Ανάπτυξη εφαρμογών σε προγραμματιστικό περιβάλλον

Γ’ Λυκείου

ΚΕΦΑΛΑΙΟ 3

Δομές δεδομένων και αλγόριθμοι

Χρήστος Μουρατίδης - Έκδοση 2020
mouratx@yahoo.com
http://users.sch.gr/mouratx

Περιεχόμενα

Δεδομένα

Δεδομένα : Κάθε στοιχείο (*σύμβολα, γράμματα, λέξεις, φράσεις, κείμενο, αριθμοί, εικόνα, video, ήχος*) που μπορεί να χρησιμοποιηθεί προς επεξεργασία για την παραγωγή πληροφορίας.

Συνήθως, από το πλήθος των δεδομένων που χαρακτηρίζουν μία οντότητα[1], μας ενδιαφέρουν ορισμένα μόνο. Για παράδειγμα, από την οντότητα «`Μαθητής`» μας ενδιαφέρουν ο αριθμός μητρώου, το επώνυμο, το όνομα, το πατρώνυμο και μητρώνυμο, το έτος γέννησης κλπ. αλλά όχι το βάρος, το ύψος κλπ.

Είναι φανερό ότι συλλέγονται, αποθηκεύονται και επεξεργάζονται, κάθε φορά, μόνο εκείνα τα που απαιτούνται για την επίλυση του προβλήματος. Και είναι επίσης φανερό ότι ο αλγόριθμος παίζει ιδιαίτερο ρόλο στην επεξεργασία των δεδομένων και την παραγωγή της πληροφορίας. Η εύρεση του κατάλληλου και συγχρόνως αποδοτικού αλγορίθμου για τα συγκεκριμένα δεδομένα είναι ιδιαίτερης σημασίας.

Σπουδαιότητα των δεδομένων

Η Πληροφορική μελετά τα δεδομένα από διαφορετικές οπτικές γωνίες, οι οποίες είναι οι εξής:

- **Από την πλευρά του Υλικού (hardware):** Τα δεδομένα αποθηκεύονται στη κύρια (RAM/ROM) και δευτερεύουσα μνήμη[2] (δίσκοι, μνήμη flash) σε διάφορες μορφές. Αυτές οι μορφές καλούνται και ***αναπαραστάσεις δεδομένων*** και εξαρτώνται από την εκάστοτε χρησιμοποιούμενη *κωδικοποίηση των δεδομένων*. Η πιο γνωστή είναι η ASCII, UNICODE, EBCDIC κ.α.

- **Από την πλευρά των Γλωσσών Προγραμματισμού:** Κάθε γλώσσα προγραμματισμού υψηλού επιπέδου (π.χ C#, Visual Basic, Java) επιτρέπει τη χρήση διαφορετικών ***τύπων μεταβλητών*** (variable types) για να περιγράψουν ένα

[1] Λέγονται και *ιδιότητες* της οντότητας.
[2] Λέγεται και *βοηθητική μνήμη*.

δεδομένο. Ο μεταφραστής της γλώσσας φροντίζει για την αποδοτικότερη μορφή αποθήκευσης, από πλευράς υλικού, κάθε μεταβλητής στον υπολογιστή. Για παράδειγμα, για έναν ακέραιο αριθμό, π.χ. τον `15`, ο προγραμματιστής στη Visual Basic έχει διαθέσιμους τους εξής τύπους μεταβλητών: `Short`, `Integer` και `Long`[3]. Ο μεταφραστής της Visual Basic θα αναλάβει να αποθηκεύσει στην αποδοτικότερη μορφή τον ακέραιο στη μνήμη.

- **Από την πλευρά των Δομών Δεδομένων:** Μία δομή δεδομένων είναι ένα *σύνολο δεδομένων* μαζί με ένα *σύνολο επιτρεπτών λειτουργιών* σε αυτό. Για παράδειγμα, ένας **πίνακας** είναι μία δημοφιλής δομή δεδομένων. Οργανώνει πολλά δεδομένα μαζί σε ένα ομοιόμορφο σύνολο πάνω στο οποίο μπορούμε να κάνουμε διάφορες λειτουργίες όπως **ανάγνωση (διάβασμα), προσθήκη, διαγραφή, αναζήτηση, ταξινόμηση**. Άλλες δομές είναι η **λίστες**, **στοίβες, ουρές, εγγραφές, αρχεία** κλπ. όπου η κάθε μία χρησιμοποιείται ανάλογα με το είδος του προβλήματος.

- **Από την πλευρά της Ανάλυσης Δεδομένων:** Μελετώνται τρόποι καταγραφής και αλληλοσυσχέτισης των δεδομένων προκειμένου να αναπαρασταθεί η γνώση για πραγματικά γεγονότα ή να εξαχθούν συμπεράσματα για γεγονότα που εκ πρώτης όψεως δεν είναι εμφανή. Εδώ υπεισέρχονται οι τεχνολογίες των Βάσεων Δεδομένων, Μοντελοποίησης Δεδομένων, Αναπαράσταση Γνώσης, Μεγάλων Δεδομένων (Big Data). Η ανάλυση των δεδομένων τελικά αποσκοπεί στην *λήψη στρατηγικών αποφάσεων* και *κατάλληλων επιχειρηματικών κινήσεων*.

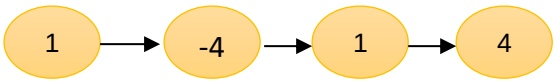
Η βασική ισότητα: Προγράμματα = Αλγόριθμοι + Δομές Δεδομένων

Δομή δεδομένων : **Είναι ένας τρόπος αποθήκευσης (οργάνωσης) δεδομένων που υφίστανται επεξεργασία από ένα σύνολο λειτουργιών.**

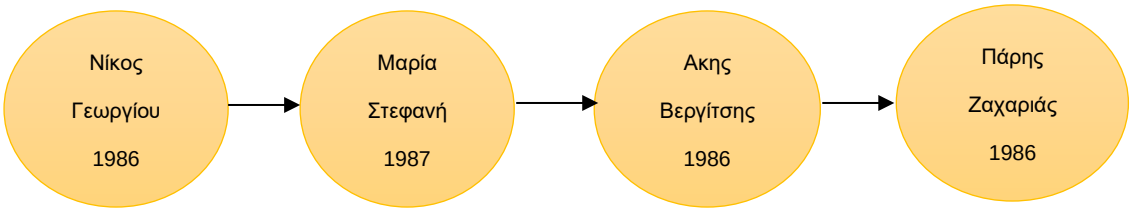
Η δομή αποτελείται από ένα σύνολο *κόμβων*. Κάθε κόμβος περιέχει μία απλή τιμή (αριθμό, αλφαριθμητικό κ.λπ.) ή μία πιο σύνθετη τιμή (π.χ. μία εγγραφή)

[3] Κάθε τέτοιος υπο-τύπος διαφοροποιείται ανάλογα με το πλήθος των bits που δεσμεύονται στη μνήμη για την αποθήκευση του ακέραιου αριθμού. `Short` = 16 bits, `Integer` = 32 bits, `Long` = 64 bits.

Π.χ. **δομή λίστας που περιέχει ακέραιους αριθμούς**:



Π.χ. **δομή λίστας που περιέχει εγγραφές** (records. *Εγγραφή = Μία ομάδα τιμών διαφορετικού τύπου σχετικά με ένα αντικείμενο. Π.χ. μία εγγραφή μαθητή περιέχει το όνομα, επώνυμο, έτος γέννησης, τάξη κλπ.*):



Βασικές **λειτουργίες (πράξεις)** επί των δομών:

Λειτουργία	Περιγραφή
Προσπέλαση (Access)	Είναι η δυνατότητα πρόσβασης σε ένα κόμβο με σκοπό την ανάγνωση ή τροποποίηση του περιεχομένου του.
Αναζήτηση (Searching)	Προσπελαύνονται οι κόμβοι μίας δομής με σκοπό να εντοπιστεί κάποιος που περιέχει μία συγκεκριμένη τιμή.
Εισαγωγή (Insertion)	Προσθήκη νέου κόμβου στη δομή.
Διαγραφή (Deletion)	Αφαιρούμε έναν κόμβο από τη δομή.
Ταξινόμηση (Sorting)	Διατάσσουμε τους κόμβους της δομής σε αύξουσα ή φθίνουσα σειρά (π.χ. αλφαβητικά κατά επώνυμο, όνομα).
Αντιγραφή (Copying)	Όλοι ή μερικοί κόμβοι αντιγράφονται σε μία άλλη δομή.
Συγχώνευση (Merging)	Δύο ή περισσότερες δομές συνενώνονται σε μία ενιαία δομή.
Διαχωρισμός (Separation)	Μία δομή διασπάται σε δύο ή περισσότερες. Είναι το αντίστροφο της συγχώνευσης.

Για κάθε λειτουργία δημιουργείται και ένας αλγόριθμος (π.χ. αλγόριθμος αναζήτησης κάποιου στοιχείου). Αρκετές φορές υπάρχουν διαφορετικοί αλγόριθμοι που υλοποιούν μία συγκεκριμένη λειτουργία (π.χ. υπάρχουν αρκετοί αλγόριθμοι για τη λειτουργία της ταξινόμησης). Όταν συμβαίνει αυτό, επιλέγουμε τον αλγόριθμο που είναι πιο αποδοτικός (π.χ. πιο γρήγορος) για τα συγκεκριμένα δεδομένα.

Υπάρχει **μεγάλη εξάρτηση μεταξύ της δομής δεδομένων και του αλγορίθμου που επεξεργάζεται τη δομή**. Το πρόγραμμα θεωρεί τη δομή δεδομένων και τον αλγόριθμο ως μία αδιάσπαστη ενότητα. Έτσι, έχουμε τον παρακάτω κανόνα:

Πρόγραμμα = Δομές δεδομένων + Αλγόριθμοι

Κατηγορίες δομών δεδομένων

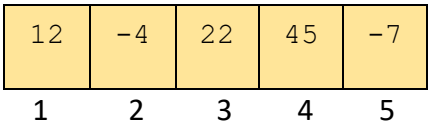
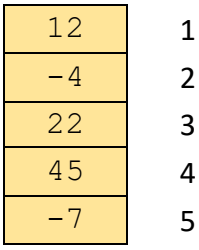
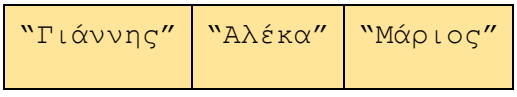
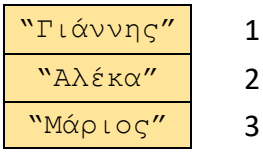
Στατικές	Δυναμικές
Το μέγεθός τους είναι καθορισμένο (σταθερό) και δεν μεταβάλλεται κατά τη διάρκεια εκτέλεσης του προγράμματος.	**Το μέγεθός τους δεν είναι καθορισμένο** (δεν είναι σταθερό) και μεταβάλλεται κατά τη διάρκεια εκτέλεσης του προγράμματος. Μπορούν να εισάγονται νέοι κόμβοι και να διαγράφονται υπάρχοντες.
Οι κόμβοι αποθηκεύονται σε συνεχόμενες θέσεις στη μνήμη.	**Οι κόμβοι δεν αποθηκεύονται σε συνεχόμενες θέσεις στη μνήμη**.

