ΘΕΩΡΙΑ ΑΕΠΠ

2022

Νίκος Παπαδόπουλος

Ανάλυση Προβλήματος

1.1 | Η έννοια πρόβλημα

ΟΡΙΣΜΟΣ
Με τον όρο Πρόβλημα εννοείται μια κατάσταση η οποία χρήζει αντιμετώπισης, απαιτεί λύση, η δε λύση της δεν είναι γνωστή, ούτε προφανής.

Παραδείγματα προβλημάτων

- Προβλήματα κοινωνικής αδικίας και εκμετάλλευσης
- Σοβαρότατα προβλήματα επιδημιών, όπως η πανούκλα, η χολέρα και η λύσσα
- Το πρόβλημα της μεταφοράς της ηλεκτρικής ενέργειας από τον τόπο παραγωγής στα σημεία κατανάλωσης
- Το πρόβλημα της τρύπας του όζοντος, και κατ' επέκταση το πρόβλημα της προστασίας του φυσικού περιβάλλοντος
- Φυσικά φαινόμενα, όπως εκρήξεις ηφαιστείων, παλιρροιακά κύματα, σεισμοί και τυφώνες
- Ο υποσιτισμός ενός πολύ μεγάλου μέρους του πληθυσμού της αφρικανικής κύρια ηπείρου, οι καθημερινοί θάνατοι πολλών ανθρώπων, ειδικά μικρών παιδιών

1.2 | Κατανόηση προβλήματος

Η οποιαδήποτε προσπάθεια αντιμετώπισης ενός προβλήματος είναι καταδικασμένη σε αποτυχία αν προηγουμένως δεν έχει γίνει απόλυτα κατανοητό το πρόβλημα που τίθεται.

Η κατανόηση ενός προβλήματος αποτελεί συνάρτηση δύο παραγόντων:

- της σωστής διατύπωσης εκ μέρους του δημιουργού του
- της αντίστοιχα σωστής ερμηνείας από τη μεριά εκείνου που καλείται να το αντιμετωπίσει

Η μορφή με την οποία παρουσιάζεται ένα πρόβλημα μπορεί να είναι οποιαδήποτε αρκεί να μπορεί να γίνει αντιληπτή από μία από τις πέντε ανθρώπινες αισθήσεις.

Τα προβλήματα μπορεί να είναι πολύπλοκα ή σχετικά απλά, πρωτόγνωρα ή συνηθισμένα. Σε κάθε όμως περίπτωση θα πρέπει να γίνουν απολύτως κατανοητά πριν γίνει κάθε προσπάθεια αντιμετώπισής τους. Η κατανόηση ενός προβλήματος εξαρτάται σε μεγάλο βαθμό από τη διατύπωσή του.

*Συνηθέστερο μέσο για να αποδοθεί η διατύπωση ενός προβλήματος είναι ο λόγος, είτε ο **προφορικός** είτε ο **γραπτός.***

Σαφήνεια διατύπωσης

Ο λόγος σαν μέσο επικοινωνίας και συνεννόησης πρέπει να χαρακτηρίζεται από σαφήνεια.

- Άστοχη χρήση ορολογίας
- λανθασμένη σύνταξη

είναι δύο στοιχεία που μπορούν να προκαλέσουν παρερμηνείες και παραπλανήσεις. Η παρερμηνεία είναι δυνατή ακόμα και σε περιπτώσεις όπου όλοι οι λεξικολογικοί και συντακτικοί κανόνες κρατούνται με ευλάβεια.

Η διατύπωση ***«Ο Γιάννης και η Μαρία είναι παντρεμένοι»*** έχει δυο ερμηνείες.

Πρώτη ερμηνεία: Ο Γιάννης και η Μαρία είναι παντρεμένοι μεταξύ τους.

Δεύτερη ερμηνεία: Ο Γιάννης είναι παντρεμένος και η Μαρία είναι παντρεμένη.

ΟΡΙΣΜΟΙ

Με τον όρο **δεδομένο** δηλώνεται οποιοδήποτε στοιχείο μπορεί να γίνει αντιληπτό από έναν τουλάχιστον παρατηρητή με μία από τις πέντε αισθήσεις του.

Με τον όρο **πληροφορία** αναφέρεται οποιοδήποτε γνωσιακό στοιχείο προέρχεται από επεξεργασία δεδομένων.

Ο όρος **επεξεργασία δεδομένων** δηλώνει εκείνη τη διαδικασία κατά την οποία ένας "μηχανισμός" δέχεται δεδομένα, τα επεξεργάζεται σύμφωνα με έναν προκαθορισμένο τρόπο και αποδίδει πληροφορίες. Επί χιλιετίες ο "μηχανισμός" επεξεργασίας των δεδομένων ήταν και εξακολουθεί να είναι ο ανθρώπινος εγκέφαλος. Στις μέρες μας, ένας άλλος "μηχανισμός" επεξεργασίας δεδομένων είναι ο υπολογιστής.

1.3 | Δομή προβλήματος

ΟΡΙΣΜΟΣ

Με τον όρο **δομή ενός προβλήματος** αναφερόμαστε στα συστατικά του μέρη, στα επιμέρους τμήματα που το αποτελούν καθώς επίσης και στον τρόπο που αυτά τα μέρη συνδέονται μεταξύ τους.

Η δυσκολία αντιμετώπισης των προβλημάτων ελαττώνεται όσο περισσότερο προχωράει η ανάλυσή τους σε απλούστερα προβλήματα. Τα νέα προβλήματα μπορούν να αναλυθούν σε άλλα, ακόμη πιο απλά. Η διαδικασία αυτή της ανάλυσης μπορεί να συνεχιστεί μέχρις ότου τα επιμέρους προβλήματα που προέκυψαν θεωρηθούν αρκετά απλά και η αντιμετώπισή τους χαρακτηριστεί ως δυνατή.

1.4 | Καθορισμός απαιτήσεων

Η σωστή επίλυση ενός προβλήματος προϋποθέτει

- τον επακριβή προσδιορισμό των δεδομένων που παρέχει το πρόβλημα
- τη λεπτομερειακή καταγραφή των ζητούμενων που αναμένονται σαν αποτελέσματα της επίλυσης του προβλήματος

Υπάρχουν πολλές περιπτώσεις προβλημάτων όπου τα δεδομένα θα πρέπει να "ανακαλυφθούν" μέσα στα λεγόμενα του προβλήματος. Η διαδικασία αυτή απαιτεί προσοχή, συγκέντρωση και σκέψη.

Το ίδιο προσεκτικά θα πρέπει να αποσαφηνιστούν και τα ζητούμενα του προβλήματος. Δεν είναι πάντοτε ιδιαίτερα κατανοητό τι ακριβώς ζητάει ένα πρόβλημα. Σε μια τέτοια περίπτωση θα πρέπει να

θέτονται μια σειρά από ερωτήσεις με στόχο τη διευκρίνιση πιθανών αποριών σχετικά με τα ζητούμενα, τον τρόπο παρουσίασής τους, το εύρος τους κ.λπ.

Τα στάδια αντιμετώπισης ενός προβλήματος είναι τρία:

- **κατανόηση**, όπου απαιτείται η σωστή και πλήρης αποσαφήνιση των δεδομένων και των ζητούμενων του προβλήματος
- **ανάλυση**, όπου το αρχικό πρόβλημα διασπάται σε άλλα επιμέρους απλούστερα προβλήματα
- **επίλυση**, όπου υλοποιείται η λύση του προβλήματος, μέσω της λύσης των επιμέρους προβλημάτων

Βασικές Έννοιες Αλγορίθμων

2.1 | Τι είναι αλγόριθμος

ΟΡΙΣΜΟΣ

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

Κάθε αλγόριθμος απαραίτητα ικανοποιεί τα επόμενα κριτήρια.

- **Είσοδος** 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** Ο αλγόριθμος πρέπει να δημιουργεί τουλάχιστον μία τιμή δεδομένων ως αποτέλεσμα προς το χρήστη ή προς έναν άλλο αλγόριθμο.
- **Καθοριστικότητα** Κάθε εντολή πρέπει να καθορίζεται χωρίς καμία αμφιβολία για τον τρόπο εκτέλεσής της. Λόγου χάριν, μία εντολή διαίρεσης πρέπει να θεωρεί και την περίπτωση όπου ο διαιρέτης λαμβάνει μηδενική τιμή ή η υπόριζος ποσότητα είναι αρνητική.
- **Περατότητα** 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.
- **Αποτελεσματικότητα** Κάθε μεμονωμένη εντολή του αλγορίθμου να είναι απλή. Αυτό σημαίνει ότι μία εντολή δεν αρκεί να έχει ορισθεί, αλλά πρέπει να είναι και εκτελέσιμη.

Η έννοια του αλγορίθμου δεν συνδέεται αποκλειστικά και μόνο με προβλήματα της Πληροφορικής.

2.2 | Σπουδαιότητα αλγορίθμων

Η Πληροφορική μπορεί να ορισθεί ως η επιστήμη που μελετά τους αλγορίθμους από τις ακόλουθες σκοπιές:

Υλικού (hardware). Η ταχύτητα εκτέλεσης ενός αλγορίθμου επηρεάζεται από τις διάφορες τεχνολογίες υλικού, δηλαδή από τον τρόπο που είναι δομημένα σε μία ενιαία αρχιτεκτονική τα διάφορα συστατικά του υπολογιστή (δηλαδή ανάλογα με το αν ο υπολογιστής έχει κρυφή μνήμη και πόση, ανάλογα με την ταχύτητα της κύριας και δευτερεύουσας μνήμης κ.ο.κ.).

Γλωσσών Προγραμματισμού (programming languages). Το είδος της γλώσσας προγραμματισμού που χρησιμοποιείται (δηλαδή, χαμηλότερου ή υψηλότερου επιπέδου) αλλάζει τη δομή και τον αριθμό των εντολών ενός αλγορίθμου. Γενικά μία γλώσσα που είναι χαμηλότερου επιπέδου (όπως η assembly ή η γλώσσα C) είναι ταχύτερη από μία άλλη γλώσσα που είναι υψηλοτέρου επιπέδου (όπως η Basic ή Pascal). Ακόμη, σημειώνεται ότι διαφορές συναντώνται μεταξύ των γλωσσών σε σχέση με το πότε εμφανίσθηκαν.

Θεωρητική (theoretical). Το ερώτημα που συχνά τίθεται είναι αν πράγματι υπάρχει ή όχι κάποιος αποδοτικός αλγόριθμος για την επίλυση ενός προβλήματος. Η προσέγγιση αυτή είναι ιδιαίτερα σημαντική, γιατί προσδιορίζει τα όρια της λύσης που θα βρεθεί σε σχέση με ένα συγκεκριμένο πρόβλημα.

Αναλυτική (analytical). Μελετώνται οι υπολογιστικοί πόροι (computer resources) που απαιτούνται από έναν αλγόριθμο, όπως για παράδειγμα το μέγεθος της κύριας και της δευτερεύουσας μνήμης, ο χρόνος για λειτουργίες CPU και για λειτουργίες εισόδου/εξόδου κ.λπ.

2.3 | Περιγραφή και αναπαράσταση αλγορίθμων

Υπάρχουν διάφοροι τρόποι αναπαράστασης ενός αλγορίθμου:

- **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, είναι το διάγραμμα ροής (flow chart). Ωστόσο η χρήση διαγραμμάτων ροής για την παρουσίαση αλγορίθμων δεν αποτελεί την καλύτερη λύση, γι' αυτό και εμφανίζονται όλο και σπανιότερα στη βιβλιογραφία και στην πράξη.
- **με φυσική γλώσσα (natural language) κατά βήματα.** Στην περίπτωση αυτή χρειάζεται προσοχή, γιατί μπορεί να παραβιασθεί το τρίτο βασικό χαρακτηριστικό ενός αλγορίθμου, όπως προσδιορίσθηκε προηγουμένως, δηλαδή το κριτήριο του καθορισμού.
- **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.

2.4 | Βασικές συνιστώσες/εντολές ενός αλγορίθμου

Οι συνιστώσες ενός αλγορίθμου, είναι οι απαραίτητες εντολές για τη συγγραφή προγραμμάτων.

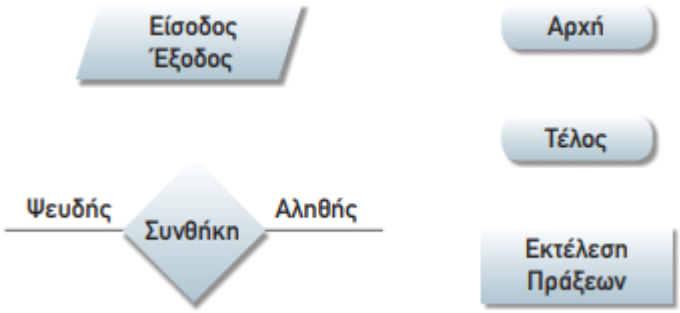
Αυτές είναι:

- σειριακές εντολές (ΔΙΑΒΑΣΕ, ΓΡΑΨΕ, ΕΜΦΑΝΙΣΕ)
- αναθέσεις τιμών (ΕΝΤΟΛΗ ΕΚΧΩΡΗΣΗΣ ←)
- επιλογής με βάση κριτήρια (ΑΝ… ΤΟΤΕ… ΑΛΛΙΩΣ…)
- διαδικασιών επανάληψης (ΟΣΟ | ΓΙΑ | ΜΕΧΡΙΣ_ΟΤΟΥ)
- ενεργειών πολλαπλών επιλογών (ΑΝ… ΤΟΤΕ… ΑΛΛΙΩΣ_ΑΝ… | ΕΠΙΛΕΞΕ)
- συνδυασμός εμφωλευμένων περιπτώσεων

Σύμβολα διαγράμματος ροής

Ένα διάγραμμα ροής αποτελείται από ένα σύνολο γεωμετρικών σχημάτων, όπου το καθένα δηλώνει μία συγκεκριμένη ενέργεια ή λειτουργία. Τα γεωμετρικά σχήματα ενώνονται μεταξύ τους με βέλη, που δηλώνουν τη σειρά εκτέλεσης των ενεργειών αυτών. Τα κυριότερα χρησιμοποιούμενα γεωμετρικά σχήματα είναι τα εξής:

- **έλλειψη**, που δηλώνει την αρχή και το τέλος του κάθε αλγορίθμου
- **ρόμβος**, που δηλώνει μία ερώτηση με δύο ή περισσότερες εξόδους για απάντηση
- **ορθογώνιο**, που δηλώνει την εκτέλεση μίας ή περισσότερων πράξεων
- **πλάγιο παραλληλόγραμμο,** που δηλώνει είσοδο ή έξοδο στοιχείων. Πολλές φορές το σχήμα αυτό μπορεί να διαφοροποιείται προκειμένου να προσδιορίζεται και το είδος της συσκευής απ' όπου γίνεται η είσοδος ή η έξοδος.



2.4.1 Δομή ακολουθίας

Η ακολουθιακή δομή εντολών (σειριακών βημάτων) χρησιμοποιείται πρακτικά για την αντιμετώπιση απλών προβλημάτων, όπου είναι δεδομένη η σειρά εκτέλεσης ενός συνόλου ενεργειών. Ένα απλό παράδειγμα από την καθημερινή ζωή είναι η ακολουθία οδηγιών μίας συνταγής μαγειρικής με στόχο την κατασκευή ενός φαγητού. Τα βήματα και οι ποσότητες που πρέπει να ακολουθηθούν είναι συγκεκριμένα και οι οδηγίες απόλυτα καθορισμένες και σαφείς.

Οι εντολές που χρησιμοποιούμε στη δομή ακολουθίας είναι

ΔΙΑΒΑΣΕ (εκτελεστέα εντολή). Διαβάζει από το πληκτρολόγιο και καταχωρεί στη μεταβλητή/ές που τη συνοδεύει/ουν μια τιμή.

ΔΙΑΒΑΣΕ όνομα, ηλικία

ΕΜΦΑΝΙΣΕ/ΓΡΑΨΕ/ΕΚΤΥΠΩΣΕ. Εμφανίζει στην οθόνη ή τυπώνει στον εκτυπωτή την τιμή της μεταβλητής που συνοδεύει την εντολή

ΓΡΑΨΕ μέσος_όρος

ΕΝΤΟΛΗ ΕΚΧΩΡΗΣΗΣ (←). Εκχωρεί στη μεταβλητή την τιμή ή το αποτέλεσμα της πράξης που βρίσκεται δεξιά από το ←

μήνυμα ← "καλημέρα"

μέσος_όρος ← (α + β) / 2

ηλικία ← 18

```
Αλγόριθμος Παράδειγμα_1
Διάβασε a
Διάβασε b
c ← a + b
Εκτύπωσε c
Τέλος Παράδειγμα_1
```

Αλγόριθμος = δηλωτική εντολή

Διάβασε= διαβάζει από το πληκτρολόγιο και καταχωρεί στη μεταβλητή/ές που τη συνοδεύει/ουν (a, b) μια τιμή)

← εντολή εκχώρησης (εισάγει το αποτέλεσμα της πράξης a+b στη μεταβλητή c)

Εκτύπωσε= τυπώνει την τιμή της μεταβλητής c. Εναλλακτικά χρησιμοποιούμε την εντολή Εμφάνισε.

