχουν το ίδιο όνομα.

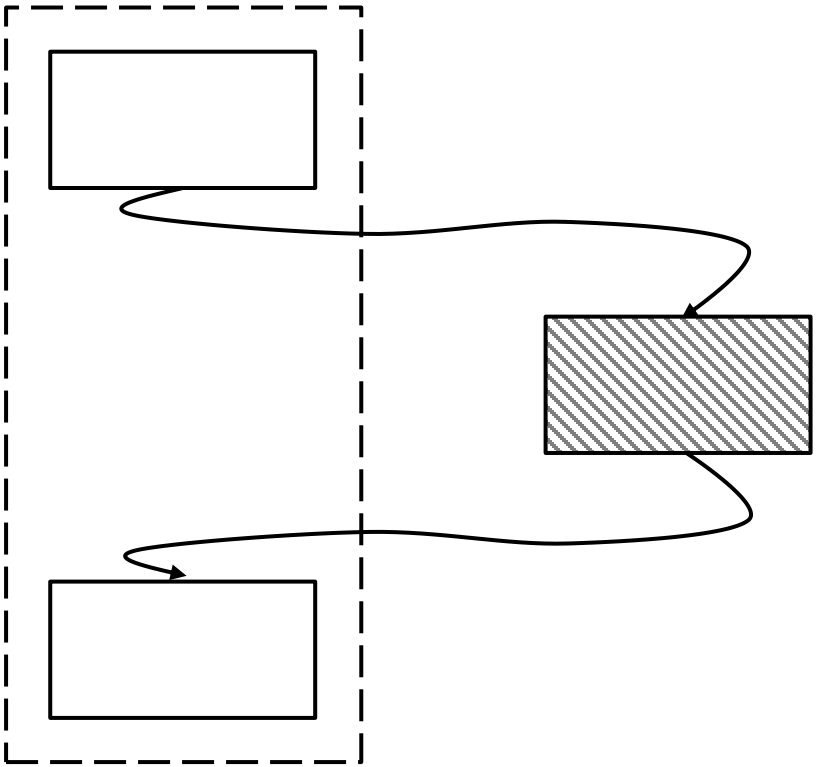
Εμβέλεια Μεταβλητών

- Κάθε συνάρτηση έχει τις δικές της τοπικές μεταβλητές.



- Για την ακρίβεια, κάθε κλήση συνάρτησης έχει τις δικές της μεταβλητές.

Ροή ελέγχου συνάρτησης



Όταν καλείται μια συνάρτηση:

- *Δεσμεύεται* χώρος στη μνήμη για τις *παραμέτρους* και τις *τοπικές μεταβλητές* της συνάρτησης.
- Οι τιμές των ορισμάτων *αντιγράφονται* στις αντίστοιχες μεταβλητές των παραμέτρων.
- Ο έλεγχος *μεταφέρεται* στο σώμα της συνάρτησης.
- *Εκτελούνται* οι εντολές της συνάρτησης
- Ο έλεγχος και τα δεδομένα εξόδου *επιστρέφονται* στην καλούσα συνάρτηση.
- *Αποδεσμεύεται* ο χώρος στη μνήμη για τις παραμέτρους και τις τοπικές μεταβλητές της συνάρτησης.

Είδη Συναρτήσεων

- Συναρτήσεις χρήστη - Ορισμένες από τον προγραμματιστή.

- Συναρτήσεις βιβλιοθήκης - Ορισμένες από το Octave/Matlab.

Συναρτήσεις Βιβλιοθήκης

- Είναι ήδη υλοποιημένες/ορισμένες από το Octave/Matlab
  - Συναρτήσεις για υπολογισμό μαθηματικών τύπων,
  - Συναρτήσεις για διανύσματα και πίνακες,
  - Συναρτήσεις για τυχαίους αριθμούς,
  - κοκ.

- Μπορούν να χρησιμοποιηθούν κατευθείαν.

