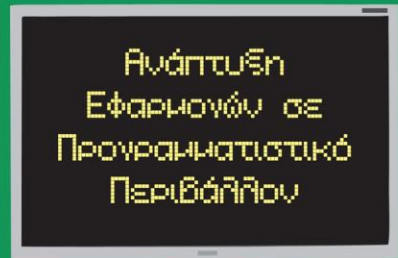


# Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον:

## Όλη η Θεωρία του ΑΕΠΠ για τις Πανελλήνιες 2023

4<sup>η</sup> έκδοση



ΒΙΒΛΙΟ ΜΑΘΗΤΗ

Γ' ΓΕΝΙΚΟΥ ΛΥΚΕΙΟΥ

Ομάδας Προσανατολισμού Θετικών Σπουδών και Σπουδών Οικονομίας & Πληροφορικής



ΙΝΣΤΙΤΟΥΤΟ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΕΚΔΟΣΕΙΣ «ΔΙΟΦΑΝΤΟΣ»

Δημήτρης Βαγιακάκος  
SVISJP

[www.youtube.com/@TuxHouseEdu](https://www.youtube.com/@TuxHouseEdu)

<https://svl.sjp.github.io/whoami/>



ΔΩΡΕΑΝ Hardcore λυμένες Ασκήσεις, η θεωρία και βοηθητικές σημειώσεις για Πανελλήνιες



# ΔΙΚΑΙΩΜΑΤΑ ΧΡΗΣΗΣ PANELLINIES\_AEPP

Η χρήση του Panellinies\_AEPP είναι δωρεάν για όλους τους αναγνώστες. Επιπλέον, επιτρέπεται η χρήση του Panellinies\_AEPP για διδασκαλία εφόσον συνοδεύεται με το logo του TuxHouse όπως και έχει δημιουργηθεί από το [επίσημο site του Panellinies\\_AEPP](#).

Απαγορεύεται οποιαδήποτε πώληση οποιουδήποτε μέρους του συγκεκριμένου έργου, όπως επίσης και η μεταφόρτωση του σε servers τρίτων χωρίς την γραπτή συγκατάθεση μου. Για οποιαδήποτε περαιτέρω πληροφορία, μπορείτε να επικοινωνήσετε μαζί μου στα social media του TuxHouse όπου θα βρείτε εδώ: <https://svlsjp.github.io/whoami/>



# ΤΙ ΕΊΝΑΙ ΑΛΓΟΡΙΘΜΟΣ;

- Αλγόριθμος είναι μια πεπερασμένη σειρά ενεργειών, αυστηρά καθορισμένων και εκτελέσιμων σε πεπερασμένο χρόνο, που στοχεύουν στην επίλυση ενός προβλήματος.





# ΚΡΙΤΗΡΙΑ ΠΟΥ ΠΡΕΠΕΙ ΝΑ ΙΚΑΝΟΠΟΙΕΪ Ο ΚΑΘΕ ΑΛΓΟΡΙΘΜΟΣ:

- Είσοδος (input). Καμία, μία ή περισσότερες τιμές δεδομένων πρέπει να δίνονται ως είσοδοι στον αλγόριθμο. Η περίπτωση που δεν δίνονται τιμές δεδομένων εμφανίζεται, όταν ο αλγόριθμος δημιουργεί και επεξεργάζεται κάποιες πρωτογενείς τιμές με τη βοήθεια συναρτήσεων παραγωγής τυχαίων αριθμών ή με τη βοήθεια άλλων απλών εντολών.
- Έξοδος (output). Ο αλγόριθμος πρέπει να δημιουργεί τουλάχιστον μία τιμή δεδομένων ως αποτέλεσμα προς το χρήστη ή προς έναν άλλο αλγόριθμο.
- Καθοριστικότητα (definiteness). Κάθε εντολή πρέπει να καθορίζεται χωρίς καμία αμφιβολία για τον τρόπο εκτέλεσής της. πχ , μία εντολή διαίρεσης πρέπει να θεωρεί και την περίπτωση όπου ο διαιρέτης λαμβάνει μηδενική τιμή. ( δεν μπορούμε να διαιρέσουμε με το 0 )
- Περαιτότητα (finiteness). Ο αλγόριθμος να τελειώνει μετά από πεπερασμένα βήματα εκτέλεσης των εντολών του. Μία διαδικασία που δεν τελειώνει μετά από ένα συγκεκριμένο αριθμό βημάτων δεν αποτελεί αλγόριθμο, αλλά λέγεται απλά υπολογιστική διαδικασία(computational procedure).
- Αποτελεσματικότητα (effectiveness). Κάθε μεμονωμένη εντολή του αλγορίθμου να είναι απλή. Αυτό σημαίνει ότι μία εντολή δεν αρκεί να έχει ορισθεί, αλλά πρέπει να είναι και εκτελέσιμη.



Με ελεύθερο κείμενο (free text), που αποτελεί τον πιο ανεπεξέργαστο και αδόμητο τρόπο παρουσίασης αλγορίθμου. Έτσι εγκυμονεί τον κίνδυνο ότι μπορεί εύκολα να οδηγήσει σε μη εκτελέσιμη παρουσίαση παραβιάζοντας το τελευταίο χαρακτηριστικό των αλγορίθμων, δηλαδή την αποτελεσματικότητα!

Με διαγραμματικές τεχνικές (diagramming techniques), που συνιστούν ένα γραφικό τρόπο παρουσίασης του αλγορίθμου. Από τις διάφορες διαγραμματικές τεχνικές που έχουν επινοηθεί, η πιο παλιά και η πιο γνωστή ίσως, είναι το διάγραμμα ροής (flow chart). Ωστόσο η χρήση διαγραμμάτων ροής για την παρουσίαση αλγορίθμων δεν αποτελεί την καλύτερη λύση.

Με φυσική γλώσσα (natural language) κατά βήματα. Στην περίπτωση αυτή χρειάζεται προσοχή, γιατί μπορεί να παραβιασθεί το τρίτο βασικό χαρακτηριστικό ενός αλγορίθμου, την καθοριστικότητα!

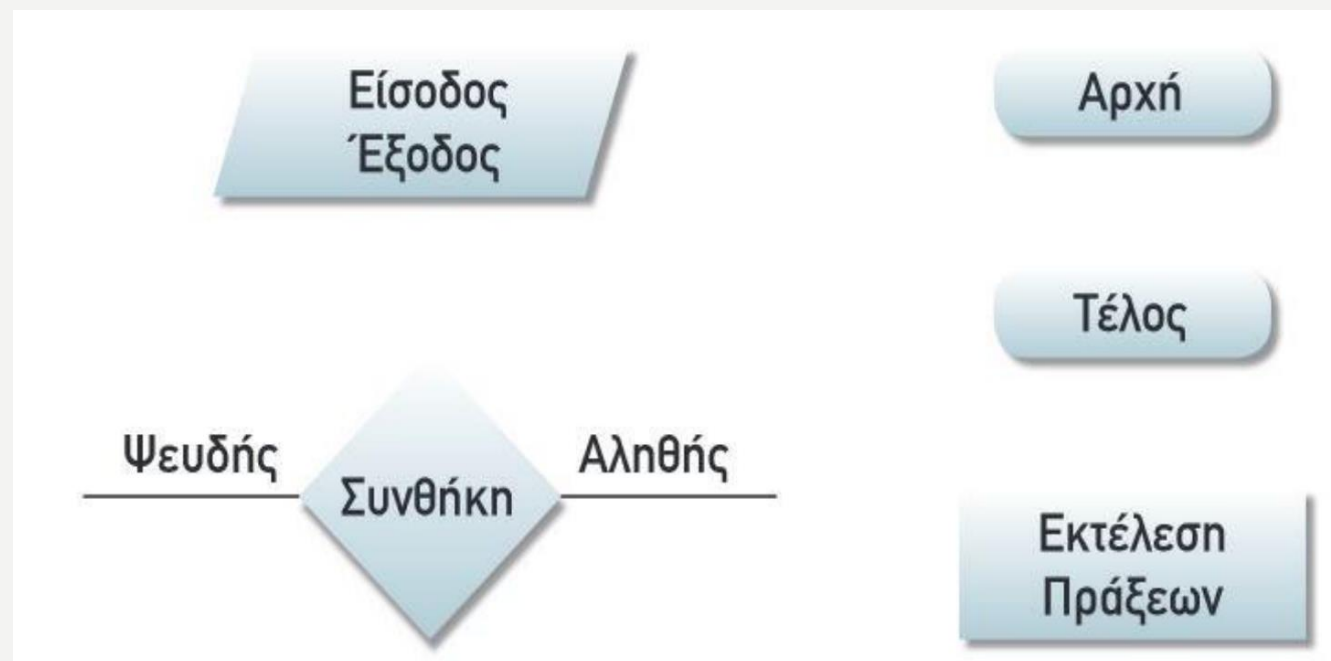
Με κωδικοποίηση (coding), δηλαδή με ένα πρόγραμμα γραμμένο είτε σε μία ψευδογλώσσα είτε σε κάποια γλώσσα προγραμματισμού που όταν εκτελεσθεί θα δώσει τα ίδια αποτελέσματα με τον αλγόριθμο.

## ΤΡΟΠΟΙ ΑΝΑΠΑΡΑΣΤΑΣΗΣ ΑΛΓΟΡΙΘΜΟΥ :



# ΔΙΑΓΡΑΜΜΑΤΑ ΡΟΪΣ:

- Έλλειψη: δηλώνει την αρχή και το τέλος του κάθε αλγορίθμου.
- Ρόμβος : δηλώνει μία ερώτηση με δύο ή περισσότερες εξόδους για απάντηση.
- Ορθογώνιο: δηλώνει την εκτέλεση μίας ή περισσότερων πράξεων.
- Πλάγιο παραλληλόγραμμο: δηλώνει είσοδο ή έξοδο στοιχείων.



# ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ ΑΛΓΟΡΙΘΜΩΝ

- Α) Σταθερές (constants). Με τον όρο αυτό αναφερόμαστε σε προκαθορισμένες τιμές που παραμένουν αμετάβλητες σε όλη τη διάρκεια της εκτέλεσης ενός αλγορίθμου. Οι σταθερές διακρίνονται σε
  - 1) Αριθμητικές, π.χ. 123, +5, -1,25
  - 2) Αλφαριθμητικές π.χ. "Τιμή", "Κατάσταση αποτελεσμάτων"
  - 3) Λογικές που είναι ακριβώς δύο, Αληθής και Ψευδής
- Β) Μεταβλητές (variables). Μια μεταβλητή είναι ένα γλωσσικό αντικείμενο, που χρησιμοποιείται για να παραστήσει ένα στοιχείο δεδομένου. Στη μεταβλητή εκχωρείται μια τιμή, η οποία μπορεί να αλλάζει κατά τη διάρκεια εκτέλεσης του αλγορίθμου. Ανάλογα με το είδος της τιμής που μπορούν να λάβουν, οι μεταβλητές διακρίνονται σε αριθμητικές, αλφαριθμητικές και λογικές.
- Γ) Τελεστές (operators). Πρόκειται για τα γνωστά σύμβολα που χρησιμοποιούνται στις διάφορες πράξεις. Οι τελεστές διακρίνονται σε αριθμητικούς, λογικούς και συγκριτικούς.
- Δ) Εκφράσεις (expressions). Οι εκφράσεις διαμορφώνονται από τους τελεστέους (operands), που είναι σταθερές και μεταβλητές και από τους τελεστές. Η διεργασία αποτίμησης μιας έκφρασης συνίσταται στην απόδοση τιμών στις μεταβλητές και στην εκτέλεση των πράξεων. Η τελική τιμή μιας έκφρασης εξαρτάται από την ιεραρχία των πράξεων και τη χρήση των παρενθέσεων. Μια έκφραση μπορεί να αποτελείται από μια μόνο μεταβλητή ή σταθερά μέχρι μια πολύπλοκη μαθηματική παράσταση.



# ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ

- **Ακέραιος τύπος.** Ο τύπος αυτός περιλαμβάνει τους ακέραιους που είναι γνωστοί από τα μαθηματικά. Οι ακέραιοι μπορούν να είναι θετικοί, αρνητικοί ή μηδέν. Παραδείγματα ακεραίων είναι οι αριθμοί **1, 3409, 0, -980**.
- **Πραγματικός τύπος.** Ο τύπος αυτός περιλαμβάνει τους πραγματικούς αριθμούς που γνωρίζουμε από τα μαθηματικά. Οι αριθμοί **3.14159, 2.71828, -112.45, 0.45** είναι πραγματικοί αριθμοί. Και οι πραγματικοί αριθμοί μπορούν να είναι θετικοί, αρνητικοί ή μηδέν.
- **Χαρακτήρας.** Ο τύπος αυτός αναφέρεται τόσο σε ένα χαρακτήρα όσο και μία σειρά χαρακτήρων. Τα δεδομένα αυτού του τύπου μπορούν να περιέχουν οποιονδήποτε χαρακτήρα παράγεται από το πληκτρολόγιο. Παραδείγματα χαρακτήρων είναι **'Κ', 'Κώστας', 'σήμερα είναι Τετάρτη', 'Τα πολλαπλάσια του 15 είναι'**. Οι χαρακτήρες πρέπει υποχρεωτικά να βρίσκονται μέσα σε απλά εισαγωγικά, **' '**. Τα δεδομένα αυτού του τύπου, επειδή περιέχουν τόσο αλφαβητικούς όσο και αριθμητικούς χαρακτήρες, ονομάζονται συχνά **αλφαριθμητικά**.
- **Λογικός.** Αυτός ο τύπος δέχεται μόνο δύο τιμές: **ΑΛΗΘΗΣ** και **ΨΕΥΔΗΣ**. Οι τιμές αντιπροσωπεύουν αληθείς ή ψευδείς συνθήκες.





# ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ

Παραδείγματα όπου απαιτούν ιδιαίτερη προσοχή:

«ΑΛΗΘΗΣ» : Αλφαριθμητικό ( μία σειρά από χαρακτήρες)

ΑΛΗΘΗΣ : Λογικό

1.23 : Πραγματικός Αριθμός

«1.23» : Αλφαριθμητικό ( μία σειρά από χαρακτήρες) (Μιας και είναι σε αυτάκια, δεν είναι αριθμός)

3 : Ακέραιος Αριθμός

«3» : Χαρακτήρας (Μιας και είναι σε αυτάκια, δεν είναι αριθμός)

Τιπ:  
Ακέραιοι είναι οι αριθμοί που δεν  
έχουν υποδιαστολή.

Πχ:  
4  
5  
2  
-21

Τιπ:  
Όταν βλέπουμε κάτι σε αυτάκια είναι σίγουρα  
χαρακτήρας.  
Όταν έχουμε περισσότερους από 1 χαρακτήρες μαζί,  
τότε το λέμε αλφαριθμητικό.

Πχ:  
«ΑΛΗΘΗΣ» ( μία σειρά από χαρακτήρες)  
«Γεια σας» ( μία σειρά από χαρακτήρες)  
« Ποκεμον» ( μία σειρά από χαρακτήρες)  
«1.34» ( μία σειρά από χαρακτήρες)  
«1» ( ένας χαρακτήρας )  
«2.0» ( μία σειρά από χαρακτήρες)

Τιπ:  
Πραγματικοί είναι οι αριθμοί που  
έχουν υποδιαστολή.

Πχ:  
4.0  
5.01  
2.23232432534  
-21.2



# ΙΕΡΑΡΧΙΑ ΠΡΑΞΕΩΝ

---

1. Ύψωση σε δύναμη

---

2. Πολλαπλασιασμός  
και διαίρεση

---

3. Πρόσθεση και  
αφαίρεση



# PRO TIPS:

Διάβασε =Εκτελεστέα  
εντολή

Αλγόριθμος = Δηλωτική  
εντολή

Η συνθήκη είναι μια λογική  
έκφραση.

$i \leftarrow i + 1 \Rightarrow$  η νέα τιμή της  
μεταβλητής  $i$  είναι η  
προηγούμενη συν ένα

Το τμήμα του αλγορίθμου  
που επαναλαμβάνεται,  
δηλαδή από την εντολή  
Όσο μέχρι το  
τέλος\_επανάληψης  
αποκαλείται βρόχος.

Ο βρόχος επανάληψης  
μπορεί να μην εκτελεσθεί  
καμία φορά, αν η πρώτη  
τιμή που διαβάζεται είναι  
αρνητική

Η εντολή  
Αρχή\_επανάληψης...  
Μέχρις\_ότου Εκτελείται  
οπωσδήποτε μια φορά

Ο βρόχος Για  $k$  από 5 μέχρι  
5 εκτελείται ακριβώς μία  
φορά

Ο βρόχος Για  $k$  από 5 μέχρι  
1 δεν εκτελείται καμία  
φορά

Η ολίσθηση προς τα  
αριστερά ισοδυναμεί με  
πολλαπλασιασμό επί δύο,  
ενώ η ολίσθηση προς τα  
δεξιά ισοδυναμεί με την  
ακέραια διαίρεση διά δύο.



# DIV MOD



$$\begin{array}{r|l} 3 & 2 \\ -2 & \\ \hline 1 & \end{array}$$

mod

div

$$3 \text{div} 2 = 1$$
$$3 \text{mod} 2 = 1$$

$$\begin{array}{r|l} 13 & 2 \\ -12 & \\ \hline 1 & \end{array}$$

mod

div

$$13 \text{div} 2 = 6$$
$$13 \text{mod} 2 = 1$$

Ο τελεστής div χρησιμοποιείται για τον υπολογισμό του πηλίκου μιας διαίρεσης ακεραίων αριθμών, ενώ ο τελεστής mod για το υπόλοιπο της διαίρεσης.

# ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΤΟΥ ΔΟΜΗΜΕΝΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ



Δημιουργία απλούστερων προγραμμάτων.

Άμεση μεταφορά των αλγορίθμων σε προγράμματα.

Διευκόλυνση ανάλυσης του προγράμματος σε τμήματα.

Περιορισμός των λαθών κατά την ανάπτυξη του προγράμματος.

Διευκόλυνση στην ανάγνωση και κατανόηση του προγράμματος από τρίτους.

Ευκολότερη διόρθωση και συντήρηση.

**// Hint! Μην πάτε καν να δώσετε αν δεν τα έχετε μάθει αυτά!!! SUPER SOS //**



# ΣΤΑΘΕΡΕΣ

- Η ΓΛΩΣΣΑ επιτρέπει την αντιστοίχιση σταθερών τιμών με ονόματα, εφόσον αυτά δηλωθούν στην αρχή του προγράμματος (στο τμήμα δήλωσης σταθερών).

Πχ. ΠΡΟΓΡΑΜΜΑ ΠΑΡΑΔΕΙΓΜΑ

ΣΤΑΘΕΡΕΣ : ΦΠΑ=24/100

Π=3.14

ΜΕΤΑΒΛΗΤΕΣ: ...

.....

.....



# ΠΟΙΑ ΟΝΟΜΑΤΑ ΜΕΤΑΒΛΗΤΩΝ/ΣΤΑΘΕΡΩΝ ΕΙΝΑΙ ΑΠΟΔΕΚΤΑ;

- Δεν μπορούν να αρχίζουν με αριθμό.
- Δεν μπορούμε να χρησιμοποιήσουμε δεσμευμένες λέξεις για όνομα μεταβλητής/σταθέρως ( π.χ ΟΣΟ ,ΠΡΟΓΡΑΜΜΑ , ΑΚΕΡΑΙΕΣ ,ΑΝ κτλπ ).
- Δεν μπορούν να έχουν κενά μεταξύ τους . Π.χ «TRAM\_ΠΕΙΡΑΙΑ» και όχι «TRAM ΠΕΙΡΑΙΑ «
- Μπορούν να χρησιμοποιηθούν γράμματα του ελληνικού και λατινικού αλφαβήτου ( Α-Ω ) , ( Α-Z ) , ψηφία (0-9) και η κάτω πάλυλα ( \_ ) .
- Δεν μπορώ να χρησιμοποιήσω “ “ (αυτάκια) ,ούτε σύμβολα όπως + - \* κτλπ .

Πχ.

α2 Δεκτό  
3χ Μη επιτρεπτό  
α\_2 Δεκτό

\_α2 Δεκτό  
«γ» Μη επιτρεπτό  
ΔΙΑΒΑΣΕ Μη επιτρεπτό  
Α\_β Δεκτό



# ΧΡΗΣΙΜΕΣ ΣΥΝΑΡΤΗΣΕΙΣ



$\text{HM}(X)$ : Υπολογισμός ημιτόνου

$\text{ΣΥΝ}(X)$ : Υπολογισμός συνημιτόνου

$\text{ΕΦ}(X)$ : Υπολογισμός εφαπτομένης

$\text{T\_P}(X)$ : Υπολογισμός τετραγωνικής ρίζας

$\text{ΛΟΓ}(X)$ : Υπολογισμός φυσικού λογαρίθμου

$\text{E}(X)$ : Υπολογισμός του  $e^x$

$\text{A\_M}(X)$ : Ακέραιο μέρος του  $X$

$\text{A\_T}(X)$ : Απόλυτη τιμή του  $X$

# TIPS AND TRICKS VOL 2



Σε ένα πρόγραμμα, αν ο πρώτος χαρακτήρας είναι το θαυμαστικό (!), σημαίνει ότι αυτή η γραμμή περιέχει σχόλια και όχι εκτελέσιμες εντολές.

Αν μία εντολή πρέπει να συνεχιστεί και στην επόμενη γραμμή, τότε ο πρώτος χαρακτήρας αυτής της γραμμής πρέπει να είναι ο χαρακτήρας &.

Κάθε εντολή γράφεται σε ξεχωριστή γραμμή.

Σε μια εντολή εκχώρησης η μεταβλητή και η έκφραση πρέπει να είναι του ιδίου τύπου.

Πάντα πρέπει να χρησιμοποιούνται ζεύγη παρενθέσεων. Διαφορετικός αριθμός αριστερών από δεξιές παρενθέσεις στην ίδια έκφραση είναι ένα από τα πιο συνηθισμένα λάθη.

Κλασσικό λάθος:  $((3 \times 2 + 3))$

Τα δεδομένα στην πραγματικότητα καταχωρούνται στην μνήμη RAM του υπολογιστή. Ανάλογα τον τύπο τους, καταλαμβάνουν διαφορετικό χώρο! Π.χ οι ακέραιοι σε 1, 2 ή 4 byte και ο πραγματικός σε 4 ή 8 bytes.

# ΠΡΟΤΕΡΑΙΟΤΗΤΑ ΠΡΑΞΕΩΝ:



## ΠΡΩΤΑ :

- ΠΑΡΕΝΘΕΣΕΙΣ
- ΔΥΝΑΜΕΙΣ
- ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΙ ΚΑΙ ΔΙΑΙΡΕΙΣΕΙΣ
- ΠΡΟΣΘΕΣΗ ΚΑΙ ΑΦΑΙΡΕΣΗ



# ΛΟΓΙΚΕΣ ΠΡΑΞΕΙΣ



ΑΛΗΘΗΣ ΚΑΙ ΑΛΗΘΗΣ = ΑΛΗΘΗΣ

ΑΛΗΘΗΣ ΚΑΙ ΨΕΥΔΗΣ = ΨΕΥΔΗΣ

ΨΕΥΔΗΣ ΚΑΙ ΨΕΥΔΗΣ = ΨΕΥΔΗΣ

Στις λογικές πράξεις που χρησιμοποιούμε το ΚΑΙ (AND) , πρέπει όλες οι συνθήκες να είναι αληθείς. Διαφορετικά, είναι ψευδής η πρόταση.

ΑΛΗΘΗΣ Η ΑΛΗΘΗΣ = ΑΛΗΘΗΣ

ΑΛΗΘΗΣ Η ΨΕΥΔΗΣ = ΑΛΗΘΗΣ

ΨΕΥΔΗΣ Η ΨΕΥΔΗΣ = ΨΕΥΔΗΣ

Στις λογικές πράξεις που χρησιμοποιούμε το Η (OR) , μας ενδιαφέρει μία συνθήκη να είναι αληθείς, έτσι ώστε να είναι αληθείς η πρόταση.

ΌΧΙ(ΑΛΗΘΗΣ) = ΨΕΥΔΗΣ

ΌΧΙ(ΨΕΥΔΗΣ) = ΑΛΗΘΗΣ

Το όχι (not) , κάνει το αντίθετο από αυτό που βγαίνει μέσα στις παρενθέσεις.

# ΟΡΙΖΟΥΜΕ ΤΗΝ ΤΑΞΙΝΟΜΗΣΗ

- Δοθέντων των στοιχείων  $a_1, a_2, \dots, a_n$  η ταξινόμηση συνίσταται στη μετάθεση (permutation) της θέσης των στοιχείων, ώστε να τοποθετη-θούν σε μία σειρά  $a_{k1}, a_{k2}, \dots, a_{kn}$  έτσι ώστε, δοθείσης μίας συνάρτησης διάταξης (ordering function),  $f$ , να ισχύει:

$$f(a_{k1}) \leq f(a_{k2}) \leq \dots \leq f(a_{kn})$$



1  
2  
4  
12  
15  
23  
343  
4893

## ΤΑΞΙΝΟΜΗΣΗ

4893  
343  
23  
15  
12  
4  
2  
1



Κατά αύξοντα  
ρυθμό :

Κατά φθείνοντα  
ρυθμό :

## ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΔΕΥΤΕΡΕΟΥΣΑΣ ΜΝΗΜΗΣ



Σε μεγάλες πρακτικές εμπορικές/επιστημονικές εφαρμογές, το μέγεθος της κύριας μνήμης δεν επαρκεί για την αποθήκευση των δεδομένων. Στην περίπτωση αυτή χρησιμοποιούνται ειδικές δομές για την αποθήκευση των δεδομένων στη δευτερεύουσα μνήμη, δηλαδή κυρίως στο μαγνητικό δίσκο. Οι ειδικές αυτές δομές ονομάζονται αρχεία (files)

- τα δεδομένα δεν χάνονται, αν διακοπεί η ηλεκτρική παροχή. Έτσι, τα δεδομένα των αρχείων διατηρούνται ακόμη και μετά τον τερματισμό ενός προγράμματος,



- Τα στοιχεία ενός αρχείου ονομάζονται εγγραφές (records), όπου κάθε εγγραφή αποτελείται από ένα ή περισσότερα πεδία (fields), που ταυτοποιούν την εγγραφή, και από άλλα πεδία που περιγράφουν διάφορα χαρακτηριστικά της εγγραφής ( **μάθημα** , **έτος γέννησης** στο παράδειγμά μας) .