Η γενική σύνταξη της εντολής εκχώρησης είναι:

μεταβλητή ← έκφραση

Σταθερές (constants). Είναι προκαθορισμένες τιμές που παραμένουν αμετάβλητες σε όλη τη διάρκεια της εκτέλεσης ενός αλγορίθμου. Οι σταθερές διακρίνονται σε

- αριθμητικές, π.χ. 123, +5, -1,25
- αλφαριθμητικές π.χ. "Ιούνιος 2022", "Κατάσταση αποτελεσμάτων"
- λογικές που είναι ακριβώς δύο, Αληθής και Ψευδής

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 διακρίνονται σε

- αριθμητικές
- αλφαριθμητικές
- λογικές

Τελεστές (operators). Πρόκειται για τα γνωστά σύμβολα που χρησιμοποιούνται στις διάφορες πράξεις. Οι τελεστές διακρίνονται σε

- αριθμητικούς (+, -, *, /, ^, div, mod)
- λογικούς (και, ή, όχι)
- συγκριτικούς (<=, <, >=, >, =, <>)

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.

Για παράδειγμα

B * 2 + Y^3

5 * X - (A + 3) > 8 * Y

X > 4 ΚΑΙ X <= 24

Πρέπει να ξέρω

Δεσμευμένες λέξεις είναι λέξεις που χρησιμοποιούνται σε έναν αλγόριθμο για να περιγράψουν συγκεκριμένες ενέργειες. Οι λέξεις αυτές δεν μπορούν να χρησιμοποιηθούν για ονόματα σταθερών ή μεταβλητών.

Μεταβλητές

- Μια μεταβλητή έχει όνομα, τύπο και τιμή.
- Ποτέ το όνομα μιας μεταβλητής δεν μπορεί να ξεκινάει με αριθμό.
- Τα ονόματα των μεταβλητών και των σταθερών επιλέγονται με τέτοιο τρόπο ώστε να εξηγούν τη σημασία και το ρόλο των μεγεθών που εκφράζουν.

Αποδεκτά ονόματα μεταβλητών

1. να ξεκινούν πάντα από γράμμα
2. να περιέχουν μόνο γράμματα, αριθμούς και την κάτω παύλα (_)
3. να μην είναι δεσμευμένη λέξη

Παραδείγματα αποδεκτών ονομάτων μεταβλητών

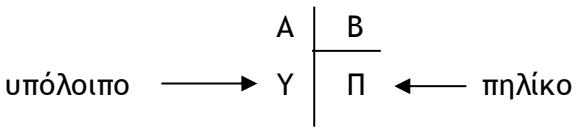
A, α	Ένα γράμμα
B1, C_1	Γράμμα και αριθμός
βαθμός, όνομα, μέσος_όρος, αριθ	Μια ή δύο λέξεις ή μέρος λέξης

Προτεραιότητα τελεστών

Παρενθέσεις > Δυνάμεις > Πολλαπλασιασμοί / Διαιρέσεις > Προσθέσεις / Αφαιρέσεις > Συγκρίσεις > Άρνηση > Σύζευξη > Διάζευξη

Τι είναι το mod και τι το div (ακέραια διαίρεση)

Έστω ότι έχουμε δύο ακέραιους αριθμούς τον Α και τον Β και ότι θέλουμε να τους διαιρέσουμε



- το πηλίκο της ακέραιας διαίρεσης συμβολίζεται με **A div B**
- υπόλοιπο της ακέραιας διαίρεσης συμβολίζεται με **A mod B**

Π.χ

10 mod 3 = 1 2 mod = 2

10 div 3 = 3 2 div = 0

Πότε χρησιμοποιούνται οι τελεστές mod και div

1. Για να δείξουμε ότι ένας ακέραιος αριθμός Α είναι πολλαπλάσιο ενός ακεραίου Β. Σε αυτή την περίπτωση **Α mod Β = 0**
2. Για να διαχωρίσουμε τους άρτιους από τους περιττούς. Ένας ακέραιος Α είναι άρτιος αν **Α mod 2 = 0,** αλλιώς είναι περιττός
3. Για να πάρουμε ως αποτέλεσμα μιας διαίρεσης μεταξύ ακεραίων πάντα έναν ακέραιο αριθμό.

Παραδείγματα εκχώρησης τιμής

α ← 2	Δώσε στη μεταβλητή α την τιμή 2
α ← β + γ	Δώσε στη μεταβλητή α το άθροισμα των τιμών των μεταβλητών β και γ
τ ← αληθής	Δώσε στη μεταβλητή τ την λογική τιμή αληθής
τ ← 'αληθής'	Δώσε στη μεταβλητή τ την τιμή-λέξη αληθής
α ← α + 1	Αύξησε την τιμή του α κατά 1
β ← x > α	Δώσε στη μεταβλητή β το αποτέλεσμα της σύγκρισης x > α δηλ. το αληθής ή το ψευδής

Στο αριστερό μέρος της εντολής εκχώρησης δεν είναι δυνατό να υπάρχει παράσταση.

*Για παράδειγμα η έκφραση 2*Α ← 3*Α+5 δεν είναι αποδεκτή*

Λογικές πράξεις

Στις παρακάτω προτάσεις

- αν βρέχει **ή** αν χιονίζει θα πάρω ομπρέλα
- αν έχει ήλιο **και** αν έχει ζέστη θα πάρω καπέλο
- αν **δεν** έχει ήλιο θα πάρω ομπρέλα

χρησιμοποιούμε τρεις λογικούς τελεστές **Ή, ΚΑΙ, ΟΧΙ** (δεν) για να συνδέσουμε 2 διαφορετικές συνθήκες (προτάσεις).

- *απάντηση = 'ναι' **Ή** απάντηση = 'ΝΑΙ'*
- *Χ >= 10 **ΚΑΙ** Χ <= 20*

Έτσι προκύπτουν οι τρεις λογικές πράξεις.

Ο επόμενος πίνακας δίνει τις τιμές των τριών αυτών λογικών πράξεων για όλους τους συνδυασμούς τιμών δύο προτάσεων Α και Β.

Πρόταση Α	Πρόταση Β	Α ή Β	Α και Β	όχι Α
Αληθής	Αληθής	Αληθής	Αληθής	Ψευδής
Αληθής	Ψευδής	Αληθής	Ψευδής	Ψευδής
Ψευδής	Αληθής	Αληθής	Ψευδής	Αληθής
Ψευδής	Ψευδής	Ψευδής	Ψευδής	Αληθής

2.4.2 Δομή Επιλογής

Στην πραγματικότητα πολύ λίγα προβλήματα μπορούν να επιλυθούν με τον προηγούμενο τρόπο της σειριακής/ακολουθιακής δομής ενεργειών.

Συνήθως πρέπει να λαμβάνονται κάποιες αποφάσεις με βάση κάποια δεδομένα κριτήρια. Η διαδικασία της επιλογής περιλαμβάνει τον έλεγχο κάποιας συνθήκης που μπορεί να έχει δύο τιμές (Αληθής ή Ψευδής) και ακολουθεί η απόφαση εκτέλεσης κάποιας ενέργειας με βάση την τιμή της συνθήκης.

Παράδειγμα : **"αν βρέχει, θα πάρω ομπρέλα, αλλιώς θα πάρω καπέλο"**.

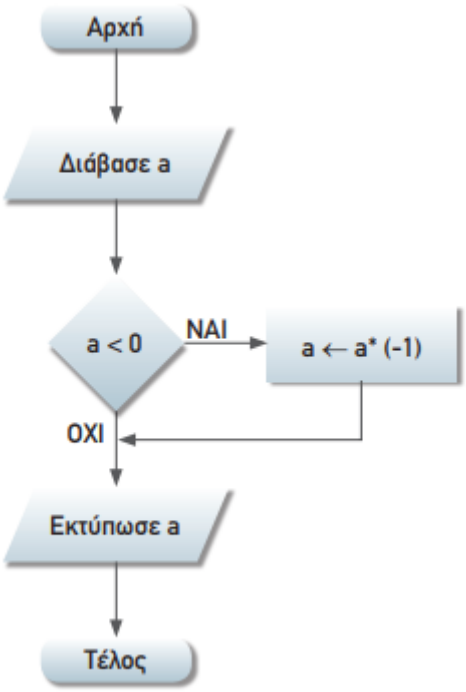
Η **συνθήκη** εδώ είναι το "αν βρέχει", ενώ η απόφαση είναι είτε να πάρω την "ομπρέλα" είτε το "καπέλο" με βάση την "τιμή" της συνθήκης.

Απλή επιλογή

```
Αν συνθήκη τότε
    εντολή_1
    εντολή_2
    ..............
    εντολή_ν
Τέλος_αν
```

```
Αλγόριθμος Παράδειγμα_2
Διάβασε a
Αν a < 0 τότε a ← a*(-1)
Εκτύπωσε a
Τέλος Παράδειγμα_2
```

Διάγραμμα ροής



Διαβάζει έναν αριθμό από το πληκτρολόγιο και τον εκχωρεί στη μεταβλητή a.

Συγκρίνει τον αριθμό a με το 0. Αν είναι μικρότερος τότε εκτελείται το ΝΑΙ το οποίο εκχωρεί στη μεταβλητή a τον αντίθετό του.

Στην περίπτωση που το a >= 0 τότε εκτελείται το ΟΧΙ.

Η εντολή Εκτύπωσε a εκτελείται και στις δύο περιπτώσεις.

Η μεταβλητή a είναι και είσοδος αλλά και έξοδος του αλγορίθμου. Επιπλέον, ο αλγόριθμος έχει καθορισμένη κάθε του εντολή (καθοριστικότητα), τελειώνει μετά από πεπερασμένο αριθμό βημάτων (περατότητα), ενώ κάθε εντολή του είναι ιδιαίτερα απλή κατά την εκτέλεσή της (αποτελεσματικότητα). Έτσι προκύπτει ότι ο αλγόριθμος αυτός πράγματι πληροί όλα τα κριτήρια.

Σύνθετη επιλογή

```
Αν συνθήκη τότε
    εντολή ή εντολές
αλλιώς
    εντολή ή εντολές
Τέλος_αν
```

```
Αλγόριθμος Παράδειγμα_3
Διάβασε a, b
Αν a < b τότε
   c ← a + b
αλλιώς
   c ← a * b
Τέλος_αν
Εκτύπωσε c
Τέλος Παράδειγμα_3
```

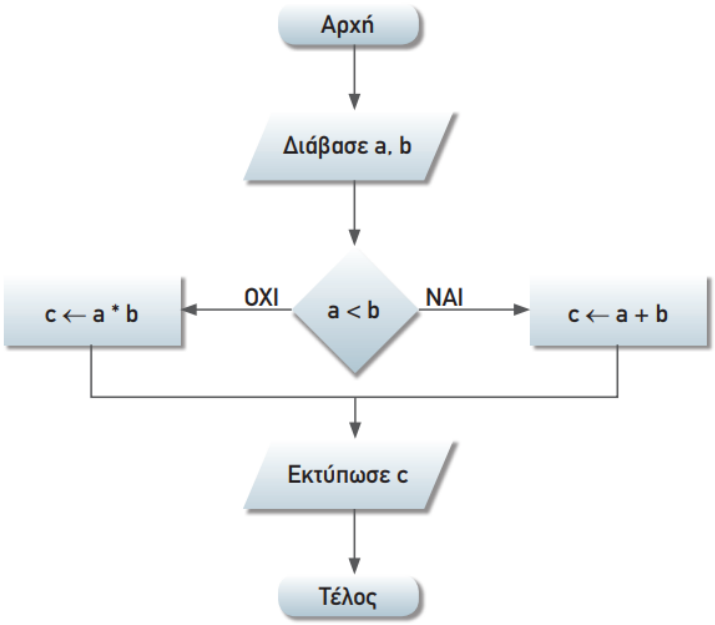
Διάγραμμα ροής

Διαβάζει δύο αριθμούς από το πληκτρολόγιο και τους εκχωρεί στις μεταβλητές a και b.

Συγκρίνει τον αριθμό a με τον b. Αν είναι μικρότερος τότε εκτελείται το ΝΑΙ το οποίο εκχωρεί στη μεταβλητή c το άθροισμα a+b.

Στην περίπτωση που το a >= b τότε εκτελείται το ΟΧΙ το οποίο εκχωρεί στη μεταβλητή c το γινόμενο a*b.

Η εντολή Εκτύπωσε c εκτελείται και στις δύο περιπτώσεις.



2.4.3 Διαδικασίες πολλαπλών επιλογών

Οι διαδικασίες των πολλαπλών επιλογών εφαρμόζονται στα προβλήματα όπου μπορεί να ληφθούν διαφορετικές αποφάσεις ανάλογα με την τιμή που παίρνει μία έκφραση.

Πολλαπλή επιλογή

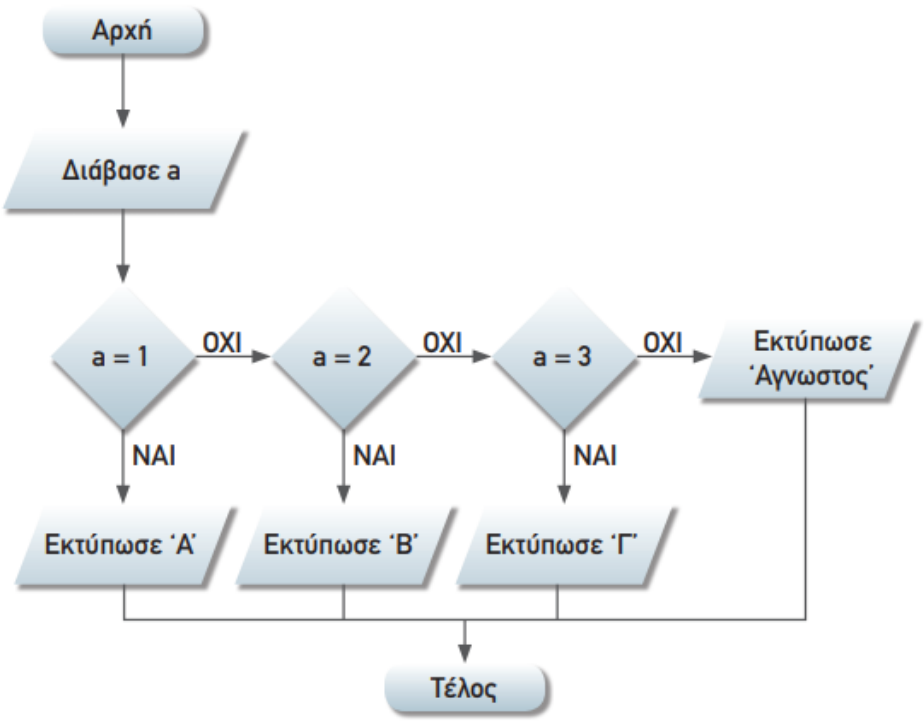
```
Αν συνθήκη τότε
     εντολή ή εντολές
αλλιώς_αν
    εντολή ή εντολές
αλλιώς_αν
    εντολή ή εντολές
……
αλλιώς
    εντολή ή εντολές
Τέλος_αν
```

```
Αλγόριθμος Παράδειγμα_4
Διάβασε a
Αν a = 1 τότε εκτύπωσε "Α"
αλλιώς_αν a = 2 τότε εκτύπωσε "Β"
αλλιώς_αν a = 3 τότε εκτύπωσε "Γ"
αλλιώς εκτύπωσε "άγνωστος"
Τέλος_αν
Τέλος Παράδειγμα_4
```

Στην πολλαπλή επιλογή έχουμε 2 ή περισσότερες συνθήκες.

Αν ισχύει η πρώτη συνθήκη (δηλ. αν είναι αληθές το a=1), εκτελείται το εκτύπωσε "Α". Αν όχι τότε ελέγχει τη δεύτερη συνθήκη a=2. Αν ισχύει, εκτυπώνει το Β. Η διαδικασία συνεχίζει μέχρι να φτάσει στο Τέλος_αν. Σε κάθε περίπτωση θα εκτελεστεί μόνο μία εντολή.

Διάγραμμα ροής



Πολλαπλή επιλογή Επίλεξε… Τέλος_επιλογών

```
Αλγόριθμος Παράδειγμα 5.
Εμφάνισε "Σε ποια ηλικία άρχισες να μαθαίνεις
προγραμματισμό;"
Διάβασε age
Επίλεξε age
   Περίπτωση < 0
      Εμφάνισε "Είπαμε ηλικία ..."
   Περίπτωση < 5
      Εμφάνισε "Μάλλον τα παραλές !!"
   Περίπτωση < 60
      Εμφάνισε "Μπράβο"
   Περίπτωση < 100
      Εμφάνισε "Ποτέ δεν είναι αργά"
   Περίπτωση αλλιώς
      Εμφάνισε "Κάλλιο αργά παρά ποτέ"
Τέλος_επιλογών
Τέλος Παράδειγμα_5
```

Στο παράδειγμα αυτό, εξετάζεται μια **έκφραση** (εδώ είναι μια μόνο μεταβλητή, η age). Ανάλογα με την τιμή της έκφρασης εκτελούνται οι εντολές μετά την Περίπτωση που αντιστοιχεί στην τιμή της έκφρασης. Οι τιμές που συνοδεύουν κάθε Περίπτωση μπορεί να είναι μία ή περισσότερες διακριτές τιμές, περιοχή τιμών από - έως ή να υπακούουν σε μια συνθήκη (όπως στο παράδειγμα αυτό). Αν η τιμή της **έκφρασης** δεν αντιστοιχεί σε καμία Περίπτωση, τότε εκτελούνται οι εντολές που ακολουθούν την Περίπτωση αλλιώς. Μετά την εκτέλεση μιας περίπτωσης, ο αλγόριθμος συνεχίζει με την εντολή, που ακολουθεί το Τέλος_επιλογών.

