ΔΟΜΗΜΕΝΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Πίνακες

Πίνακες

- Η απλούστερη δομή (συλλογή) δεδομένων

- Ιδιότητες του πίνακα:
  - Τα στοιχεία αποθηκεύονται ***στη σειρά***
  - Όλα τα δεδομένα είναι του ***ίδιου τύπου***

- Μας βοηθάνε να αποθηκεύουμε τα δεδομένα μας ώστε να τα χρησιμοποιήσουμε στο πρόγραμμα μας αργότερα

- Για παράδειγμα:
  - Όταν θέλουμε *απλώς* να προσθέσουμε 10 αριθμούς μπορούμε να χρησιμοποιήσουμε μια μόνο μεταβλητή για όλα τα δεδομένα…
  - … όταν *όμως* θέλουμε να συγκρίνουμε καθένα από τους 10 αριθμούς με τον μέσο όρο τους, πρέπει πρώτα να τους αποθηκεύσουμε σε ένα *πίνακα* 10 θέσεων

Παραδείγματα χρήσης

- Έστω ότι διαβάζουμε 10 αριθμούς από το χρήστη με τη συνάρτηση `input()`.

- Σε ποιες από τις παρακάτω περιπτώσεις είναι αναγκαστικό να αποθηκεύσουμε τους αριθμούς σε πίνακα και σε ποιες όχι?

  - θέλω απλώς να προσθέσω τους 10 αριθμούς → χρησιμοποιώ μια μόνο μεταβλητή για όλα τα δεδομένα μου

  - θέλω να συγκρίνω καθένα από τους 10 αριθμούς με τον μέσο όρο τους → χρησιμοποιώ ένα πίνακα 10 θέσεων

  - θέλω να ταξινομήσω τους 10 αριθμούς κατά αύξουσα σειρά → χρησιμοποιώ ένα πίνακα 10 θέσεων

Δομή πινάκων

- Μονοδιάστατοι πίνακες ή διανύσματα

1	2	3

- Δισδιάστατοι πίνακες

1,1	1,2	1,3
2,1	2,2	2,3
3,1	3,2	3,3

Δημιουργία πίνακα

- Μονοδιάστατος Πίνακας

```
v = [1, 2, 3, 2, 1];
```

τα στοιχεία ενός πίνακα βρίσκονται ανάμεσα σε αγκύλες (όχι παρενθέσεις) και διαχωρίζονται με κόμμα ή με κενό

- Δισδιάστατος Πίνακας

```
A = [1, 2, 3; 4.1, 5, 2.3;];
```

Οι δισδιάστατοι πίνακες ορίζονται με παρόμοιο τρόπο: δίνουμε τα στοιχεία κάθε γραμμής και για να υποδείξουμε την αλλαγή γραμμής χρησιμοποιούμε το σύμβολο `;'

Παράδειγμα

- Πως θα ορίσουμε τους ακόλουθους πίνακες?

$$a = [-3 \quad 4 \quad 0] \qquad \mathrm{b} = \begin{bmatrix} 1 \\ 2 \\ -4 \end{bmatrix} \qquad A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 0 & 1 \\ -1 & 2 & -2 \end{bmatrix}$$

```
a = [-3, 4, 0];

b = [1; 2; -4];

A = [1, 2, 3; 1, 0, 1; -1, 2, -2];
```

Εκτύπωση πινάκων με `disp()`

- Εκτύπωση μονοδιάστατου πίνακα:

```
a = [1 2 5 1];

disp(a);
```

- Εκτύπωση δισδιάστατου πίνακα:

```
b = [1 2 3; 5 1 4; 3 2 -1];

disp(b);
```

Μονοδιάστατοι πίνακες

Μονοδιάστατοι πίνακες (διανύσματα)

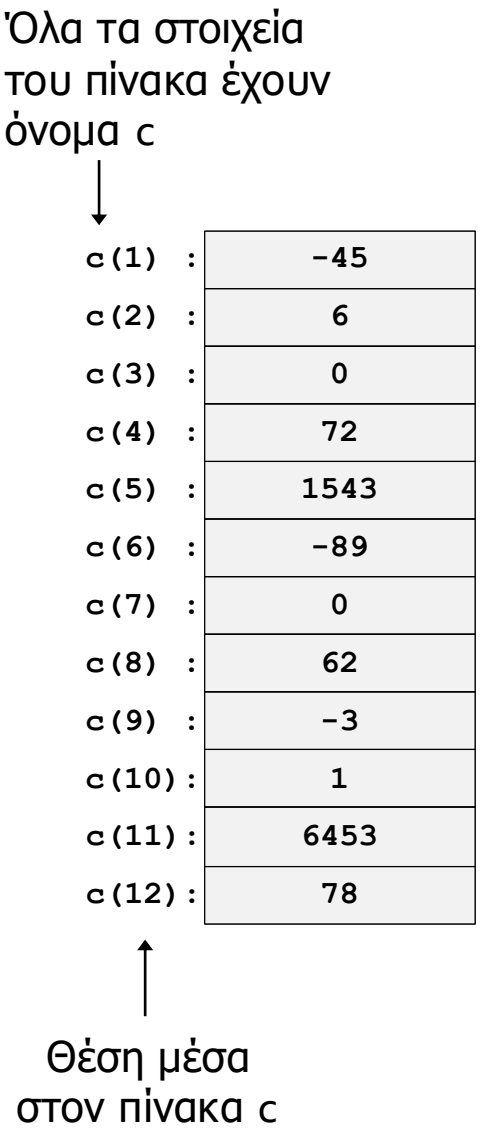
- Σύνολο από συνεχόμενες θέσεις μνήμης

- Για να προσπελάσουμε ένα στοιχείο ενός πίνακα χρειαζόμαστε:
  - Το όνομα του πίνακα
  - Τη θέση του στοιχείου
  - π.χ., `c(2) = 6;`

- Μορφή:

  arrayname`(`*position_number*`)`

  - Το πρώτο στοιχείο είναι στη θέση 1
  - Για έναν πίνακα `n` στοιχείων με όνομα `c`:
    - `c(1), c(2), ..., c(n)`



Παράδειγμα

- Θέλουμε να διαχειριστούμε τις ημερήσιες θερμοκρασίες μιας εβδομάδας:

ΚΥ	ΔΕ	ΤΡ	ΤΕ	ΠΕ	ΠΑ	ΣΑ
16	18	14	12	19	21	15

- … χρησιμοποιείται ο μονοδιάστατος πίνακας Τ.

T(1)	T(2)	T(3)	T(4)	T(5)	T(6)	T(7)
16	18	14	12	19	21	15

- … όπου η θερμοκρασία της Κυριακής είναι η `T(1)`, της Δευτέρας η `T(2)`, κοκ.

- Τα στοιχεία του πίνακα αποθηκεύονται σε διαδοχικές και συνεχόμενες θέσεις μνήμης.

Αυτόματη αρχικοποίηση

- Αρχικοποίηση πίνακα 10 στοιχείων με μηδενικά
  - Παράδειγμα: `A = zeros(1, 10);`

0	0	0	0	0	0	0	0	0	0

- Αρχικοποίηση πινάκα 10 στοιχείων με άσους
  - Παράδειγμα: `B = ones(1, 10);`

1	1	1	1	1	1	1	1	1	1

Προσπέλαση στοιχείων

- Τα στοιχεία του πίνακα χρησιμοποιούνται σαν κανονικές μεταβλητές:
  - χρησιμοποιούμε το όνομα του πίνακα και μέσα σε παρένθεση την θέση του στοιχείου που επιθυμούμε να προσπελάσουμε,
  - εκτελούμε κανονικά αριθμητικές, λογικές ή συγκριτικές πράξεις.
- Παράδειγμα:

```
A = [1, 2, 3];

A(1) =  3;
x = A(1) + A(2);

index = 2;

A(index) = 3;

disp(A(2));
```

Παράδειγμα

- Τι θα τυπώσει το παρακάτω πρόγραμμα?

```
a = [3 6 7];
b = [1 9 4 5];

c = a(2) + b(4);

disp(c);
```

- **Απάντηση:** Θα τυπώσει 11. Το b είναι ένας μονοδιάστατος πίνακας, οπότε το b(n) είναι το *n*-ιοστό στοιχείο του. Άρα το a(2) είναι 6 και το b(4) είναι 5, οπότε η μεταβλητή c θα πάρει την τιμή 6+5 = 11.