Πίνακες

Είναι ο **χαρακτηριστικότερος εκπρόσωπος των στατικών δομών**. Περιέχει ένα σταθερό σύνολο κόμβων (θέσεις) που αποθηκεύονται σε συνεχόμενες θέσεις στη μνήμη. Επίσης, όλα τα δεδομένα που αποθηκεύει είναι του **ιδίου τύπου** (δηλαδή ακέραιοι, πραγματικοί κ.λπ.).

Μονοδιάστατοι πίνακες (μίας γραμμής ή μίας στήλης)

Παραδείγματα:

Μίας γραμμής	Μίας στήλης
Πίνακας ακεραίων **Π[1:5]**	Πίνακας ακεραίων **Π[1:5]**
	
Πίνακας αλφαριθμητικών **Π[1:3]**	Πίνακας αλφαριθμητικών **Π[1:3]**
	

Δεν έχει σημασία η οπτική αναπαράσταση του πίνακα (αν θα είναι γραμμής ή στήλης) καθότι οι λειτουργίες του είναι ανεξάρτητες από αυτήν. Μπορεί κάποιος να θεωρήσει τη δομή του μονοδιάστατου πίνακα σε όποια οπτική επιθυμεί.

Δείκτης θέσης: Οι αριθμοί κάτω από τις θέσεις του πίνακα (*στην οπτική της γραμμής*) ή δίπλα (*στην οπτική της στήλης*) δηλώνουν τον **αριθμό θέσης** του πίνακα. Ο αριθμός θέσης λέγεται και **δείκτης θέσης**.

Πώς ορίζεται ο μονοδιάστατος πίνακας: Ο πίνακας **ορίζεται ως εξής**: **τύπος *όνομα_πίνακα [διαστάσεις].*** Π.χ. ακέραιος `Π[1:5]` δηλώνει έναν μονοδιάστατο πίνακα μίας γραμμής πέντε θέσεων (στήλες) που περιέχει ακεραίους (*ή στην οπτική της στήλης*

δηλώνει έναν μονοδιάστατο πίνακα μίας στήλης πέντε θέσεων (γραμμές) που περιέχει ακεραίους).

Γενικά, ορίζεται ως Π [1:N].

Αναφορά στο περιεχόμενο μίας θέσης: Για να αναφερθούμε στο περιεχόμενο μία θέσης του πίνακα βάζουμε το ***όνομα[θέση]***.

Για παράδειγμα, `Π[1]` = το περιεχόμενο της θέσης `1`
`Π[6]` = το περιεχόμενο της θέσης `6`

Το περιεχόμενο της θέσης `1` στον παραπάνω πίνακα αλφαριθμητικών είναι
`Π[1] = "Γιάννης"`

Στις παρούσες σημειώσεις θα χρησιμοποιήσουμε την *οπτική της γραμμής* για τον μονοδιάστατο πίνακα.

Δισδιάστατοι πίνακες (πολλών γραμμών και στηλών)

Εδώ έχουμε περισσότερες από μία γραμμές. Κάθε γραμμή έχει ένα σύνολο θέσεων (στήλες).

Παραδείγματα:

Πίνακας ακεραίων **Π[1:3, 1:5]**

12	-4	22	45	-7
8	12	3	0	0
15	22	12	4	-6

Πίνακας αλφαριθμητικών **Π[1:4, 1:3]**

"Γιάννης"	"Αλέκα"	"Μάριος"
"Σωτήρης"	"Λεωνίδας"	"Περικλής"
"Ιωάννα"	"Μαρία"	"Νίκος"
"Δέσποινα"	"Άγγελος"	"Χρήστος"

Η **πρώτη διάσταση αναφέρεται στις γραμμές και η δεύτερη στις στήλες**.

Δείκτης θέσης: Ο δείκτης θέσης του πίνακα προσδιορίζεται πλέον με δύο αριθμούς. Ο πρώτος υποδηλώνει τον αριθμό γραμμής και ο δεύτερος τον αριθμό στήλης Π.χ. `[1,3]` σημαίνει την θέση που βρίσκεται στην 1η γραμμή και 3η στήλη.

	1	2	3	4
1	Π[1,1]	Π[1,2]	Π[1,3]	Π[1,4]
2	Π[2,1]	Π[2,2]	Π[2,3]	Π[2,4]
3	Π[3,1]	Π[3,2]	Π[3,3]	Π[3,4]

Πώς ορίζεται ο δισδιάστατος πίνακας: Ο πίνακας **ορίζεται ως εξής**: **τύπος *όνομα_πίνακα [διάσταση1, διάσταση2].*** Π.χ. ακέραιος `Π[1:3, 1:5]` δηλώνει έναν δισδιάστατο πίνακα 3 γραμμών και πέντε στηλών που περιέχει ακεραίους.

Γενικά, ορίζεται ως Π [1:M, 1:N] για έναν πίνακα MxN.

Αναφορά στο περιεχόμενο μίας θέσης: Για να αναφερθούμε στο περιεχόμενο μία θέσης του πίνακα βάζουμε το ***όνομα[θέση]***.

Για παράδειγμα, `Π[1,3]` = το περιεχόμενο της θέσης `[1,3]`
`Π[6,2]` = το περιεχόμενο της θέσης `[6,2]`

Το περιεχόμενο της θέσης `[3,1]` στον παραπάνω πίνακα αλφαριθμητικών είναι
`Π[3,1] = "Ιωάννα"`

Τυπικές επεξεργασίες (λειτουργίες) σε έναν πίνακα.

- **Προσπέλαση** των στοιχείων του πίνακα, προκειμένου να αναγνώσουμε τις τιμές[4] του.
- **Εισαγωγή τιμών**.
- **Διαγραφή μίας ή περισσοτέρων τιμών**.

[4] Με τον όρο τιμή μίας θέσης εννοούμε το περιεχόμενο της θέσης, το οποίο μπορεί να είναι αριθμός ή αλφαριθμητικό.

- **Εκτύπωση** των στοιχείων του πίνακα
- Υπολογισμός του **αθροίσματος** των στοιχείων του (αν έχει αριθμούς).
- Υπολογισμός του **μέσου όρου** των στοιχείων του (αν έχει αριθμούς).
- Εύρεση του **ελάχιστου ή μέγιστου** στοιχείου του (αν έχει αριθμούς).
- **Αναζήτηση** ενός στοιχείου.
- **Ταξινόμηση** του πίνακα.
- **Συγχώνευση** δύο πινάκων, δηλαδή συνδυάζοντας τα στοιχεία δύο πινάκων με τέτοιο τρόπο ώστε να προκύψει ένας νέος ενιαίος πίνακας. Στο παράρτημα, υπάρχει ένας σχετικός αλγόριθμος.

Στους αλγόριθμους επεξεργασίας των πινάκων (μονοδιάστατους ή δισδιάστατους) χρησιμοποιείται εκτενώς ή δομή επανάληψης `Για.. από..μέχρι…`, όπως θα δούμε παρακάτω.

Παρακάτω, θα δούμε μερικούς τυπικούς γενικούς αλγόριθμους επεξεργασίας πινάκων, πρώτα για μονοδιάστατο και κατόπιν για δισδιάστατο πίνακα, κάνοντας χρήση της ψευδογλώσσας, στοιχεία της οποίας είδαμε στο 2ο Κεφάλαιο.

Αλγόριθμοι μονοδιάστατου πίνακα

Θεωρούμε ότι οι παρακάτω αλγόριθμοι επενεργούν επί ενός πίνακα `Π[1:N]` όπου τα στοιχεία του είναι ακέραιοι αριθμοί.

Διάβασμα στοιχείων

```
Αλγόριθμος Διάβασμα_στοιχείων
Δεδομένα // Π[1:N] //
```

```
    Για  i από 1 μέχρι N
        Διάβασε Π[i]
    Τέλος_επανάληψης
Τέλος Διάβασμα_στοιχείων
```

Ο πίνακας έχει τόσες θέσεις όσα τα δεδομένα που θα διαβαστούν. Με μία επανάληψη `N` φορές διαβάζονται ένα-ένα τα στοιχεία στην αντίστοιχη θέση του πίνακα. **Ο μετρητής `i` χρησιμοποιείται ως δείκτης θέσης**.

Μπορούμε να κατασκευάσουμε τον **πίνακα τιμών** για να δούμε **βήμα-βήμα την εκτέλεση του αλγορίθμου**:

Π =	Π[1]	Π[2]	Π[3]	Π[4]	...	Π[N]

Επανάληψη	Τιμές Μετρητή `i`	Θέση Πίνακα `Π[i]`
1η	1	Π[1]
2η	2	Π[2]
3η	3	Π[3]
…	…	…
Νη	N	Π[N]

Με τον πίνακα τιμών ελέγχουμε την «καλή» εκτέλεση του αλγορίθμου μας, ιδιαίτερα όταν έχει δομές επανάληψης.

Ο παραπάνω αλγόριθμος είναι γενικός. Αντί για ακεραίους θα μπορούσαμε να είχαμε αλφαριθμητικά.

Εκτύπωση στοιχείων

```
Αλγόριθμος Εκτύπωση_στοιχείων
Δεδομένα // Π[1:N] //
     Για  i από 1 μέχρι N
          Εκτύπωσε Π[i]
     Τέλος_επανάληψης
Τέλος Εκτύπωση_στοιχείων
```

Υπολογισμός αθροίσματος

```
Αλγόριθμος Άθροισμα_στοιχείων
Δεδομένα // Π[1:N] //
     SUM ← 0        !αρχικοποίηση πριν μπει στην επανάληψη
     Για  i από 1 μέχρι N
          SUM ← SUM +  Π[i]
     Τέλος_επανάληψης
     Εκτύπωσε SUM
Τέλος Άθροισμα_στοιχείων
```

Η μεταβλητή SUM είναι συσσωρευτής (ή αθροιστής). Πρέπει να πάρει **αρχική τιμή το 0** πριν ξεκινήσει η επαναληπτική διαδικασία. Σε κάθε επανάληψη προσθέτει τον ακέραιο που βρίσκεται στη θέση `Π[i]`.