2.4.4 Εμφωλευμένες Διαδικασίες

Μία πολλαπλή επιλογή, όπως η παρακάτω, μπορεί να γίνει και με μία εμφωλευμένη δομή (Ένα ή περισσότερα Αν μέσα σε ένα άλλο Αν).

Παράδειγμα

Πολλαπλή επιλογή

Χαρακτηρισμός ατόμων ανάλογα με το βάρος και το ύψος τους

```
Διάβασε βάρος, ύψος
Αν βάρος<80 και ύψος<1.70 τότε
      εκτύπωσε "ελαφρύς-κοντός"
αλλιώς_αν βάρος<80 και ύψος>=1.70 τότε
      εκτύπωσε "ελαφρύς-ψηλός"
αλλιώς_αν βάρος>=80 και ύψος<1.70 τότε
      εκτύπωσε "βαρύς-κοντός"
αλλιώς_αν βάρος>=80 και ύψος>=1.70 τότε
      εκτύπωσε "βαρύς-ψηλός"
Τέλος_αν
```

Εμφωλευμένη επιλογή

Χαρακτηρισμός ατόμων ανάλογα με το βάρος και το ύψος τους

```
Αλγόριθμος Παράδειγμα_6
Διάβασε βάρος, ύψος
Αν βάρος < 80 τότε
   Αν ύψος < 1.70 τότε
      εκτύπωσε "ελαφρύς-κοντός"
   αλλιώς
      εκτύπωσε "ελαφρύς-ψηλός"
   Τέλος_αν
αλλιώς
   Αν ύψος < 1.70 τότε
      εκτύπωσε "βαρύς-κοντός"
   αλλιώς
      εκτύπωσε "βαρύς-ψηλός"
   Τέλος_αν
Τέλος_αν
Τέλος Παράδειγμα_6
```

2.4.5 Δομή Επανάληψης

Η διαδικασία της επανάληψης είναι ιδιαίτερα συχνή, αφού πλήθος προβλημάτων μπορούν να επιλυθούν με κατάλληλες επαναληπτικές διαδικασίες. Μια επαναληπτική διαδικασία εφαρμόζεται στις περιπτώσεις όπου μία ακολουθία εντολών πρέπει να εφαρμοσθεί πολλές φορές όσο ισχύει μια συνθήκη.

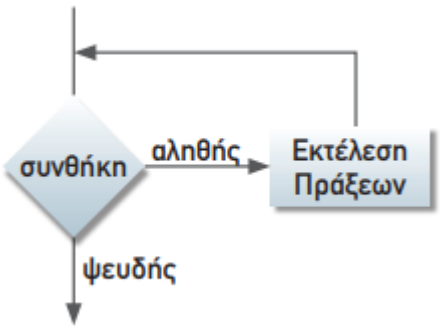
Παράδειγμα

- Το άθροισμα των εξόδων κάθε οικογένειας μιας πόλης
- Ο μέσος όρος των βαθμών κάθε μαθητή ενός σχολείου

Όσο

Επαναλαμβάνονται οι εντολές, όσο η συνθήκη είναι αληθής. Όταν η συνθήκη γίνει ψευδής, τότε ο αλγόριθμος συνεχίζεται με την εντολή που ακολουθεί το Τέλος_επανάληψης.

```
Όσο συνθήκη επανάλαβε
    εντολές
Τέλος_επανάληψης
```



Το τμήμα του αλγορίθμου που επαναλαμβάνεται, δηλαδή από την εντολή Όσο μέχρι το Τέλος_επανάληψης αποκαλείται **βρόχος.**

```
Αλγόριθμος Παράδειγμα_7
i ← 1
Όσο i ≤ 100 επανάλαβε
    Εμφάνισε i
    i ← i + 1
Τέλος_επανάληψης
Τέλος Παράδειγμα_7
```

Εκτύπωση διαδοχικών αριθμών
Το i είναι μια μεταβλητή-μετρητής που παίρνει ως αρχική τιμή το 1.
Η συνθήκη i<=100 ελέγχει την τρέχουσα τιμή του i με το 100 και όσο είναι μικρότερη ή ίση ο βρόχος επαναλαμβάνεται.
Η εντολή **i ← i + 1** εκχωρεί στο i νέα τιμή, αυτήν που είχε + 1

Στο παραπάνω παράδειγμα ξέρουμε ότι η διαδικασία θα τερματιστεί μετά από 100 επαναλήψεις.

Υπάρχουν όμως περιπτώσεις που ο τερματισμός καθορίζεται από τον χρήστη, που σημαίνει ότι δεν ξέρουμε το πλήθος των επαναλήψεων από την αρχή.

Στο παρακάτω παράδειγμα διαβάζονται και εκτυπώνονται όσοι θετικοί αριθμοί δίνονται από το πληκτρολόγιο. Ο αλγόριθμος τελειώνει, όταν δοθεί ένας αρνητικός αριθμός ή το μηδέν.

```
Αλγόριθμος Παράδειγμα_8
Διάβασε x
Όσο x > 0 επανάλαβε
    Εμφάνισε x
    Διάβασε x
Τέλος_επανάληψης
Τέλος Παράδειγμα_8
```

Η συνθήκη συνέχειας είναι στην αρχή. Αυτό σημαίνει ότι ο βρόχος επανάληψης μπορεί να μην εκτελεσθεί καμία φορά αν η αρχική τιμή της μεταβλητής x είναι αρνητική.

Το Όσο χρησιμοποιεί δύο Διάβασε, ένα έξω από την επανάληψη πριν το Όσο, και ένα μέσα σε αυτήν στο τέλος του βρόχου.

Η εντολή Όσο .... Επανάλαβε

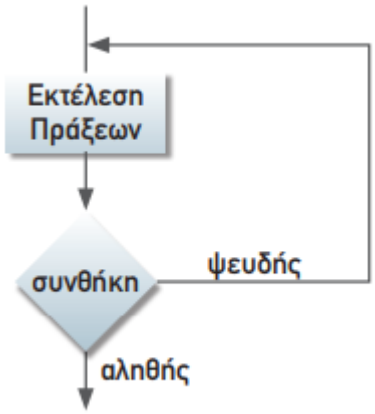
- μπορεί να μην εκτελεσθεί καμία φορά, αν η πρώτη τιμή που διαβάζεται δεν επαληθεύει τη συνθήκη συνέχειας
- τερματίζεται όταν η συνθήκη γίνει ψευδής

Στην περίπτωση που η **μεταβλητή** που διαβάζουμε βρίσκεται στη συνθήκη του **Όσο** τότε έχουμε δύο εντολές **ΔΙΑΒΑΣΕ**. Η πρώτη είναι έξω από το Όσο και η δεύτερη μέσα στο Όσο αλλά στο τέλος του βρόχου (πριν από το Τέλος_επανάληψης).

Μέχρις_ότου

Επαναλαμβάνονται οι εντολές, όσο η συνθήκη είναι ψευδής. Όταν η συνθήκη γίνει αληθής, τότε ο αλγόριθμος συνεχίζεται με την εντολή που ακολουθεί το Μέχρις_ότου.

```
Αρχή_επανάληψης
Εντολές
Μέχρις_ότου συνθήκη
```



Το παρακάτω παράδειγμα_9 λύνει το ίδιο πρόβλημα με το παράδειγμα_8 χρησιμοποιώντας την εντολή Μέχρις_ότου.

```
Αλγόριθμος Παράδειγμα_9
Αρχή_επανάληψης
    Διάβασε x
    Εμφάνισε x
Μέχρις_ότου x < 0
Τέλος Παράδειγμα_9
```

Η συνθήκη τερματισμού είναι στο τέλος. Αυτό σημαίνει ότι ο βρόχος επανάληψης θα εκτελεσθεί οπωσδήποτε τουλάχιστον μία φορά ακόμα και αν η αρχική τιμή της μεταβλητής x είναι αρνητική.

Το Μέχρις_ότου χρησιμοποιεί μόνο ένα Διάβασε μέσα στην επανάληψη, στην αρχή του βρόχου.

Αυτά τα δύο προγράμματα, 8 και 9, δίνουν το ίδιο αποτέλεσμα στην περίπτωση που η πρώτη τιμή του x είναι αρνητικός αριθμός;

Η εντολή Αρχή_επανάληψης… Μέχρις_ότου

- εκτελείται οπωσδήποτε μια φορά
- τερματίζεται όταν η συνθήκη γίνει αληθής

Για…από…μέχρι

Όταν ο αριθμός των φορών που θα εκτελεστεί μια επαναληπτική διαδικασία είναι γνωστός εκ των προτέρων, τότε είναι προτιμότερο να χρησιμοποιείται η εντολή Για…από…μέχρι.

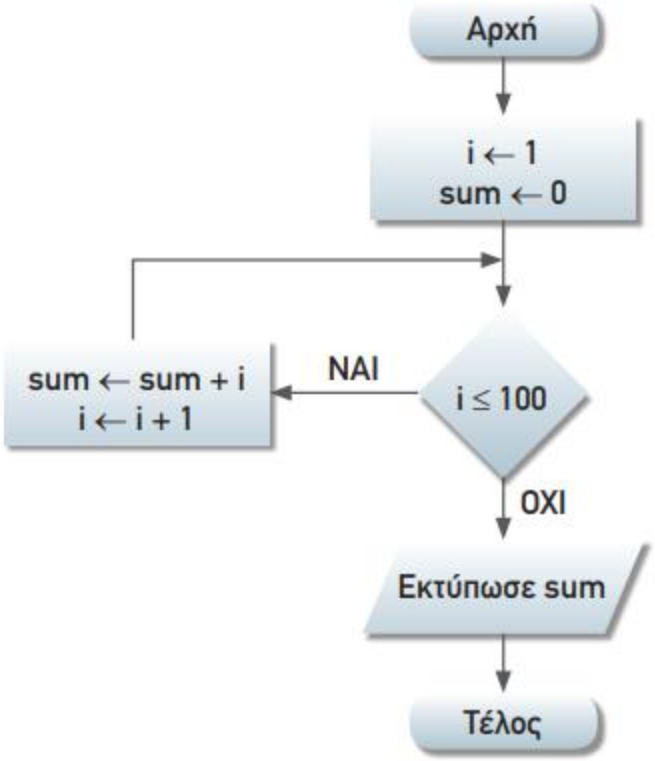
```
Για μεταβλητή από τ1 μέχρι τ2 με_βήμα β
    Διαδικασία
Τέλος_επανάληψης
```

μεταβλητή είναι ένας μετρητής που ξεκινά από μια τιμή **τ1** και αυξανόμενος ή μειούμενος κατά το βήμα **β** φτάνει στην τιμή **τ2** όπου και σταματά.

```
Αλγόριθμος Παράδειγμα_10
Sum ← 0
Για i από 1 μέχρι 100
    Sum ← Sum + i
Τέλος_επανάληψης
Εκτύπωσε Sum
Τέλος Παράδειγμα_10
```

Υπολογισμός αθροίσματος των αριθμών 1-100.

Η μεταβλητή-αθροιστής sum, στην αρχή παίρνει την τιμή 0. Η Για επαναλαμβάνεται 100 φορές, όσοι και οι αριθμοί που θα προσθέσει. Η εντολή **sum ← sum + i** κάθε φορά εκχωρεί στη sum τη νέα τιμή του i. Οι τιμές που παίρνει το i είναι οι αριθμοί 1, 2, 3, 4, …100



Παρατηρούμε ότι το διάγραμμα ροής του **Για** είναι ίδιο με αυτό του **Όσο**.

Έξω από την επανάληψη δίνουμε αρχική τιμή στον μετρητή i ίση με 1. Η συνθήκη συνέχειας **i <= 100** είναι στην αρχή της επανάληψης και όσο ισχύει επαναλαμβάνονται οι εντολές του βρόχου.

Ο μετρητής i πρέπει σε κάθε επανάληψη να αυξάνεται κατά μια τιμή (εδώ αυξάνεται κατά 1).

Παραδείγματα

Ο βρόχος

Για k **από** 5 **μέχρι** 5	εκτελείται ακριβώς μία φορά
Για k **από** 5 **μέχρι** 1	δεν εκτελείται καμία φορά
Για k **από** 0 **μέχρι** 1 **με_βήμα** 0,1	εκτελείται 11 φορές
Για k **από** 4 **μέχρι** -2 **με_βήμα** -1	εκτελείται 7 φορές
Για k **από** 14 **μέχρι** 1 **με_βήμα** -3	εκτελείται 5 φορές

Πολλαπλασιασμός αλά ρωσικά

45	19	45
90	9	90
180	4	
360	2	
720	1	720
Άθροισμα = 855		

Έστω ότι δίνονται δύο θετικοί ακέραιοι αριθμοί, οι αριθμοί 45 και 19. Οι αριθμοί γράφονται δίπλα-δίπλα και ο πρώτος διπλασιάζεται, ενώ ο δεύτερος υποδιπλασιάζεται αγνοώντας το δεκαδικό μέρος, μέχρις ότου στη δεύτερη στήλη να προκύψει μονάδα.

Τελικώς, το γινόμενο ισούται με το άθροισμα των στοιχείων της πρώτης στήλης, όπου αντίστοιχα στη δεύτερη στήλη υπάρχει περιττός αριθμός.

Η μέθοδος αυτή χρησιμοποιείται πρακτικά στους υπολογιστές, γιατί υλοποιείται πολύ πιο απλά απ' ό,τι ο γνωστός μας χειρωνακτικός τρόπος πολλαπλασιασμού. Πιο συγκεκριμένα, απαιτεί πολλαπλασιασμό επί δύο, διαίρεση διά δύο και πρόσθεση. Σε επίπεδο κυκλωμάτων υπολογιστή ο πολλαπλασιασμός επί δύο και η διαίρεση διά δύο μπορούν να υλοποιηθούν ταχύτατα με μία απλή εντολή ολίσθησης (shift).

```
Αλγόριθμος Πολλαπλασιασμός_αλά_ρωσικά
Δεδομένα // Μ1,Μ2 ακέραιοι //
Ρ ← 0
Όσο Μ2 > 0 επανάλαβε
   Αν Μ2 mod 2 = 1 τότε Ρ ← Ρ+Μ1
    Μ1 ← Μ1*2
    Μ2 ← Μ2 div 2
Τέλος_επανάληψης
Αποτελέσματα // Ρ, το γινόμενο των ακεραίων Μ1,Μ2 //
Τέλος Πολλαπλασιασμός_αλά_ρωσικά
```

Ολίσθηση

Στα κυκλώματα του υπολογιστή τα δεδομένα αποθηκεύονται με δυαδική μορφή, δηλαδή 0 και 1. Έτσι ο αριθμός 17 του δεκαδικού συστήματος ισοδυναμεί με τον αριθμό 00010001 του δυαδικού συστήματος.

Αν μετακινήσουμε τα ψηφία αυτά κατά μία θέση προς τα αριστερά, δηλαδή αν προσθέσουμε ένα 0 στο τέλος του αριθμού και αγνοήσουμε το αρχικό 0, τότε προκύπτει ο αριθμός 00100010 του δυαδικού συστήματος, που ισοδυναμεί με τον αριθμό 34 του δεκαδικού συστήματος.

00010001 **17**

00100010 **34**

Επίσης, με παρόμοιο τρόπο, αν μετακινήσουμε τα ψηφία κατά μία θέση δεξιά, δηλαδή αποκόψουμε το τελευταίο ψηφίο 1 και θεωρήσουμε ένα ακόμη αρχικό 0, τότε προκύπτει ο αριθμός 00001000 του δυαδικού συστήματος, που ισοδυναμεί με τον αριθμό 8 του δεκαδικού συστήματος.