Η συνάρτηση `length`

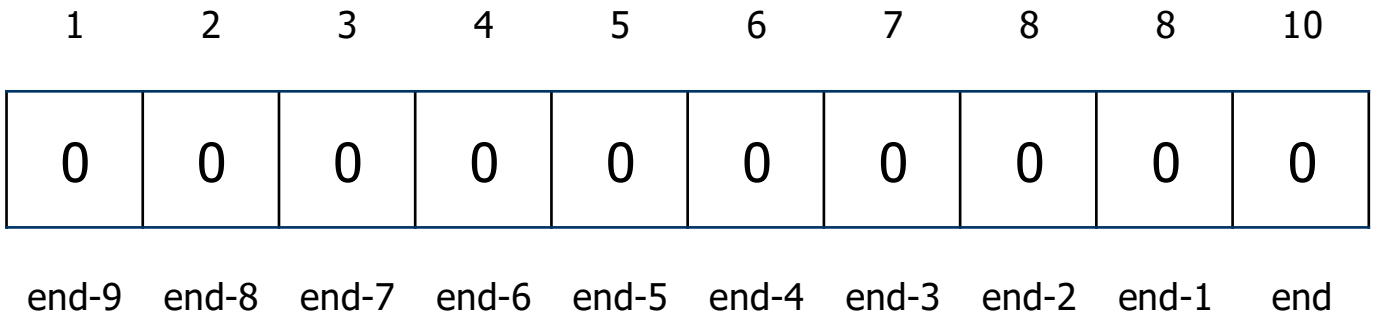
- Επιστρέφει τον αριθμό στοιχείων ενός πίνακα.

- Παραδείγματα:

```
a = [5 4 2 5 6];
len = length(a);          5

b = zeros(1, 20);
len = length(b);          20

c = ones(1, 42);
len = length(c);          42
```

Η θέση end

- Η τελευταία θέση ενός πίνακα μπορεί να προσπελαστεί μέσω της ειδικής λέξης-κλειδί end :



- Παράδειγμα:

```
arr = [ 2, 5, -1, 4, 8, 4, 2, -2, 7.5 ];

last_elem = arr(end);
```

Προσθήκη στοιχείων

- Μπορούμε να προσθέτουμε στοιχεία δυναμικά σε έναν υπάρχον μονοδιάστατο πίνακα (διάνυσμα), αμέσως μετά τη την τελευταία θέση

- Παράδειγμα:

```
x = [ 1, 2, 3 ];
x(end + 1) = 4;

disp(x);
```

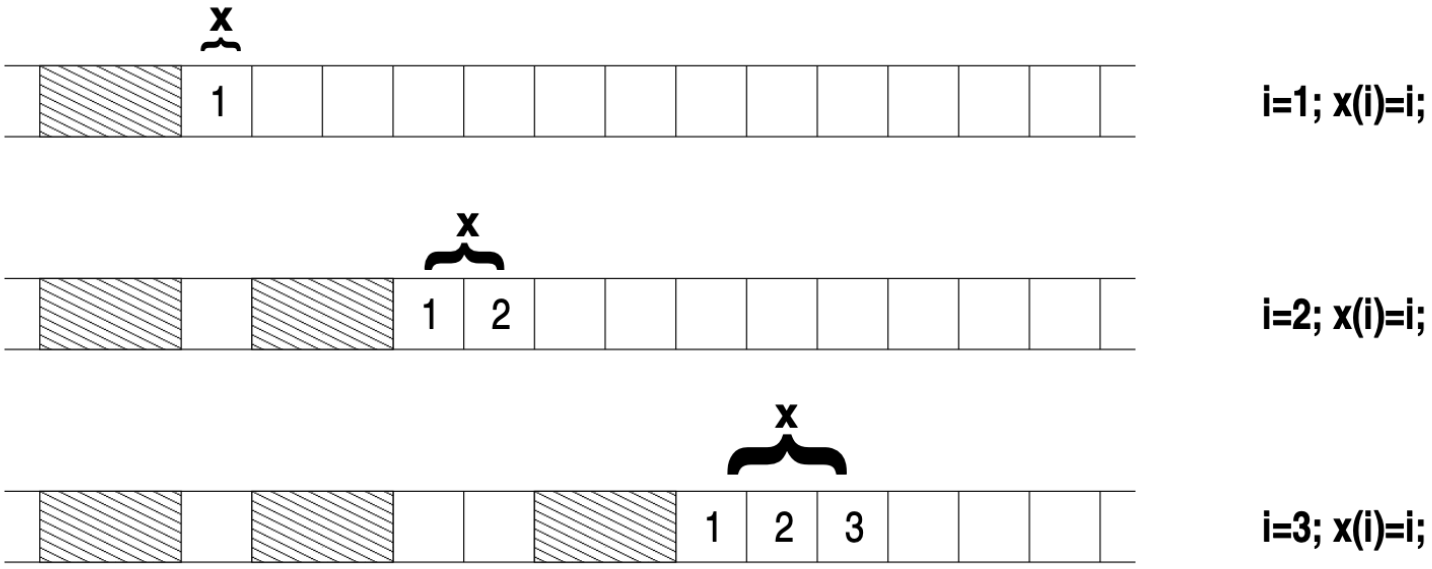
Προσθέτουμε τον αριθμό 4 στην θέση `end + 1`.

Το x εχει πλέον τέσσερα στοιχεία.

Δυναμική αύξηση του πίνακα

```
x = [ ];
for i = 1:100
        x(i) = i;
end
```

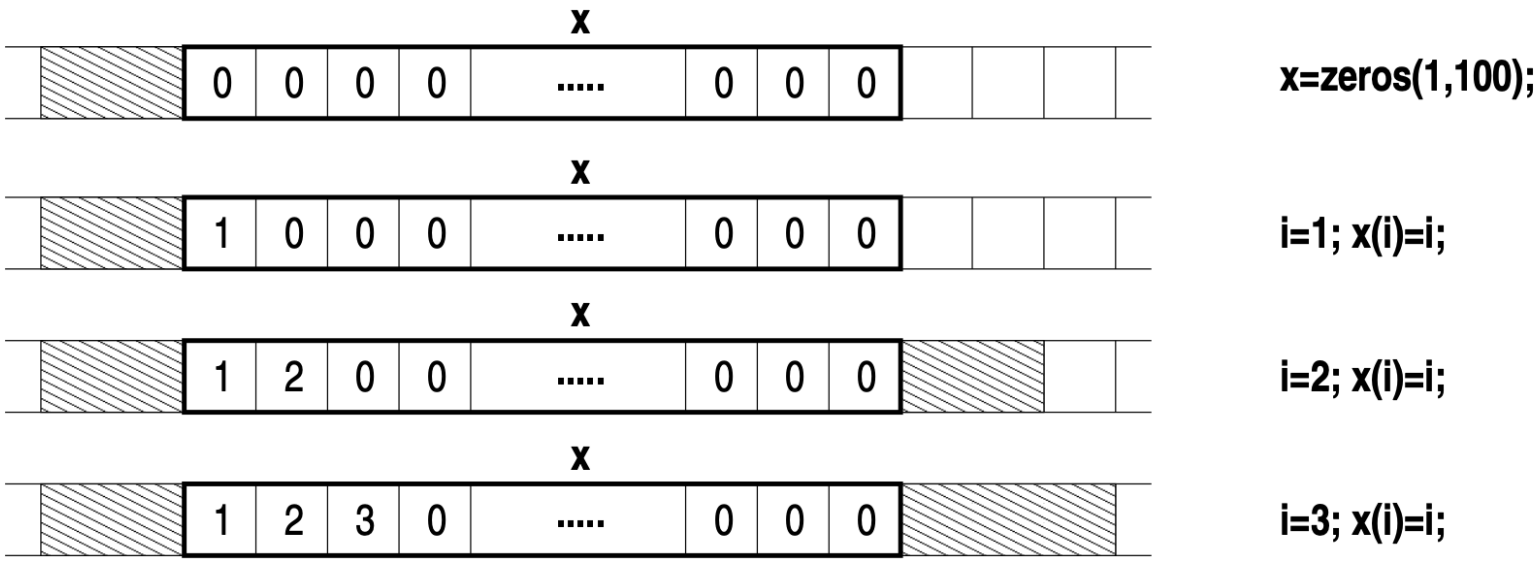
- Η εικόνα της μνήμης (ο χώρος δεσμεύεται δυναμικά):