## ΜΝΗΜΗ RAM

- Τα δεδομένα χάνονται αν διακοπεί η ηλεκτρική παροχή στην μνήμη **RAM**
- Η μνήμη **RAM** έχει περιορισμένο μέγεθος – δεν μπορεί να χωρέσει πάρα πολλά δεδομένα

### Π.χ Αρχείο Φοιτητές\_ΠΑΠΕΙ

Secondary key Γιατί μπορεί να υπάρχουν και 2 άτομα με ίδιο όνομα		Primary key Το μήτρω είναι υποχρεωτικά μοναδικό	
Όνομα	Μήτρω	Μάθημα	Έτος γέννησης
Δημήτρης	E5301	Ανάλυση	2000
Γιάννης	E5348	Πιθανότητες	1999
Δημήτρης	E6894	Λογική	2000
Αποστόλης	E3950	Ανάλυση	1999

κελί

εγγραφές

εγγραφή

# ΑΠΟ ΤΙ ΠΡΟΣΔΙΟΡΙΖΕΤΑΙ ΜΙΑ ΓΛΩΣΣΑ;

**Αλφάβητο :** Αλφάβητο μίας γλώσσας καλείται το σύνολο των στοιχείων που χρησιμοποιείται από τη γλώσσα

**Λεξιλόγιο :** Αποτελείται από ένα υποσύνολο όλων των ακολουθιών που δημιουργούνται από τα στοιχεία του αλφαβήτου, τις λέξεις που είναι δεκτές από τη γλώσσα

**Γραμματική:**  
Αποτελείται από το τυπικό ή τυπολογικό (accidence) και το συντακτικό (syntax).

**Σημασιολογία :** Η σημασιολογία (Semantics) είναι το σύνολο των κανόνων που καθορίζει το νόημα των λέξεων και κατά επέκταση των εκφράσεων και προτάσεων που χρησιμοποιούνται σε μία γλώσσα. Στις γλώσσες προγραμματισμού οι οποίες είναι τεχνητές γλώσσες, ο δημιουργός της γλώσσας αποφασίζει τη σημασιολογία των λέξεων της γλώσσας.

**Τυπικό :** Είναι το σύνολο των κανόνων που ορίζει τις μορφές με τις οποίες μία λέξη είναι αποδεκτή.

**Συντακτικό :** Είναι το σύνολο των κανόνων που καθορίζει τη νομιμότητα της διάταξης και της σύνδεσης των λέξεων της γλώσσας για τη δημιουργία προτάσεων. Η γνώση του συντακτικού επιτρέπει τη δημιουργία σωστών προτάσεων στις φυσικές γλώσσες, ενώ στις γλώσσες προγραμματισμού τη δημιουργία σωστών εντολών

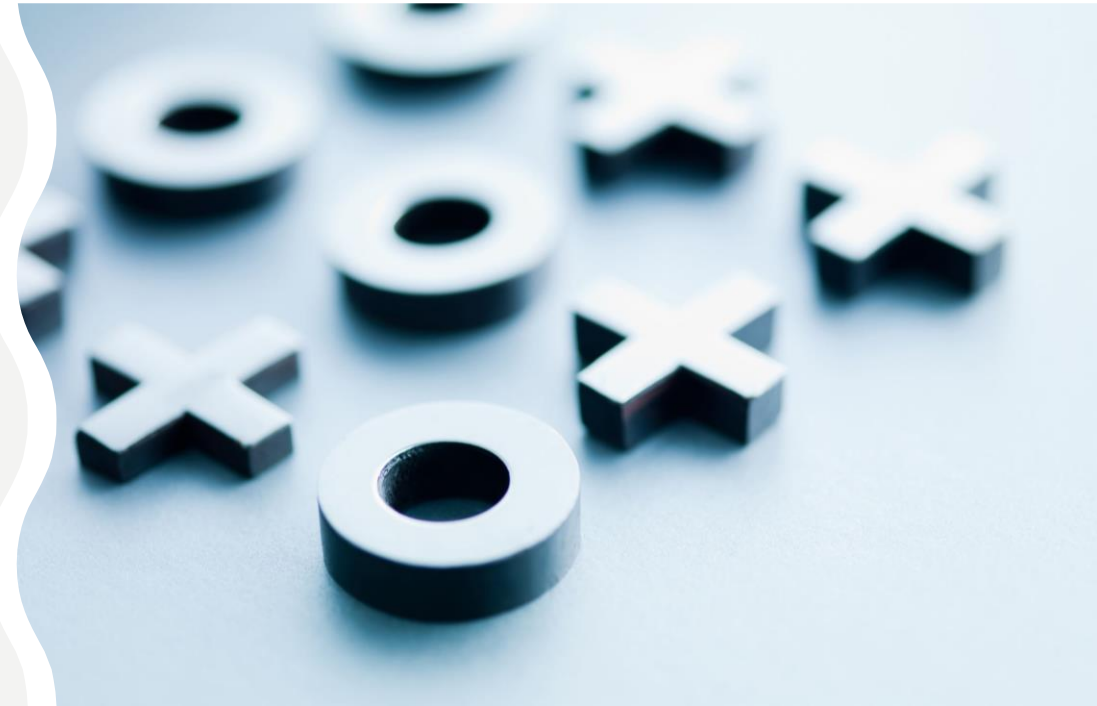




# ΤΟ ΑΛΦΑΒΗΤΟ ΤΗΣ ΓΛΩΣΣΑΣ



- Γράμματα:
  - ✓ Κεφαλαία ελληνικού αλφαβήτου (Α-Ω)
  - ✓ Πεζά ελληνικού αλφαβήτου (α-ω)
  - ✓ Κεφαλαία λατινικού αλφαβήτου (Α-Z)
  - ✓ Πεζά λατινικού αλφαβήτου (a-z)
- Ψηφία: 0-9
- Ειδικοί χαρακτήρες: + - \* / = ( ) . , ' ! & κενός χαρακτήρας ^



# ΤΙ ΕΊΝΑΙ ΔΟΜΗ ΔΕΔΟΜΕΝΩΝ;

- Δομή Δεδομένων είναι ένα σύνολο αποθηκευμένων δεδομένων που υφίστανται επεξεργασία από ένα σύνολο λειτουργιών.

(Αλγόριθμοι+ Δομές δεδομένων = ΠΡΟΓΡΑΜΜΑ )



Προσπέλαση (access), πρόσβαση σε έναν κόμβο με σκοπό να εξετασθεί ή να τροποποιηθεί το περιεχόμενό του.

Εισαγωγή (insertion), δηλαδή η προσθήκη νέων κόμβων σε μία υπάρχουσα δομή.

Διαγραφή (deletion), που αποτελεί το αντίστροφο της εισαγωγής, δηλαδή ένας κόμβος αφαιρείται από μία δομή.

Αναζήτηση (searching), κατά την οποία προσπελούνται οι κόμβοι μιας δομής, προκειμένου να εντοπιστούν ένας ή περισσότεροι που έχουν μια δεδομένη ιδιότητα.

Ταξινόμηση (sorting), όπου οι κόμβοι μιας δομής διατάσσονται κατά αύξουσα ή φθίνουσα σειρά.

Αντιγραφή (copying), κατά την οποία όλοι οι κόμβοι ή μερικοί από τους κόμβους μίας δομής αντιγράφονται σε μία άλλη δομή.

Συγχώνευση (merging), κατά την οποία δύο ή περισσότερες δομές συνενώνονται σε μία ενιαία δομή.

Διαχωρισμός (separation), που αποτελεί την αντίστροφη πράξη της συγχώνευσης.

# ΒΑΣΙΚΕΣ ΛΕΙΤΟΥΡΓΙΕΣ ΤΩΝ ΔΟΜΩΝ ΔΕΔΟΜΕΝΩΝ



# ΚΑΤΗΓΟΡΙΕΣ ΔΟΜΩΝ ΔΕΔΟΜΕΝΩΝ:

- Οι δομές δεδομένων διακρίνονται σε δύο μεγάλες κατηγορίες:
  - τις στατικές(static)
  - Τις δυναμικές(dynamic)

Οι δυναμικές δομές δεν αποθηκεύονται σε συνεχόμενες θέσεις μνήμης αλλά στηρίζονται στην τεχνική της λεγόμενης δυναμικής παραχώρησης μνήμης (dynamic memory allocation). Με άλλα λόγια, οι δομές αυτές δεν έχουν σταθερό μέγεθος, αλλά ο αριθμός των κόμβων τους μεγαλώνει και μικραίνει καθώς στη δομή εισάγονται νέα δεδομένα ή διαγράφονται κάποια δεδομένα αντίστοιχα. Όλες οι σύγχρονες γλώσσες προγραμματισμού προσφέρουν τη δυνατότητα δυναμικής παραχώρησης μνήμης.



# ΟΡΙΖΟΥΜΕ ΤΟΝ ΠΙΝΑΚΑ

- Είναι μια δομή που περιέχει στοιχεία του ίδιου τύπου (δηλαδή ακέραιους, πραγματικούς κ.λπ).

Η δήλωση των στοιχείων ενός πίνακα και η μέθοδος αναφοράς τους εξαρτάται από τη συγκεκριμένη γλώσσα υψηλού επιπέδου που χρησιμοποιείται.

Όμως, γενικά η αναφορά στα στοιχεία ενός πίνακα γίνεται με τη χρήση του συμβολικού ονόματος του πίνακα ακολουθούμενου από την τιμή ενός ή περισσότερων δεικτών (indexes) σε παρένθεση ή αγκύλη.

Π.χ  $\text{Ονόματα}[ i ]$

← αγκύλες

↑ δείκτης

↑ Όνομα πίνακα





# ΠΩΣ ΛΕΙΤΟΥΡΓΟΥΝ ΟΙ ΠΙΝΑΚΕΣ

Πίνακας M[10] (ένας μονοδιάστατος πίνακας)

1	2	3	4	5	6	7	8	9	10
43	34	535	5	483	453	477	3	63	7

Στοιχείο M[2]

Στοιχείο M[6]

Όνομα Πίνακα [θέση]



# ΠΩΣ ΛΕΙΤΟΥΡΓΟΥΝ ΟΙ ΠΙΝΑΚΕΣ

Στήλες

Πίνακας A[7,9] (ένας δισδιάστατος πίνακας)

Γραμμές

	1	2	3	4	5	6	7	8	9
1	32	23123	3213	321	12441	25324	345	545	5345
2	4543	44675	73	32	64	654	34	3	2
3	534	4534	76	765	776	45523	567	1223	234
4	43534	4353	54	6	32234	45654	674	23232	645
5	54	54	34	546	32	354	567	233	564
6	35	53	534	6	4	33543	56	2323	342
7	453	5	56	54	243	54	3	321	245

Στοιχείο A[4,3]

Όνομα Πίνακα [Γραμμή, Στήλη]

Στοιχείο A[7,8]



# ΑΝΑΖΗΤΗΣΗ



# ΣΕΙΡΙΑΚΉ ΑΝΑΖΉΤΗΣΗ



Η σειριακή μέθοδος αναζήτησης είναι η πιο απλή, αλλά και η λιγότερη αποτελεσματική μέθοδος αναζήτησης. Έτσι, δικαιολογείται η χρήση της μόνο σε περιπτώσεις όπου:

ο πίνακας είναι **μη** ταξινομημένος,

ο πίνακας είναι μικρού μεγέθους (για παράδειγμα,  $n \leq 20$ ),

η αναζήτηση σε ένα συγκεκριμένο πίνακα γίνεται σπάνια,

# ΔΥΑΔΙΚΗ ΑΝΑΖΗΤΗΣΗ



Ο αλγόριθμος της δυαδικής αναζήτησης (binary search) εφαρμόζεται μόνο σε πίνακες που έχουν ταξινομημένα στοιχεία. Αν τα στοιχεία **δεν** είναι ταξινομημένα τότε **δεν μπορεί να εφαρμοστεί**.

Ο αλγόριθμος λειτουργεί ως εξής:

- ✓ Βρίσκουμε το μεσαίο στοιχείο του ταξινομημένου πίνακα.
- ✓ Εάν το προς αναζήτηση στοιχείο είναι ίσο με το μεσαίο στοιχείο τότε σταματάμε την αναζήτηση αφού το στοιχείο βρέθηκε
- ✓ Εάν δεν βρέθηκε, τότε ελέγχουμε αν το στοιχείο που αναζητούμε είναι μικρότερο ή μεγαλύτερο από το μεσαίο στοιχείο του πίνακα.
- ✓ Αν είναι μικρότερο, περιορίζουμε την αναζήτηση στο πρώτο μισό του πίνακα (με την προϋπόθεση ότι τα στοιχεία είναι διατεταγμένα κατά αύξουσα σειρά), ενώ αν είναι μεγαλύτερο περιορίζουμε την αναζήτηση στο δεύτερο μισό του πίνακα.



# ΔΥΑΔΙΚΗ ΑΝΑΖΗΤΗΣΗ



27

Το βρήκαμε!

**Π.χ** Θέλουμε να βρούμε σε ποιά θέση του πίνακα είναι ο αριθμός 27 .

Ο παρών πίνακας έχει 9 στοιχεία ,ταξινομημένα με αύξοντα ρυθμό.

Το μέσο του πίνακα είναι Συνολικά\_Στοιχεία/2 ,άρα  $9/2=4.5$  ,οπότε παίρνουμε το 5<sup>ο</sup> στοιχείο

Τώρα θα πάρουμε τα άκρα (στην προκειμένη ,το πρώτο στοιχείο του πίνακα) και το τέλος,το τελευταίο στοιχείο του πίνακα (η 9<sup>η</sup> θέση δηλαδή)

Αφού ο αριθμός που ψάχνουμε ,είναι σε θέση μεγαλύτερη του μέσου ,θα κόψουμε τον πίνακα στα 2 ,και θα κρατήσουμε ως άκρο , το μέσο (Δηλαδή η αναζήτηση μας από εδώ και πέρα θα είναι μεταξύ της θέσης 5 και 9)

Έτσι θα έχουμε το νέο μεσο ,δηλαδή  $5/2=2.5$  ,άρα 3 ! Πάλι , ο αριθμός που ψάχνουμε ,είναι σε θέση μεγαλύτερη του μέσου ,θα ξανακόψουμε τον πίνακα στα 2 ,και θα κρατήσουμε ως άκρο , το μέσο (Δηλαδή το 7 θα γίνει το ένα άκρο ,και το άλλο άκρο παραμένει το 9)

Και έτσι στο συγκεκριμένο παράδειγμα ,μείναμε με ένα υποπίνακα 3 στοιχείων !