00010001 **17**

00001000 **8**

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

Πρέπει να ξέρω

Δομές επανάληψης

1. Αν ξέρουμε το πλήθος των επαναλήψεων, τότε χρησιμοποιούμε τη **Για**
2. Αν θα γίνει τουλάχιστον μία επανάληψη, μπορούμε να χρησιμοποιήσουμε τη **Μέχρις_ότου**
3. Αν δεν ξέρουμε πόσες επαναλήψεις θα γίνουν, επιλέγουμε την **Όσο**

Εντολή Για ... από ... μέχρι

Για μεταβλητή από τ1 μέχρι τ2 [με_βήμα β]

1. Βήμα > 0 : πρέπει να ισχύει τ1 <= τ2 για να γίνει τουλάχιστον μία επανάληψη
2. Βήμα < 0 : πρέπει να ισχύει τ1 >= τ2 για να γίνει τουλάχιστον μία επανάληψη
3. Βήμα = 0 : τότε θα έχουμε άπειρες επαναλήψεις (ατέρμων βρόχος)
4. Βήμα =1 : μπορώ να παραλείψω το με_βήμα 1. Θεωρώ ότι η μεταβλητή αυξάνεται κατά 1

Εντολή Όσο επανάλαβε

1. Αν η συνθήκη της **Όσο** είναι αρχικά **Ψευδής**, τότε η Όσο δεν θα εκτελεστεί.
2. Αν η συνθήκη της **Όσο** είναι αρχικά **Αληθής**, και δεν μεταβληθεί, τότε η Όσο θα εκτελείται συνεχώς (Ατέρμονας βρόχος). Άρα, για να σταματήσει η Όσο να εκτελείται, θα πρέπει κάπου μέσα στην επανάληψη να υπάρχουν εντολές που αλλάζουν τις τιμές των μεταβλητών που υπάρχουν στη συνθήκη της Όσο (ανανέωση).

Εντολή Αρχή_επανάληψης ... μέχρις_ότου

1. Επειδή ο έλεγχος της συνθήκης γίνεται στο τέλος, οι εντολές θα εκτελεστούν τουλάχιστον 1 φορά.
2. Κάνει επανάληψη όσο η συνθήκη είναι ΨΕΥΔΗΣ, σταματάει μόλις γίνει ΑΛΗΘΗΣ.
3. Η αρχικοποίηση και η ανανέωση των μεταβλητών που συμμετέχουν στη συνθήκη, πολλές φορές ταυτίζονται.

Βασικές Χρήσεις

1. Έλεγχος ορθότητας – εγκυρότητας των τιμών που εισάγονται.
2. Δημιουργία Μενού επιλογών.
3. Ερώτηση “Θες συνέχεια ΝΑΙ / ΟΧΙ”

Μετατροπή από ΓΙΑ... σε ΟΣΟ... ΕΠΑΝΑΛΑΒΕ

<table>
<tr><td colspan="2">ΓΙΑ μετ ΑΠΟ τ1 ΜΕΧΡΙ τ2 ΜΕ_ΒΗΜΑ β
εντολές
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ</td></tr>
<tr><td>Αν τ1<= τ2 τότε η μετατροπή γίνεται :

μετ ← τ1
ΟΣΟ μετ <= τ2 ΕΠΑΝΑΛΑΒΕ
εντολές
μετ ← μετ + β
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ</td><td>Αν τ1 >= τ2 τότε η μετατροπή γίνεται :

μετ ← τ1
ΟΣΟ μετ >= τ2 ΕΠΑΝΑΛΑΒΕ
εντολές
μετ ← μετ + β
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ</td></tr>
</table>

1. Πριν την εντολή ΟΣΟ εκχωρούμε στη μεταβλητή (μετ) της ΓΙΑ... την αρχική τιμή δηλ. την τ1.
2. Στη συνθήκη συγκρίνουμε την τιμή της μετ. με την τ2.
3. Πριν το ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ της ΟΣΟ... μεταβάλλουμε την τιμή της μεταβλητής κατά την τιμή
4. του βήματος. Αν δεν υπάρχει βήμα, τότε θεωρούμε ότι είναι το 1.
5. Η μετατροπή της ΟΣΟ... σε ΓΙΑ..., γίνεται μόνο στην περίπτωση που στην ΟΣΟ... είναι γνωστός ο
6. αριθμός των επαναλήψεων, σε οποιαδήποτε άλλη περίπτωση δεν μετατρέπεται η ΟΣΟ... σε ΓΙΑ....

Καλό είναι να μετατρέπουμε το Για σε Όσο, και στη συνέχεια το Όσο.. σε Μέχρις_ότου, για να μην μπερδευτούμε.

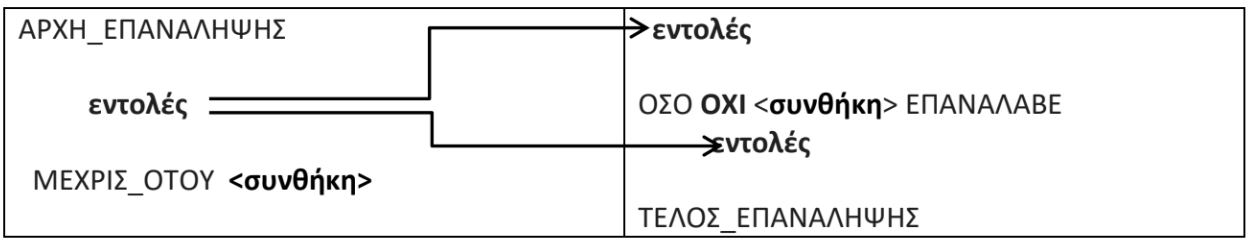
Μετατροπή ΟΣΟ... σε ΑΡΧΗ ΕΠΑΝΑΛΗΨΗΣ ... ΜΕΧΡΙΣ_ΟΤΟΥ

ΟΣΟ <**συνθήκη**> ΕΠΑΝΑΛΑΒΕ εντολές ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ	ΑΝ <**συνθήκη**> ΤΟΤΕ ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ εντολές ΜΕΧΡΙΣ_ΟΤΟΥ **ΟΧΙ** <**συνθήκη**> ΤΕΛΟΣ_ΑΝ

Πρέπει να θυμόμαστε ότι η ΟΣΟ... μπορεί να μην εκτελεστεί ούτε μία φορά, ενώ η ΜΕΧΡΙΣ_ΟΤΟΥ θα εκτελεστεί τουλάχιστον μία φορά.

Για να το διασφαλίσουμε αυτό βάζουμε την ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ μέσα σε ένα ΑΝ...ΤΟΤΕ που ελέγχει την ίδια συνθήκη με την Όσο.

Μετατροπή **ΑΡΧΗ ΕΠΑΝΑΛΗΨΗΣ** ... **ΜΕΧΡΙΣ_ΟΤΟΥ** σε **ΟΣΟ...**



Όσο ... επανάλαβε

Βασικές Χρήσεις

- Έλεγχος ορθότητας – εγκυρότητας των τιμών που εισάγονται.
- Δημιουργία Μενού επιλογών.
- Ερώτηση "Θες συνέχεια ΝΑΙ / ΟΧΙ"

Διαφορές Μεταξύ των Όσο….επανάλαβε και Αρχή_επανάληψης…

Όσο …επανάλαβε	Αρχή_επανάληψης
Ο έλεγχος της συνθήκης γίνεται στην αρχή	Ο έλεγχος της συνθήκης γίνεται στο τέλος
Η επανάληψη μπορεί να μην γίνει ούτε μία φορά	Η επανάληψη γίνεται τουλάχιστον μία φορά
Η επανάληψη σταματάει μόλις η συνθήκη γίνει ψευδής	Η επανάληψη σταματάει μόλις η συνθήκη γίνει αληθής

Εμφωλευμένες Δομές Επανάληψης (εμφωλευμένοι Βρόχοι)

Αν ένας βρόχος βρίσκεται μέσα σε άλλον, τότε έχουμε εμφωλευμένους βρόχους.

Στην περίπτωση αυτή ισχύουν οι παρακάτω κανόνες:

- Η είσοδος σε βρόχο υποχρεωτικά γίνεται από την αρχή του
- Δεν πρέπει να χρησιμοποιηθεί η ίδια μεταβλητή ως μετρητής δύο ή περισσοτέρων βρόχων που ο ένας βρίσκεται στο εσωτερικό του άλλου
- Ο εσωτερικός βρόχος πρέπει να βρίσκεται ολόκληρος μέσα στον εξωτερικό βρόχο

Για ι από 1 μέχρι 10 Για κ από 1 μέχρι 5 με_βήμα 2 **Εντολές** Τέλος_επανάληψης Τέλος_επανάληψης	Το διπλανό πρόγραμμα ικανοποιεί τους παραπάνω κανόνες. Οι **εντολές** θα εκτελεστούν 10 φορές (εξωτερικό Για) επί 3 (εσωτερικό Για) = 30 φορές.

3.1 | Δεδομένα

Η Πληροφορική θεωρείται η επιστήμη που μελετά τα δεδομένα από τις ακόλουθες σκοπιές:

Υλικού. Το υλικό, δηλαδή η μηχανή, επιτρέπει στα δεδομένα ενός προγράμματος να αποθηκεύονται στην κύρια μνήμη και στις περιφερειακές συσκευές του υπολογιστή με διάφορες αναπαραστάσεις. Τέτοιες μορφές είναι η δυαδική, ο κώδικας ASCII, ο κώδικας EBCDIC, το συμπλήρωμα του 1 ή του 2 κ.λπ.

Γλωσσών προγραμματισμού. Οι γλώσσες προγραμματισμού υψηλού επιπέδου επιτρέπουν τη χρήση διάφορων τύπων μεταβλητών για να περιγράψουν ένα δεδομένο. Ο μεταφραστής κάθε γλώσσας φροντίζει για την αποδοτικότερη μορφή αποθήκευσης, από πλευράς υλικού, κάθε μεταβλητής στον υπολογιστή.

Δομών Δεδομένων. Δομή δεδομένων είναι ένα σύνολο δεδομένων μαζί με ένα σύνολο επιτρεπτών λειτουργιών επί αυτών. Για παράδειγμα, μία τέτοια δομή είναι η εγγραφή (record), που μπορεί να περιγράφει ένα είδος, ένα πρόσωπο κ.λπ. Η εγγραφή αποτελείται από τα πεδία (fields) που αποθηκεύουν χαρακτηριστικά διαφορετικού τύπου, όπως για παράδειγμα ο κωδικός, η περιγραφή κ.λπ. Άλλη μορφή δομής δεδομένων είναι το αρχείο που αποτελείται από ένα σύνολο εγγραφών. Μία επιτρεπτή λειτουργία σε ένα αρχείο είναι η σειριακή προσπέλαση όλων των εγγραφών του.

Ανάλυσης Δεδομένων. Τρόποι καταγραφής και αλληλοσυσχέτισης των δεδομένων μελετώνται έτσι ώστε να αναπαρασταθεί η γνώση για πραγματικά γεγονότα. Οι τεχνολογίες των Βάσεων Δεδομένων, της Μοντελοποίησης Δεδομένων και της Αναπαράστασης Γνώσης ανήκουν σε αυτή τη σκοπιά μελέτης των δεδομένων.

3.2 | Αλγόριθμοι + Δομές Δεδομένων = Προγράμματα

Τα δεδομένα ενός προβλήματος αποθηκεύονται στον υπολογιστή, είτε στην κύρια μνήμη του είτε στη δευτερεύουσα μνήμη του. Η αποθήκευση αυτή δεν γίνεται κατά έναν τυχαίο τρόπο αλλά συστηματικά, δηλαδή χρησιμοποιώντας μία δομή.

ΟΡΙΣΜΟΣ

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

Κάθε μορφή δομής δεδομένων αποτελείται από ένα σύνολο κόμβων (nodes). Οι βασικές λειτουργίες (ή αλλιώς πράξεις) επί των δομών δεδομένων είναι οι ακόλουθες:

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 προσπελαύνονται 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.

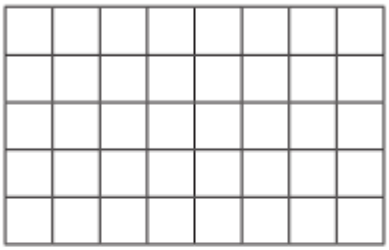
Οι δομές δεδομένων διακρίνονται σε δύο μεγάλες κατηγορίες:

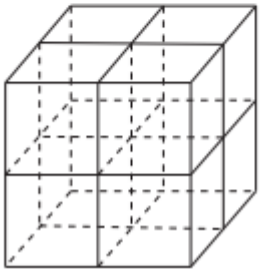
στατικές (static)	δυναμικές (dynamic)
τα στοιχεία των στατικών δομών αποθηκεύονται σε συνεχόμενες θέσεις μνήμης	δεν αποθηκεύονται σε συνεχόμενες θέσεις μνήμης αλλά στηρίζονται στην τεχνική της λεγόμενης δυναμικής παραχώρησης μνήμης
το ακριβές μέγεθος της απαιτούμενης κύριας μνήμης καθορίζεται κατά τη στιγμή του προγραμματισμού τους, και κατά συνέπεια κατά τη στιγμή της μετάφρασής τους και όχι κατά τη στιγμή της εκτέλεσης τους προγράμματος	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.
υλοποιούνται με πίνακες	υλοποιούνται με αρχεία

3.3 | Πίνακες

Μπορούμε να ορίσουμε τον πίνακα ως μια δομή που περιέχει στοιχεία του ίδιου τύπου (δηλαδή ακέραιους, πραγματικούς κ.λπ). Η δήλωση των στοιχείων ενός πίνακα γίνεται με τη χρήση του συμβολικού ονόματος του πίνακα ακολουθούμενου από την τιμή ενός ή περισσότερων δεικτών (indexes) σε αγκύλη.

Για παράδειγμα βαθμός[I], θερμοκρασία[I, j]





Ένας πίνακας μπορεί να είναι

- μονοδιάστατος
- δισδιάστατος
- τρισδιάστατος
- ν-διάστατος

Όσον αφορά στους δισδιάστατους πίνακες, αν το μέγεθος των δύο διαστάσεων είναι ίσο, τότε ο πίνακας λέγεται **τετραγωνικός** και γενικά συμβολίζεται ως πίνακας **n x n**.

Μάλιστα μπορούμε να θεωρήσουμε το δισδιάστατο πίνακα ότι είναι ένας μονοδιάστατος πίνακας, όπου κάθε θέση του περιέχει ένα νέο μονοδιάστατο πίνακα.

Ο μονοδιάστατος πίνακας μαθ[24] μπορεί να περιέχει τα ονόματα των μαθητών μιάς τάξης.

Πρέπει να ξέρω

Η αναφορά στα στοιχεία ενός πίνακα

γίνεται με τη χρήση του ονόματος του πίνακα ακολουθούμενου από την τιμή ενός ή περισσοτέρων δεικτών σε παρένθεση ή αγκύλη.

Η δήλωση ενός πίνακα γίνεται ως εξής:

Τύπος_δεδομένων: όνομα_πίνακα[N]

Η δήλωση του πίνακα Μαθητής που περιέχει τα ονόματα 100 μαθητών είναι:

Χαρακτήρες: Μαθητής[100]

Ο δείκτης ενός πίνακα μπορεί να είναι

- μια σταθερά
- μια μεταβλητή
- μια αριθμητική έκφραση με ακέραιο πάντα αποτέλεσμα

Ο πίνακας Α[10]

είναι ένας γραμμικός ή μονοδιάστατος πίνακας με 10 στοιχεία.

Ο πίνακας Β[5, 7]

είναι ένας ορθογώνιος ή δισδιάστατος πίνακας που αποτελείται από 5 γραμμές και 7 στήλες δηλ. συνολικά 35 στοιχεία.

Με τη χρήση των πινάκων

μειώνεται το πλήθος των μεταβλητών που χρειάζεται να ορίσουμε σε ένα αλγόριθμο.

Όμως η χρήση πίνακα

δικαιολογείται μόνο στην περίπτωση όπου τα δεδομένα που έχουμε εισάγει θέλουμε να τα έχουμε διαθέσιμα σε διαφορετικά σημεία του προγράμματος οπότε το μόνο που μας μένει είναι να τα αποθηκεύσουμε σε έναν πίνακα.

Οι εργασίες στους πίνακες

(διάβασμα, εμφάνιση, επεξεργασία στοιχείων κλπ) γίνονται με μία δομή επανάληψης (στους μονοδιάστατους) ή δύο εμφωλευμένες στους πίνακες δύο διαστάσεων.

Σειριακή Αναζήτηση

Είναι η πιο απλή μέθοδος αναζήτησης η οποία ελέγχει όλα τα στοιχεία του πίνακα ξεκινώντας από το πρώτο και συγκρίνοντάς τα με την τιμή που αναζητούμε. Η αναζήτηση μπορεί να είναι επιτυχής, αν βρούμε αυτό που ζητάμε, ή ανεπιτυχής, αν εξαντληθούν όλα τα στοιχεία του πίνακα χωρίς να βρούμε το στοιχείο που ζητάμε.

Στοίβα

Η στοίβα (stack) είναι μία **στατική δομή δεδομένων** τα στοιχεία της οποίας τοποθετούνται με τέτοιο τρόπο, ώστε να παίρνουμε πρώτα αυτά που βρίσκονται στο πάνω μέρος (κορυφή) της στοίβας και τελευταία αυτά που βρίσκονται στο κάτω μέρος.
Η παραπάνω μέθοδος ονομάζεται **Τελευταίο Μέσα, Πρώτο Έξω ή LIFO (Last In First Out).**

Χαρακτηριστικό παράδειγμα στοίβας, είναι μία στοίβα από πιάτα. Βάζουμε ένα νέο πιάτο πάντα στην κορυφή της στοίβας ενώ, παίρνουμε (βγάζουμε) το πιάτο που βάλαμε τελευταίο.