Ας δούμε ένα δείγμα της εκτέλεσης του αλγορίθμου για `N=4` με κάποια δοκιμαστικά δεδομένα, κάνοντας χρήσης του πίνακα των τιμών:

Π =

2	21	16	-2

Επανάληψη	Τιμές Μετρητή i	Θέση Πίνακα Π[i]	SUM
1η	1	Π[1] έχει 2	0+2 = 2
2η	2	Π[2] έχει 21	2+21 = 23
3η	3	Π[3] έχει 16	23+16 = 39
4η	4	Π[4] έχει -2	39+(-2) = 37

Με τα συγκεκριμένα δεδομένα του πίνακα τιμών, η τελική τιμή της SUM είναι 37. Μπορεί ο αναγνώστης να βάλει τα δικά του δοκιμαστικά δεδομένα και να ελέγξει κατ' αυτόν τον τρόπο την καλή λειτουργία του αλγορίθμου.

Υπολογισμός μέσου όρου (ΜΟ)

Ο αλγόριθμος αυτός διαφέρει ελάχιστα από τον προηγούμενο. Πρέπει να βρούμε πρώτα το άθροισμα των ακεραίων του πίνακα και κατόπιν να διαιρέσουμε με το πλήθος τους.

```
Αλγόριθμος ΜΟ_στοιχείων
Δεδομένα // Π[1:N] //
      SUM ← 0          !αρχικοποίηση πριν μπει στην επανάληψη
      ΜΟ ← 0
      Για  i από 1 μέχρι N
            SUM ← SUM +  Π[i]
      Τέλος_επανάληψης
      ΜΟ ← SUM / N
      Εκτύπωσε ΜΟ
Τέλος ΜΟ_στοιχείων
```

Μία μικρή λεπτομέρεια: Στις διαιρέσεις πρέπει να προσέχουμε ο παρονομαστής να μην είναι μηδενικός. Εδώ, σαφώς υποθέτουμε ότι N>0. Αλλά θα μπορούσαμε να προσθέσουμε τον εξής έλεγχο:

Αντί :

```
MO ← SUM / N
```

να γράφαμε:

```
Αν N > 0 τότε
    MO ← SUM / N
Τέλος_Αν
```

Εύρεση του μέγιστου στοιχείου

```
Αλγόριθμος Μέγιστος_στοιχείων
Δεδομένα // Π[1:N] //
    MAX ← Π[1]          !Θέτουμε σαν αρχική τιμή στην μεταβλητή MAX
                        !το πρώτο στοιχείο του πίνακα.
    Για  i από 2 μέχρι N
        Αν Π[i] > MAX  τότε
            MAX ← Π[i]
        Τέλος_Αν
    Τέλος_επανάληψης
    Εκτύπωσε MAX
Τέλος Μέγιστος_στοιχείων
```

Ως αρχική τιμή τίθεται στην μεταβλητή MAX ο ακέραιος της πρώτης θέσης του πίνακα Π[1]. Θεωρείται η ιδανικότερη αρχική τιμή σε σχέση με άλλες (π.χ. το 0).[5]

Σε κάθε επανάληψη, ελέγχουμε αν το τρέχον στοιχείο του πίνακα Π[i] είναι μεγαλύτερο από την τρέχουσα τιμή της MAX. Αν ναι, τότε θέτουμε αυτό το στοιχείο στην MAX. Έτσι, στο τέλος της επαναληπτικής διαδικασίας, η μεταβλητή MAX θα έχει την μεγαλύτερη τιμή του πίνακα.

[5] Σε κάποιες γλώσσες προγραμματισμού μπορούμε να θέσουμε ως αρχική τιμή την μεγαλύτερη που μπορεί να υποστηρίξει ο συγκεκριμένος τύπος αριθμητικών δεδομένων. Για παράδειγμα, στην Visual Basic, η μέγιστη τιμή του τύπου Integer είναι η Integer.MaxValue.

Ας δούμε ένα δείγμα της εκτέλεσης του αλγορίθμου για N=5 με κάποια δοκιμαστικά δεδομένα, κάνοντας χρήσης του πίνακα των τιμών:

Π =	2	21	-2	26	12

Επανάληψη	Τιμές Μετρητή i	Θέση Πίνακα Π[i]	MAX
		Π[1] έχει 2	γίνεται 2
1η	2	Π[2] έχει 21	γίνεται 21
2η	3	Π[3] έχει -2	παραμένει 21
3η	4	Π[4] έχει 26	γίνεται 26
4η	5	Π[5] έχει 12	παραμένει 26

Με τα συγκεκριμένα δεδομένα του πίνακα τιμών, η τελική τιμή της MAX είναι 26. Μπορεί ο αναγνώστης να βάλει τα δικά του δοκιμαστικά δεδομένα και να ελέγξει κατ' αυτόν τον τρόπο την καλή λειτουργία του αλγορίθμου.

Εύρεση του ελάχιστου στοιχείου

Παρόμοιος είναι και ο τυπικός αλγόριθμος εύρεσης του μικρότερου στοιχείου στον μονοδιάστατο πίνακα Π[1:N] :

```
Αλγόριθμος Ελάχιστος_στοιχείων
Δεδομένα // Π[1:N] //
    MIN ← Π[1]          !Θέτουμε σαν αρχική τιμή στην μεταβλητή MAX
                        !το πρώτο στοιχείο του πίνακα.
    Για  i από 2 μέχρι N
        Αν Π[i] < MIN τότε
            MIN ← Π[i]
        Τέλος_Αν
```

```
        Τέλος_επανάληψης
        Εκτύπωσε MAX
Τέλος Ελάχιστος_στοιχείων
```

Ως αρχική τιμή τίθεται στην μεταβλητή MIN ο ακέραιος της πρώτης θέσης του πίνακα Π[1]. Θεωρείται η ιδανικότερη αρχική τιμή σε σχέση με άλλες (π.χ. το 0).[6]

Σε κάθε επανάληψη, ελέγχουμε αν το τρέχον στοιχείο του πίνακα Π[i] είναι μικρότερο από την τρέχουσα τιμή της MIN. Αν ναι, τότε θέτουμε αυτό το στοιχείο στην MIN. Έτσι, στο τέλος της επαναληπτικής διαδικασίας, η μεταβλητή MIN θα έχει την μικρότερη τιμή του πίνακα.

Ας δούμε ένα δείγμα της εκτέλεσης του αλγορίθμου για N=5 με κάποια δοκιμαστικά δεδομένα, κάνοντας χρήσης του πίνακα των τιμών:

Π =

2	21	-2	26	12

Επανάληψη	Τιμές Μετρητή i	Θέση Πίνακα Π[i]	MIN
		Π[1] έχει 2	γίνεται 2
1η	2	Π[2] έχει 21	παραμένει 2
2η	3	Π[3] έχει -2	γίνεται -2
3η	4	Π[4] έχει 26	παραμένει -2
4η	5	Π[5] έχει 12	παραμένει -2

Με τα συγκεκριμένα δεδομένα του πίνακα τιμών, η τελική τιμή της MIN είναι -2. Μπορεί ο αναγνώστης να βάλει τα δικά του δοκιμαστικά δεδομένα και να ελέγξει κατ' αυτόν τον τρόπο την καλή λειτουργία του αλγορίθμου.

Οι αλγόριθμοι για το μέγιστο και ελάχιστο *θα μπορούσαν να συνδυαστούν σε έναν* που να υπολογίζει ταυτόχρονα το ελάχιστο και μέγιστο των στοιχείων του μονοδιάστατου πίνακα, χρησιμοποιώντας βέβαια τις δύο μεταβλητές MAX και MIN.

[6] Σε κάποιες γλώσσες προγραμματισμού μπορούμε να θέσουμε ως αρχική τιμή την μικρότερη που μπορεί να υποστηρίξει ο συγκεκριμένος τύπος αριθμητικών δεδομένων. Για παράδειγμα, στην Visual Basic, η μικρότερη τιμή του τύπου Integer είναι η Integer.MinValue.

Αναζήτηση στοιχείου

Η αναζήτηση στοιχείου σε πίνακα μπορεί να γίνει με διάφορους τρόπους. Εδώ, θα δούμε την **σειριακή μέθοδο**. Η λογική είναι η εξής: Σαρώνουμε[7] τον πίνακα και εξετάζουμε σε κάθε θέση αν το στοιχείο της θέσης αυτής είναι ίσο με αυτό που ψάχνουμε. Αν ναι, τότε σταματάμε.

Πότε χρησιμοποιούμε τη μέθοδο αυτή;

- Όταν ο πίνακας είναι μη ταξινομημένος
- Ο πίνακας είναι μικρού μεγέθους (π.χ. `N <=20`)
- Η αναζήτηση να πραγματοποιείται σπάνια, διότι η μέθοδος αυτή είναι σχετικά αργή.