Χρήσιμες Μαθηματικές Συναρτήσεις

Συνάρτηση	Ερμηνεία
`round`	Στρογγυλοποίηση αριθμού
`floor`	Στρογγυλοποίηση αριθμού προς τα κάτω
`ceil`	Στρογγυλοποίηση αριθμού προς τα πάνω
`sqrt`	Τετραγωνική ρίζα πίνακα
`exp`	Εκθετική συνάρτηση
`log`	Λογάριθμος
`cos`	Συνημίτονο
`sin`	Ημίτονο
`tan`	Εφαπτομένη
`atan`	Αντίστροφη εφαπτομένη
`rem`	Υπόλοιπο ακέραιας διαίρεσης

Συναρτήσεις βιβλιοθήκης - Παραδείγματα

```
a = sqrt(5);

b = floor(8.9);

c = ceil(1.1);

z = log(3) + exp(12);

x = sin(12) + sqrt(cos(15));

m = rem(18, 3);

o = (sin(5))^2 + (cos(5))^2;
```

Συναρτήσεις βιβλιοθήκης - Παραδείγματα

- Υπολογισμός της παράστασης $\sin^2(27) + \sqrt{\cos(13)} + \tan^3(32)$

```
a = sin(27)^2 + sqrt(cos(13)) + tan(32)^3;
```

Χρήσιμες Συναρτήσεις για Διανύσματα

Συνάρτηση	**Περιγραφή**
`size`	Μέγεθος διανύσματος
`length`	Μήκος διανύσματος
`min`	Ελάχιστο στοιχείο διανύσματος
`max`	Μέγιστο στοιχείο διανύσματος
`sort`	Ταξινόμηση σε αύξουσα σειρά
`sum`	Άθροισμα στοιχείων
`prod`	Γινόμενο στοιχείων
`norm`	Ευκλείδεια απόσταση διανύσματος
`mean`	Μέση τιμή
`std`	Τυπική απόκλιση

Συναρτήσεις βιβλιοθήκης - Παραδείγματα

- Εκτύπωση μέσου όρου βαθμολογίας:

```
grades = [6, 5, 5.5, 8.5, 8, 5, 7, 9, 7, 5.5];

m = mean(grades);

disp(m);
```

- Εκτύπωση ελάχιστου και μέγιστου βαθμού:

```
grades = [3, 5, 5.5, 2.5, 8, 5, 3];

disp( min(grades) );
disp( max(grades) );
```

Συναρτήσεις βιβλιοθήκης - Παραδείγματα

- Εκτύπωση των βαθμών με αύξουσα σειρά:

```
grades = [3, 5, 5.5, 2.5, 8, 5, 3];

gradesSorted = sort(grades);

disp(gradesSorted);
```

- Εκτύπωση των τριών μεγαλύτερων βαθμών:

```
grades = [3, 5, 5.5, 2.5, 8, 5, 3];

gradesSorted = sort(grades);

fprintf('Οι 3 μεγαλύτεροι βαθμοί είναι %d %d %d\n',
    gradesSorted(end), gradesSorted(end-1), gradesSorted(end-2));
```

Χρήσιμες Συναρτήσεις για Πίνακες

Συνάρτηση	**Περιγραφή**
`size`	Μέγεθος πίνακα
`length`	Η μεγαλύτερη διάσταση πίνακα
`min`	Διάνυσμα ελάχιστων στοιχείων κάθε στήλης
`max`	Διάνυσμα μέγιστων στοιχείων κάθε στήλης
`sort`	Ταξινόμηση σε αύξουσα σειρά κάθε στήλης
`sum`	Άθροισμα στοιχείων κάθε στήλης
`prod`	Γινόμενο στοιχείων κάθε στήλης
`norm`	Ευκλείδεια απόσταση πίνακα
`mean`	Μέση τιμή κάθε στήλης
`std`	Τυπική απόκλιση κάθε στήλης

Συναρτήσεις βιβλιοθήκης - Παραδείγματα

- Εκτύπωση των θετικών αριθμών ενός πίνακα δύο διαστάσεων:

```
arr = [8, 4, 13, -5; -1 4 6 2];

s = size(arr);

height = s(1);
width = s(2);

for i=1:height
    for j=1:width
        if arr(i, j) > 0
            disp(arr(i, j));
        end
    end
end
```

Συναρτήσεις για Τυχαίους Αριθμούς

Συνάρτηση	**Ερμηνεία**
`randn`	Επιστρέφει έναν τυχαίο μεταξύ `-1` και `1`
`rand`	Επιστρέφει έναν τυχαίο ακέραιο μεταξύ `0` και `1`
`randi(imax)`	Επιστρέφει έναν τυχαίο ακέραιο μεταξύ `1` και `imax`

Συναρτήσεις βιβλιοθήκης - Παραδείγματα

- Ρίψη κέρματος:

```
coin = randi(0, 1);
```

- Ρίψη ζαριού:

```
dice = randi(1, 6);
```