Οι **κύριες λειτουργίες** σε μία στοίβα είναι δύο:

Η ώθηση (push) στοιχείου στην κορυφή της στοίβας δηλ. το να βάλουμε ένα στοιχείο σε μία στοίβα.
Στη διαδικασία της ώθησης πρώτα ελέγχουμε αν η στοίβα είναι γεμάτη.
Αν προσπαθήσουμε να «βάλουμε» ένα στοιχείο σε μία ήδη γεμάτη στοίβα, τότε έχουμε **υπερχείλιση (overflow)** της στοίβας, πράγμα το οποίο αποφεύγουμε.

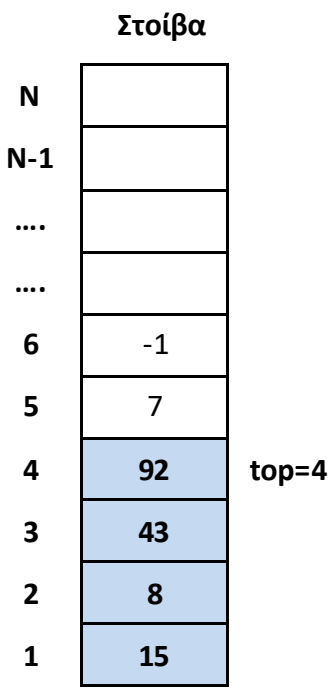
Η απώθηση (pop) στοιχείου από τη στοίβα, δηλ. το να βγάλουμε ένα στοιχείο από τη στοίβα. Στη διαδικασία της απώθησης πρώτα ελέγχουμε αν υπάρχει ένα τουλάχιστον στοιχείο στη στοίβα. Αν προσπαθήσουμε να «αφαιρέσουμε» ένα στοιχείο από μία άδεια στοίβα, έχουμε **υποχείλιση (underflow)** της στοίβας.

Η υλοποίηση της στοίβας γίνεται με χρήση μονοδιάστατου πίνακα.
Χρησιμοποιούμε μία βοηθητική μεταβλητή - **δείκτη** - (**top**), που δείχνει τη θέση του στοιχείου που τοποθετήθηκε τελευταίο στην **κορυφή** της στοίβας.
Κατά την **ώθηση** ενός νέου στοιχείου στη στοίβα (εισαγωγή στοιχείου στον πίνακα)

- ελέγχουμε πρώτα αν η στοίβα είναι γεμάτη, δηλ. αν top < N
- ο δείκτης **top** αυξάνεται κατά ένα: **top←top+1**
- γίνεται η ώθηση του στοιχείου στη νέα θέση που δείχνει ο δείκτης **top**

Κατά την **απώθηση** ενός στοιχείου από τη στοίβα (εξαγωγή στοιχείου)

- ελέγχουμε πρώτα αν υπάρχει τουλάχιστον ένα στοιχείο στη στοίβα, δηλ. **top>=1**
- εξάγεται το στοιχείο που δείχνει ο δείκτης **top**
- ο δείκτης **top** μειώνεται κατά ένα: **top←top-1**



ΠΡΟΣΟΧΗ!

- Σε μία κενή **στοίβα** θεωρούμε ότι η αρχική τιμή του δείκτη **top** είναι **μηδέν** (**top=0**).
- Ο δείκτης **top** είναι ουσιαστικά μία μεταβλητή που δείχνει τη θέση που τοποθετήθηκε το τελευταίο στοιχείο στη στοίβα/πίνακα. Επίσης μας δείχνει και το **πλήθος των ενεργών στοιχείων** σε μία στοίβα.
- Στην **απώθηση δεν διαγράφεται το στοιχείο**, ουσιαστικά δεν γίνεται καμία παρέμβαση (αλλαγή) στα στοιχεία του πίνακα. Απλώς ο δείκτης **top** δείχνει στην προηγούμενη θέση. **Στη στοίβα της παραπάνω εικόνας θεωρούμε ότι υπάρχουν 4 στοιχεία.**

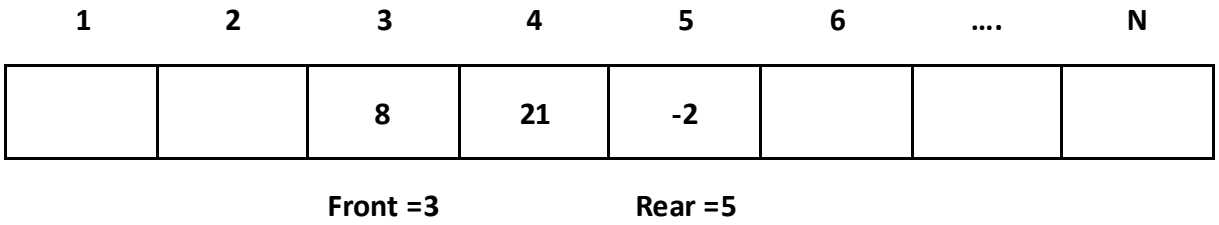
Ώθηση	**Απώθηση**
Διάβασε x Αν **top < N** τότε top ← top + 1 stack [top] ← x done ← Αληθής αλλιώς done ← Ψευδής γράψε ‘ ΥΠΕΡΧΕΙΛΙΣΗ!!’ Τέλος_αν	Αν **top>=1** τότε x ← stack[top] top ← top-1 done ← Αληθής αλλιώς done ← Ψευδής γράψε ‘ΥΠΟΧΕΙΛΙΣΗ!!!!’ Τέλος_αν

Ουρά

Ουρά (Queue), ονομάζεται μία δομή δεδομένων το σύνολο των στοιχείων της οποίας είναι διατεταγμένο με τέτοιο τρόπο, ώστε τα στοιχεία που τοποθετήθηκαν πρώτα στην ουρά να λαμβάνονται επίσης πρώτα.
Η παραπάνω μέθοδος ονομάζεται **Πρώτο Μέσα, Πρώτο Έξω ή FIFO (=First In First Out).**

Ο πελάτης μιας τράπεζας που μπαίνει πρώτος σε μία ουρά για να εξυπηρετηθεί, είναι αυτός που εξυπηρετείται και πρώτος, με την έξοδό του από την ουρά αναμονής.
Άλλα παραδείγματα είναι η ουρά στα λεωφορεία ή η ουρά στα ταμεία, όπου ο πρώτος που στέκεται στην ουρά εξυπηρετείται και πρώτος.

Υλοποίηση ουράς με χρήση μονοδιάστατου πίνακα



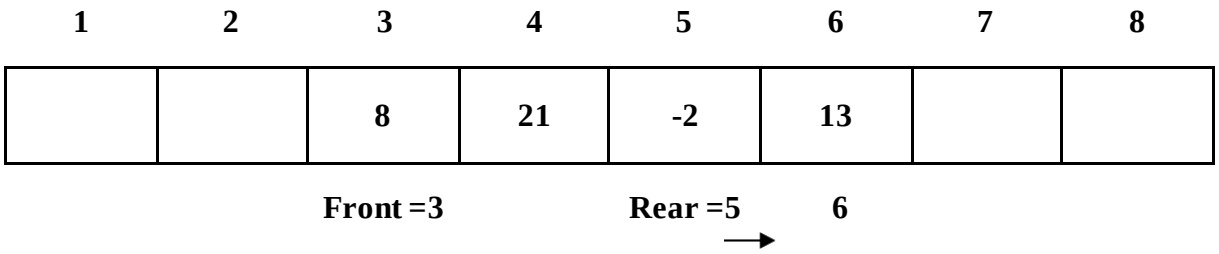
Χρησιμοποιούμε δύο **μεταβλητές - δείκτες**, τον **front** που δείχνει τη θέση του 1ου στοιχείου της ουράς ή καλύτερα του στοιχείου που είναι έτοιμο να βγει από την ουρά και τον **rear** που δείχνει τη θέση του στοιχείου που μπήκε τελευταίο στην ουρά.
Ως αρχικές τιμές των δεικτών **rear και front θεωρούμε το μηδέν**.

Βασικές λειτουργίες σε μία ουρά

Εισαγωγή

Η **εισαγωγή** ενός νέου στοιχείου γίνεται από το **πίσω άκρο** της ουράς

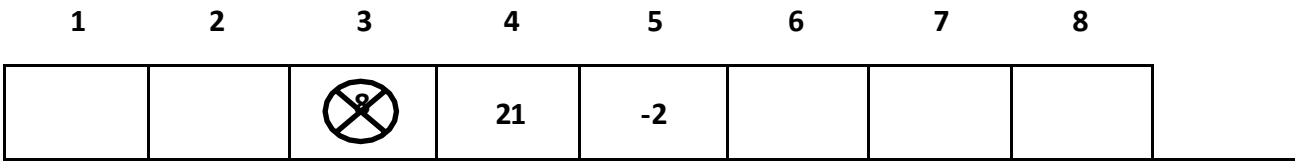
- ελέγχουμε πρώτα αν η ουρά είναι γεμάτη, δηλ. αν rear = N
- πρώτα αυξάνουμε τον δείκτη **rear** κατά ένα ως εξής: **rear ← rear + 1** (rear = 6)
- μετά εισάγουμε το στοιχείο (13) στον πίνακα, στη θέση (6) που δείχνει ο rear



Εξαγωγή

Η **εξαγωγή** ενός στοιχείου γίνεται από το **εμπρός άκρο της ουράς**

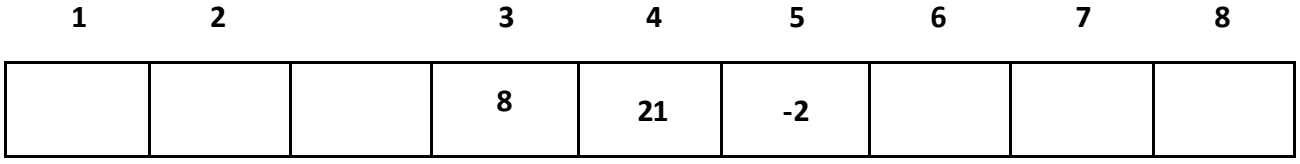
- ελέγχουμε πρώτα αν υπάρχει τουλάχιστον ένα στοιχείο στην ουρά, όχι(front =0 και rear = 0)
- πρώτα παίρνουμε το στοιχείο (8) που δείχνει ο δείκτης front (3)
- μετά η τιμή του δείκτη front αυξάνεται κατά ένα ως εξής: **front ← front +1** (front = 4)



Front=4 Rear =5

Εισαγωγή_σε_Ουρά	**Εξαγωγή_από_Ουρά**
Διάβασε x Αν **rear = N** τότε ΓΡΑΨΕ 'Γεμάτη ουρά' done ← Ψευδής αλλιώς_αν **front = 0 ΚΑΙ rear = 0** τότε front← 1 rear←1 queue[rear] ← x done ← Αληθής αλλιώς rear ← rear+1 queue[rear] ← x done ← Αληθής Τέλος_αν	Αν **front =0 και rear** = **0** τότε ΓΡΑΨΕ 'Άδεια ουρά' done ← Ψευδής αλλιώς_αν **front = rear** τότε x ← queue[front] done ← Αληθής front← 0 rear← 0 αλλιώς x ← queue[front] front ← front+1 done ← Αληθής Τέλος_αν

Κατά την εξαγωγή ενός στοιχείου, ο οποίος στη συνέχεια αυξάνεται κατά ένα (δείχνει στην επόμενη θέση του πίνακα) **χωρίς** στην πραγματικότητα να γίνεται καμία παρέμβαση στα περιεχόμενα του πίνακα (δηλ. **χωρίς να διαγράφεται κάποιο στοιχείο**).



Front=4 Rear =5

ΠΡΟΣΟΧΗ!

- Αν front = rear (και front <>0), τότε η ουρά έχει μόνο ένα στοιχείο.
- Αν front > rear, τότε η ουρά άδειασε (απλή ουρά).
- Πλήθος στοιχείων σε ουρά: rear - front +1 (ΔΕΝ ισχύει για ΑΔΕΙΑ ΟΥΡΑ- front=0, rear =0).
- Πάντα οι αρχικές τιμές για front, rear είναι μηδέν (ξεκινάω με άδεια ουρά).
- Μπορούμε να ολισθήσουμε τα στοιχεία μιας ουράς κατά μία θέση προς τα

αριστερά, όταν γίνεται εξαγωγή στοιχείου ή όταν η ουρά έχει front >1 και rear = N (δηλ. ενώ φαίνεται ότι έχει γεμίσει, ενώ υπάρχει χώρος ακόμα).

Ο κώδικας για αυτό είναι:

Κάθε φορά, μετά από εξαγωγή	**Συνολική μεταφορά όλων στην αρχή**
Για i από front μέχρι rear A[i-1] ← A[i] Τέλος_επανάληψης Front ←1 A[rear]←"*" rear←rear -1	K←1 Για i από front μέχρι rear A[K]← A[i] A[i]←"*" K← K+1 Τέλος_επανάληψης Front ←1 rear←K - 1

Λίστες

Η έννοια της λίστας συναντάται αρκετά συχνά στην καθημερινότητά μας.

Έτσι έχουμε λίστες με

- τα αγαπημένα μας τραγούδια
- τις επαφές μας
- τα ψώνια που θέλουμε να κάνουμε

Η λίστα δεν είναι τίποτα άλλο παρά μία συλλογή από αντικείμενα του ίδιου τύπου.

Μπορούμε να έχουμε δηλαδή λίστες από λέξεις, από ονόματα αλλά και από αριθμούς.

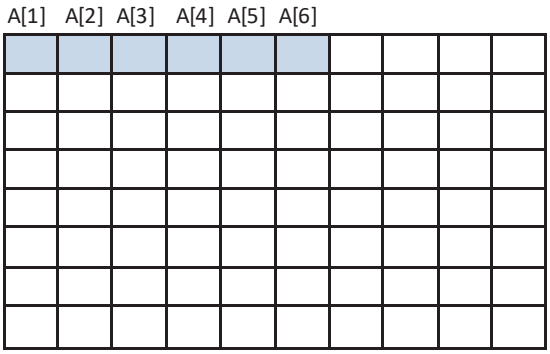
Ας υποθέσουμε ότι πρέπει να αποθηκεύσετε σε μια δομή δεδομένων τα μέρη που θέλετε να επισκεφτείτε. Πολύ απλά και εύκολα μπορείτε να χρησιμοποιήσετε έναν πίνακα.

Το γεγονός, όμως, ότι το **μέγεθος ενός πίνακα παραμένει σταθερό** (στατική δομή δεδομένων), **δυσχεραίνει την προσθήκη και τη διαγραφή στοιχείων**. Θεωρείστε την περίπτωση που θέλετε να προσθέσετε στον πίνακα και άλλα μέρη στα οποία θέλετε να ταξιδέψετε. Τι θα κάνετε αν ο πίνακας είναι γεμάτος;

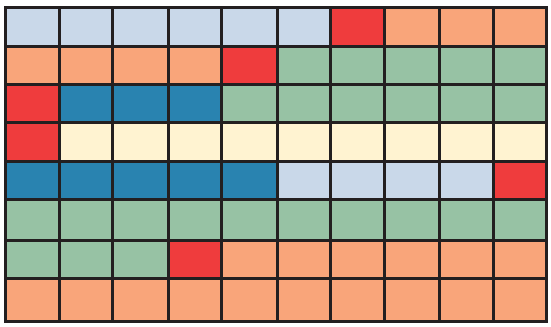
Μια πρώτη απάντηση θα ήταν να δημιουργήσουμε έναν νέο μεγαλύτερο πίνακα και να αντιγράψουμε σε αυτόν τα στοιχεία του προηγούμενου πίνακα. Έτσι, θα είχαμε μεγαλύτερο χώρο για να προσθέσουμε και νέα στοιχεία. Αυτή όμως δεν είναι η ιδανική λύση, όταν η προσθήκη και η διαγραφή στοιχείων αποτελεί συχνό φαινόμενο.

Ας θεωρήσουμε το πλέγμα των κελιών που παρουσιάζεται στις παρακάτω εικόνες ως τη μνήμη του υπολογιστή. Στην **Εικόνα α** παρουσιάζεται ο **χώρος που καταλαμβάνει ο πίνακας Α** με το γαλάζιο χρώμα. Ο πίνακας αυτός αποτελείται από έξι στοιχεία τα οποία είναι αποθηκευμένα σε **συνεχόμενες θέσεις μνήμης**. Συνήθως όμως, η μνήμη του υπολογιστή δεν είναι τόσο καθαρή και τακτοποιημένη. Στην Εικόνα β φαίνονται τα **διάσπαρτα (μη συνεχόμενα) ελεύθερα κελιά** με το

κόκκινο χρώμα. Στην προκειμένη περίπτωση, **δεν μπορείτε να «τοποθετήσετε» έναν πίνακα** στη μνήμη, **διότι δεν υπάρχουν συνεχόμενες ελεύθερες θέσεις**. Μπορείτε, όμως, να χρησιμοποιήσετε τις διάσπαρτες αυτές ελεύθερες θέσεις για να αποθηκεύσετε τα δεδομένα σας, αν θεωρήσετε ότι αποτελούν «στοιχεία» μίας **συνδεδεμένης λίστας** (δυναμική δομή δεδομένων).