Θα χρειαστούμε μία ***μεταβλητή*** **`βρέθηκε`** *λογικού τύπου*, που θα γίνει `Αληθής` αν το στοιχείο βρεθεί. Επίσης, δεν γνωρίζουμε σε ποιά θέση βρίσκεται. Μπορεί να είναι στην 1η θέση αλλά μπορεί να είναι στη `N`στή θέση. Θα χρησιμοποιήσουμε και μία ***μεταβλητή*** **`θέση`** που θα κρατήσει τη θέση όπου βρέθηκε. Το στοιχείο που αναζητούμε το ονομάζουμε **`key`** και εδώ είναι ένας ακέραιος.

```
Αλγόριθμος Σειριακή_αναζήτηση_στοιχείου
Δεδομένα // Π[1:N], key //
    βρέθηκε ← Ψευδής        !Αρχικοποίηση μεταβλητών.
    θέση ← 0
    i ← 1

    Όσο (i < = N) ΚΑΙ (βρέθηκε = Ψευδής) επανάλαβε !όσο δεν είμαστε στο
                                                    !τέλος του πίνακα και δεν βρέθηκε
        Αν Π[i] = key τότε      !αν το i στοιχείο ισούται με αυτό που ψάχνουμε τότε
            βρέθηκε ← Αληθής    !βρέθηκε
            θέση ← i            !βάλε τον αριθμό θέσης στην μεταβλητή θέση.
        αλλιώς                  !αλλιώς
            i ← i + 1 !προχώρα στην επόμενη θέση.
        Τέλος_αν
```

[7] Δηλαδή, διαβάζουμε ένα προς ένα τα στοιχεία του πίνακα, ξεκινώντας από την πρώτη θέση.

```
        Τέλος_επανάληψης
        Αν βρέθηκε = Αληθής τότε
                Εκτύπωσε "Το στοιχείο βρέθηκε στη θέση ", θέση
            αλλιώς
                Εκτύπωσε "Το στοιχείο δεν βρέθηκε "
        Τέλος_αν

Τέλος Σειριακή_αναζήτηση_στοιχείου
```

Ας δούμε ένα δείγμα της εκτέλεσης του αλγορίθμου για N=5 με κάποια δοκιμαστικά δεδομένα, κάνοντας χρήσης του πίνακα των τιμών:

Π =	2	21	-2	26	12

key = 26

Επανάληψη	Τιμές Μετρητή i	Θέση Πίνακα Π[i]	βρέθηκε	θέση
	0		γίνεται Ψευδής	γίνεται 0
1η	1	Π[1] έχει 2	παραμένει Ψευδής	παραμένει 0
2η	2	Π[2] έχει 21	παραμένει Ψευδής	παραμένει 0
3η	3	Π[3] έχει -2	παραμένει Ψευδής	παραμένει 0
4η	4	Π[4] έχει 26	γίνεται Αληθής	γίνεται 4
5η	5	Π[5] έχει 12	παραμένει Αληθής	παραμένει 4

Με τα συγκεκριμένα δεδομένα του πίνακα τιμών, η τελική τιμή της βρέθηκε είναι Αληθής και της θέση είναι 4. Μπορεί ο αναγνώστης να βάλει τα δικά του δοκιμαστικά δεδομένα και να ελέγξει κατ' αυτόν τον τρόπο την καλή λειτουργία του αλγορίθμου.

Όταν ο πίνακας είναι ***ταξινομημένος*** είναι προτιμότερο να χρησιμοποιείται μία πιο αποδοτική (γρήγορη) μέθοδος που ονομάζεται ***δυαδική αναζήτηση***.

Ταξινόμηση στοιχείων

Υπάρχουν αρκετοί αλγόριθμοι ταξινόμησης. Αρκετά δημοφιλής είναι ο αλγόριθμος της **ευθείας ανταλλαγής ή φυσαλίδας (Bubble Sort)**. Βασίζεται στην αρχή της σύγκρισης γειτονικών στοιχείων του πίνακα και ανταλλαγής τους μέχρι να διαταχθούν όλα σε μία σειρά (αύξουσα ή φθίνουσα).

Η λογική είναι η εξής : Κάνουμε διαδοχικές σαρώσεις στον πίνακα. Σε κάθε σάρωση, το μικρότερο στοιχείο (για αύξουσα ταξινόμηση) μετακινείται προς την κορυφή του πίνακα Αφού γίνουν όλες οι σαρώσεις θα έχει επιτευχθεί η ταξινόμηση.

```
Αλγόριθμος Φυσαλίδα_Bubble_Sort
Δεδομένα // Π[1:N] //
    Για i από 2 μέχρι N          !το i σαρώνει από τη 2η θέση έως το τέλος
        Για j από N μέχρι i με_βήμα -1 !το j σαρώνει από το τέλος προς τη θέση i
            Αν Π[j-1] > Π[j] τότε    !αν το j-1 στοιχείο είναι μεγαλύτερο από το j
                                     !στοιχείο τότε
                temp ← Π[j-1]        !κάνε αντιμετάθεση στοιχείων. Δηλαδή, το j
                Π[j-1] ← Π[j]        !στοιχείο θα πάει στη θέση του j-1 και το j-1 στη
                Π[j] ← temp          !θέση του j. Επειδή δεν μπορεί να γίνει άμεσα
                                     !διότι θα χαθεί το στοιχείο j-1, προσωρινά
                                     !τοποθετείται στη βοηθητική μεταβλητή temp
            Τέλος_αν
        Τέλος_επανάληψης
    Τέλος_επανάληψης
Τέλος Φυσαλίδα_Bubble_Sort
```

Παράδειγμα: Δίνεται ο παρακάτω μονοδιάστατος πίνακας:

Π =

12	7	5	3	10

Ζητείται, λοιπόν, να ταξινομηθούν οι ακέραιοι `12, 7, 5, 3, 10` του πίνακα. Πώς διαμορφώνεται ο πίνακας σε κάθε βήμα του αλγορίθμου;

Δείκτες		*Θέσεις πίνακα*				
i	**j**	**1**	**2**	**3**	**4**	**5**
2	*5*	*12*	*7*	*5*	3	10
	4	*12*	*7*	3	5	*10*
	3	*12*	3	*7*	*5*	*10*
	2	3	12	*7*	*5*	*10*
3	*5*	*3*	*12*	*7*	5	10
	4	*3*	*12*	5	7	*10*
	3	*3*	5	12	*7*	*10*
4	*5*	*3*	*5*	*12*	7	10
	4	*3*	*5*	7	12	*10*
5	*5*	*3*	*5*	*7*	10	12

Όπως βλέπουμε, στο *πρώτο πέρασμα του δείκτη* `j`, ο αριθμός `3`, μετά από διαδοχικές συγκρίσεις με τα γειτονικά του στοιχεία (που χρωματίζονται πορτοκαλί) ανεβαίνει, σαν τη φυσαλίδα, στην πρώτη θέση του πίνακα.

Στο *δεύτερο πέρασμα του δείκτη* `j`, ο αριθμός `5`, ανεβαίνει, σαν τη φυσαλίδα, στην δεύτερη θέση του πίνακα.

Μετά από όλα τα περάσματα, η *τελική κατάσταση του πίνακα* θα έχει διαμορφωθεί ως εξής:

`Π =`

3	5	7	10	12

Ο αλγόριθμος ταξινόμησης Bubble-Sort είναι χρήσιμος όταν τα στοιχεία του πίνακα είναι αρκετά έως πολύ αταξινόμητα. Όταν ο πίνακας είναι σχεδόν ταξινομημένος δεν θεωρείται αποδοτικός διότι κάνει πολλές συγκρίσεις με σχεδόν καθόλου μεταβολή θέσεων των στοιχείων.

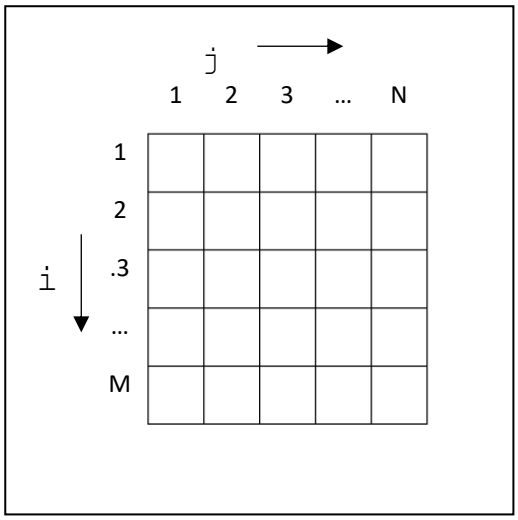
Να σημειώσουμε ότι **αν επιθυμούμε φθίνουσα ταξινόμηση** τότε αρκεί να αλλάξουμε τη συνθήκη `Π[j-1] > Π[j]` σε `Π[j-1] < Π[j]`.

Αλγόριθμοι δισδιάστατου πίνακα

Θεωρούμε ότι οι παρακάτω αλγόριθμοι επενεργούν επί ενός πίνακα `Π[1:M, 1:N]` όπου τα στοιχεία του είναι ακέραιοι αριθμοί.

Διάβασμα στοιχείων

```
Αλγόριθμος Διάβασμα_στοιχείων
Δεδομένα // Π[1:M, 1:N] //
     Για  i από 1 μέχρι M
          Για  j από 1 μέχρι N
               Διάβασε Π[i,j]
          Τέλος_επανάληψης
     Τέλος_επανάληψης
Τέλος Διάβασμα_στοιχείων
```



Ο δείκτης θέσης `i` διατρέχει τις γραμμές και ο δείκτης θέσης `j` διατρέχει τις στήλες. Λόγω του ότι η δομή επανάληψης `Για j από 1 μέχρι N` είναι εμφωλευμένη της `Για i από 1 μέχρι M`, αυτό σημαίνει ότι η **σάρωση γίνεται γραμμή προς γραμμή**, δηλαδή `Π[1,1] Π[1,2] Π[1,3] Π[1,4]… Π[1,N] Π[2,1] Π[2,2] Π[2,3] Π[2,4] .. Π[2,M] Π[3,1]` κλπ.

Εάν αντιστρέψουμε τις δομές επανάληψης τότε η **σάρωση θα γίνεται στήλη προς στήλη**, δηλαδή `Π[1,1] Π[2,1] Π[3,1] Π[4,1]… Π[M,1] Π[1,2] Π[2,2] Π[3,2] Π[4,2] .. Π[M,2] Π[1,3]` κλπ.

Το πώς θα γίνει η σάρωση του δισδιάστατου πίνακα εξαρτάται από το είδος του προβλήματος που καλούμαστε να επιλύσουμε. Σε πολλές περιπτώσεις, εφόσον δεν έχει σημασία η σειρά σάρωσης, προτιμούμε τον παραπάνω αλγόριθμο, δηλαδή τη σάρωση *γραμμή προς γραμμή*.

Εκτύπωση στοιχείων

```
Αλγόριθμος Εκτύπωση_στοιχείων
Δεδομένα // Π[1:M, 1:N] //
     Για  i από 1 μέχρι M
          Για  j από 1 μέχρι N
               Εκτύπωσε Π[i,j]
          Τέλος_επανάληψης
     Τέλος_επανάληψης
Τέλος Εκτύπωση_στοιχείων
```

Ισχύουν, φυσικά, όσα αναφέραμε στον προηγούμενο αλγόριθμο του διαβάσματος για τη σειρά σάρωσης του πίνακα.