Το περιεχόμενο του νέου μέσου είναι ο αριθμός που ψάχνουμε ,οπότε ο δείκτης είναι ο αριθμός που ψάχναμε! (η θέση του στοιχείου στον πίνακα είναι η 8<sup>η</sup> θέση )

# ΔΥΑΔΙΚΗ ΑΝΑΖΗΤΗΣΗ



Άρα, η λογική της δυαδικής αναζήτησης, είναι να κόβουμε τον ταξινομημένο πίνακα σε μικρότερους υποπίνακες με κριτήριο αν το μέσο είναι μεγαλύτερο ή μικρότερο του στοιχείου, μέχρι να καταλήξουμε στο στοιχείο που θέλουμε!

32

Το βρήκαμε!

**Π.χ** Θέλουμε να βρούμε σε ποιά θέση του πίνακα είναι ο αριθμός 27 .

Ο παρών πίνακας έχει 9 στοιχεία ,ταξινομημένα με φθίνοντα ρυθμό.

Το μέσο του πίνακα είναι Συνολικά\_Στοιχεία/2 ,άρα  $9/2=4.5$  ,οπότε παίρνουμε το 5<sup>ο</sup> στοιχείο

Τώρα θα πάρουμε τα άκρα (στην προκειμένη ,το πρώτο στοιχείο του πίνακα) και το τέλος, το τελευταίο στοιχείο του πίνακα (η 9<sup>η</sup> θέση δηλαδή)

Αφού ο αριθμός που ψάχνουμε ,είναι σε θέση μικρότερη του μέσου ,θα κόψουμε τον πίνακα στα 2 ,και θα κρατήσουμε ως τέλος , το μέσο (Δηλαδή η αναζήτηση μας από εδώ και πέρα θα είναι μεταξύ της θέσης 1 και 5)

Έτσι θα έχουμε το νέο μεσο ,δηλαδή  $5/2=2.5$  ,άρα 3 ! Πάλι , ο αριθμός που ψάχνουμε ,είναι σε θέση μικρότερη του μέσου ,θα ξανακόψουμε τον πίνακα στα 2 ,και θα κρατήσουμε ως τέλος , το μέσο (Δηλαδή το 1 θα γίνει το ένα άκρο ,και το άλλο άκρο 3) Και έτσι στο συγκεκριμένο παράδειγμα ,μείναμε με ένα υποπίνακα 3 στοιχείων !

Το περιεχόμενο του νέου μέσου είναι ο αριθμός που ψάχνουμε ,οπότε ο δείκτης είναι ο αριθμός που ψάχναμε! (η θέση του στοιχείου στον πίνακα είναι η 2<sup>η</sup> θέση )



# ΛΟΓΙΚΑ ΛΑΘΗ VS ΣΥΝΤΑΚΤΙΚΑ



- Λογικό είναι ένα λάθος που προκύπτει μετά από λάθος στον τρόπο σκέψης.

**Πχ.**  $a = c/d$  με τιμές  $c=4$   $d=0$  – Δεν μπορούμε να διαιρέσουμε με το 0.

Το πρόγραμμα στα λογικά λάθη μεν μεταγλωττίζεται χωρίς πρόβλημα, αλλά δεν τρέχει σωστά κατά την εκτέλεση του λόγω αυτών των λαθών ή δεν τρέχει σωστά όταν δοθούν κάποιες τιμές (Όπως το παράδειγμα με την διαίρεση όταν ο χρήστης δώσει την τιμή 0).

- Συντακτικό είναι ένα λάθος στην συγγραφή του προγράμματος όταν παραβιάζεται κάποιο ή κάποια από τα προσδιοριζόμενα της γλώσσας προγραμματισμού που χρησιμοποιούμε (αλφάβητο, λεξιλόγιο γραμματική, σημασιολογία). Τα συντακτικά λάθη οδηγούν στην αδυναμία μεταγλώττισης του προγράμματος.

**Πχ.** ΔΙΑΒΑΖΕ (δεν υπάρχει τέτοια δεσμευμένη λέξη), ΓΡΑΠΣΕ «Στροφή του Γιώργη» (είναι ΓΡΑΨΕ όχι ΓΡΑΠΣΕ) ή αν  $x=3$  και  $y=$ “Γεια σου” κι εμείς προσπαθήσουμε να εκχωρίσουμε στην  $z$ , το  $x+y$  (Δεν μπορούμε να κάνουμε πράξεις μεταξύ χαρακτήρων και ακεραίων)

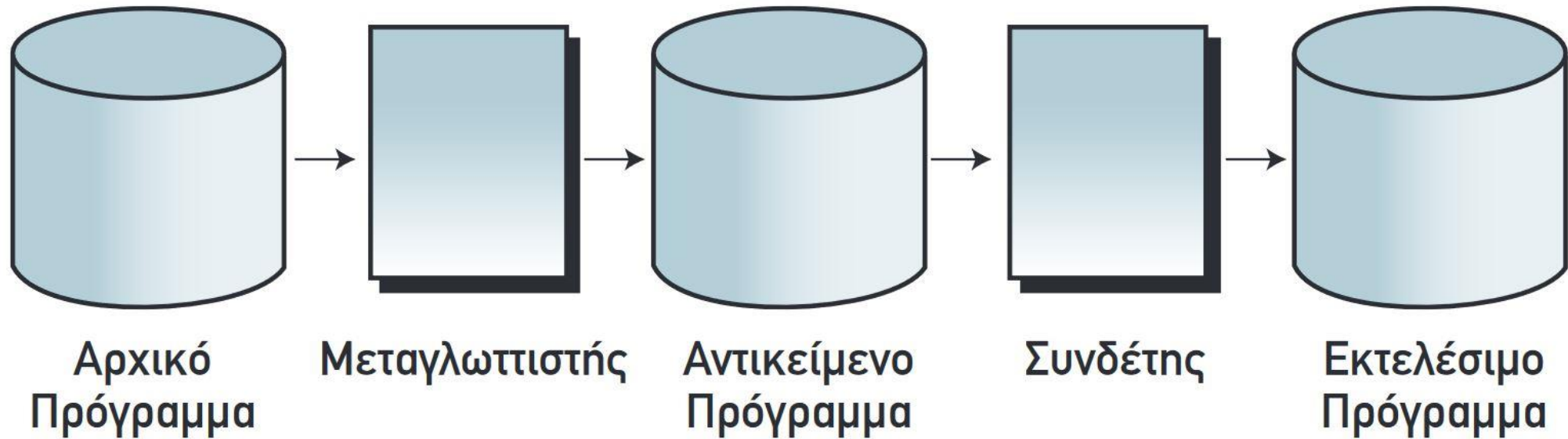
# ΣΥΝΟΨΊΖΟΝΤΑΣ...

Σαν συντακτικό λάθος θεωρείται αυτό που διακόπτει την μεταγλώττιση του προγράμματος δηλαδή υπάρχει ένα λάθος στην σύνταξη μιας εντολής που οδηγεί στην μη εκτέλεση της. (Η εντολή δεν εκτελείται ποτέ)

Ένα λογικό λάθος, δεν έχει να κάνει με την λανθασμένη σύνταξη μιας εντολής. (Η εντολή μπορεί να εκτελείται κανονικά ή να μην εκτελείται για ορισμένες μόνο τιμές των μεταβλητών αλλά να μην δίνει τα αναμενόμενα αποτελεσματα.)

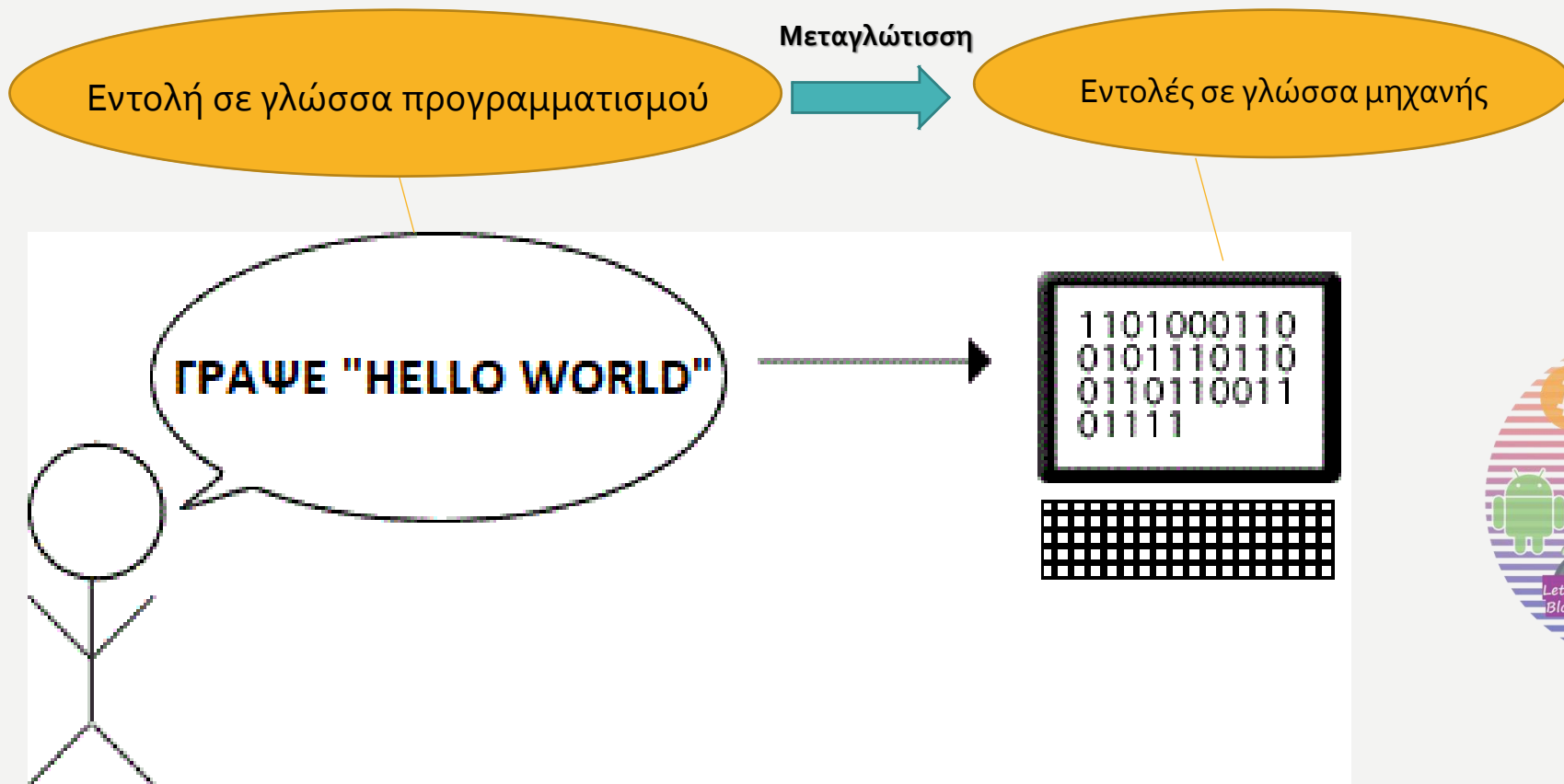


# ΜΕΤΑΓΛΩΤΙΣΤΕΣ ΚΑΙ ΔΙΕΡΜΗΝΕΥΤΕΣ:



Κρατήστε στο μυαλό αυτό το σχήμα!





- Κάθε πρόγραμμα που γράφτηκε σε οποιαδήποτε γλώσσα προγραμματισμού πρέπει να μετατραπεί σε μορφή αναγνωρίσιμη και εκτελέσιμη από τον υπολογιστή, δηλαδή σε εντολές γλώσσας μηχανής.

# ΜΕΤΑΓΛΩΤΤΙΣΤΗΣ

- Δέχεται στην είσοδο ένα πρόγραμμα γραμμένο σε μια γλώσσα υψηλού επιπέδου και παράγει ένα ισοδύναμο πρόγραμμα σε γλώσσα μηχανής. Το τελευταίο μπορεί να εκτελείται οποτεδήποτε από τον υπολογιστή και είναι τελείως ανεξάρτητο από το αρχικό πρόγραμμα. (Άρα ο μεταγλωττιστής χρησιμοποιείται αφού ολοκληρώσουμε την σύνταξη του κώδικά μας.)

# ΔΙΕΡΜΗΝΕΥΤΗΣ

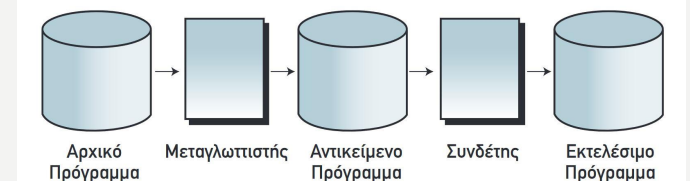
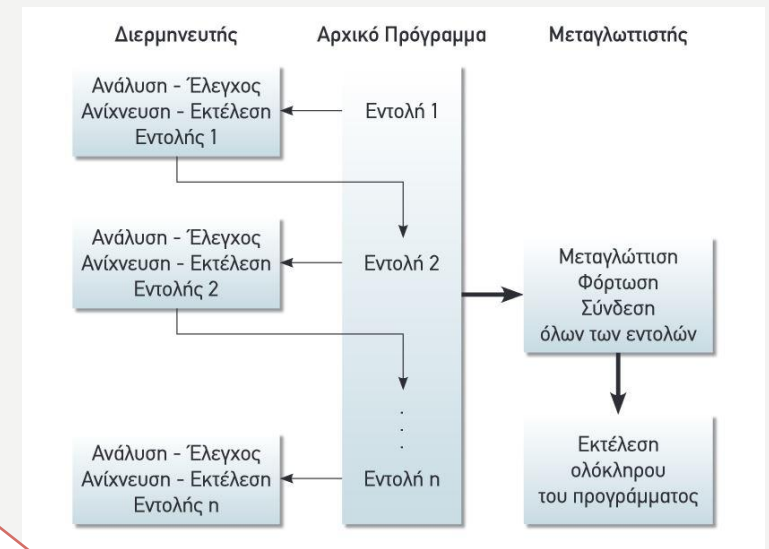
- Ο διερμηνευτής διαβάζει μία προς μία τις εντολές του αρχικού προγράμματος και για καθεμία εκτελεί αμέσως μια ισοδύναμη ακολουθία εντολών μηχανής.