Εικόνα α: Ο πίνακας Α στη μνήμη



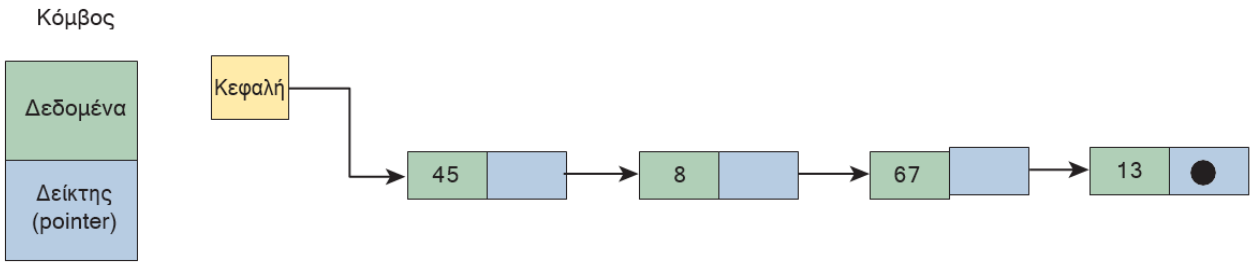
β: Με το χρώμα κόκκινο αναπαριστάνουμε τα ελεύθερα κελιά

Η **συνδεδεμένη λίστα** αποτελείται από μία σειρά από κόμβους, που συνήθως βρίσκονται σε απομακρυσμένες θέσεις μνήμης. Κάθε κόμβος αποτελείται από δύο κύρια τμήματα.

Το **πρώτο τμήμα** περιέχει τα **δεδομένα** και το **δεύτερο** τη **διεύθυνση** του επόμενου κόμβου με τον οποίο συνδέεται δηλ. τον **δείκτη** (pointer) που δείχνει στον επόμενο κόμβο.

Το πεδίο **Δεδομένα** μπορεί να περιέχει μία ή περισσότερες **αλφαριθμητικές ή αριθμητικές** πληροφορίες.

Ο **δείκτης** (pointer) είναι ένας ιδιαίτερος τύπος δεδομένων που **δε 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.



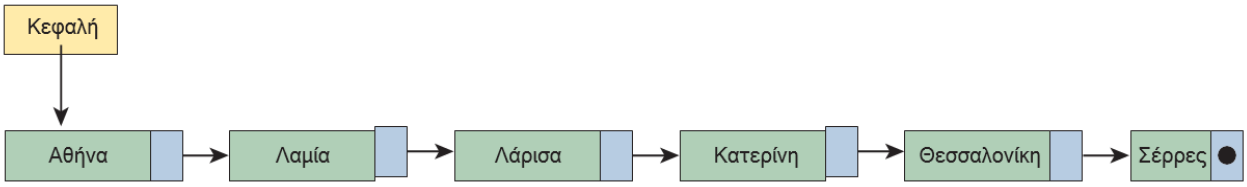
Η παραπάνω λίστα έχει τέσσερις κόμβους, που περιέχουν ακεραίους. Ο **δείκτης του τελευταίου κόμβου** της λίστας **έχει ως τιμή το NULL** (κενό) και τον αναπαριστούμε συμβολικά με την τελεία. Οι **δείκτες των υπολοίπων κόμβων** περιέχουν τη διεύθυνση του επόμενου κόμβου αντίστοιχα και τους **απεικονίζουμε συμβολικά με ένα βέλος** για να υποδηλώσουμε τη σύνδεση μεταξύ του προηγούμενου και του επόμενου κόμβου. Κάθε λίστα συνοδεύεται από έναν δείκτη με το όνομα «**Κεφαλή**» (head), που **δείχνει στον πρώτο κόμβο της λίστας**, δηλαδή περιέχει τη διεύθυνση του πρώτου κόμβου της λίστας.

Οι κόμβοι μιας λίστας δεν έχουν ονόματα. Επομένως, αν θελήσουμε να **έχουμε πρόσβαση στον τέταρτο κόμβο μιας λίστας**, για να επεξεργαστούμε τα δεδομένα που περιέχει, θα πρέπει να **ξεκινήσουμε από τον πρώτο κόμβο της λίστας**, η διεύθυνση του οποίου περιέχεται στον δείκτη

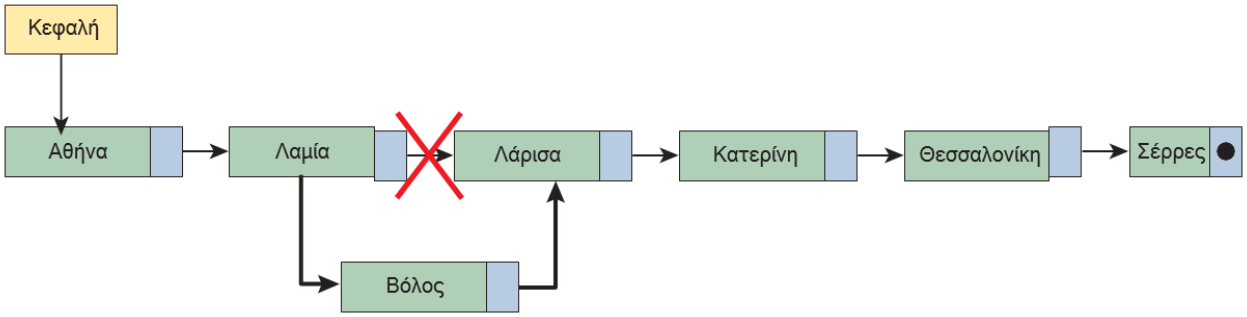
Κεφαλή. Ξεκινώντας από τον πρώτο κόμβο της λίστας μπορούμε να έχουμε πρόσβαση στη διεύθυνση μνήμης του δεύτερου κόμβου. Από τον δεύτερο κόμβο μπορούμε να έχουμε πρόσβαση στη διεύθυνση μνήμης του τρίτου κόμβου. Και συνεχίζουμε έτσι μέχρι να φτάσουμε στον τέταρτο κόμβο. Αυτός είναι ο μόνος τρόπος για να διασχίσουμε μία συνδεδεμένη λίστα.

Παράδειγμα 1

Έχουμε αποθηκεύσει στην παρακάτω συνδεδεμένη λίστα τις πόλεις που είναι πιθανό να επισκεφθούμε σε ένα ταξίδι μας.



Λίγο πριν το ταξίδι, αποφασίζουμε να επισκεφθούμε τον Βόλο, αμέσως μετά τη Λαμία και πριν φτάσουμε στη Λάρισα. Επομένως, χρειάζεται να εισαγάγουμε ένα νέο κόμβο στη λίστα μας, κάνοντας την κατάλληλη διευθέτηση των δεικτών.

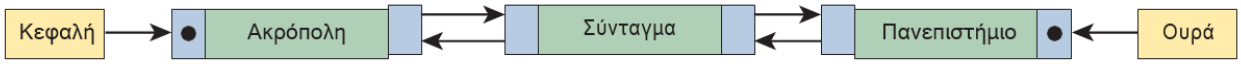


Απλά συνδεδεμένη λίστα (μπορούμε να κινηθούμε προς μία μόνο κατεύθυνση).

Οι αθλητές που λαμβάνουν μέρος σε έναν αγώνα σκυταλοδρομίας είναι τοποθετημένοι με μια συγκεκριμένη σειρά και ο κάθε αθλητής για να ξεκινήσει να τρέχει πρέπει να περιμένει τον προηγούμενο του.

Διπλά συνδεδεμένη λίστα (μπορούμε να τη διατρέξουμε και προς τις δύο κατευθύνσεις).

Κάθε σταθμός του μετρό συνδέεται με τον αμέσως επόμενο και τον αμέσως προηγούμενο κόμβο της λίστας.



Διαφορές Λίστας σε σχέση με τον Πίνακα

- Ο **πίνακας** θεωρείται μια **δομή τυχαίας προσπέλασης**, σε αντίθεση με μια **λίστα** που είναι στην ουσία μια **δομή ακολουθιακής ή σειριακής προσπέλασης**. Για να φθάσουμε, δηλαδή, σ' έναν κόμβο μιας λίστας πρέπει να περάσουμε από όλους τους προηγούμενους ξεκινώντας από τον πρώτο.
- **Ο πίνακας έχει σταθερό μέγεθος**, το οποίο δηλώνεται εξαρχής κατά την υλοποίηση. Αυτό γίνεται, διότι ο πίνακας είναι στατική δομή δεδομένων σε αντίθεση με τη **λίστα που είναι δυναμική δομή και το μέγεθός της μπορεί να μεταβάλλεται** καθώς εισέρχονται νέοι κόμβοι στη λίστα ή διαγράφονται κάποιοι άλλοι.
- Οι **κόμβοι της λίστας αποθηκεύονται σε μη συνεχόμενες θέσεις μνήμης** σε αντιδιαστολή με τους **πίνακες**, όπου τα στοιχεία αποθηκεύονται σε **συνεχόμενες θέσεις μνήμης**.

Πλεονεκτήματα των λιστών (έναντι των πινάκων)

- Το δυναμικό τους μέγεθος
- η ευκολία εισαγωγής και διαγραφής από οποιοδήποτε μέρος της λίστας
- η μη αναγκαιότητα δήλωσης του μεγέθους τους

Μειονεκτήματα των λιστών (έναντι των πινάκων)

- **Η τυχαία πρόσβαση στη λίστα δεν επιτρέπεται**. Είναι αδύνατο να φτάσετε στον n-οστό κόμβο μιας απλά συνδεδεμένης λίστας χωρίς πρώτα να περάσετε από όλους τους κόμβους διαδοχικά μέχρι τον συγκεκριμένο κόμβο ξεκινώντας από τον πρώτο κόμβο. Εναλλακτικά, στην περίπτωση της διπλά συνδεμένης λίστας μπορείτε να ξεκινήσετε και από τον τελευταίο κόμβο. Επομένως, **δεν μπορούμε να πραγματοποιήσουμε με αποτελεσματικό τρόπο δυαδική αναζήτηση** σε συνδεδεμένες λίστες.

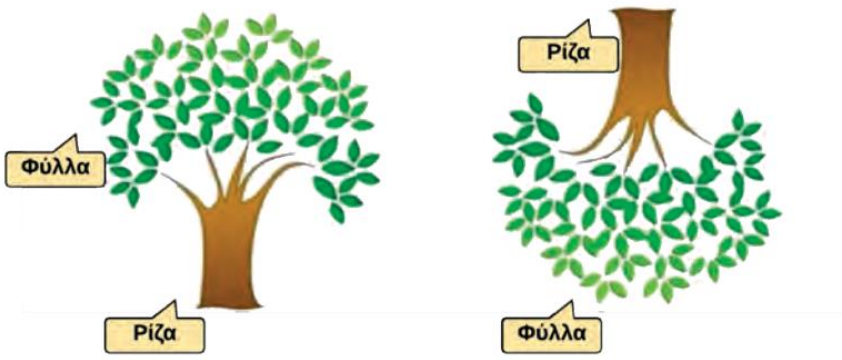
- Οι συνδεδεμένες λίστες έχουν πολύ **μεγαλύτερη επιβάρυνση από τους πίνακες**, αφού οι **συνδεδεμένοι κόμβοι της λίστας είναι δυναμικά κατανεμημένοι** (οι οποίοι είναι λιγότερο αποτελεσματικοί στη χρήση της μνήμης) και **κάθε κόμβος στη λίστα πρέπει, επιπλέον, να αποθηκεύσει έναν πρόσθετο δείκτη** που θα δείχνει στον επόμενο κόμβο. Στην περίπτωση των διπλά συνδεδεμένων λιστών χρειαζόμαστε επιπλέον έναν δεύτερο δείκτη που θα δείχνει στον προηγούμενο κόμβο.

Βασικές πράξεις των συνδεδεμένων λιστών

- Εισαγωγή κόμβου στη λίστα (εισαγωγή κόμβου στην αρχή, στο τέλος της λίστας ή ενδιάμεσα).
- Διαγραφή κόμβου από τη λίστα (διαγραφή από την αρχή, το τέλος της λίστας ή ενδιάμεσα).
- Έλεγχος για το αν η λίστα είναι κενή.
- Αναζήτηση κόμβου για την εύρεση συγκεκριμένου στοιχείου.
- Διάσχιση της λίστας και προσπέλαση των στοιχείων της (π.χ. εκτύπωση των δεδομένων που περιέχονται σε όλους τους κόμβους της λίστας).

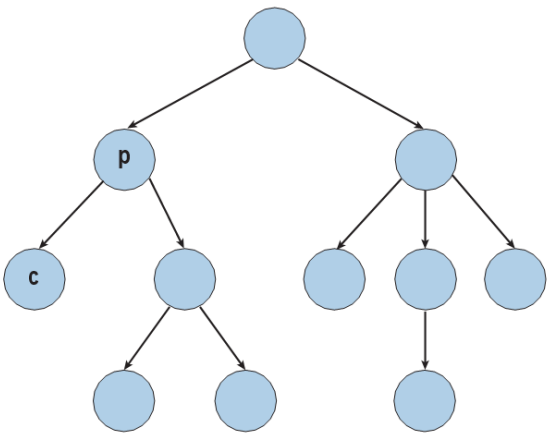
Δένδρα

Μερικές φορές τα πράγματα δεν είναι γραμμικά, όπως έχουμε δει μέχρι τώρα. Σε μία γραμμική δομή, μετά από κάθε στοιχείο ακολουθεί ένα άλλο στοιχείο εκτός και αν είναι το τελευταίο. Τι συμβαίνει όμως στις περιπτώσεις όπου μετά από ένα στοιχείο ακολουθεί όχι ένα, αλλά δύο, τρία ή και περισσότερα στοιχεία;



Τα δένδρα στη φύση και στην Πληροφορική

Ένα δένδρο αποτελείται από κόμβους, οι οποίοι συνδέονται μεταξύ τους με ακμές.



Γονέα (p) ονομάζουμε τον κόμβο από τον οποίο ξεκινάει η ακμή και **παιδί** (c) τον κόμβο στον οποίο καταλήγει η ακμή.

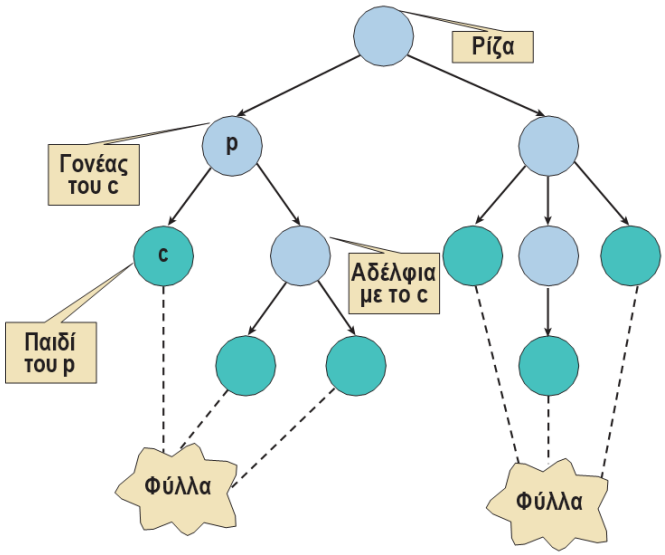
Ο κόμβος χωρίς γονέα ονομάζεται **ρίζα** και βρίσκεται στην κορυφή του δένδρου.

Όλοι οι κόμβοι, εκτός από την **ρίζα**, έχουν ακριβώς έναν γονέα.

Ένας κόμβος μπορεί να έχει κανένα, ένα ή περισσότερα παιδιά.

Κόμβοι με τον ίδιο γονέα ονομάζονται **αδέλφια**.

Οι κόμβοι χωρίς παιδιά ονομάζονται **φύλλα**.



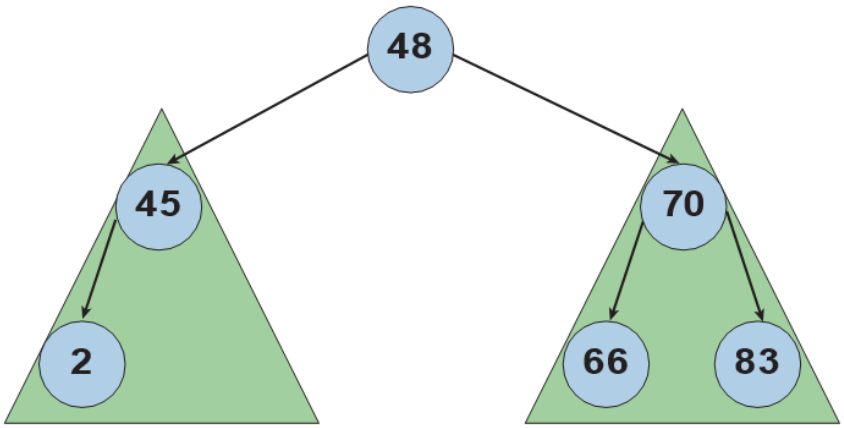
Μπορούμε να έχουμε ένα **απλό δένδρο**, το οποίο να απαρτίζεται από **έναν μόνο κόμβο**. Αυτός ο κόμβος είναι και ρίζα του απλού αυτού δένδρου, διότι δεν έχει γονέα και φύλλο, και διότι δεν έχει παιδιά.