Υπολογισμός συνολικού αθροίσματος

Με άλλα λόγια, πρέπει να αθροίσουμε όλους τους ακέραιους που περιέχει ο δισδιάστατος πίνακας. Όπως γνωρίζουμε ήδη, θα χρειαστούμε μία **μεταβλητή αθροιστή ή συσσωρευτή SUM** για να κρατάει το τρέχον άθροισμα, η οποία αρχικά έχει 0.

```
Αλγόριθμος Άθροισμα_στοιχείων
Δεδομένα // Π[1:M, 1:N] //
     SUM ← 0        !αρχικοποίηση πριν μπει στην επανάληψη
     Για  i από 1 μέχρι M
          Για  j από 1 μέχρι N
               SUM ← SUM +  Π[i,j]
          Τέλος_επανάληψης
     Τέλος_επανάληψης
     Εκτύπωσε SUM
```

Τέλος Άθροισμα_στοιχείων

Ας δούμε ένα δείγμα της εκτέλεσης του αλγορίθμου για M=2 και N=4 με κάποια δοκιμαστικά δεδομένα, κάνοντας χρήσης του πίνακα των τιμών:

Π =

2	21	6	10
4	5	3	8

Επανάληψη	i	j	Θέση Πίνακα Π[i,j]	SUM
				0
1η	1	1	Π[1,1] έχει 2	0+2 = 2
2η	1	2	Π[1,2] έχει 21	2+21 = 23
3η	1	3	Π[1,3] έχει 6	23+6 = 29
4η	1	4	Π[1,4] έχει 10	29+10 = 39
5η	2	1	Π[2,1] έχει 4	39+4 = 43
6η	2	2	Π[2,2] έχει 5	43+5 = 48
7η	2	3	Π[2,3] έχει 3	48+3 = 51
8η	2	4	Π[2,4] έχει 8	51+8 = 59

Με τα συγκεκριμένα δεδομένα του πίνακα τιμών, η τελική τιμή της SUM είναι 59. Μπορεί ο αναγνώστης να βάλει τα δικά του δοκιμαστικά δεδομένα και να ελέγξει κατ' αυτόν τον τρόπο την καλή λειτουργία του αλγορίθμου.

Υπολογισμός μέσου όρου (ΜΟ)

Ο αλγόριθμος αυτός διαφέρει ελάχιστα από τον προηγούμενο. Πρέπει να βρούμε πρώτα το άθροισμα των ακεραίων του πίνακα και κατόπιν να διαιρέσουμε με το πλήθος τους.

Αλγόριθμος ΜΟ_στοιχείων

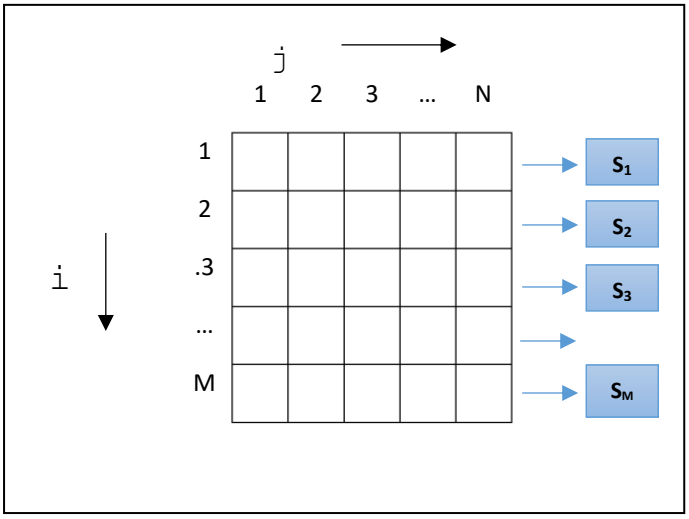
Δεδομένα // Π[1:M, 1:N] //

```
      SUM ← 0          !αρχικοποίηση πριν μπει στην επανάληψη
      Για  i από 1 μέχρι M
            Για  j από 1 μέχρι N
                  SUM ← SUM +  Π[i,j]
            Τέλος_επανάληψης
      Τέλος_επανάληψης
      MO ← SUM / (M * N)
      Εκτύπωσε MO
Τέλος MO_στοιχείων
```

Στο παράρτημα, περιγράφουμε τον ίδιο αλγόριθμο με διάγραμμα ροής.

Υπολογισμός αθροίσματος ανά γραμμή

Εδώ το ζητούμενο είναι να υπολογίζεται και εκτυπώνεται το **άθροισμα σε κάθε γραμμή** που σαρώνεται:

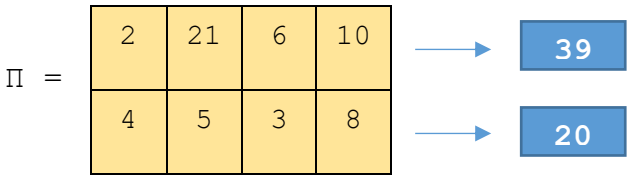



```
Αλγόριθμος Άθροισμα_στοιχείων_ανά_γραμμή
Δεδομένα // Π[1:M, 1:N] //
      Για  i από 1 μέχρι M
            SUM ← 0      !αρχικοποίηση πριν μπει στην επανάληψη του j
```

```
        Για  j από 1 μέχρι N
             SUM ← SUM +  Π[i,j]
        Τέλος_επανάληψης
        Εκτύπωσε SUM      !αφορά το άθροισμα της γραμμής
    Τέλος_επανάληψης
Τέλος Άθροισμα_στοιχείων_ανά_γραμμή
```

Μόλις ολοκληρωθεί η σάρωση (μέσω του μετρητή `j`) των θέσεων της πρώτης γραμμής τυπώνουμε το άθροισμα `SUM`. Στη συνέχεια, ο μετρητής `i` αυξάνεται και πάει στην επόμενη γραμμή. Πριν ξεκινήσει η σάρωση των θέσεων της δεύτερης γραμμής, η μεταβλητή `SUM` μηδενίζεται ώστε να ετοιμαστεί για τον υπολογισμό του αθροίσματος της δεύτερης γραμμής κ.ο.κ.

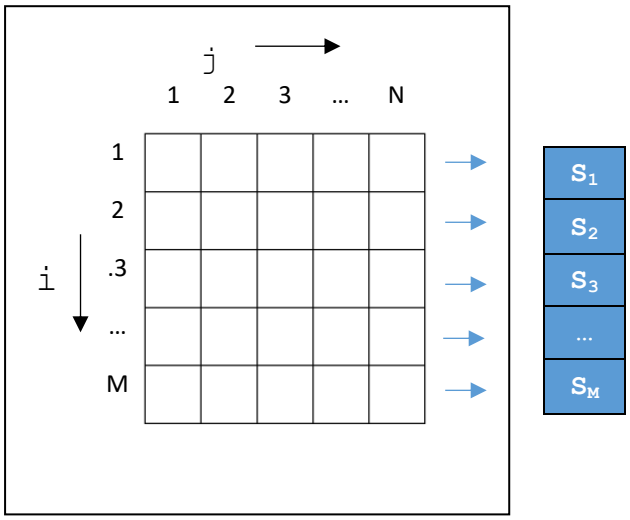
Ας δούμε ένα δείγμα της εκτέλεσης του αλγορίθμου για `M=2` και `N=4` με κάποια δοκιμαστικά δεδομένα, κάνοντας χρήσης του πίνακα των τιμών:



Επανάληψη	i	j	Θέση Πίνακα Π[i,j]	SUM
				0
1η	1	1	Π[1,1] έχει 2	0+2 = 2
2η	1	2	Π[1,2] έχει 21	2+21 = 23
3η	1	3	Π[1,3] έχει 6	23+6 = 29
4η	1	4	Π[1,4] έχει 10	29+10 = **39**
5η	2	4		0
5η	2	1	Π[2,1] έχει 4	0+4 = 4
6η	2	2	Π[2,2] έχει 5	4+5 = 9
7η	2	3	Π[2,3] έχει 3	9+3 = 12
8η	2	4	Π[2,4] έχει 8	12+8 = **20**

Διάφορες **παραλλαγές** μπορούμε να έχουμε σε αυτόν τον αλγόριθμο, όπως:

- Να υπολογιστεί το **άθροισμα ανά στήλη**.
- Να **μην χάνονται τα αθροίσματα αλλά να αποθηκεύονται σε έναν μονοδιάστατο πίνακα**:



Στη συνέχεια, μπορεί να ζητείται επιπλέον επεξεργασία του προκύψαντα μονοδιάστατου πίνακα S (π.χ. εύρεση του μέγιστου κλπ.). Παρακάτω, θα δούμε έναν σχετικό αλγόριθμο με τον μέσο όρο ανά γραμμή.

Υπολογισμός μέσου όρου (ΜΟ) ανά γραμμή

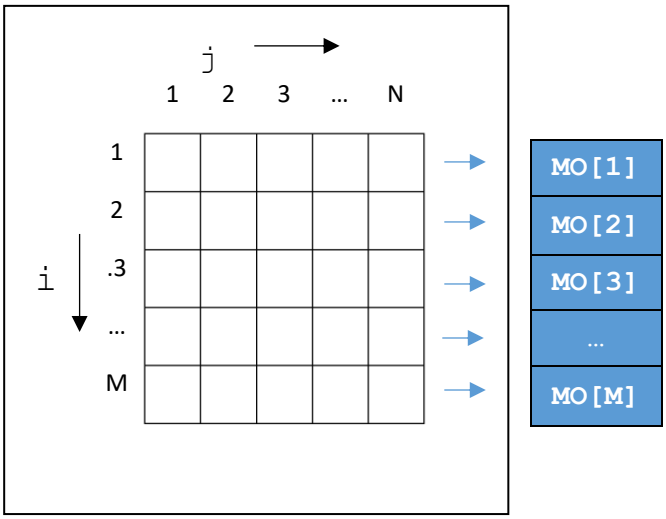
Ο αλγόριθμος αυτός διαφέρει ελάχιστα από τον προηγούμενο. Πρέπει να βρούμε πρώτα το άθροισμα των ακεραίων ανά γραμμή του πίνακα και κατόπιν να διαιρέσουμε με το πλήθος τους.

```
Αλγόριθμος ΜΟ_στοιχείων_ανά_γραμμή
Δεδομένα // Π[1:M, 1:N] //
    Για  i από 1 μέχρι M
         SUM ← 0      !αρχικοποίηση πριν μπει στην επανάληψη του j
         Για  j από 1 μέχρι N
              SUM ← SUM +  Π[i,j]
         Τέλος_επανάληψης
```

```
        ΜΟ ← SUM / N    !αφορά το ΜΟ της γραμμής
        Εκτύπωσε ΜΟ
    Τέλος_επανάληψης
Τέλος ΜΟ_στοιχείων_ανά_γραμμή
```