Προ-δέσμευση του πίνακα

```
x = zeros(1, 100);
for i=1:100
        x(i) = i;
end
```

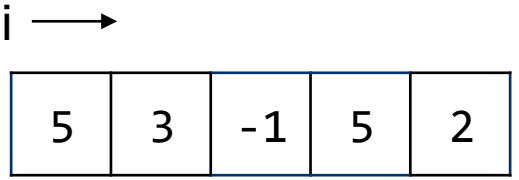
- Η εικόνα της μνήμης (προδέσμευση χώρου):



Διάσχιση πίνακα

- Χρησιμοποιούμε εντολές επανάληψης (`for`, `while`) για να διατρέξουμε όλα τα στοιχεία ενός πίνακα ή ένα τμήμα του.
  - π.χ. το πρώτο μισό του πίνακα, τα πρώτα 5 στοιχεία, τα στοιχεία από τη θέση 2 έως τη θέση 10, κλπ.
- **Παράδειγμα:**

```
array = [ 5, 3, -1, 5, 2 ];
for i = 1:length(array)
    disp(array(i));
end
```



- **Προσοχή:** οι θέσεις που προσπελαύνουμε πρέπει να είναι έγκυρες:
  - οι έγκυρες θέσεις ενός πίνακα `a` είναι μεταξύ `1` και `length(a)`,
  - αν προσπελάσουμε θέση εκτός ορίων του πίνακα, το πρόγραμμα θα πετάξει λάθος και θα τερματίσει.

Παράδειγμα 1

- Διάσχιση μονοδιάστατου πίνακα:

```
array = [ 4, -1, 9, 1, 0, -3, 6, 1, -5];
```

```
for i = 1:length(array)
    disp(array(i));
end
```

Διάσχιση πίνακα από την αρχή έως το τέλος

```
sum = 0;
for i = 1:length(array)
    sum = sum + array(i);
end
```

Διάσχιση πίνακα από την αρχή έως το τέλος και πρόσθεση όλων των στοιχείων

Παράδειγμα 2

- Τι θα τυπώσει το παρακάτω πρόγραμμα?

```
b = [3, 8, 9, 4, 7, 5];

sum1 = 0;

for k = 1:4
    sum1 = sum1 + b(k);
end

disp(sum1);
```

Απάντηση: Θα τυπώσει 24. Το πρόγραμμα προσθέτει τα στοιχεία από την θέση 1 έως και τη θέση 4 του πίνακα b.
Οπότε `sum1 = b(1) + b(2) + b(3) + b(4) = 3 + 8 + 9 + 4 = 24.`

Παράδειγμα 3

- Τι θα τυπώσει το παρακάτω πρόγραμμα?

```
b = [3, 8, 9, 4, 7, 5];

sum1 = 0;

for k = 1:2:5
    sum1 = sum1 + b(k);
end

disp(sum1);
```

Απάντηση: Θα τυπώσει 19. Το πρόγραμμα θα πραγματοποιήσει την πρόσθεση `sum1 = b(1) + b(3) + b(5) = 3 + 9 + 7 = 19`

Είσοδος πίνακα από το χρήστη

- Σειριακή αποθήκευση των στοιχείων:

```
for i = 1:7
    array(i) = input('Enter value: ');
end
```

- Ο χρήστης εισάγει ένα-ένα τα στοιχεία τα οποία αποθηκεύονται στην αντίστοιχη θέση του πίνακα `array`.

Αναζήτηση στοιχείου/ων

- Χρησιμοποιούμε εντολές επανάληψης (`for`, `while`) μαζί με ένθετες `if`.

- Η επανάληψη μπορεί να γίνει ανάλογα τις ανάγκες μας:
  - για όλα τα στοιχεία (από την αρχή μέχρι το τέλος),
  - για όσο ισχύει μια συνθήκη (π.χ., εύρεση ενός στοιχείου, κλπ.).

- Χρησιμοποιούμε μία (ή περισσότερες) ένθετες `if`:
  - οι συνθήκες επιλέγουν τα στοιχεία που επιθυμούμε τα οποία επεξεργαζόμαστε αναλόγως

Παράδειγμα

- Αναζήτηση και εύρεση στοιχείου/ων:

```
numbers = [2, -4, 4, -7, 15];
for i = 1:length(numbers)
    if numbers(i) < 0
        disp(numbers(i));
    end
end
```

Εκτύπωση όλων των αρνητικών αριθμών

```
numbers = [2, -4, 4, -7, 15];
for i = 1:length(numbers)
    if numbers(i) < 0
        disp(numbers(i));
        break;
    end
end
```

Εκτύπωση του πρώτου αρνητικού αριθμού

Άσκηση 1

- Να γράψετε ένα πρόγραμμα που να βρίσκει το μεγαλύτερο αριθμό που περιέχεται σε ένα μονοδιάστατο πίνακα.

- Λύση:

```
array = [3, -2, 6, -7, 3, 5];

maximum = array(1);

for i = 2:length(array)
    if array(i) > maximum
        maximum = array(i);
    end
end
```

Άσκηση 2

- Να γράψετε ένα πρόγραμμα που να βρίσκει το μικρότερο αριθμό που περιέχεται σε ένα μονοδιάστατο πίνακα.

- Λύση:

```
array = [3, -2, 6, -7, 3, 5];

minimum = array(1);

for i = 2:length(array)
    if array(i) < minimum
        minimum = array(i);
    end
end
```

Πράξεις με πίνακες

- Το MATLAB/Octave επιτρέπει αριθμητικές πράξεις μεταξύ πινάκων (σύμφωνα με τους κανόνες της γραμμικής άλγεβρας)
  - Πρόσθεση
  - Αφαίρεση
  - Πολλαπλασιασμός
  - Διαίρεση
  - Αναστροφή
  - Ύψωση σε δύναμη

- **Προσοχή**, οι διαστάσεις των πινάκων θα πρέπει να είναι σωστές.

Παράδειγμα 1

- Πρόσθεση πινάκων:

```
a  = [3 6 7];
b  = [1 9 4];

c  = a + b;
```

Επεξήγηση: Το `a` και το `b` είναι πίνακες τριών στοιχείων. Στην τρίτη γραμμή, η λειτουργία του `a + b` έχει ως αποτέλεσμα την προσθήκη των `a` και `b` στοιχείο-προς-στοιχείο. Επομένως, η τιμή της μεταβλητής `c` μετά την εκτέλεση του παραπάνω κώδικα είναι `[4 15 11]`.

Παράδειγμα 2

- Πρόσθεση πινάκων:

```
u = [4 0 -1 2];
v = [1, 2, -3, 1];

z = u + v;
```

Επεξήγηση: Η τιμή της μεταβλητής `c` μετά την εκτέλεση του παραπάνω κώδικα είναι `[5 2 -4 3]`. Στον πίνακα `u` χωρίσαμε τα στοιχεία με κενά, ενώ στο `v` τα χωρίσαμε με κόμματα. Οι δύο τρόποι είναι ισοδύναμοι.