Ένα δένδρο (tree) είναι μία δομή με τους εξής κανόνες:

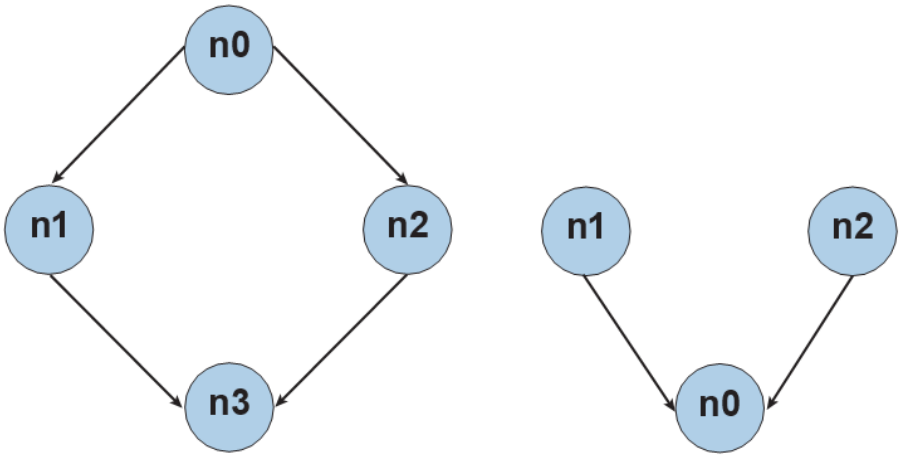
- Υπάρχει ένας ξεχωριστός κόμβος που ονομάζεται ρίζα. Αυτός είναι ένας κόμβος χωρίς γονέα.
- Για κάθε κόμβο c, εκτός από τη ρίζα, υπάρχει μόνο μια ακμή που καταλήγει στον κόμβο αυτόν ξεκινώντας από κάποιον άλλον κόμβο p. Ο κόμβος p ονομάζεται γονέας του c και ο κόμβος c παιδί του p.
- Για κάθε κόμβο υπάρχει μία μοναδική διαδρομή, δηλαδή, μια ακολουθία διαδοχικών ακμών, που ξεκινάει από τη ρίζα και τερματίζει σε αυτόν τον κόμβο.
- Δένδρο θεωρούμε και το κενό δένδρο, δηλαδή το δένδρο που δεν έχει ούτε κόμβους, ούτε ακμές. Το κενό δένδρο είναι το μόνο δένδρο χωρίς ρίζα.

Μέσα σε ένα δένδρο μπορούμε να εντοπίσουμε και άλλα μικρότερα δένδρα, που ονομάζονται **υποδένδρα**. Κάθε **κόμβος ενός δένδρου** μπορεί να **θεωρηθεί ως ρίζα** ενός υποδένδρου, δηλαδή ενός άλλου μικρότερου δένδρου, που ξεκινάει από τον κόμβο αυτόν.

Στο παρακάτω δένδρο, ο κόμβος **48** είναι ρίζα και **έχει δύο υποδένδρα** που ξεκινούν από τους κόμβους 45 και 70 αντίστοιχα. Ο κόμβος **45 έχει ένα υποδένδρο** που αποτελείται από τον κόμβο 2. Ο κόμβος **70 έχει δύο υποδένδρα** που αποτελούνται από τους κόμβους 66 και 83 αντίστοιχα. Τα υποδένδρα των κόμβων **2, 66 και 83 είναι κενά**.



Οι παρακάτω δομές δεν είναι δένδρα επειδή



Στην αριστερή δομή

- ο κόμβος **n3** έχει δύο γονείς, τους **n1** και **n2**
- υπάρχουν **δύο διαδρομές** από την ρίζα **n0** προς τον κόμβο **n3**

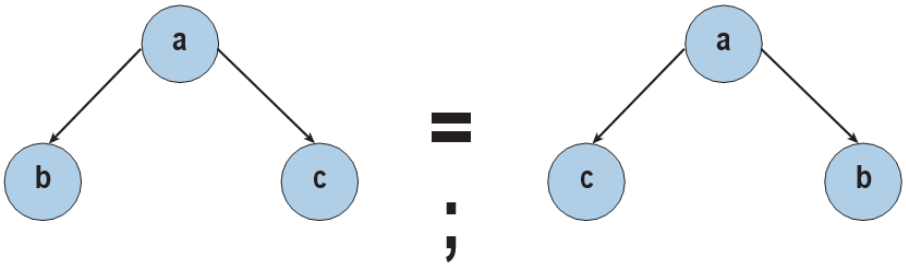
Στη δεξιά δομή

υπάρχουν **δύο κόμβοι χωρίς γονέα, οι n1 και n2**. Είναι σαν να λέμε ότι έχουμε δύο ρίζες στο δένδρο αυτό.

Τα δύο παρακάτω δένδρα είναι ίδια ή όχι;

Έχει σημασία η σειρά των αδελφών b και c;

Όχι πάντοτε. Αν θέλουμε να μοντελοποιήσουμε την ιεραρχική σχέση των μελών μιας οικογένειας και μας ενδιαφέρει να οργανώσουμε τα αδέλφια σύμφωνα με την ηλικία τους, τότε τα αδέλφια που θα έχουν γεννηθεί νωρίτερα θα τοποθετηθούν στην δενδρική δομή πιο αριστερά σε σχέση με αυτά που θα έχουν γεννηθεί αργότερα. Σε αυτή την περίπτωση, που για **κάθε κόμβο** υπάρχει μία **γραμμική σχέση μεταξύ των παιδιών του κόμβου** αυτού, αναφερόμαστε σε ένα **διατεταγμένο δένδρο.**



Τα δένδρα είναι μία μη-γραμμική ευέλικτη δομή δεδομένων.

Χρησιμοποιούνται σε πολλούς τομείς της επιστήμης των υπολογιστών:

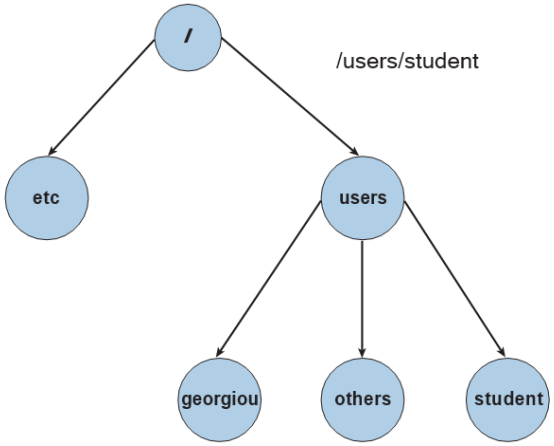
- λειτουργικά συστήματα
- γραφικά
- συστήματα βάσεων δεδομένων
- παιχνίδια
- τεχνητή νοημοσύνη
- δικτύωση υπολογιστών

Αναπαραστάσεις δεδομένων που υλοποιούνται με δένδρα

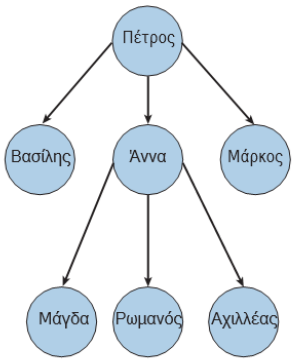
- το οικογενειακό δένδρο
- η δομή ενός οργανισμού
- ο πίνακας περιεχομένων ενός βιβλίου
- τα αρχεία και οι φάκελοι ενός υπολογιστή
- ένα λεξικό
- τα μέρη που απαρτίζουν την μηχανή ενός αυτοκινήτου
- τα συστατικά μιας πρότασης

Υπάρχουν δύο λόγοι για τους οποίους τα δένδρα είναι τόσο ισχυρά.

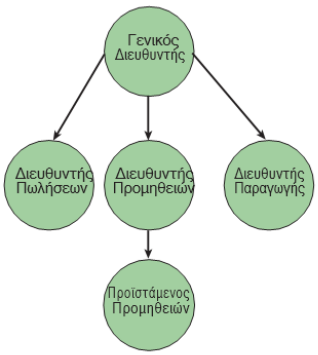
- είναι πολύ εύκολο να προσθέσετε, να αφαιρέσετε ή να αναζητήσετε ένα στοιχείο σε ένα δένδρο
- η δομή των δένδρων μεταφέρει πληροφορίες



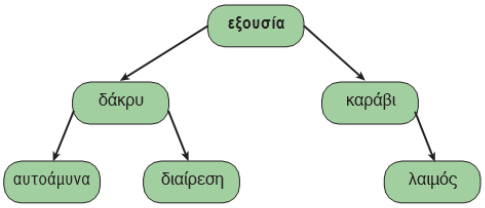
Ας θεωρήσουμε το σύστημα αρχείων του υπολογιστή, όπως φαίνεται στη διπλανή εικόνα. Είναι πολύ εύκολο να προσθέσετε ένα νέο κατάλογο για τον καθηγητή **georgiou**. Επίσης, λόγω του ότι ο κατάλογος **users** είναι παιδί της ρίζας **/** και ότι ο κατάλογος **student** είναι παιδί του **users** μπορούμε να συμπεράνουμε ότι υπάρχει η διαδρομή **/users/student**.



α. Οικογενειακό δένδρο



β. Οργανόγραμμα μιας εταιρείας

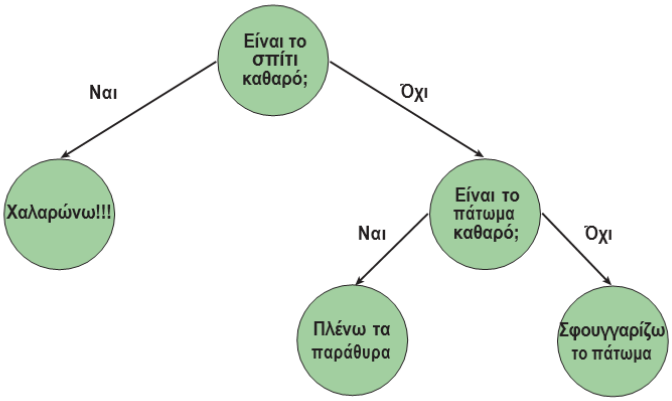


γ. Οργάνωση ενός λεξικού

Τα δένδρα, πέρα από μια αποτελεσματική οργάνωση και διαχείριση δεδομένων, αποτελούν τη βάση αρκετών αλγορίθμων επίλυσης προβλήματος, όπως:

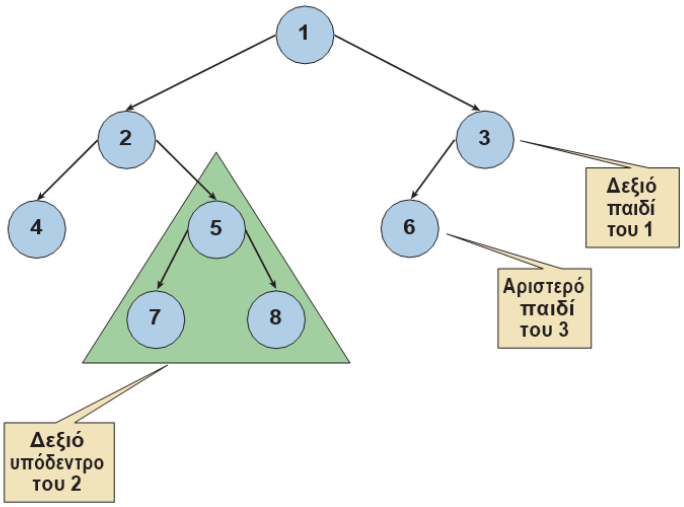
- η συμπίεση εικόνων
- η ταξινόμηση
- η αυτόματη συμπλήρωση λέξεων σε συσκευές κινητών τηλεφώνων
- η μεταγλώττιση ενός προγράμματος
- η λήψη αποφάσεων (μηχανική μάθηση)

Στα δένδρα απόφασης, κάθε κόμβος αντιπροσωπεύει ένα **χαρακτηριστικό (ιδιότητα)**, κάθε ακμή αντιπροσωπεύει μια **απόφαση (κανόνα)** και κάθε φύλλο αντιπροσωπεύει ένα **αποτέλεσμα**.



Δυαδικά Δένδρα

Ένα δυαδικό δένδρο (binary tree) είναι ένα διατεταγμένο δένδρο, στο οποίο κάθε κόμβος έχει το πολύ δύο παιδιά, το αριστερό και το δεξί παιδί. Στο παρακάτω δυαδικό δένδρο, ο κόμβος 3 έχει ως αριστερό υποδένδρο, το δένδρο με μοναδικό κόμβο το 6 και ως δεξιό υποδένδρο, το κενό δένδρο. Προφανώς, αν ανταλλάξουμε το αριστερό με το δεξιό υποδένδρο ενός κόμβου παίρνουμε ένα διαφορετικό δένδρο.

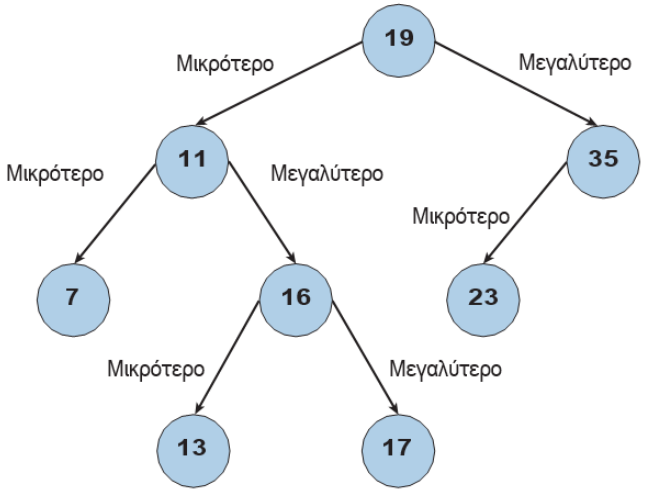


Δυαδικά Δένδρα Αναζήτησης

Οι αλγόριθμοι της αναζήτησης, για παράδειγμα όλων των email που έχετε λάβει ή στείλει σε μία συγκεκριμένη περίοδο ή της εύρεσης όλων των λέξεων, που αρχίζουν από την συμβολοσειρά «πληρ», αξιοποιούν μία ειδική κατηγορία δυαδικών δένδρων, αυτών των δυαδικών δένδρων αναζήτησης.

Θα χρησιμοποιήσουμε τώρα τα δυαδικά δένδρα με έναν τρόπο που θα μας επιτρέψει να τα βρίσκουμε πιο εύκολα. Η ιδέα πίσω από ένα δυαδικό δένδρο αναζήτησης είναι παρόμοια με αυτήν της δυαδικής αναζήτησης σε έναν ταξινομημένο πίνακα.

Ένα δυαδικό δένδρο αναζήτησης είναι ένα δυαδικό δένδρο, όπου για κάθε κόμβο u, όλοι οι κόμβοι του αριστερού υποδένδρου έχουν τιμές μικρότερες της τιμής του κόμβου u και όλοι οι κόμβοι του δεξιού υποδένδρου έχουν τιμές μεγαλύτερες (ή ίσες) της τιμής του κόμβου u.

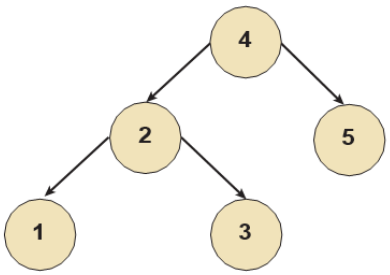


Το δυαδικό δένδρο της Εικόνας α πληροί τα κριτήρια του δυαδικού δένδρου αναζήτησης, σε αντίθεση με το δυαδικό δένδρο της Εικόνας β.

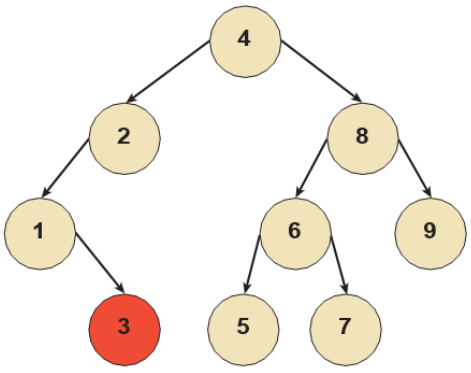
Στο υποδένδρο με **ρίζα το 1 ο κόμβος** 3 βρίσκεται στο **δεξιό** υποδένδρο αυτού και είναι **μεγαλύτερος** από τον κόμβο/ρίζα 1.

Όμως στο υποδένδρο με **ρίζα το 2 ο κόμβος 3** βρίσκεται στο **αριστερό** υποδένδρο αυτού αλλά δεν είναι **μικρότερος** από τον κόμβο/ρίζα 2.