Υπολογισμός μέσου όρου (ΜΟ) ανά γραμμή και αποθήκευσή του σε ξεχωριστό πίνακα

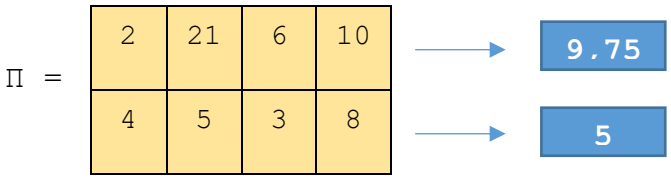
Εδώ, επιθυμούμε να αποθηκεύεται ξεχωριστά σε έναν πίνακα ο ΜΟ της γραμμής. Δηλαδή, μόλις υπολογίσουμε τον ΜΟ της πρώτης γραμμής να τον καταχωρήσουμε στην πρώτη θέση ενός μονοδιάστατου πίνακα μέσων όρων. Ακολούθως, για τον ΜΟ της δεύτερης γραμμής κ.ο.κ.




```
Αλγόριθμος ΜΟ_στοιχείων_ανά_γραμμή_και_αποθήκευση
Δεδομένα // Π[1:M, 1:N], ΜΟ[1:M] //
    Για  i από 1 μέχρι M
        Μ[i] ← 0     !αρχικοποίηση πριν μπει στην επανάληψη του j
        SUM ← 0
        Για  j από 1 μέχρι N      !πρώτα το άθροισμα της γραμμής
            SUM ← SUM +  Π[i,j]
```

```
        Τέλος_επανάληψης
        ΜΟ[i] ← SUM / N        !αφορά το ΜΟ της γραμμής i
        Εκτύπωσε ΜΟ[i]
    Τέλος_επανάληψης
Τέλος ΜΟ_στοιχείων_ανά_γραμμή_και_αποθήκευση
```

Ας δούμε ένα δείγμα της εκτέλεσης του αλγορίθμου για Μ=2 και Ν=4 με κάποια δοκιμαστικά δεδομένα, κάνοντας χρήσης του πίνακα των τιμών:



Επανάληψη	i	j	Θέση Πίνακα Π[i,j]	SUM	Θέση πίνακα Μ[i]
1η	1			0	Μ[1] γίνεται 0
1η	1	1	Π[1,1] έχει 2	0+2 = 2	Μ[1] παραμένει 0
2η	1	2	Π[1,2] έχει 21	2+21 = 23	Μ[1] παραμένει 0
3η	1	3	Π[1,3] έχει 6	23+6 = 29	Μ[1] παραμένει 0
4η	1	4	Π[1,4] έχει 10	29+10 = 39	**Μ[1]** γίνεται 39/4 = **9,75**
5η	2	4		0	Μ[2] γίνεται 0
5η	2	1	Π[2,1] έχει 4	0+4 = 4	Μ[2] παραμένει 0
6η	2	2	Π[2,2] έχει 5	4+5 = 9	Μ[2] παραμένει 0
7η	2	3	Π[2,3] έχει 3	9+3 = 12	Μ[2] παραμένει 0
8η	2	4	Π[2,4] έχει 8	12+8 = 20	**Μ[2]** γίνεται 20/4 = **5**

Με τα συγκεκριμένα δεδομένα του πίνακα τιμών, η τελική κατάσταση του πίνακα ΜΟ φαίνεται παραδίπλα:

ΜΟ
9,75
5

Μπορεί ο αναγνώστης να βάλει τα δικά του δοκιμαστικά δεδομένα και να ελέγξει κατ' αυτόν τον τρόπο την καλή λειτουργία του αλγορίθμου.

Εύρεση του μέγιστου στοιχείου

```
Αλγόριθμος Μέγιστος_στοιχείων
Δεδομένα // Π[1:M, 1:N] //
    MAX ← Π[1,1]        !Θέτουμε σαν αρχική τιμή στην μεταβλητή MAX
                        !το πρώτο στοιχείο του πίνακα.
    Για  i από 2 μέχρι M
        Για  j από 1 μέχρι N
            Αν Π[i,j] > MAX  τότε
                MAX ← Π[i,j]
            Τέλος_Αν
        Τέλος_επανάληψης
    Τέλος_επανάληψης
    Εκτύπωσε MAX
Τέλος Μέγιστος_στοιχείων
```

Ας δούμε ένα δείγμα της εκτέλεσης του αλγορίθμου για M=2 και N=4 με κάποια δοκιμαστικά δεδομένα, κάνοντας χρήσης του πίνακα των τιμών:

Π =

2	21	6	10
21	5	32	8

Επανάληψη	i	j	Θέση Πίνακα Π[i,j]	MAX
			Π[1,1] έχει 2	γίνεται 2
1η	1	2	Π[1,2] έχει 21	γίνεται 21
2η	1	3	Π[1,3] έχει 6	παραμένει 21
3η	1	4	Π[1,4] έχει 10	παραμένει 21
5η	2	1	Π[2,1] έχει 21	παραμένει 21
5η	2	2	Π[2,2] έχει 5	παραμένει 21
6η	2	3	Π[2,3] έχει 32	γίνεται 32
7η	2	3	Π[2,4] έχει 8	παραμένει **32**

Με τα συγκεκριμένα δεδομένα του πίνακα τιμών, η τελική τιμή της MAX είναι 32. Μπορεί ο αναγνώστης να βάλει τα δικά του δοκιμαστικά δεδομένα και να ελέγξει κατ' αυτόν τον τρόπο την καλή λειτουργία του αλγορίθμου.

Εύρεση του ελάχιστου στοιχείου

Παρόμοιος είναι και ο αλγόριθμος για την εύρεση του ελάχιστου στοιχείου:

```
Αλγόριθμος Ελάχιστος_στοιχείων
Δεδομένα // Π[1:M, 1:N] //
    MIN ← Π[1,1]        !Θέτουμε σαν αρχική τιμή στην μεταβλητή MAX
                        !το πρώτο στοιχείο του πίνακα.
    Για  i από 2 μέχρι M
        Για  j από 1 μέχρι N
            Αν Π[i,j] < MIN τότε
                MIN ← Π[i,j]
            Τέλος_Αν
        Τέλος_επανάληψης
    Τέλος_επανάληψης
```

```
        Εκτύπωσε MIN
Τέλος Ελάχιστος_στοιχείων
```

Όπως για τον υπολογισμό για το άθροισμα και μέσο όρο *ανά γραμμή* (*ή στήλη*) έτσι κι εδώ, **μπορούμε να βρούμε τον μέγιστο και ελάχιστο *ανά γραμμή*** (ή *στήλη*, αντίστοιχα). Οι αλγόριθμοι είναι παρόμοιοι αφού βασίζονται στην ίδια λογική, όπως περιγράψαμε στα σχετικά εδάφια.

Αναζήτηση στοιχείου

Όπως και στον μονοδιάστατο πίνακα, η αναζήτηση στοιχείου σε πίνακα μπορεί να γίνει με διάφορους τρόπους. Εδώ, θα δούμε την **σειριακή μέθοδο, η οποία είναι ίδιας λογικής**: Σαρώνουμε[8] τον πίνακα και εξετάζουμε σε κάθε θέση αν το στοιχείο της θέσης αυτής είναι ίσο με αυτό που ψάχνουμε. Αν ναι, τότε σταματάμε.

Εδώ, επειδή η θέση προσδιορίζεται από δύο δείκτες, τον `i` και `j`, θα χρειαστούμε δύο αντίστοιχες μεταβλητές για να κρατήσουμε σε ποιά θέση βρέθηκε το στοιχείο `key`.

```
Αλγόριθμος Σειριακή_αναζήτηση_στοιχείου
Δεδομένα // Π[1:M, Π[1:N], key //
        βρέθηκε ← Ψευδής          !Αρχικοποίηση μεταβλητών.
        θέση_i ← 0                !σε ποιά γραμμή βρέθηκε.
        θέση_j ← 0                !σε ποιά στήλη βρέθηκε.
        i ← 1

        Όσο (i <= M) ΚΑΙ (βρέθηκε = Ψευδής) επανάλαβε !όσο δεν είμαστε στο
                                                       !τέλος των γραμμών και δεν βρέθηκε
                j ← 1
                Όσο (j <= N) ΚΑΙ (βρέθηκε = Ψευδής) επανάλαβε !όσο δεν είμαστε
                                                               !στο τέλος των θέσεων της γραμμής και δεν βρέθηκε
```

[8] Δηλαδή, διαβάζουμε ένα προς ένα τα στοιχεία του πίνακα, ξεκινώντας από την πρώτη θέση.

```
            Αν Π[i,j] = key τότε      !αν το i στοιχείο ισούται με αυτό που
                                      !ψάχνουμε τότε
                βρέθηκε ← Αληθής      !βρέθηκε
                θέση_i ← i            !κράτησε τον δείκτη γραμμής .
                θέση_j ← j            !και τον δείκτη στήλης όπου βρέθηκε .
            αλλιώς                    !αλλιώς
                j ← j + 1             !προχώρα στην επόμενη θέση της γραμμής..
            Τέλος_αν
        Τέλος_επανάληψης
        i ← i + 1                     !προχώρα στην επόμενη γραμμή.
    Τέλος_επανάληψης
    Αν βρέθηκε = Αληθής τότε
            Εκτύπωσε "Το στοιχείο βρέθηκε στη θέση ",
                                        θέση_i, θέση_j
        αλλιώς
            Εκτύπωσε "Το στοιχείο δεν βρέθηκε "
    Τέλος_αν

Τέλος Σειριακή_αναζήτηση_στοιχείου
```

Ο αλγόριθμος μοιάζει με αυτόν του μονοδιάστατου πίνακα με τη διαφορά ότι εδώ έχουμε δύο δομές επανάληψης **Όσο..επανάλαβε** λόγω των δύο διαστάσεων του πίνακα. Η **εξωτερική δομή επανάληψης** διατρέχει τις γραμμές του πίνακα μέσω του **δείκτη i** και η **εσωτερική (εμφωλευμένη) δομή επανάληψης** διατρέχει τις στήλες(ή κατά μία άλλη άποψη τις θέσεις της συγκεκριμένης γραμμής) μέσω του **δείκτη j**.

Σε κάθε περίπτωση, όταν βρεθεί το στοιχείο key οι επαναλήψεις σταματούν αμέσως (επειδή οι λογικές συνθήκες ελέγχου αποτιμώνται Ψευδής).