Το αρχικό πρόγραμμα λέγεται **πηγαίο πρόγραμμα** (source), ενώ το πρόγραμμα που παράγεται από το **μεταγλωττιστή** λέγεται αντικείμενο πρόγραμμα (object). Το αντικείμενο πρόγραμμα είναι μεν σε μορφή κατανοητή από τον υπολογιστή, αλλά συνήθως **δεν** είναι σε θέση να εκτελεστεί. Χρειάζεται να συμπληρωθεί και να συνδεθεί με άλλα τμήματα προγράμματος απαραίτητα για την εκτέλεσή του, τμήματα που είτε τα γράφει ο προγραμματιστής είτε βρίσκονται στις βιβλιοθήκες (libraries) της γλώσσας. Το πρόγραμμα που επιτρέπει αυτή τη σύνδεση ονομάζεται **συνδέτης – φορτωτής** (linker- loader). Το αποτέλεσμα του συνδέτη είναι η παραγωγή του **εκτελέσιμου προγράμματος** (executable), το οποίο είναι το τελικό πρόγραμμα που εκτελείται από τον υπολογιστή. Για το λόγο αυτό η συνολική διαδικασία αποκαλείται **μεταγλώττιση και σύνδεση**. Η δημιουργία του εκτελέσιμου προγράμματος γίνεται μόνο στην περίπτωση που το αρχικό πρόγραμμα δεν περιέχει **συντακτικά λάθη**. Τις περισσότερες φορές κάθε πρόγραμμα αρχικά θα έχει λάθη. Τα λάθη του προγράμματος είναι γενικά δύο ειδών, **λογικά** και **συντακτικά**. Τα λογικά λάθη εμφανίζονται μόνο στην εκτέλεση, ενώ τα συντακτικά λάθη στο στάδιο της

**μεταγλώττισης.**



# POINTS




Κάθε γλώσσα προσδιορίζεται από το αλφάβητό της, το λεξιλόγιό της, τη γραμματική της και τη σημασιολογία της.

Η ιεραρχική σχεδίαση ή ιεραρχικός προγραμματισμός χρησιμοποιεί τη στρατηγική της συνεχούς διαίρεσης του προβλήματος σε υποπροβλήματα.

Τα σύγχρονα ολοκληρωμένα προγραμματιστικά περιβάλλοντα δεν παρέχουν απλώς ένα μεταφραστή μιας γλώσσας προγραμματισμού. Περιέχουν όλα τα προγράμματα και τα εργαλεία που απαιτούνται και βοηθούν τη συγγραφή, την εκτέλεση και κύρια τη διόρθωση των προγραμμάτων.



YouTube GR Αναζήτηση



Μαθαίνουμε τον  
Πολλαπλασιασμό Αλά  
Ρωσικά  
...Σε 3 Λεπτά

Πολλαπλασιασμός Αλά Ρωσικά - Μαθαίνουμε τον Πολλαπλασιασμό Αλά Ρωσικά σε 3 λεπτά - Panellinies\_AEPP

TuxHouse  
6,94 χιλ. εγγεγραμμένοι

Αναλυτικά στοιχεία Επεξεργασία βίντεο

62 Κοινοποίηση Λήψη Σας ευχαριστούμε

1,2 χιλ. προβολές πριν από 2 έτη eLearning Greek Videos  
Πολλαπλασιασμός Αλά Ρωσικά - Μαθαίνουμε τον Πολλαπλασιασμό Αλά Ρωσικά σε 3 λεπτά -  
Το συγκεκριμένο βίντεο δημιουργήθηκε για τις ανάγκες του Panellinies\_AEPP, ωστόσο περιέχει και υλοποίηση με Python για να ανήκει και στην κατηγορία "In Less Than 10 Minutes".  
Εμφάνιση περισσότερων



Δεν ξεχνάμε να διαβάσουμε και τον Πολλαπλασιασμό Αλά Ρωσικά !!!!!

<https://www.youtube.com/watch?v=y57gXCblpMA>

## ΥΠΟΠΡΟΓΡΑΜΜΑΤΑ

Τμηματικός προγραμματισμός ονομάζεται η τεχνική σχεδίασης και ανάπτυξης των προγραμμάτων ως ένα σύνολο από απλούστερα τμή-ματα προγραμμάτων.



# ΙΔΙΟΤΗΤΕΣ ΥΠΟΠΡΟΓΡΑΜΜΑΤΩΝ

Κάθε υποπρόγραμμα έχει μόνο **μία είσοδο** και **μία έξοδο**. Στην πραγματικότητα κάθε υποπρόγραμμα ενεργοποιείται με την είσοδο σε αυτό που γίνεται πάντοτε από την αρχή του, εκτελεί ορισμένες ενέργειες, και απενεργοποιείται με την έξοδο από αυτό που γίνεται πάντοτε από το τέλος του.

Κάθε υποπρόγραμμα πρέπει να είναι ανεξάρτητο από τα άλλα. Αυτό σημαίνει ότι κάθε υποπρόγραμμα μπορεί να σχεδιαστεί, να αναπτυχθεί και να συντηρηθεί αυτόνομα χωρίς να επηρεαστούν άλλα υποπρογράμματα. Στην πράξη βέβαια η απόλυτη ανεξαρτησία είναι δύσκολο να επιτευχθεί.

Κάθε υποπρόγραμμα πρέπει να μην είναι πολύ μεγάλο. Η έννοια του μεγάλου προγράμματος είναι υποκειμενική, αλλά πρέπει κάθε υποπρόγραμμα να είναι τόσο, ώστε να είναι εύκολα κατανοητό για να μπορεί να ελέγχεται. Γενικά κάθε υποπρόγραμμα πρέπει να **εκτελεί μόνο μία λειτουργία**. Αν εκτελεί περισσότερες λειτουργίες, τότε συνήθως μπορεί και πρέπει να διασπαστεί σε ακόμη μικρότερα υποπρογράμματα.



# ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΤΟΥ ΤΜΗΜΑΤΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

01

Διευκολύνει την ανάπτυξη του αλγορίθμου και του αντιστοίχου προγράμματος.

02

Διευκολύνει την κατανόηση και διόρθωση του προγράμματος.

03

Απαιτεί λιγότερο χρόνο και προσπάθεια στη συγγραφή του προγράμματος.

04

Επεκτείνει τις δυνατότητες των γλωσσών προγραμματισμού.





Ορισμός : Μία παράμετρος είναι μία μεταβλητή που επιτρέπει το πέρασμα της τιμής της από ένα τμήμα προγράμματος σε ένα άλλο.

## ΠΑΡΑΜΕΤΡΟΙ

Το υποπρόγραμμα είναι αυτόνομο και ανεξάρτητο τμήμα προγράμματος, αλλά συχνά πρέπει να επικοινωνεί με το υπόλοιπο πρόγραμμα. Συνήθως δέχεται τιμές από το τμήμα προγράμματος που το καλεί και μετά την εκτέλεση επιστρέφει σε αυτό νέες τιμές, αποτελέσματα.

Οι τιμές αυτές που περνούν από το ένα υποπρόγραμμα στο άλλο λέγονται παράμετροι. Οι παράμετροι λοιπόν είναι σαν τις κοινές μεταβλητές ενός προγράμματος με μία ουσιώδη διαφορά, χρησιμοποιούνται για να περνούν τιμές στα υποπρογράμματα.

Η λίστα των τυπικών παραμέτρων (**formal parameter list**) καθορίζει τις παραμέτρους στη δήλωση του υποπρογράμματος. Η λίστα των πραγματικών **παραμέτρων (actual parameter list)** καθορίζει τις παραμέτρους στην κλήση του υποπρογράμματος.

!Μερικές γλώσσες προγραμματισμού ονομάζουν ορίσματα τις τυπικές παραμέτρους και απλά παραμέτρους τις πραγματικές παραμέτρους.

# ΟΙ ΚΑΝΟΝΕΣ ΤΩΝ ΠΑΡΑΜΕΤΡΩΝ:



01

Ο αριθμός των πραγματικών και των τυπικών παραμέτρων πρέπει να είναι ίδιος.

02

Η τυπική παράμετρος και η αντίστοιχέ της πραγματική πρέπει να είναι του ιδίου τύπου.

03

Κάθε πραγματική παράμετρος αντιστοιχεί στην τυπική παράμετρο που βρίσκεται στην αντίστοιχη θέση. Για παράδειγμα, η πρώτη της λίστας των τυπικών παραμέτρων στην πρώτη της λίστας των πραγματικών παραμέτρων κ.ο.κ.



# ΔΙΑΔΙΚΑΣΙΕΣ ΚΑΙ ΣΥΝΑΡΤΗΣΕΙΣ

1

Η συνάρτηση είναι ένας τύπος υποπρογράμματος που υπολογίζει και επιστρέφει μόνο μία τιμή με το όνομά της (όπως οι μαθηματικές συναρτήσεις).

```
ΣΥΝΑΡΤΗΣΗ πάνω_από_μο(Π, ν): ΑΚΕΡΑΙΑ
ΜΕΤΑΒΛΗΤΕΣ
  ΠΡΑΓΜΑΤΙΚΕΣ: Π[100], άθροισμα, μο
  ΑΚΕΡΑΙΕΣ: ν, i, πλ
ΑΡΧΗ
  άθροισμα ← 0
  ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ ν
    άθροισμα ← άθροισμα + Π[i]
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  μο ← άθροισμα/ν
```

Μία συνάρτηση υπολογισμού μέσου όρου. Μία συνάρτηση δεν μπορεί να διαβάσει ή να προβάλει στον χρήστη, παρά μόνο να επιστρέψει μία τιμή με το όνομά της

2

Η διαδικασία είναι ένας τύπος υποπρογράμματος που μπορεί να εκτελεί όλες τις λειτουργίες ενός προγράμματος.

Τυπικές παραμέτροι είναι αυτές οι παραμέτροι που δίνονται στα υποπρογράμματα { (Π, ν) στην προκειμένη περίπτωση }

Εκείνες οι παραμέτροι που υπάρχουν στο κάλεσμα του υποπρογράμματος μας από τον κύριο κώδικά μας, ονομάζονται **πραγματικές παραμέτροι**.