Παράδειγμα 3

- Αφαίρεση πινάκων:

```
p = [7, -3, 4, 8, 10];
q = [3, 2, -9, 3, -8];

r = p - q;
```

Επεξήγηση: Η τιμή της μεταβλητής `r` μετά την εκτέλεση του παραπάνω κώδικα είναι `[4 -5 13 5 18]`. Από κάθε στοιχείο του πίνακα `p` αφαιρέσαμε το αντίστοιχο στοιχείο του πίνακα `q`.

Παράδειγμα 4

- Πολλαπλασιασμός πινάκων:

```
m = [1, 2, 3, 4; 5, 6, 7, 8];
n = [6, 7; 8, 9; 10, 11; 12, 13];

o = m * n;
```

Επεξήγηση: Η τιμή της μεταβλητής o μετά την εκτέλεση του παραπάνω κώδικα είναι `[100 110 ; 244 270]`. Ο πολλαπλασιασμός πινάκων γίνεται όπως τον γνωρίζουμε από τη γραμμική άλγεβρα.

Προσοχή, θα πρέπει το πλήθος των στηλών του πρώτου πίνακα πίνακα πρέπει να είναι ίσο με το πλήθος των γραμμών του δεύτερου πίνακα.

Ειδικοί Τελεστές

- Το MATLAB/Octave μπορεί να εκτελεί μία πράξη **ανά στοιχείο** σε ολόκληρο τον πίνακα με τη μία:

```
a = [1 2 3];
b = [4 5 6];
```

Πολλαπλασιασμός ανά στοιχείο .* :

```
a .* b
```

```
[4 10 18]
```

Διαίρεση ανά στοιχείο ./ :

```
a ./ b
```

```
[0.25 0.4 0.5]
```

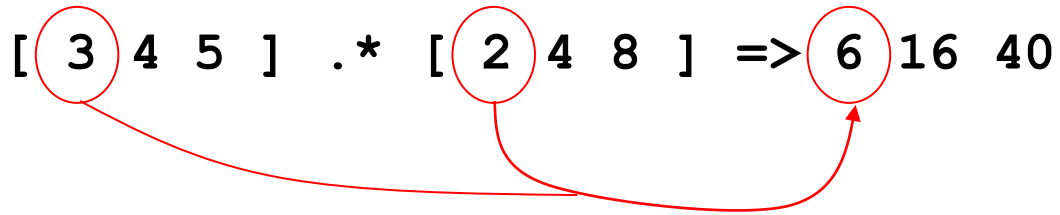
Ύψωση σε δύναμη ανά στοιχείο .^ :

```
a .^ b
```

```
[1 32 729]
```

Παράδειγμα 1

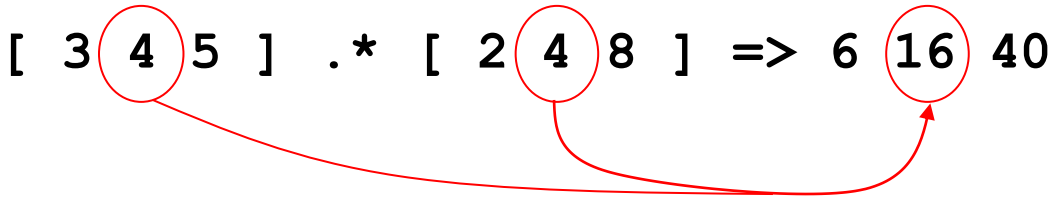
```
[3 4 5] .* 2 => 6 8 10
```



Προσέξτε την τελεία πριν τον τελεστή του πολλαπλασιασμού

Παράδειγμα 1

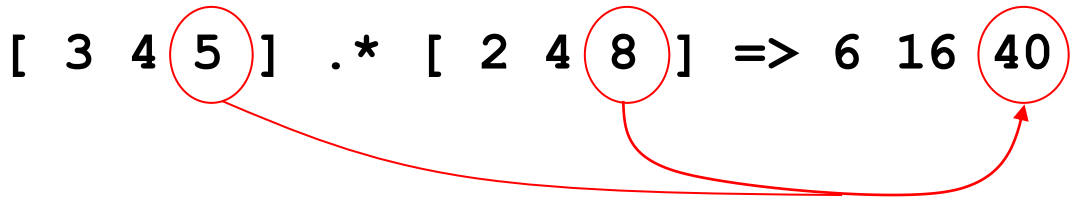
```
[3 4 5] .* 2 => 6 8 10
```



Προσέξτε την τελεία πριν τον τελεστή του πολλαπλασιασμού

Παράδειγμα 1

```
[3 4 5] .* 2 => 6 8 10
```



Προσέξτε την τελεία πριν τον τελεστή του πολλαπλασιασμού

Συχνά Λάθη (1/3)

- Όρια πίνακα: τι γίνεται αν βγούμε έξω από τον πίνακα;
- Περίπτωση 1:

```
x = [ 3, 8, -3, 15 ];

elem = x(5);

--> Error: index is out of bounds
```

Συχνά Λάθη (2/3)

- Όρια πίνακα: τι γίνεται αν βγούμε έξω από τον πίνακα;

- Περίπτωση 2:

```
x = [ 3, 8, -3, 15 ];

sum = 0;
for i=1:5
        sum = sum + x(i);
end

disp(sum);
```

```
--> Error: index is out of bounds
```

Συχνά Λάθη (3/3)

- Όρια πίνακα: τι γίνεται αν βγούμε έξω από τον πίνακα;

- Περίπτωση 3:

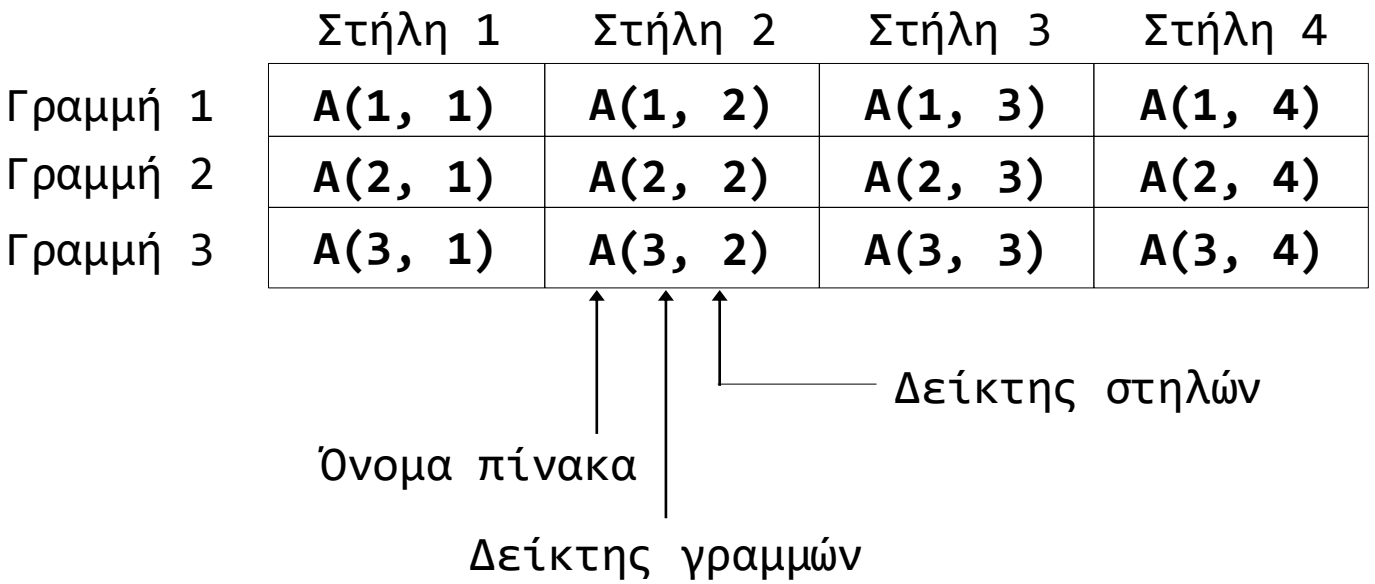
```
x = [ 3, 8, -3, 15 ];

a = x(0);
```

```
Error: index values must be >= 1
```