Παραλλαγή: **Μία παραλλαγή** θα ήταν να μην σταματάει ο αλγόριθμος αμέσως όταν το στοιχείο βρεθεί αλλά να συνεχίσει μέχρι το τέλος του πίνακα και να μας εμφανίσει ***πόσες φορές υπάρχει το στοιχείο key***.

```
Αλγόριθμος Σειριακή_αναζήτηση_στοιχείου_v2
Δεδομένα // Π[1:M, Π[1:N], key //
```

```
βρέθηκε ← Ψευδής          !Αρχικοποίηση μεταβλητών.
φορές ← 0                 !πόσες φορές βρέθηκε το key.
i ← 1

Όσο (i < = M) επανάλαβε          !όσο δεν είμαστε στο τέλος των γραμμών
    j ← 1
    Όσο (j < = N) επανάλαβε      !όσο δεν είμαστε
                                 !στο τέλος των θέσεων της γραμμής
        Αν Π[i,j] = key τότε     !αν το i στοιχείο ισούται με αυτό που
                                 !ψάχνουμε τότε
            βρέθηκε ← Αληθής     !βρέθηκε
            φορές ← φορές +1
            Εκτύπωσε "Το στοιχείο βρέθηκε στη θέση ",
                                 i, j
                                 !πού βρέθηκε.
        αλλιώς                   !αλλιώς
            j ← j + 1            !προχώρα στην επόμενη θέση της γραμμής..
        Τέλος_αν
    Τέλος_επανάληψης
    i ← i + 1                    !προχώρα στην επόμενη γραμμή.
Τέλος_επανάληψης
Αν βρέθηκε = Αληθής τότε
        Εκτύπωσε "Αριθμός φορών που το στοιχείο βρέθηκε:
                                 ", φορές
    αλλιώς
        Εκτύπωσε "Το στοιχείο δεν βρέθηκε"
Τέλος_αν

Τέλος Σειριακή_αναζήτηση_στοιχείου_v2
```

Όπως παρατηρούμε, στις λογικές συνθήκες των δομών επανάληψης, δεν χρειάζεται έλεγχος της μεταβλητής βρέθηκε αφού ο πίνακας θα σαρωθεί όλος. Επίσης, θα ήταν προτιμότερο να χρησιμοποιηθεί η δομή επανάληψης **Για..από..μέχρι**.

Πλεονεκτήματα / Μειονεκτήματα των πινάκων

Πλεονεκτήματα	Μειονεκτήματα
▪ Είναι ένας **βολικός κι εύκολος τρόπος διαχείρισης δεδομένων του *ιδίου τύπου*** *(π.χ. όλοι ακέραιοι)*	▪ **Απαιτούν μνήμη**. Ο πίνακας δεσμεύει από την αρχή του προγράμματος αρκετές θέσεις μνήμης. Έτσι, αν χρησιμοποιούμε πολλούς πίνακες σε ένα πρόγραμμα το επιβαρύνουμε όσον αφορά τη μνήμη.
	▪ **Περιορίζουμε τις δυνατότητες του προγράμματος**. Επειδή ο πίνακας είναι στατική δομή, δεν μπορεί να αλλάξει το μέγεθός του κατά την εκτέλεση του προγράμματος.

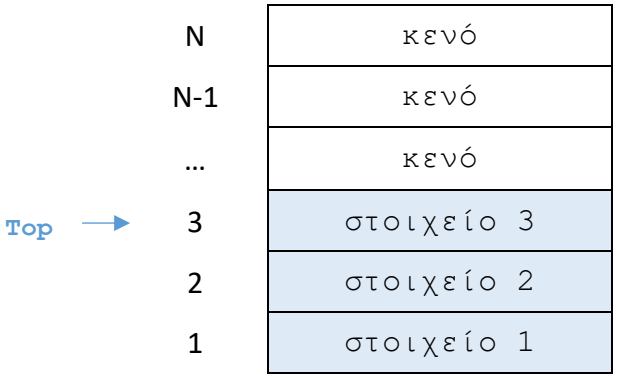
Πότε είναι προτιμότερο να χρησιμοποιούμε του πίνακες; Όταν τα δεδομένα που εισάγουμε, απαιτείται να βρίσκονται στη μνήμη RAM καθ' όλη τη διάρκεια εκτέλεσης του προγράμματος.

Στοίβα

Η στοίβα υλοποιεί τη **λογική LIFO** (**Last In First Out**). Τα δεδομένα εισάγονται στην κορυφή της στοίβας ενώ η αφαίρεση ενός στοιχείου γίνεται πάντα από την κορυφή της στοίβας *(όπως συμβαίνει σε μία στοίβα από πιάτα. Τοποθετούμε ένα νέο πιάτο στην κορυφή και αφαιρούμε ένα πιάτο πάλι από την κορυφή*).

Η στατική στοίβα[9] υλοποιείται με μονοδιάστατο πίνακα και χρησιμοποιεί ένα ***δείκτη Top που δείχνει στην κορυφή της στοίβας***.

Γενικά, μπορούμε να αναπαραστήσουμε τη στοίβα με N στοιχεία, όπου ο **δείκτης Top** δείχνει, για παράδειγμα, στη θέση **3**, ως εξής (*οπτική κατακόρυφη)*:



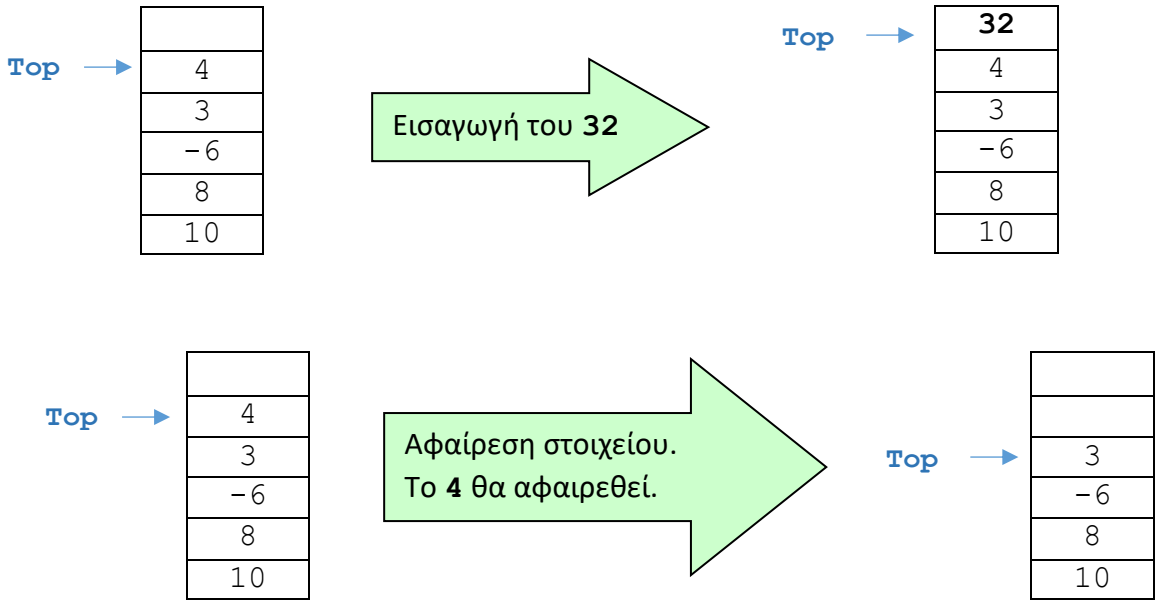
Το γεγονός ότι ο **δείκτης Top** δείχνει στη θέση **3** σημαίνει ότι η στοίβα είναι γεμάτη μέχρι εκεί. Οι πιο πάνω θέσεις είναι κενές.

Οι **λειτουργίες της στοίβας** είναι δύο:

- Η ***εισαγωγή-ώθηση (Push)*** ενός στοιχείου στην κορυφή της στοίβας.
- Η ***αφαίρεση-απώθηση (Pop)*** ενός στοιχείου από την κορυφή της στοίβας.

[9] Με τον όρο *στατική στοίβα* εννοούμε ότι έχει *προκαθορισμένο σταθερό μέγεθος* και ως εκ τούτου υλοποιείται με μονοδιάστατο πίνακα. Η *δυναμική στοίβα* υλοποιείται με μία άλλη δομή δεδομένων που ονομάζεται λίστα και το μέγεθός της μεταβάλλεται δυναμικά.

Παράδειγμα (η στοίβα περιέχει ακέραιους):



Υπερχείλιση - Υποχείλιση

- Κατά την εισαγωγή ενός στοιχείου (ώθηση), πρέπει να γίνεται έλεγχος μήπως η στοίβα είναι γεμάτη (δηλαδή δεν υπάρχει άλλη ελεύθερη θέση στον πίνακα). Δηλαδή, να ελέγχει μήπως συμβεί ***υπερχείλιση (overflow).***

- Αντίστοιχα, κατά την αφαίρεση ενός στοιχείου (απώθηση), πρέπει να γίνεται έλεγχος μήπως η στοίβα είναι άδεια. Δηλαδή, να ελέγχει μήπως συμβεί ***υποχείλιση (underflow).***

Οι αλγόριθμοι για τις λειτουργίες **push** και **pop** είναι πολύ απλοί:

```
Αλγόριθμος Push_item
Δεδομένα // Π[1:N], Top, item //
    Αν Top + 1 > N τότε
        Εκτύπωσε "Δεν μπορεί να εισαχθεί το στοιχείο item. Θα
                                        γίνει υπερχείλιση"
    αλλιώς
        Top ← Top + 1
        Π[Top] ← item
    Τέλος_Αν
Τέλος Push_item
```

```
Αλγόριθμος Pop_item
Δεδομένα // Π[1:N], Top //
     Αν Top - 1 < 0 τότε
          Εκτύπωσε "Δεν μπορεί να εξαχθεί στοιχείο. Θα
                                        γίνει υποχείλιση"
     αλλιώς
          Item ← Π[Top]
          Top ← Top - 1
     Τέλος_Αν
Αποτελέσματα // item
Τέλος Pop_item
```

Μερικά παραδείγματα πρακτικών εφαρμογών της λογικής LIFO είναι:

- Το **ιστορικό στο πρόγραμμα πλοήγησης στο Διαδίκτυο**. Κάθε φορά που πατάμε το κουμπί `Back` μεταβαίνουμε στην τελευταία σελίδα που επισκεφθήκαμε (κι όχι στην πρώτη).
- Οι **αναιρέσεις (Undo) σε ένα πρόγραμμα** π.χ. επεξεργασίας κειμένου.
- Στην **υλοποίηση των αναδρομικών διαδικασιών και συναρτήσεων** που χρησιμοποιούνται στις γλώσσες προγραμματισμού.