Δηλαδή, το πρόγραμμα, κατά την εκτέλεσή του, οδηγεί τις πραγματικές παραμέτρους και τις συνδέει με τις τυπικές παραμέτρους του υποπρογράμματος. Εκεί, εκτελεί τις εντολές και όταν ολοκληρωθεί η εκτέλεση του υποπρογράμματος, γυρνάει πίσω τα αποτελέσματα στις πραγματικές παραμέτρους, και συνεχίζεται η εκτέλεση του προγράμματος.

```
ΔΙΑΔΙΚΑΣΙΑ Είσοδος_Δεδομενων ( x )
ΜΕΤΑΒΛΗΤΕΣ
  ΠΡΑΓΜΑΤΙΚΕΣ : x
ΑΡΧΗ
  ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΡΑΨΕ 'Δώσε έναν αριθμό'
    ΔΙΑΒΑΣΕ x
  ΜΕΧΡΙΣ_ΟΤΟΥ x > 0
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Μία διαδικασία. Οι διαδικασίες μπορούν να εκτελέσουν όλες τις λειτουργίες των προγραμμάτων, δηλαδή μπορούν να διαβάσουν ή να προβάλουν στον χρήστη.



# ΕΜΒΕΛΕΙΑ ΜΕΤΑΒΛΗΤΩΝ

- Το τμήμα του προγράμματος που ισχύουν οι μεταβλητές λέγεται εμβέλεια (scope) μεταβλητών.





# ΤΥΠΟΙ ΕΜΒΕΛΕΙΑΣ

- **Απεριόριστη εμβέλεια:**

Σύμφωνα με αυτή την αρχή όλες οι μεταβλητές και όλες οι σταθερές είναι γνωστές και μπορούν να χρησιμοποιούνται σε οποιοδήποτε τμήμα του προγράμματος, άσχετα που δηλώθηκαν. Όλες οι μεταβλητές είναι καθολικές.

Η απεριόριστη εμβέλεια καταστρατηγεί την αρχή της αυτονομίας των υποπρογραμμάτων, δημιουργεί πολλά προβλήματα και τελικά είναι αδύνατη για μεγάλα προγράμματα με πολλά υποπρογράμματα, αφού ο καθένας που γράφει κάποιο υποπρόγραμμα πρέπει να γνωρίζει τα ονόματα όλων των μεταβλητών που χρησιμοποιούνται στα υπόλοιπα υποπρογράμματα.

- **Περιορισμένη εμβέλεια:**

Η περιορισμένη εμβέλεια υποχρεώνει όλες τις μεταβλητές που χρησιμοποιούνται σε ένα τμήμα προγράμματος, να δηλώνονται σε αυτό το τμήμα. Όλες οι μεταβλητές είναι τοπικές, ισχύουν δηλαδή για το υποπρόγραμμα στο οποίο δηλώθηκαν. Στη ΓΛΩΣΣΑ έχουμε περιορισμένη εμβέλεια. Τα πλεονεκτήματα της περιορισμένης εμβέλειας είναι η απόλυτη αυτονομία όλων των υποπρογραμμάτων και η δυνατότητα να χρησιμοποιείται οποιοδήποτε όνομα, χωρίς να ενδιαφέρει αν το ίδιο χρησιμοποιείται σε άλλο υποπρόγραμμα.

- **Μερικώς περιορισμένη εμβέλεια:**

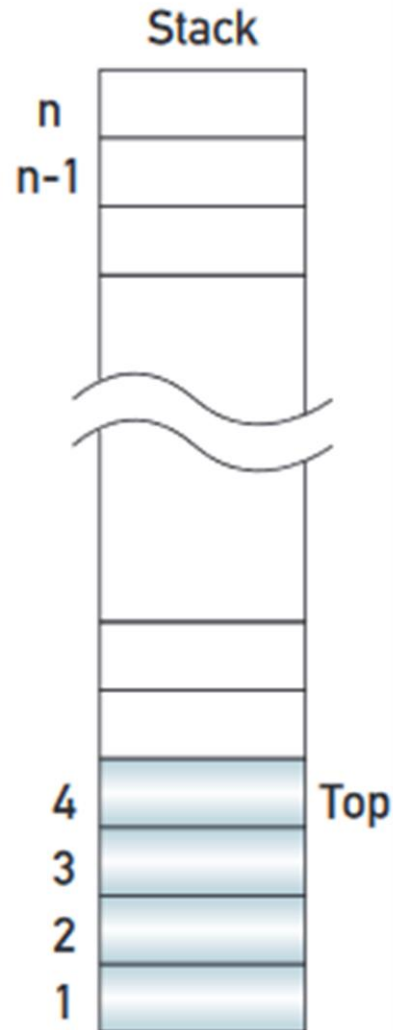
Σύμφωνα με αυτή την αρχή άλλες μεταβλητές είναι τοπικές και άλλες καθολικές. Κάθε γλώσσα προγραμματισμού έχει τους δικούς της κανόνες και μηχανισμούς για τον τρόπο και τις προϋποθέσεις που ορίζονται οι μεταβλητές ως τοπικές ή καθολικές. Η μερικώς περιορισμένη εμβέλεια προσφέρει μερικά πλεονεκτήματα στον πεπειραμένο προγραμματιστή, αλλά για τον αρχάριο περιπλέκει το πρόγραμμα δυσκολεύοντας την ανάπτυξή του.

# ΑΝΑΔΡΟΜΗ

- Αναδρομή ονομάζεται η δυνατότητα ενός υποπρογράμματος να καλεί τον εαυτό του.



# ΣΤΟΪΒΑ ( STACK )



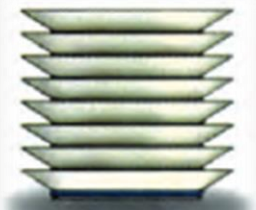
LIFO (Last In, First Out) Δομή Δεδομένων , **δηλαδή:**

**Πώς να σκεφτώ μια στοίβα;**

Μια στοίβα με ταχυδρομικές επιστολές  
Εξέταση των επιστολών από πάνω προς τα κάτω.



Μια στοίβα από πιάτα  
Πλύσιμο των πιάτων από πάνω προς τα κάτω.



Άρα ο,τι εισέρχεται τελευταίο (Last in) ,  
εξέρχεται πρώτο (First out)

**Βασικές ενέργειες μιας στοίβας :**

Τοποθέτηση στοιχείου στην κορυφή μιας στοίβας.

Ονομάζεται εισαγωγή (**push**).

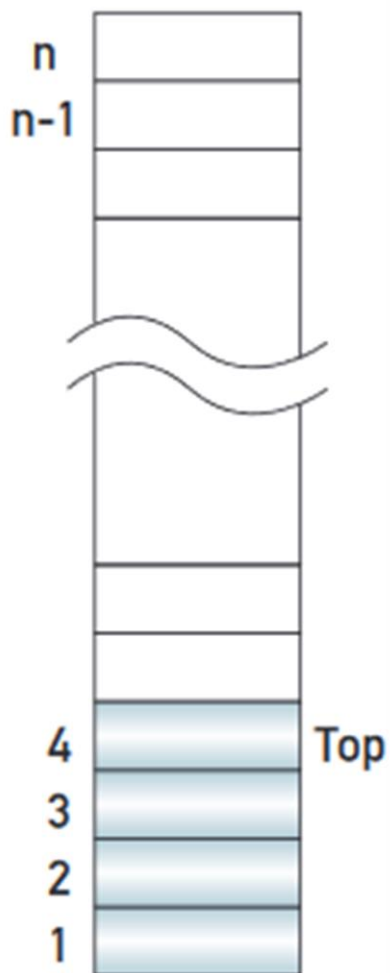
Απομάκρυνση στοιχείου από την κορυφή.

Ονομάζεται εξαγωγή (**pop**).





Stack



# ΣΤΟΪΒΑ ( STACK )

Η διαδικασία της ώθησης πρέπει οπωσδήποτε να ελέγχει, αν η στοίβα είναι γεμάτη, οπότε λέγεται ότι συμβαίνει υπερχείλιση (overflow) της στοίβας.

Αντίστοιχα, η διαδικασία απώθησης ελέγχει, αν υπάρχει ένα τουλάχιστον στοιχείο στη στοίβα, δηλαδή ελέγχει αν γίνεται υποχείλιση (underflow) της στοίβας

Η δομή στοίβα μπορεί να υλοποιηθεί με :  
έναν πίνακα και  
έναν δείκτη στο στοιχείο της κορυφής της στοίβας. (μια βοηθητική μεταβλήτη)

Μια βοηθητική μεταβλητή (με όνομα συνήθως top) χρησιμοποιείται για να δείχνει το στοιχείο που τοποθετήθηκε τελευταίο στην κορυφή της στοίβας. Για την εισαγωγή ενός νέου στοιχείου στη στοίβα (**ώθηση**) αρκεί να αυξηθεί η μεταβλητή top κατά ένα και στη θέση αυτή να εισέλθει το στοιχείο. Αντίθετα για την εξαγωγή ενός στοιχείου από τη στοίβα (**απώθηση**) εξέρχεται πρώτα το στοιχείο που δείχνει η μεταβλητή top και στη συνέχεια η top μειώνεται κατά ένα για να δείχνει τη νέα κορυφή.

# ΠΑΡΑΔΕΙΓΜΑ ΣΤΟΙΒΑΣ

push(10)	push(5)	pop()	push(15)	push(7)	pop()
				7	
	5		15	15	15
10	10	10	10	10	10
Εισαγωγή του 10	Εισαγωγή του 5	Αφαίρεση του πιο πάνω στοιχείου	Εισαγωγή του 15	Εισαγωγή του 7	Αφαίρεση του πιο πάνω στοιχείου





# ΟΥΡΑ ( QUEUE )



**FIFO (First In, First Out)** Δομή Δεδομένων , **δηλαδή:**  
**Πώς να σκεφτώ μια ουρά;**

Μια ουρά αναμονής για επιβίβαση στον Προαστιακό Σιδηρόδρομο.

Μία ουρά αναμονής στο Super Market.

Ουρά εκτυπωτή με εργασίες εκτύπωσης που αναμένουν επεξεργασία.

Ουρά πακέτων δεδομένων που περιμένουν την σειρά τους για να μεταδοθούν στο Διαδίκτυο.

**Άρα ο,τι εισέρχεται πρώτο (First in) ,  
εξέρχεται πρώτο (First out)**

**Βασικές ενέργειες μιας ουράς :**

Εισαγωγή (**enqueue**) στοιχείου στο πίσω άκρο της ουράς.