Δισδιάστατοι πίνακες

Διδιάστατοι Πίνακες

- Πίνακες με *m* γραμμές και *n* στήλες (*m* x *n* πίνακας)
  - Όπως και στα μαθηματικά: πρώτα γραμμή και μετά στήλη



Παράδειγμα χρήσης

- Αν *n* είναι το πλήθος των φοιτητών/τριών που εξετάστηκαν στα 5 μαθήματα του Β εξαμήνου, τότε οι βαθμολογίες αναπαρίστανται σαν ένας πίνακας *nx5* (*n* γραμμών και 5 στηλών):

Α.Μ	Δομημένος Προγραμ/μός	Αρχές Λογιστικής	Πιθανότητες και Στατιστική	Μάρκετινγκ	Γραμ. Άλγεβρα κ Διακριτά Μαθ/κά
11000	5	9	2	7	6
22222	10	8	6	0	7
50001	7	2	5	8	8
….	…	…	…	…	6
30712	6	6	6	4	3

- Κάθε γραμμή του πίνακα έχει υποχρεωτικά τον ίδιο αριθμό στηλών:
  - ο φοιτητής 11000 είτε έχει πάρει 9 στο μάθημα 'Αρχές Λογιστικής'
  - ο φοιτητής 22222 είτε έχει πάρει 0 στο μάθημα 'Μάρκετινγκ' ή δεν έχει εξεταστεί σε αυτό

Βασικές λειτουργίες

- **Αρχικοποίηση:** `array = [1 2; 2 3; 4 4];`
  - *Προσέξτε* ότι για τη δήλωση του δισδιάστατου πίνακα, χωρίσαμε τις γραμμές με ερωτηματικό.

- **Προσπέλαση τιμής:** `x = array(3, 2);`
  - Διαβάζουμε την τιμή του πίνακα που βρίσκεται στην τρίτη γραμμή και δεύτερη στήλη και την αποθηκεύουμε στη μεταβλητή x

- **Αποθήκευση τιμής:** `array(1, 2) = 3;`
  - Αποθηκεύουμε την τιμή 3 στη θέση του πίνακα που βρίσκεται στην πρώτη γραμμή και δεύτερη στήλη

Πίνακες – Δημιουργία / Αρχικοποίηση

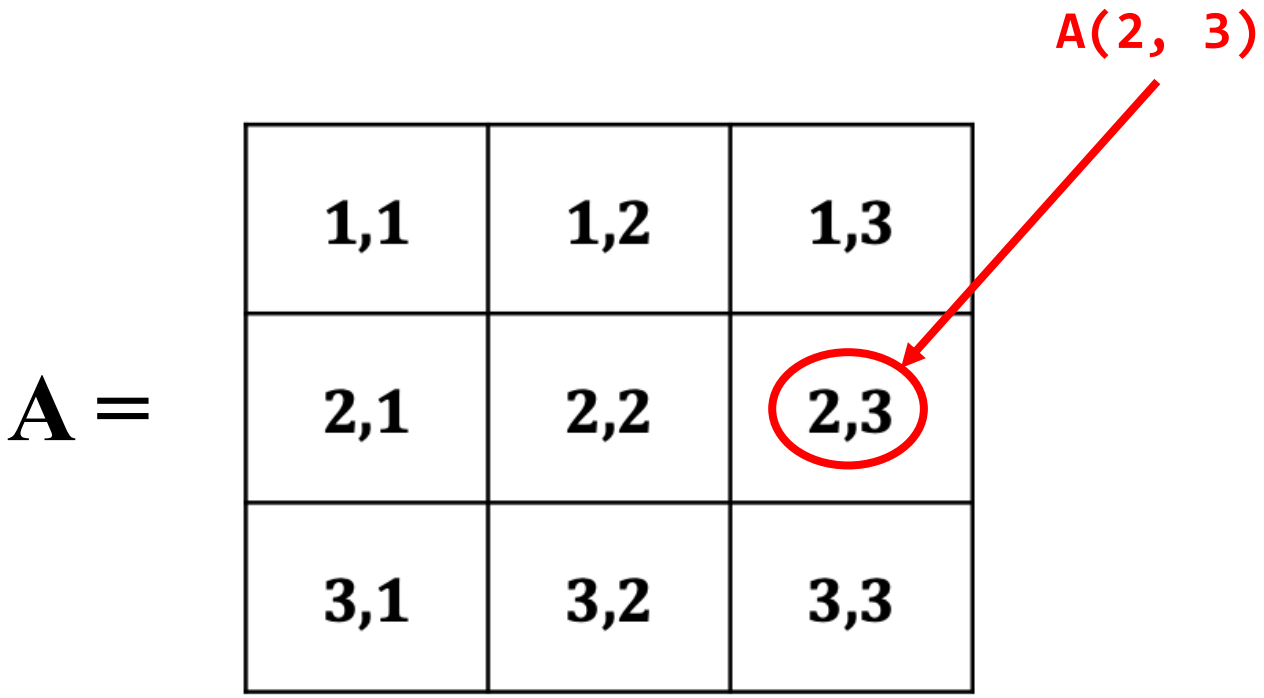
- Παράδειγμα: Αρχικοποίηση πίνακα 5x5 στοιχείων με μηδενικά

```
B = zeros(5,5);
```

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Διδιάστατοι Πίνακες – Προσπέλαση

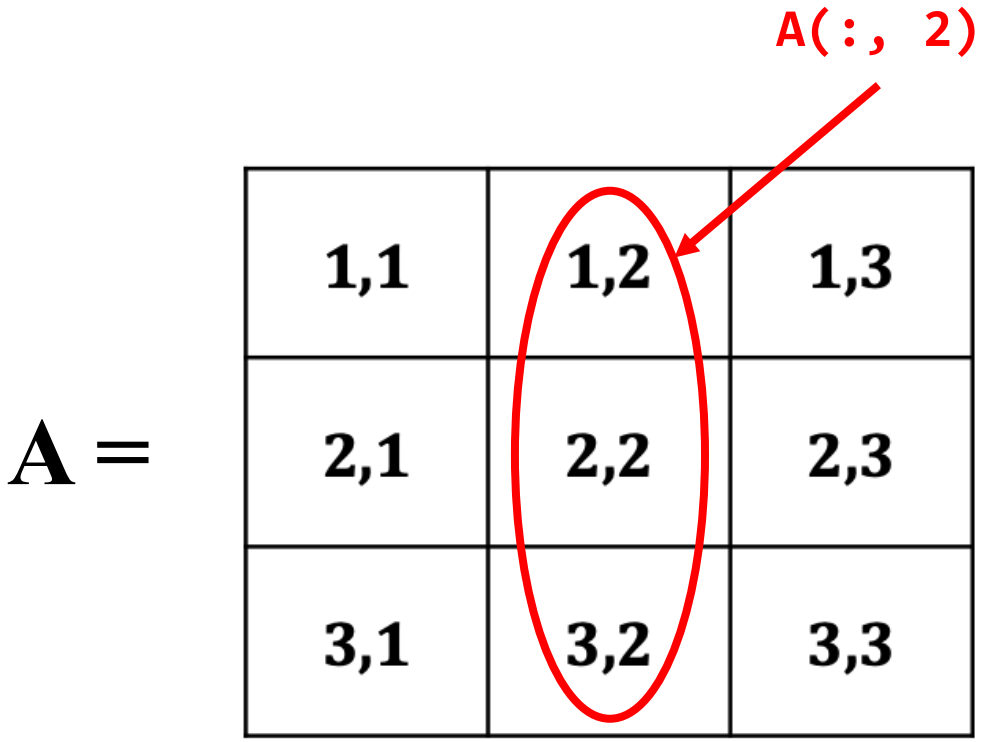
- Προσπέλαση ενός συγκεκριμένου στοιχείου:




```
x = A(2, 3);
disp(x);
```