Εξαιτίας του τελευταίου γεγονότος, δηλαδή της ύπαρξης ενός τουλάχιστον κόμβου (στην περίπτωσή μας του κόμβου 3) που δεν πληροί το κριτήριο του δυαδικού δένδρου αναζήτησης, συμπεραίνουμε ότι δεν μπορούμε να κατατάξουμε το δυαδικό αυτό δένδρο στα δυαδικά δένδρα αναζήτησης.



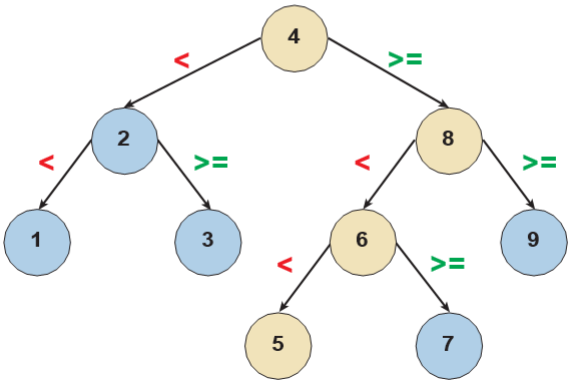
Εικόνα α: Δυαδικό δένδρο αναζήτησης



β: Μη δυαδικό δένδρο αναζήτησης

Το πλεονέκτημα των δυαδικών δένδρων αναζήτησης είναι η ταχύτητα αναζήτησης για μια συγκεκριμένη τιμή χάρη στον τρόπο αποθήκευσης των τιμών.

1	8	2	9	3	6	7	4	5



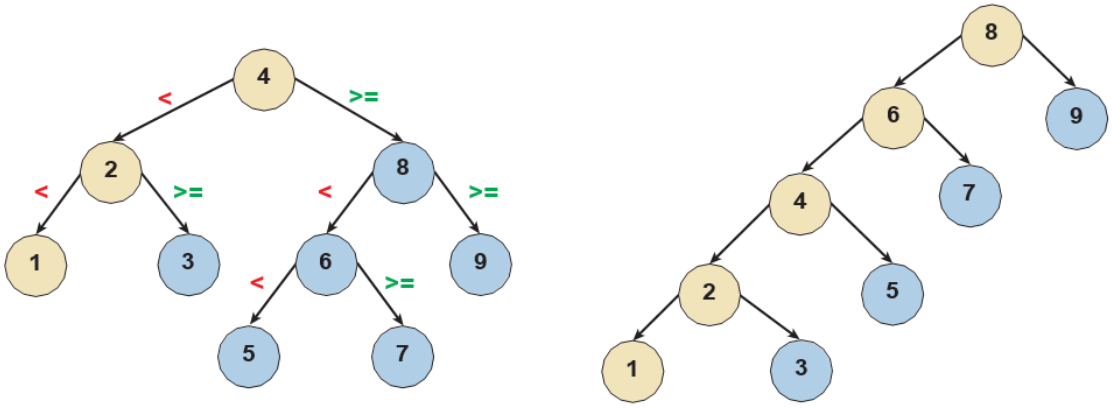
Θεωρείστε ότι ψάχνετε το στοιχείο 5 στον παραπάνω πίνακα (σειριακή δομή). Στην περίπτωση αυτή, θα χρειαζόταν να διασχίσουμε ολόκληρο τον πίνακα για να βρούμε το στοιχείο 5, δηλ. στην 9η σύγκριση.

Τώρα, θεωρείστε ότι ψάχνετε το στοιχείο 5 στο παραπάνω δένδρο.
Αρχικά, συγκρίνετε τη ρίζα του δένδρου (το 4) με το υπό αναζήτηση στοιχείο (το 5). Το **5 > 4** συνεπώς συνεχίζετε με το δεξιό υποδένδρο και αγνοείτε το αριστερό υποδένδρο.
Στη συνέχεια, συγκρίνετε το 8 με το 5. Επειδή το **8 > 5**, συνεχίζετε με το αριστερό υποδένδρο. Κατόπιν, συναντάτε το 6 όπου **6 > 5**.
Μεταβαίνετε, λοιπόν, στο αριστερό υποδένδρο του 6, όπου βρίσκετε το στοιχείο 5 σε μόλις 3 συγκρίσεις.

Στην περίπτωση που δεν υπάρχει το στοιχείο στο δένδρο (για παράδειγμα αν ψάχναμε το στοιχείο 12), η αναζήτηση καταλήγει σε ένα κενό υποδένδρο και εκεί τερματίζει η όλη διαδικασία.
Με βάση τον αλγόριθμο αυτό, μειώνουμε δραματικά τον χρόνο για να βρούμε το στοιχείο που ψάχνουμε και αυτό διότι περιορίζουμε αισθητά τους κόμβους τους οποίους επισκεπτόμαστε. Κάθε φορά αφήνουμε στην άκρη ένα υποδένδρο και συνεχίζουμε με το άλλο.

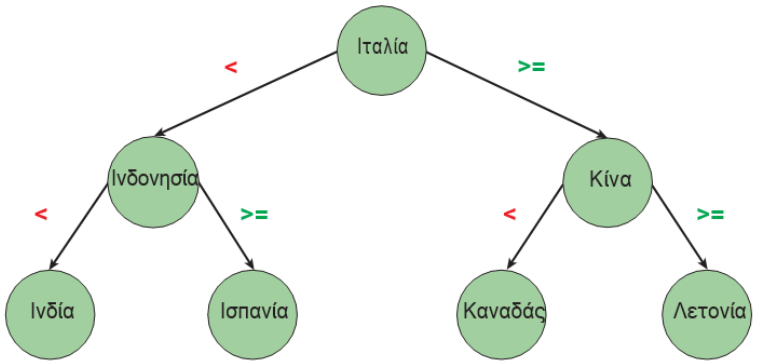
Θεωρείστε ότι αναζητούμε το στοιχείο 1 στα παρακάτω δυαδικά δένδρα αναζήτησης. Σε ποιο από τα δύο δυαδικά δένδρα αναζήτησης θα βρούμε πιο γρήγορα την τιμή 1;
Μήπως στο πρώτο, διότι σε κάθε βήμα απορρίπτουμε όσο περισσότερους κόμβους είναι δυνατόν; Οπτικά, θα μπορούσαμε να θεωρήσουμε ότι το πρώτο δένδρο είναι πιο «**ισορροπημένο**» σε σχέση με το δεύτερο; Παρατηρούμε, λοιπόν, χωρίς να αναφερθούμε διεξοδικά στα ισορροπημένα δένδρα, ότι αν θέλουμε να έχουμε γρήγορους αλγόριθμους αναζήτησης πρέπει να αποθηκεύουμε τις τιμές στα δυαδικά δένδρα αναζήτησης με έναν

συγκεκριμένο τρόπο.



Τα **δυαδικά δένδρα αναζήτησης** συνδυάζουν τα **πλεονεκτήματα των λιστών**, όσον αφορά τις πράξεις της εισαγωγής και της διαγραφής, αλλά και τα **πλεονεκτήματα των ταξινομημένων πινάκων**, όσον αφορά την πράξη της αναζήτησης.

Ινδία	Ινδονησία	Ισπανία	Ιταλία	Καναδάς	Κίνα	Λετονία



ΒΑΣΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ ΣΕ ΜΟΝΟΔΙΑΣΤΑΤΟΥΣ ΠΙΝΑΚΕΣ

Εισαγωγή στοιχείων σε μονοδιάστατο πίνακα Α[Ν]

```
Για Ι από 1 μέχρι Ν
    Εμφάνισε΄Δώσε το΄, i,΄στοιχείο του πίνακα΄
    Διάβασε Α[Ι]
Τέλος_επανάληψης
```

Εύρεση αθροίσματος και μέσου όρου πίνακα Π[100]

```
Σ ←0
Για i από 1 μέχρι 100
     Σ Σ+Π[i]
Τέλος_επανάληψης
Εμφάνισε΄Το άθροισμα είναι:΄, Σ
ΜΟ ← Σ/100
Εμφάνισε΄Ο μέσος όρος είναι:΄, ΜΟ
```

Εύρεση μέγιστου και θέσης σε πίνακα Π[100]

```
ΜΑΧ ←Π[1]
Pos ←1
Για i από 2 μέχρι 100
    Αν Π[i]>ΜΑΧ τότε
       ΜΑΧ ← Π[i]
       Pos ← i
    Τέλος_αν
Τέλος_επανάληψης
Εμφάνισε΄Το μέγιστο είναι:΄, ΜΑΧ, "στη θέση", Pos
```

Αναζήτηση στοιχείου και όλων των θέσεων, σε πίνακα Π[100]

```
Εμφάνισε΄Δώσε την τιμή που αναζητάς:΄
Διάβασε x
ΒΡΕΘΗΚΕ ← ΨΕΥΔΗΣ
Για i από 1 μέχρι 100
     Αν Π[i] = x τότε
        ΒΡΕΘΗΚΕ ←ΑΛΗΘΗΣ
        Εμφάνισε΄Η τιμή΄, x,΄βρέθηκε στη θέση΄, i,΄του πίνακα΄
     Τέλος_αν
Τέλος_επανάληψης
Αν ΒΡΕΘΗΚΕ = ΨΕΥΔΗΣ τότε
     Εμφάνισε΄Δεν βρέθηκε στον πίνακα΄
Τέλος_αν
```

Αναζήτηση στοιχείου και της 1ης θέσης, σε πίνακα Π[Ν]

```
pos ← 0
```

```
F ← ΨΕΥΔΗΣ
I ← 1
όσο i<= N και F = ΨΕΥΔΗΣ επανάλαβε
      αν A[i] = key τότε
          pos ← i
          F ← ΑΛΗΘΗΣ
      τέλος_αν
      i ←i + 1
τέλος_επανάληψης
Αν ΒΡΕΘΗΚΕ = ΨΕΥΔΗΣ τότε
      Εμφάνισε΄Δεν βρέθηκε στον πίνακα΄
Αλλιώς
      Εμφάνισε΄Βρέθηκε στον πίνακα στη θέση΄, pos
Τέλος_αν
```

Αύξουσα ταξινόμηση των στοιχείων πίνακα Α[100]

```
Για i από 2 μέχρι 100
      Για j από 100 μέχρι k με_βήμα -1
          Αν A[j-1] > A[j] τότε
                Temp ← A[ j-1 ]
                A[ j-1 ] ← A[ j ]
                A [ j ] ← Temp
          Τέλος_αν
      Τέλος_επανάληψης
Τέλος_επανάληψης
```

Στην περίπτωση **παράλληλων πινάκων** (π.χ. Α και Β), όταν θέλουμε σε περίπτωση ισότητας των στοιχείων του πρώτου πίνακα να ταξινομούμε ως προς τα στοιχεία του δεύτερου, τότε ο κώδικάς μας μετατρέπεται ως εξής:

```
   για i από 2 μέχρι 100
         για j από 100 μέχρι i με_βήμα –1
               αν A[ j-1 ] > A[ j ] τότε
                     temp← A[ j-1 ]
                     A[ j-1 ] ← A[ j ]
                     A[ j ] ←temp
                     Temp1← B[ j-1 ]
                     B[ j-1 ] ← B[ j ]
                     B[ j ] ←temp1
               αλλιώς_αν A[ j-1 ] = A[ j ] τότε
                     αν B[ j-1 ] > B[ j ] τότε
                           temp1← B[ j-1 ]
                           B[ j-1 ] ←B[ j ]
                           B[ j ] ←temp1
                     τέλος_αν
```

```
            τέλος_αν
        τέλος_επανάληψης
    τέλος_επανάληψης
```

Βελτίωση του κώδικα ταξινόμησης

Ο αρχικός αλγόριθμος της ταξινόμησης θα διατρέξει τον πίνακα Ν-1 φορές (Ν το μέγεθος του πίνακα) ακόμη και αν ο πίνακας ήταν σε τέτοια μορφή, ώστε να έχει ταξινομηθεί από το πρώτο κιόλας πέρασμα.
Αν θέλουμε να κάνουμε τον κώδικά μας πιο "έξυπνο", ώστε να αποφεύγει τα επιπλέον άσκοπα περάσματα θα πρέπει να αντικαταστήσουμε την πρώτη επαναληπτική δομή **για Ι από 2 μέχρι Ν** η οποία καθορίζει το πλήθος των περασμάτων του πίνακα με μία δομή όσο … επανάλαβε στην οποία θα ελέγχεται και το αν ο πίνακας είναι ήδη ταξινομημένος.
Για τον σκοπό αυτό, ελέγχουμε με μία λογική μεταβλητή, αν γίνεται έστω και μία αντιμετάθεση σε κάποιο πέρασμα.

```
i ← 2
done ← ΨΕΥΔΗΣ
Όσο i ≤ 100 και done = ΨΕΥΔΗΣ επανάλαβε
    done ← ΑΛΗΘΗΣ
    για j από 100 μέχρι i με_βήμα –1
        αν A[ j-1 ] > A[ j ] τότε
            temp ← A[ j-1 ]
            A[ j-1 ] ← A[ j ]
            A[ j ] ← temp
            done ← ΨΕΥΔΗΣ
        τέλος_αν
    τέλος_επανάληψης
    i ← i + 1
τέλος_επανάληψης
```

ΒΑΣΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ ΔΙΣΔΙΑΣΤΑΤΩΝ ΠΙΝΑΚΩΝ

Εισαγωγή στοιχείων σε πίνακα Α[100, 50] ανά γραμμές

```
Για Ι από 1 μέχρι 100
    Για J από 1 μέχρι 50
        Διάβασε Α[Ι, J]
    Τέλος_επανάληψης
Τέλος_επανάληψης
```

Εισαγωγή στοιχείων σε πίνακα Α[Κ, Ν] ανά στήλες

```
Για J από 1 μέχρι 50
    Για Ι από 1 μέχρι 100
        Διάβασε Α[Ι, J]
    Τέλος_επανάληψης
Τέλος_επανάληψης
```

Υπολογισμός αθροίσματος όλων των στοιχείων πίνακα Α[100,50]

Σ ← 0
Για Ι από 1 μέχρι 100
    Για J από 1 μέχρι 50
    Σ ← Σ + Α[Ι,J]
    Τέλος_επανάληψης
Τέλος_επανάληψης
Εμφάνισε "Το άθροισμα των στοιχείων του πίνακα είναι:", Σ

Υπολογισμός αθροίσματος κατά γραμμή πίνακα Α[100,50]

Για Ι από 1 μέχρι 100
    Σ ← 0
    Για J από 1 μέχρι 50
        Σ ← Σ + Α[Ι,J]
    Τέλος_επανάληψης
    S[Ι] ← Σ
Τέλος_επανάληψης

Υπολογισμός αθροίσματος κατά στήλη πίνακα Α[100,50]

Για J από 1 μέχρι 50
    Σ ← 0
    Για Ι από 1 μέχρι 100
        Σ ← Σ + Α[Ι,J]
    Τέλος_επανάληψης
    S[j] ← Σ
Τέλος_επανάληψης

Υπολογισμός αθροίσματος της 5ης γραμμής πίνακα Α[100,50]

Σ ← 0
Για J από 1 μέχρι 50
    Σ ← Σ + Α[5,J]
Τέλος_επανάληψης
Εμφάνισε "Το άθροισμα της 5ης γραμμής είναι:", Σ

Υπολογισμός αθροίσματος της 3ης στήλης πίνακα Α[100,50]

Σ ← 0
Για Ι από 1 μέχρι 100
    Σ ← Σ + Α[Ι,3]
Τέλος_επανάληψης
Εμφάνισε "Το άθροισμα της 3ης στήλης είναι:", Σ

Εύρεση μέγιστου σε πίνακα Π[100,50] με καθορισμό θέσης

ΜΕΓ ← Π[1,1]
ΓΡΑΜΜΗ ← 1
ΣΤΗΛΗ ← 1

```
Για Ι από 1 μέχρι 100
    Για J από 1 μέχρι 50
        Αν Π[Ι,J] > ΜΕΓ τότε
            ΜΕΓ ← Π[Ι,J]
            ΓΡΑΜΜΗ ← Ι
            ΣΤΗΛΗ ← J
        Τέλος_αν
    Τέλος_επανάληψης
Τέλος_επανάληψης
Εμφάνισε "Μέγιστο:", ΜΕΓ, "και είναι στη γραμμή", ΓΡΑΜΜΗ", και στη στήλη", ΣΤΗΛΗ
```

Σειριακή αναζήτηση δισδιάστατου πίνακα Α[100,50]

```
posi ← 0
posj ← 0
F ← ΨΕΥΔΗΣ
i← 1
όσο (i <= 100) και (F = ΨΕΥΔΗΣ) επανάλαβε
    j← 1
    όσο (j <= 50) και (F = ΨΕΥΔΗΣ) επανάλαβε
    αν A[i, j] = key τότε
    posi ← i
    posj ← j
    F ← ΑΛΗΘΗΣ
    τέλος_αν
        j ←j + 1
    τέλος_επανάληψης
    i←i + 1
τέλος_επανάληψης
```