Εξαγωγή (**dequeue**) στοιχείου από το εμπρός άκρο της ουράς.





# ΟΥΡΑ ( QUEUE )

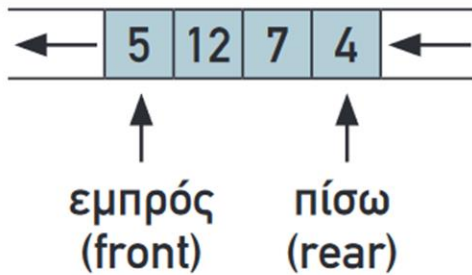


Άρα, σε αντίθεση με τη δομή της στοίβας, στην περίπτωση της ουράς απαιτούνται δύο δείκτες: ο εμπρός (**front**) και ο πίσω (**rear**) δείκτης, που μας δίνουν τη θέση του στοιχείου που σε πρώτη ευκαιρία θα εξαχθεί και τη θέση του στοιχείου που μόλις εισήλθε.

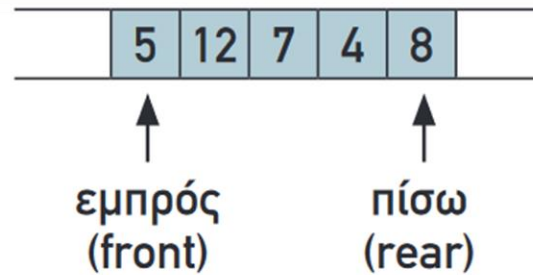
Μία ουρά μπορεί να υλοποιηθεί με τη βοήθεια ενός μονοδιάστατου πίνακα. Για την εισαγωγή ενός νέου στοιχείου στην ουρά **αυξάνεται ο δείκτης rear κατά ένα** και στη θέση αυτή αποθηκεύεται το στοιχείο. Αντίστοιχα για τη λειτουργία της εξαγωγής, εξέρχεται το στοιχείο που δείχνει ο δείκτης front, ο οποίος στη συνέχεια **αυξάνεται κατά ένα, για να δείχνει το επόμενο στοιχείο που πρόκειται να εξαχθεί**. Σε κάθε περίπτωση όμως, πρέπει να ελέγχεται πριν από οποιαδήποτε ενέργεια, αν υπάρχει ελεύθερος χώρος στον πίνακα για την εισαγωγή και αν υπάρχει ένα τουλάχιστον στοιχείο για την εξαγωγή.



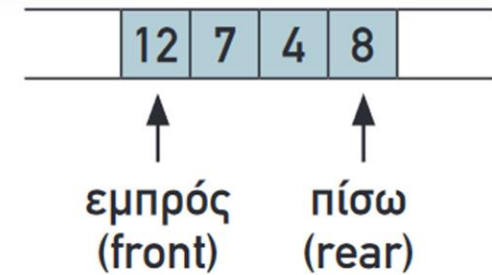
# ΠΑΡΑΔΕΙΓΜΑ ΟΥΡΑΣ



α) Μια ουρά  
με 4 στοιχεία



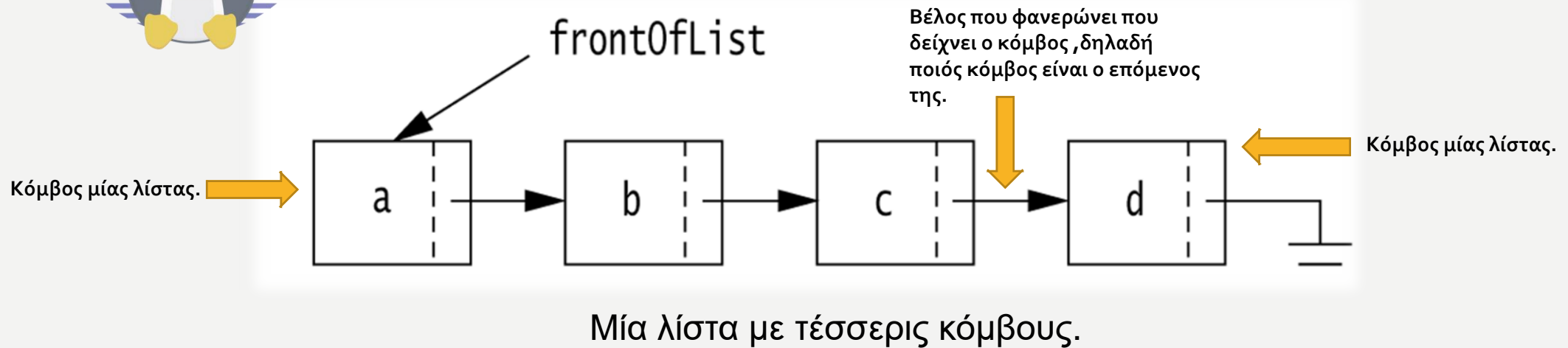
β) Η ουρά μετά  
την εισαγωγή του  
στοιχείου 8



γ) Η ουρά μετά  
την εξαγωγή του  
στοιχείου 5



# ΛΙΣΤΑ ( LIST )



Στις λίστες το κύριο χαρακτηριστικό είναι ότι οι κόμβοι τους συνήθως βρίσκονται σε απομακρυσμένες θέσεις μνήμης και η σύνδεσή τους γίνεται με δείκτες (pointers).

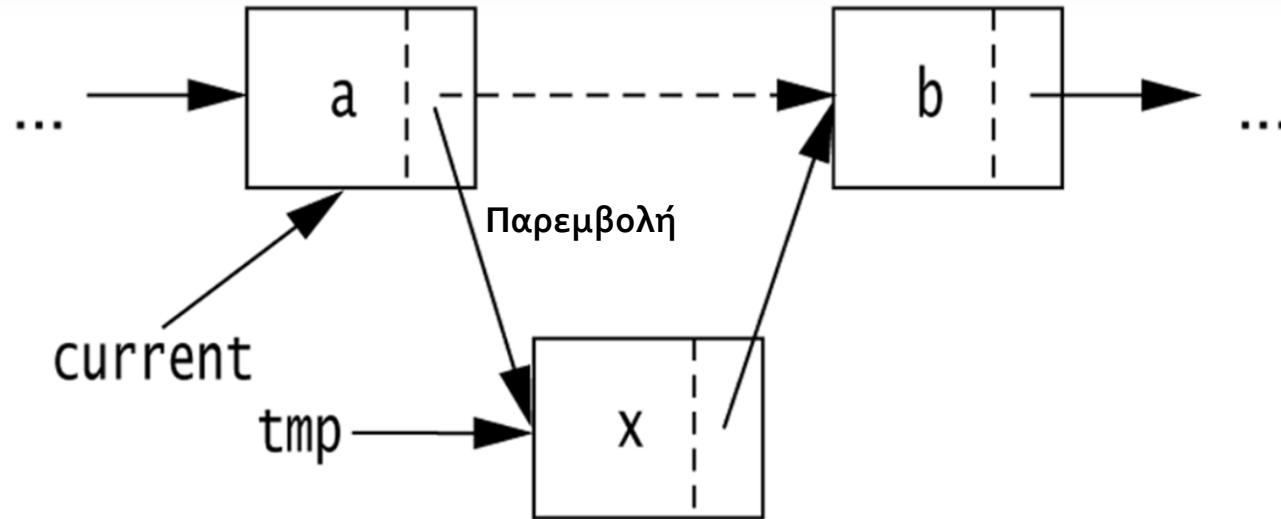
Ο δείκτης **δεν λαμβάνει αριθμητικές τιμές** όπως ακέραιες, πραγματικές κ.ά., αλλά **οι τιμές του είναι διευθύνσεις στην κύρια μνήμη** και χρησιμοποιείται ακριβώς για τη σύνδεση των διαφόρων στοιχείων μιας δομής, που είναι αποθηκευμένα σε μη συνεχόμενες θέσεις μνήμης. Συνήθως ο δείκτης είναι ένα πεδίο κάθε κόμβου της δομής, όπως φαίνεται στο διπλανό σχήμα.

Δεδομένα	Δείκτης
----------	---------

! Οι όροι index και pointer αποδίδονται στα ελληνικά ως δείκτης. Και οι δύο παραπέμπουν σε θέσεις, πίνακα ο πρώτος και μνήμης ο δεύτερος.



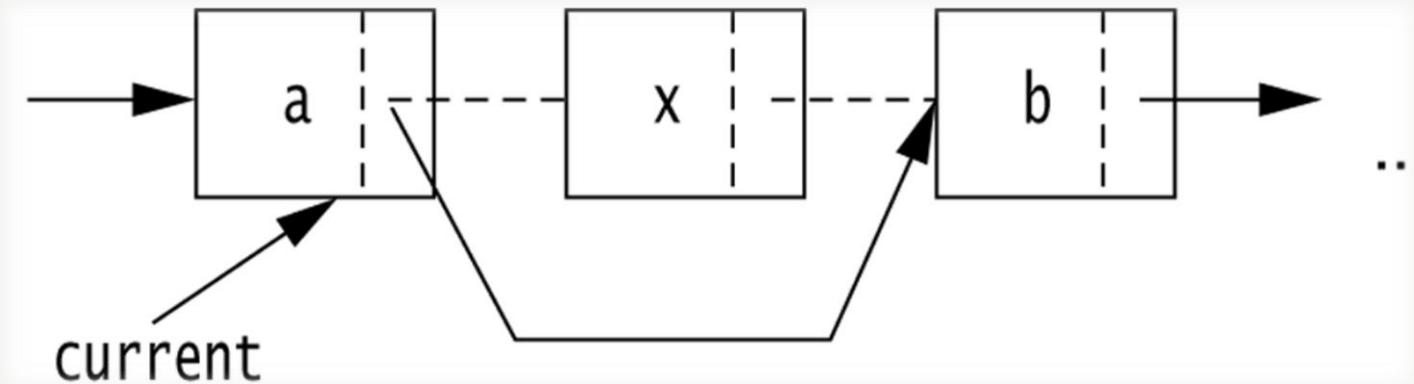
# ΕΙΣΑΓΩΓΗ ΣΕ ΛΙΣΤΑ



Όπως φαίνεται και στο σχήμα, οι απαιτούμενες ενέργειες για την εισαγωγή (**παρεμβολή**) του νέου κόμβου είναι ο δείκτης του δεύτερου κόμβου να δείχνει το νέο κόμβο και ο δείκτης του νέου κόμβου να δείχνει τον τρίτο κόμβο (δηλαδή να πάρει την τιμή που είχε πριν την εισαγωγή ο δείκτης του δεύτερου κόμβου). Έτσι οι κόμβοι της λίστας διατηρούν τη λογική τους σειρά, αλλά οι φυσικές θέσεις στη μνήμη μπορεί να είναι τελείως διαφορετικές



# ΔΙΑΓΡΑΦΗ ΣΕ ΛΙΣΤΑ

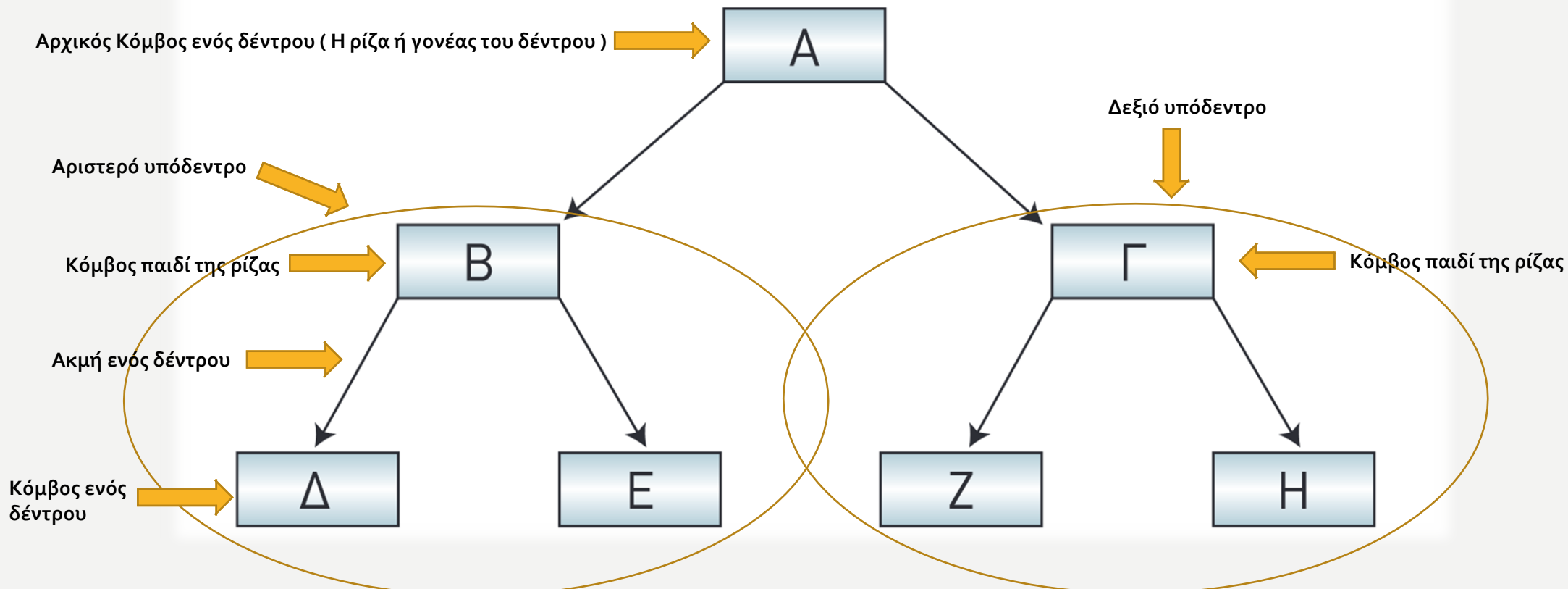


Αντίστοιχα για τη διαγραφή ενός κόμβου αρκεί να αλλάξει τιμή ο δείκτης του προηγούμενου κόμβου και να δείχνει πλέον τον επόμενο αυτού που διαγράφεται. Ο κόμβος που διαγράφηκε (ο τρίτος) αποτελεί "άχρηστο δεδομένο" και ο χώρος μνήμης που καταλάμβανε, παραχωρείται για άλλη χρήση.



# ΔΕΝΤΡΑ

Το κύριο χαρακτηριστικό των δένδρων είναι, ότι από έναν κόμβο δεν υπάρχει ένας μόνο επόμενος κόμβος, αλλά περισσότεροι. Υπάρχει ένας μόνο κόμβος, που λέγεται ρίζα, από τον οποίο ξεκινούν όλοι οι άλλοι κόμβοι.



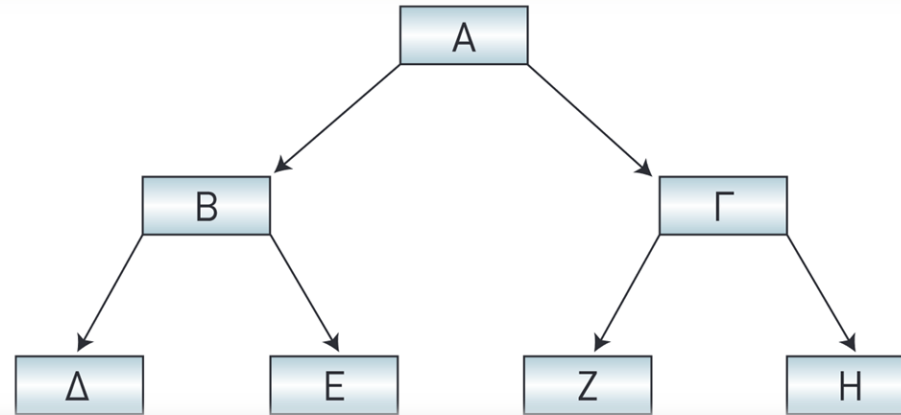
Τα δέντρα είναι μια από τις βασικές δομές δεδομένων που χρησιμοποιούνται στον προγραμματισμό.

Ένα δέντρο συνδυάζει τα πλεονεκτήματα δύο άλλων δομών:

- Ενός ταξινομημένου πίνακα
- Μιας συνδεδεμένης λίστας

# ΔΕΝΤΡΑ

Από τη ρίζα ξεκινούν δύο κόμβοι. Οι κόμβοι αυτοί λέγονται παιδιά της ρίζας. Με την ίδια λογική, από κάθε παιδί της ρίζας ξεκινούν άλλα παιδιά.



Μετακίνηση στους κόμβους του δέντρου μέσω των ακμών:

- Σε μία διαδρομή.
- Από τη ρίζα προς τα κάτω (από πάνω προς τα κάτω).

! Οι δομές δεδομένων που χρησιμοποιούν δείκτες αποκαλούνται **δυναμικές (dynamic)**, γιατί η υλοποίησή τους γίνεται έτσι, ώστε να μην απαιτείται εκ των προτέρων καθορισμός του μέγιστου αριθμού κόμβων. Είναι φανερό ότι οι δομές αυτές είναι πιο ευέλικτες από τη στατική δομή του πίνακα, επειδή επεκτείνονται και συρρικνώνονται κατά τη διάρκεια εκτέλεσης του προγράμματος.



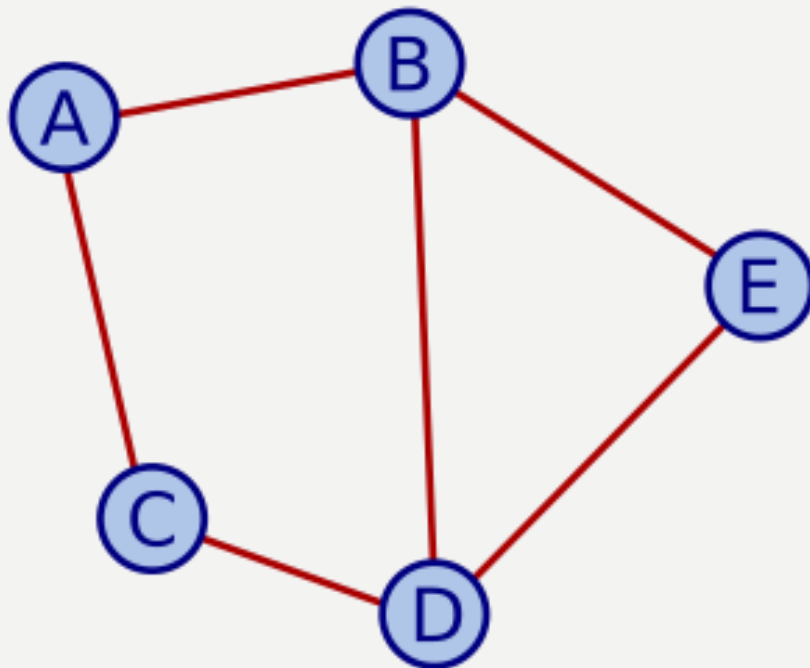


# ΓΡΑΦΟΙ

Ένας γράφος (**graph**) αποτελείται από **ένα σύνολο κόμβων** (ή σημείων ή κορυφών) και ένα σύνολο γραμμών (ή ακμών ή τόξων) που ενώνουν μερικούς ή όλους τους κόμβους.

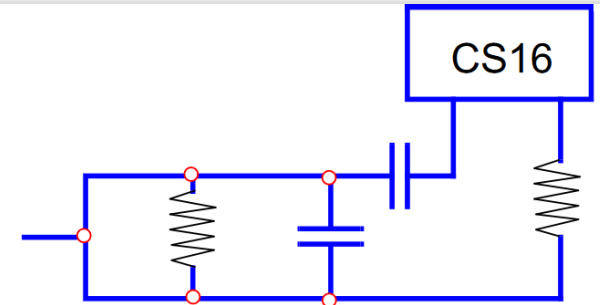
Ο γράφος αποτελεί την πιο γενική δομή δεδομένων, με την έννοια ότι όλες οι προηγούμενες δομές που παρουσιάστηκαν μπορούν να θεωρηθούν περιπτώσεις γράφων.

Πολλά προβλήματα και καταστάσεις της καθημερινής μας ζωής μπορούν να περιγραφούν με τη βοήθεια γράφων.

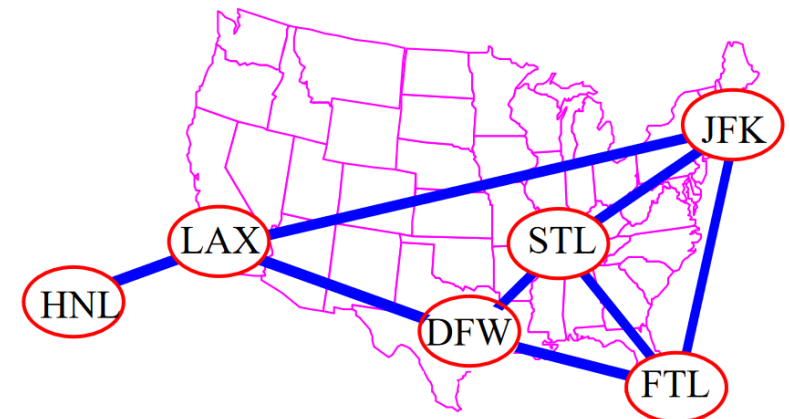


## Παραδείγματα:

Ηλεκτρονικά  
κυκλώματα



Δίκτυα (δρόμων,  
πτήσεων,  
επικοινωνιών)



# ΜΕΘΟΔΟΣ ΔΙΑΙΡΕΙ ΚΑΙ ΒΑΣΙΛΕΥΕ

Στην κατηγορία "*Διαίρει και Βασίλευε*" (divide and conquer) εντάσσονται οι τεχνικές που **υποδιαιρούν ένα πρόβλημα σε μικρότερα υποπροβλήματα**, που έχουν την ίδια τυποποίηση με το αρχικό πρόβλημα αλλά είναι μικρότερα σε μέγεθος.