Δισδιάστατοι Πίνακες – Προσπέλαση

- Προσπέλαση μιας ολόκληρης στήλης




```
x = A(:, 2);
disp(x);
```

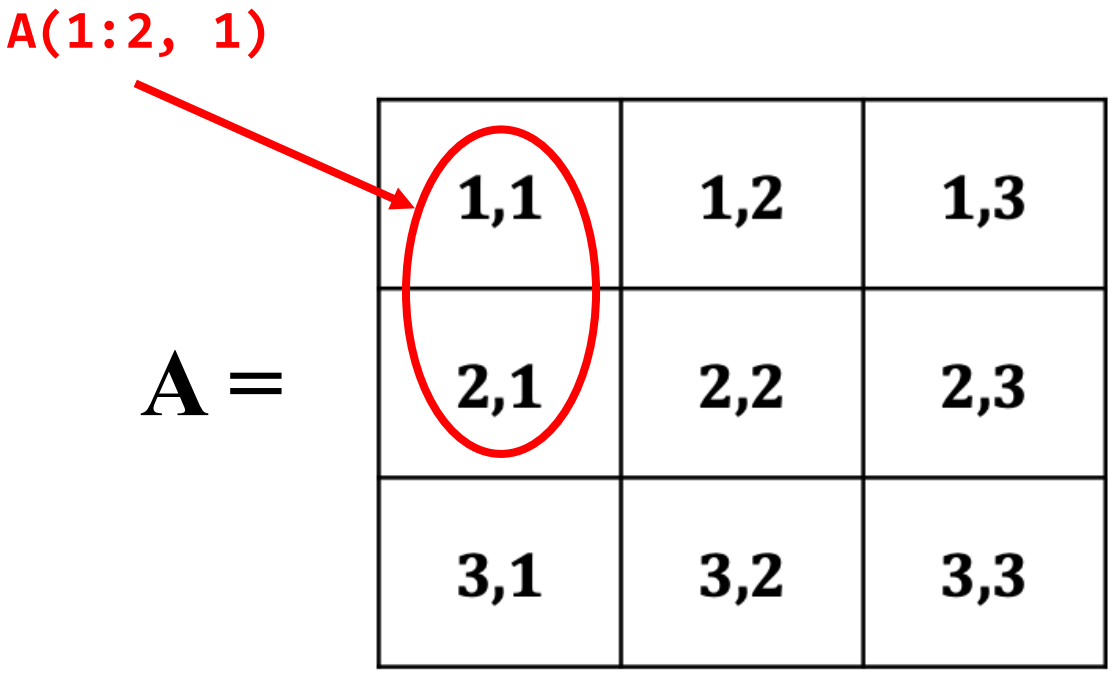
Διδιάστατοι Πίνακες – Προσπέλαση

- Προσπέλαση μιας ολόκληρης γραμμής


```
x = A(1, :);
disp(x);
```

Δισδιάστατοι Πίνακες – Προσπέλαση

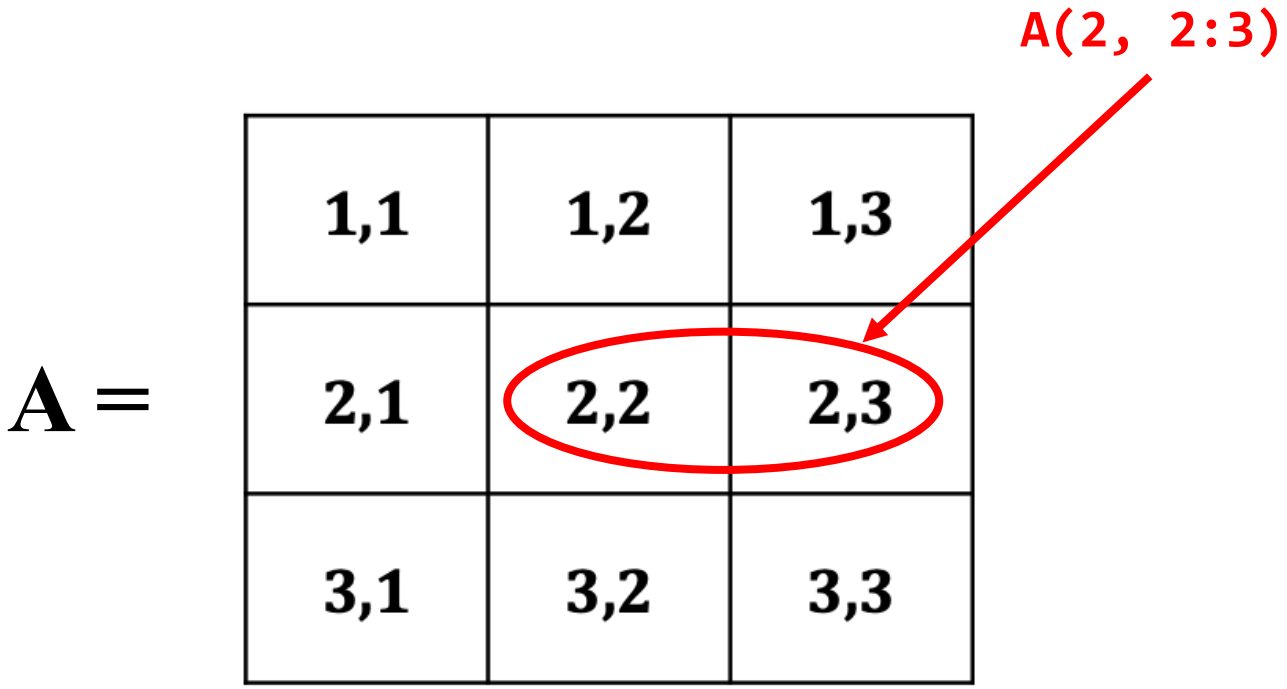
- Προσπέλαση υπο-στήλης




```
x = A(1:2, 1);
disp(x);
```

Διδιάστατοι Πίνακες – Προσπέλαση

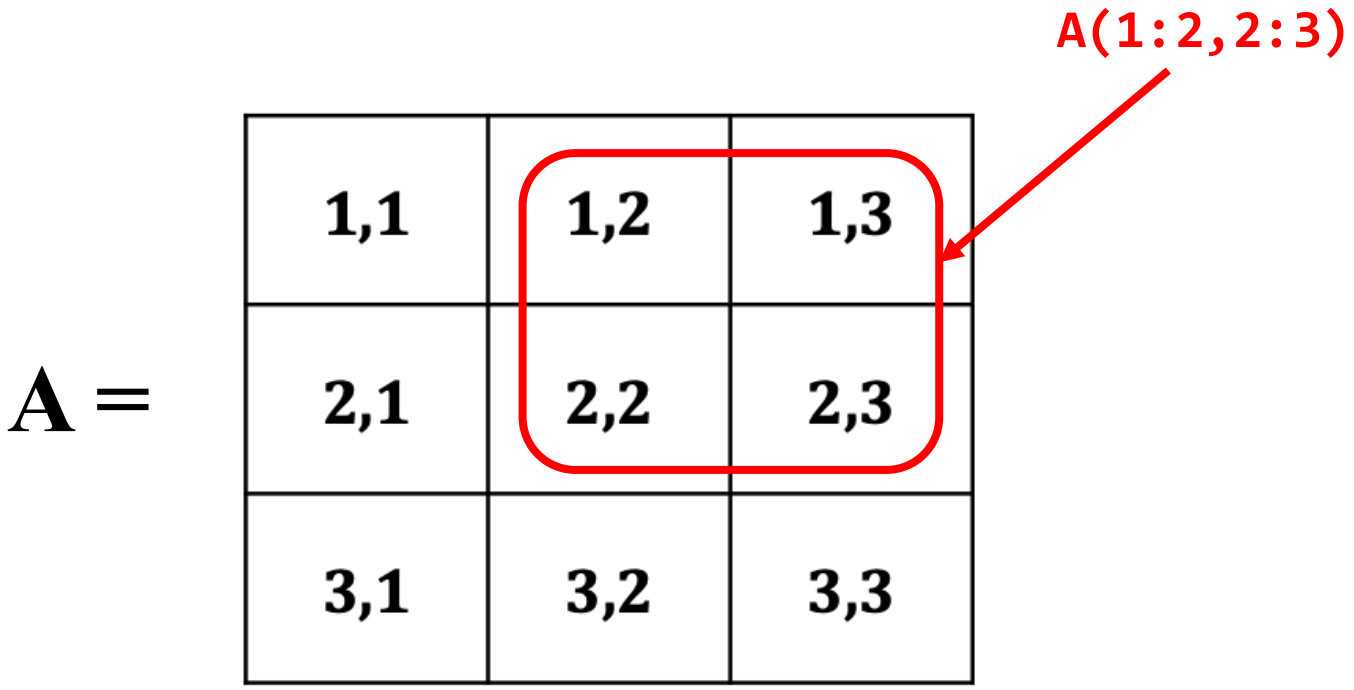
- Προσπέλαση υπο-γραμμής




```
x = A(2, 2:3);
disp(x);
```

Δισδιάστατοι Πίνακες – Προσπέλαση

- Προσπέλαση υπο-πίνακα




```
x = A(1:2, 2:3);
disp(x);
```

Ερώτηση κατανόησης

- Έστω ο πίνακας:

A =

3	6	8	2
8	5	1	3
5	4	7	1

  - Τι θα τυπώσουν οι παρακάτω εκφράσεις?

    - `A(1,2)`
    - `A(1, end)`
    - `A(end,end)`
    - `A(end-2,4)`
    - `A(3,:)`
    - `A(:,2)`
    - `A(:,:)`
    - `A(A(1,1),A(3,4))`

Είσοδος στοιχείων πίνακα από το χρήστη

- Ο χρήστης εισάγει ένα-ένα τα στοιχεία:

```
for i = 1:100
  for j = 1:3
    M(i,j) = input('Δώσε τιμή: ');
  end
end
```

- Κάθε στοιχείο αποθηκεύεται στην θέση που προσδιορίζεται από τη μεταβλητή `i` και τη μεταβλητή `j`.

Πράξεις με πίνακες

- Το Matlab/Octave επιτρέπει αριθμητικές πράξεις μεταξύ δισδιάστατων πινάκων, όπως και πράξεις ανά στοιχείο (αντίστοιχα με τους μονοδιάστατους)
  - Πρόσθεση
  - Αφαίρεση
  - Πολλαπλασιασμός
  - Διαίρεση
  - Αναστροφή
  - Ύψωση σε δύναμη

- **Προσοχή**, οι διαστάσεις των πινάκων θα πρέπει κάθε φορά να είναι σωστές για να γίνουν οι πράξεις

Παράδειγμα 1

- Πρόσθεση πινάκων:

```
X = [1 2 ; 3 4];
Y = [3 1 ; 1 3];

Z = X + Y;

disp(Z);
```

- Έξοδος

```
Z =
        4     3
        4     7
```

Παρατήρηση: Το X και το Y είναι πίνακες 2x2, επομένως η πράξη X+Y έχει ως αποτέλεσμα την πρόσθεση των X και Y στοιχείο-προς-στοιχείο.

Παράδειγμα 2

- Αφαίρεση πινάκων:

```
X = [1 2 ; 3 4];
Y = [3 1 ; 1 3];

Z = X – Y;

disp(Z);
```

- Έξοδος

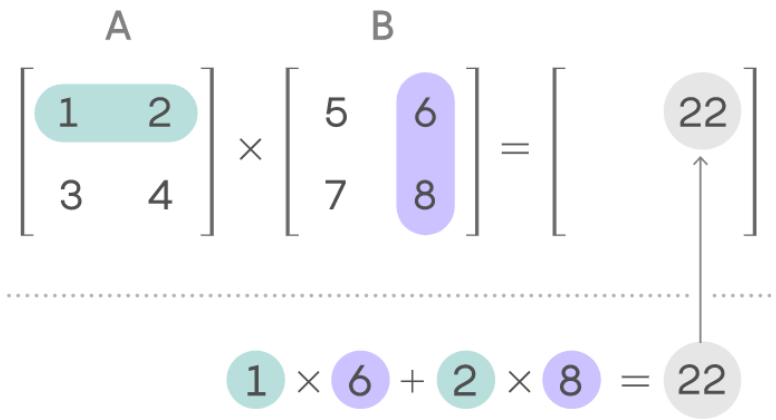
```
Z =
      -2     1
       2     1
```

Παρατήρηση: Το X και το Y είναι πίνακες 2x2, επομένως η πράξη X-Y έχει ως αποτέλεσμα την αφαίρεση των X και Y στοιχείο-προς-στοιχείο.

Παράδειγμα 3

- Πολλαπλασιασμός πινάκων:

```
A = [1 2; 3 4];
B = [5 6; 7 8];

C = A * B;

disp(C);
```



- Έξοδος

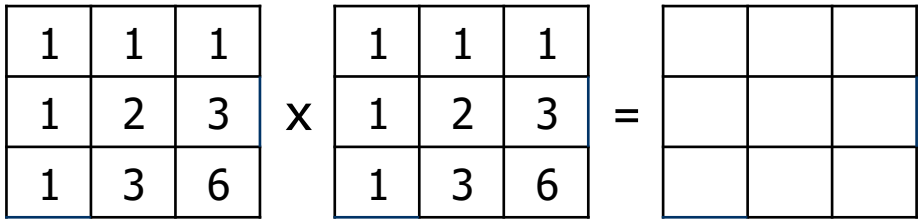
```
C =

          19          22
          43          50
```

Παρατήρηση: Ο πολλαπλασιασμός πινάκων γίνεται σύμφωνα με τους κανόνες της γραμμικής άλγεβρας.

Παράδειγμα 4

- Πολλαπλασιασμός πινάκων:

```
A = [1 1 1; 1 2 3; 1 3 6];

X = A * A;

disp(X);
```



- Έξοδος

```
X =

     3     6    10
     6    14    25
    10    25    46
```

Παρατήρηση: Το A*A είναι ισοδύναμο με A^2

Παράδειγμα 5

- Ύψωση των στοιχείων ενός πίνακα σε δύναμη:

```
A = [1 1 1; 1 2 3; 1 3 6];

X = A .^ 2;

disp(X);
```

1	1	1
1	2	3
1	3	6

.^2 =

1^2	1^2	1^2
1^2	2^2	3^2
1^2	3^2	6^2

- Έξοδος

```
X =
     1     1     1
     1     4     9
     1     9    36
```

Παρατήρηση: Το `A^2` είναι διαφορετικό από το `A.^2`

Ένωση πινάκων

- Μπορούμε να ενώσουμε ή να επεκτείνουμε πίνακες χρησιμοποιώντας τον τελεστή αγκύλης [ ]
  - Για παράδειγμα, η έκφραση [A, B] (ή [A B]) ενώνει τους πίνακες A και B οριζόντια, ενώ η έκφραση [A; B] τους ενώνει κατακόρυφα

- Παράδειγμα:

```
A = [1, 1, 1, 1];
B = [0, 0, 0, 0];

C = [A B];
```

1	1	1	1	0	0	0	0

```
D = [A; B];
```

1	1	1	1
0	0	0	0

Οι συναρτήσεις `length` και `size`

- `length(…)`
  - *Αν η παράμετρος είναι ένας μονοδιάστατος πίνακας*, τότε επιστρέφει το μήκος του πίνακα (δηλαδή τον αριθμό των στοιχείων που περιέχει).
  - *Αν η παράμετρος είναι ένας δισδιάστατος πίνακας*, τότε επιστρέφει το μήκος της μεγαλύτερης από τις δύο διαστάσεις.