Με όμοιο τρόπο, τα υποπροβλήματα αυτά μπορούν να διαιρεθούν σε ακόμη μικρότερα υποπροβλήματα κοκ. Έτσι η επίλυση ενός προβλήματος έγκειται στη σταδιακή επίλυση των όσο το δυνατόν μικρότερων υποπροβλημάτων, ώστε τελικά να καταλήξουμε στη συνολική λύση του αρχικού ευρύτερου προβλήματος. Αυτή η προσέγγιση ονομάζεται από επάνω προς τα κάτω (top-down).

Η Δυναμική Αναζήτηση αποτελεί έναν αλγόριθμο που ακολουθεί την φιλοσοφία της μεθόδου Διαίρει και Βασίλευε.



# ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΕΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ



Η αντικειμενοστραφής σχεδίαση εκλαμβάνει ως πρωτεύοντα δομικά στοιχεία ενός προγράμματος τα δεδομένα, από τα οποία δημιουργούνται με κατάλληλη μορφοποίηση τα **αντικείμενα** (objects). Αυτή η σχεδίαση αποδείχθηκε ότι επιφέρει καλύτερα αποτελέσματα, αφού τα προγράμματα που δημιουργούνται είναι περισσότερο ευέλικτα και επαναχρησιμοποιήσιμα.

Φυσικά ο αντικειμενοστραφής προγραμματισμός χρησιμοποιεί την ιεραρχική σχεδίαση, τον τμηματικό προγραμματισμό και ακολουθεί τις αρχές του δομημένου προγραμματισμού.

## ANIMAL CLASS

Σε αυτή την εικόνα το **ANIMAL** είναι μία κλάση!!! Τα **DOG**, **CAT** και **COW** είναι **Αντικείμενα** της κλάσης **ANIMAL**.

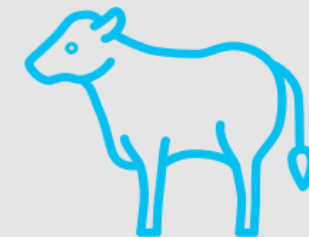
### DOG CLASS



### CAT CLASS

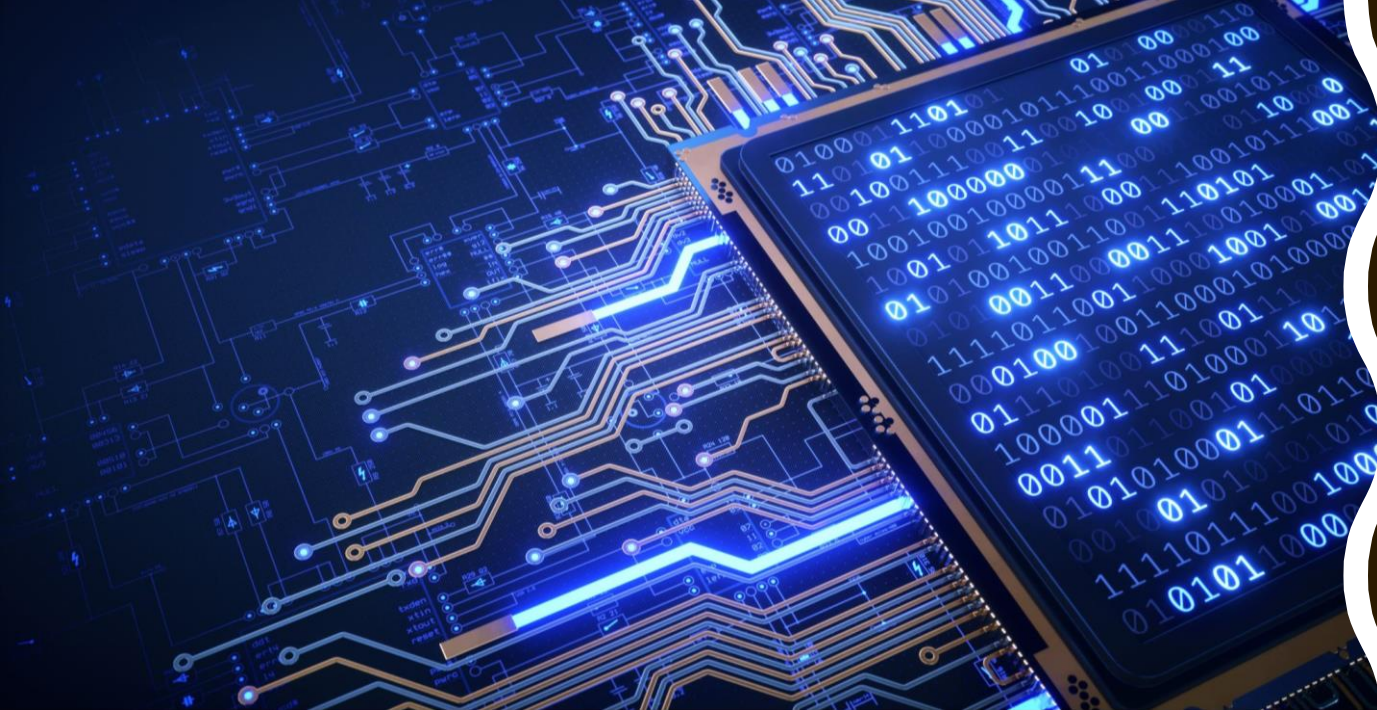


### COW CLASS



### Επεξήγηση των παραπάνω:

Όλα τα ζώα μοιράζονται κάποια κοινά χαρακτηριστικά (π.χ. , έχουν μία μύτη, δύο μάτια) . Για να μην φτιάχνουμε ξανά και ξανά τα ίδια, φτιάχνουμε αυτά τα κοινά χαρακτηριστικά ως μία κλάση. Έπειτα, φτιάχνουμε αντικείμενα που παίρνουν ως default όλα τα κοινά χαρακτηριστικά από την κλάση που ορίζουμε και έπειτα, προσθέτουμε στο αντικείμενο τα δικά του χαρακτηριστικά (π.χ. ο σκύλος γαβγίζει , η γάτα νιαουρίζει κλπ.)



# ΕΚΣΦΑΛΜΑΤΩΣΗ

## Ορισμός:

Η διαδικασία ελέγχου, εντοπισμού και διόρθωσης των σφαλμάτων ενός προγράμματος καλείται εκσφαλμάτωση (debugging). Στόχος της διαδικασίας εκσφαλμάτωσης είναι ο εντοπισμός των σημείων του προγράμματος που προκαλούν προβλήματα στη λειτουργία του.





# ΤΙ ΠΡΕΠΕΙ ΝΑ ΕΧΩ ΣΤΟ ΜΥΑΛΟ ΜΟΥ ΓΙΑ ΤΗΝ ΕΚΣΦΑΛΜΑΤΩΣΗ :

## 01

Η εργασία της εκσφαλμάτωσης δεν είναι εύκολη, απαιτεί βαθιά γνώση της γλώσσας προγραμματισμού και φυσικά αντίστοιχες ικανότητες από τον προγραμματιστή. Για τον εντοπισμό ενός λάθους δεν υπάρχουν ιδιαίτερα μυστικά και τρυκ. Η εκσφαλμάτωση είναι ένα πρόβλημα λογικής και όσο πιο καλά αντιλαμβάνεται ο προγραμματιστής τον τρόπο που εργάζεται το πρόγραμμα, τόσο πιο εύκολα και σύντομα θα εντοπίσει λάθη που προκαλούν δυσλειτουργίες.

## 02

Σε ένα σύγχρονο προγραμματιστικό περιβάλλον δεν χρειάζεται ιδιαίτερα μνεία για τα λάθη που παρουσιάζονται κατά το χρόνο σχεδιασμού, αφού αυτά, όπως αναφέρθηκε, είναι συντακτικά λάθη και τις περισσότερες φορές το περιβάλλον προγραμματισμού τα ανιχνεύει αυτόματα και προτείνει τη διόρθωσή τους. Ακόμη και αν το περιβάλλον δεν προτείνει τη διόρθωση, ο μεταγλωττιστής συλλαμβάνει και περιγράφει το λάθος και στη συνέχεια ο προγραμματιστής μπορεί πολύ εύκολα να το διορθώσει.

## 03

Τα λάθη που κυρίως μας απασχολούν στη φάση της εκσφαλμάτωσης είναι τα λογικά λάθη και τα λάθη που παρουσιάζονται κατά το χρόνο εκτέλεσης του προγράμματος. Η εκσφαλμάτωση τέτοιων λαθών μπορεί να γίνει μέσα από εργαλεία εκσφαλμάτωσης ή από ειδικές εντολές ή συναρτήσεις που προσφέρει το περιβάλλον προγραμματισμού.



## ΕΛΕΓΧΟΣ ΜΑΥΡΟΥ ΚΟΥΤΙΟΥ



Μια δημοφιλής τεχνική ελέγχου είναι ο έλεγχος μαύρου κουτιού (black-box testing).



Ονομάζεται έτσι διότι τα δεδομένα εισόδου στα σενάρια ελέγχου προκύπτουν από τις προδιαγραφές του προγράμματος, αγνοώντας εντελώς τον κώδικα. Δηλαδή το πρόγραμμα μοιάζει σαν να βρίσκεται μέσα σε ένα μαύρο κουτί που κρύβει το περιεχόμενό του.

YouTube GR 250 ΣΛ

Activities Document Viewer Feb 19 17:32 Σ\_Λ\_version4.0.pdf 400%

## Λίστα με προτεινόμενα Σ/Λ για εξέταση στις Πανελλήνιες στο μάθημα «Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον»

Βαγιακάκος Δημήτριος  
DimitirsV SV1SJP  
<https://www.youtube.com/LinuxOSblog>

Απαγορεύεται η εμπορική αναπαραγωγή αυτού του φυλλαδίου!

Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον:

ΔΩΡΕΑΝ 250+ Λυμένα Σ/Λ για Πανελλήνιες ΑΕΠΠ - 4η έκδοση

TuxHouse 6,94 χιλ. εγγεγραμμένοι

897 προβολές 20 Φεβ 2021  
ΔΩΡΕΑΝ 250+ Λυμένα Σ/Λ για Πανελλήνιες ΑΕΠΠ - 4η έκδοση

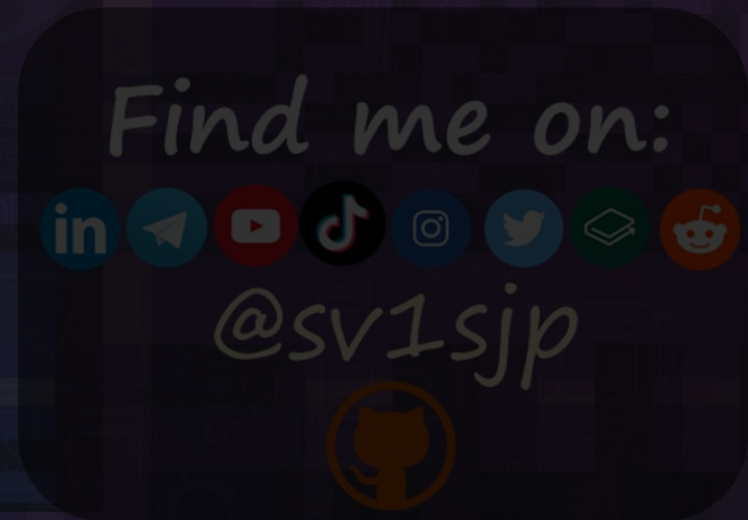


Δεν ξεχνάμε να διαβάσουμε και τα 250++ Λυμένα Σ/Λ !!!!!  
[https://www.youtube.com/watch?v=YB\\_nCsKwhcw](https://www.youtube.com/watch?v=YB_nCsKwhcw)



# ΑΠΟΡΙΕΣ;

Για οποιαδήποτε απορία ή διευκρίνηση,  
στείλτε μας μήνυμα στο Instagram  
@TuxHouseEdu



Και μία εγγραφή στο κανάλι TuxHouse στο [YouTube](#), [LBRY](#) & [TikTok](#) μας βοηθάει να εξελιχθούμε περισσότερο, παράγοντας ακόμη περισσότερο και ποιοτικότερο δωρεάν υλικό!