- `size(…)`
  - Επιστρέφει έναν πίνακα δύο θέσεων με τις διαστάσεις (μέγεθος) του πίνακα:
    - `size(A)` επιστρέφει έναν πίνακα δύο στοιχείων: το πρώτο στοιχείο είναι ο αριθμός γραμμών, και το δεύτερο στοιχείο ο αριθμός στηλών.
  - Αν θέλουμε μόνο τον αριθμό των γραμμών:
    - `size(A, 1)` επιστρέφει τον αριθμό γραμμών (p) του `A`
  - Αν θέλουμε μόνο τον αριθμό των στηλών:
    - `size(A, 2)` επιστρέφει τον αριθμό των στηλών (q) του `A`.
  - Σημείωση: Ένας μονοδιάστατος πίνακας x είναι ουσιαστικά ένας πίνακας 1 x n (με μία γραμμή και n στήλες.
    - Άρα το πλήθος των στοιχείων του προκύπτει ως: `size(x, 2)`

Παράδειγμα 1

- Τί θα επιστρέψει η `length` σε κάθε μία από τις παρακάτω περιπτώσεις?

```
a = [7 1 -3 2 1 5];
len = length(a);
```
→ 6

```
b = [-2 3 5; 3 -5 3; 1 0 -2];
len = length(b);
```
→ 3

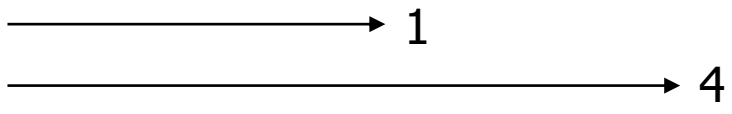
```
d = [-3 1 8 5; 6 7 -9 3];
len = length(d);
```
→ 4

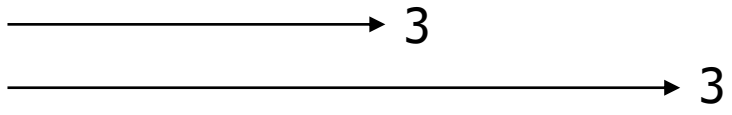
```
e = [-3; 6; 1; 8; 4; -2; 0];
len = length(e);
```
→ 7

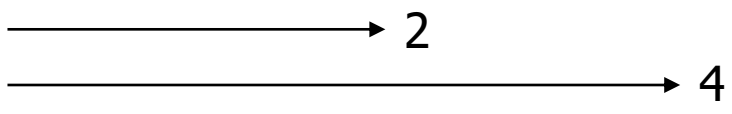
Παράδειγμα 2

- Τί θα επιστρέψει η `size` σε κάθε μία από τις παρακάτω περιπτώσεις?

```
a = [4 1 5 8];
rows = size(a, 1);
cols = size(a, 2);

b = [2 -4 5; 7 -1 5; 9 10 8];
rows = size(b, 1);
cols = size(b, 2);

d = [-3 1 8 5; 6 7 -9 3];
rows = size(d, 1);
cols = size(d, 2);
```



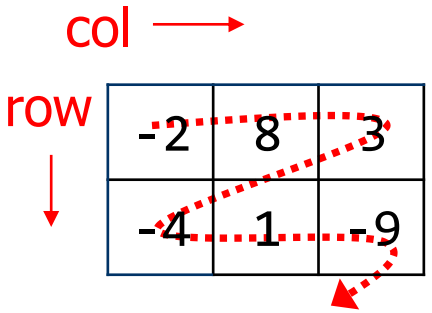




Διάσχιση δισδιάστατου πίνακα

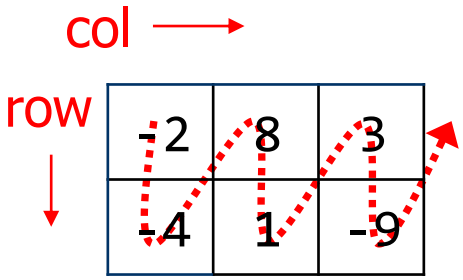
- Διάσχιση δισδιάστατου πίνακα *κατά γραμμή*

```
array = [ -2, 8, 3; -4, 1, -9 ];

for row = 1:size(array, 1)
    for col = 1:size(array, 2)
        disp(array(row, col));
    end
end
```



- Διάσχιση δισδιάστατου πίνακα *κατά στήλη*

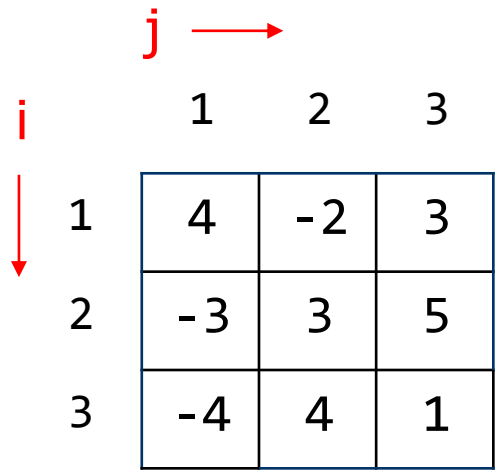
```
for col = 1:size(array, 2)
    for row = 1:size(array, 1)
        disp(array(row, col));
    end
end
```



Παράδειγμα 1

- Μέγιστο και ελάχιστο στοιχείο δισδιάστατου πίνακα

```
array = [4, -2, 3; -3, 3, 5; -4, 4, 1];

minimum = array(1, 1);
maximum = array(1, 1);

for i = 1:size(array, 1)
    for j = 1:size(array, 2)
        if array(i,j) < minimum
            minimum = array(i, j);
        end
        if array(i,j) > maximum
            maximum = array(i, j);
        end
    end
end
```



Άσκηση 1

- Να γράψετε ένα πρόγραμμα που να υπολογίζει το άθροισμα όλων των στοιχείων ενός δισδιάστατου πίνακα.

- Λύση:

```
array = [1 2 4; 5 -4 2; 2 -1 2];

sum = 0;

for i=1:size(array, 1)
    for j=1:size(array, 2)
        sum = sum + array(i,j);
    end
end
```

Άσκηση 2

- Να γράψετε ένα πρόγραμμα που να υπολογίζει το άθροισμα όλων των θετικών στοιχείων ενός δισδιάστατου πίνακα.

- Λύση:

```
array = [1 2 4; 5 -4 2; 2 -1 2];

sum = 0;

for i = 1:size(array, 1)
    for j = 1:size(array, 2)
        if array(i,j) > 0
            sum = sum + array(i,j);
        end
    end
end
```

Άσκηση 3

- Να γράψετε ένα πρόγραμμα που να υπολογίζει το πλήθος όλων των θετικών στοιχείων ενός δισδιάστατου πίνακα.

- Λύση:

```
array = [1 2 4; 5 -4 2; 2 -1 2];

count = 0;

for i = 1:size(array, 1)
    for j = 1:size(array, 2)
        if array(i,j) > 0
            count = count + 1;
        end
    end
end
```

Άσκηση 4: Μέσος όρος μαθημάτων

- Να γράψετε ένα πρόγραμμα που να ζητάει από το χρήστη τους βαθμούς 30 μαθητών σε 5 μαθήματα και να τους καταχωρεί σε ένα δισδιάστατο πίνακα.

- Στη συνέχεια το πρόγραμμα:
  - Θα υπολογίζει το μέσο όρο του κάθε μαθητή.
  - Θα υπολογίζει τη μέση βαθμολογία κάθε μαθήματος.

Άσκηση 4: Μέσος όρος μαθημάτων

- Λύση:

```
grades = zeros(30, 5);

for i = 1:30
    fprintf('Φοιτητής/τρια %d \n', i);
    for j = 1:5
        grades(i, j) = input('Βαθμός: ');
    end
end

for i = 1:30
    sum = 0;
    for j = 1:5
        sum = sum + grades(i, j);
    end
    fprintf('Μέσος όρος Φοιτητής/τρια %d \n', sum/5);
end

 for j = 1:5
    sum = 0;
    for i = 1:30
        sum = sum + grades(i, j);
    end
    fprintf('Μέσος όρος μαθήματος %d \n', sum/30);
end
```