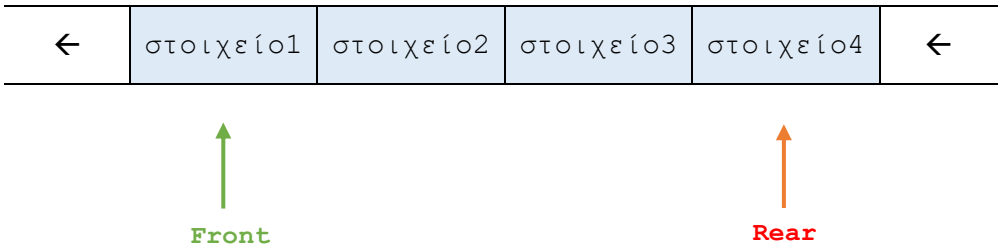
Ουρά

Η ουρά υλοποιεί τη **λογική FIFO** (First In First Out). Τα δεδομένα εισάγονται στο πίσω μέρος της ουράς ενώ η εξαγωγή ενός στοιχείου γίνεται πάντα από το μπροστινό μέρος της ουράς (όπως συμβαίνει σε μία ουρά σε τράπεζα!).

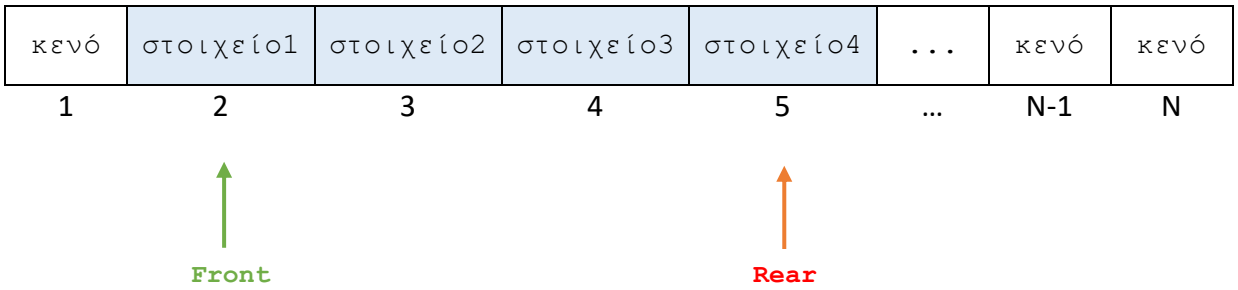
Η στατική ουρά [10]υλοποιείται με μονοδιάστατο πίνακα και χρησιμοποιεί **δύο δείκτες** :

- Τον **Front** που δείχνει στο μπροστινό μέρος της ουράς και
- τον **Rear** που δείχνει στο πίσω μέρος της ουράς.

Η **γενική μορφή μίας ουράς**, που περιέχει για παράδειγμα 4 στοιχεία, είναι η εξής:



Γενικά, μπορούμε **να αναπαραστήσουμε την ουρά με N στοιχεία σε μονοδιάστατο πίνακα**, όπου ο δείκτης **Front** δείχνει, για παράδειγμα, στη θέση 2, και ο δείκτης **Rear**, για παράδειγμα, στη θέση 5, ως εξής (*οπτική οριζόντια):*

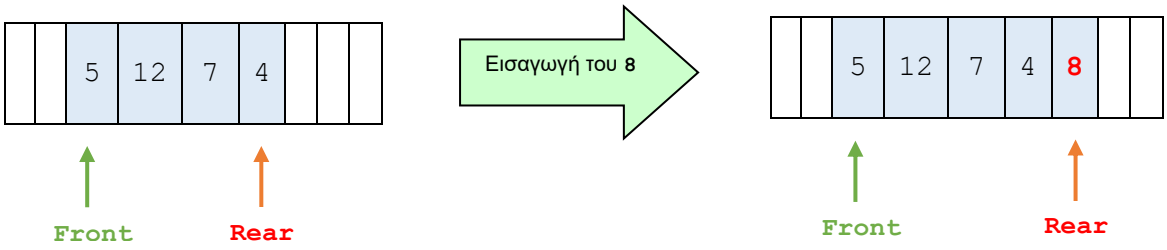


Στο παραπάνω σχήμα, οι σκιασμένες θέσεις του πίνακα υποδηλώνουν ότι εκεί υπάρχει περιεχόμενο. Με άλλα λόγια, η ουρά περιέχει τόσα στοιχεία όσα βρίσκονται μεταξύ των δεικτών **Front** και **Rear**.

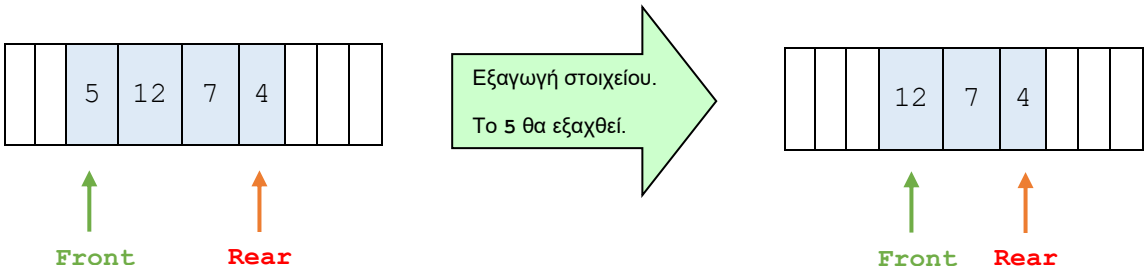
Οι **λειτουργίες της ουράς** είναι δύο :

- Η ***εισαγωγή*** ενός στοιχείου γίνεται στο πίσω μέρος της ουράς.
- Η ***εξαγωγή*** ενός στοιχείου γίνεται από το μπροστινό μέρος της ουράς.

Παράδειγμα (η ουρά περιέχει ακέραιους):



[10] Όπως και με τη στοίβα, υπάρχει και η *δυναμική ουρά* που υλοποιείται με *λίστα*, το μέγεθος της οποίας μεταβάλλεται δυναμικά.



Διαθεσιμότητα – Άδεια ουρά

- **Κατά την εισαγωγή ενός στοιχείου, πρέπει να γίνεται έλεγχος μήπως η ουρά δεν έχει διαθέσιμο χώρο να επεκταθεί** *(δηλαδή, δεν υπάρχουν ελεύθερες θέσεις στο πίσω μέρος του πίνακα***)**. Από τεχνικής άποψης, αυτό συμβαίνει όταν ο δείκτης `Rear` ξεπεράσει το μέγεθος του πίνακα.

- **Κατά την εξαγωγή ενός στοιχείου, πρέπει να γίνεται έλεγχος μήπως η ουρά είναι άδεια**. Από τεχνικής άποψης, αυτό συμβαίνει όταν ο δείκτης `Front` ξεπεράσει τον `Rear`.

Όπως και με τη στοίβα, οι αλγόριθμοι για τις λειτουργίες **εισαγωγής** και **εξαγωγής** είναι πολύ απλοί:

```
Αλγόριθμος Εισαγωγή_item
Δεδομένα // Π[1:N], Rear, item //
    Αν Rear = N τότε    !ελέγχουμε αν υπάρχουν διαθέσιμες θέσεις,.
        Εκτύπωσε "Δεν υπάρχουν διαθέσιμες θέσεις!"
    αλλιώς
        Rear ← Rear + 1
        Π[Rear] ← item
    Τέλος_Αν
Τέλος Εισαγωγή_item

Αλγόριθμος Εξαγωγή_item
Δεδομένα // Π[1:N], Front, Rear //
```

```
    Αν Front > Rear τότε    ! ελέγχουμε αν ο Front έχει ξεπεράσει τον Rear.
        Εκτύπωσε "Η ουρά είναι άδεια!"
    αλλιώς
        item ← Π[Front]
        Front ← Front + 1
    Τέλος_Αν
Αποτελέσματα // item
Τέλος Εξαγωγή_item
```

Δομές δεδομένων δευτερεύουσας μνήμης

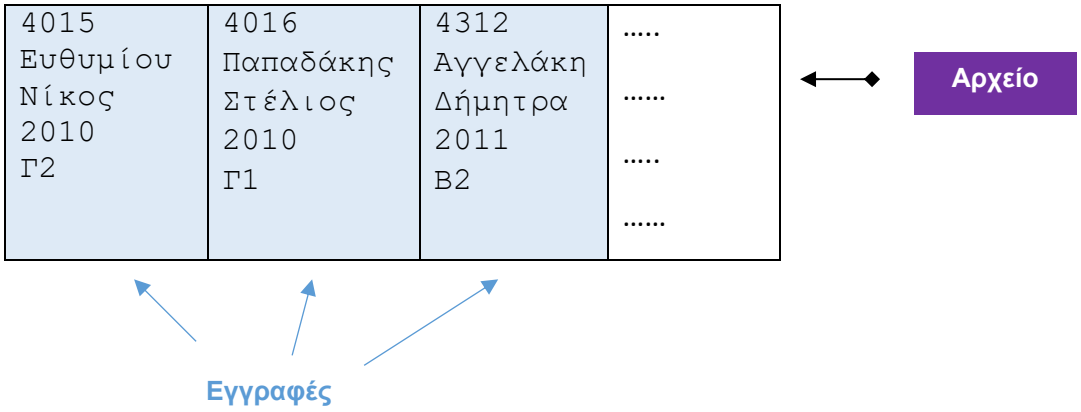
Επειδή η RAM δεν επαρκεί για την αποθήκευση των δεδομένων καθώς και το ότι δεν αποθηκεύει μόνιμα, χρησιμοποιούμε τους **μαγνητικούς δίσκους** (ταινίες, σκληροί δίσκοι κ.α.), **οπτικούς δίσκους** (CD-ROM, DVD-ROM κ.α.) και **μνήμες Flash** (USB sticks, κάρτες μνήμης, SSD). Αυτά αποτελούν την **περιφερειακή ή δευτερεύουσα μνήμη**[11].

Για την αποθήκευση δεδομένων σε αυτή κάνουμε χρήση **ειδικών δομών** που λέγονται ***αρχεία (files)*.**

- Κάθε αρχείο περιέχει μία ***συλλογή εγγραφών*** (π.χ. ένα αρχείο με εγγραφές μαθητών).
- Μία ***εγγραφή (Record)*** αποτελεί τη συλλογή στοιχειωδών πληροφοριών σχετικά με ένα αντικείμενο (π.χ. μαθητής, ένα βιβλίο κ.λπ.).
- Μία στοιχειώδης πληροφορία συνιστά ένα ***πεδίο (field).*** Π.χ. μία εγγραφή μαθητή περιέχει στοιχειώδεις πληροφορίες (πεδία) όπως : Επώνυμο, Όνομα, Πατρώνυμο, Έτος γέννησης, Τάξη κλπ.

Στο παρακάτω σχήμα, βλέπουμε ένα παράδειγμα αρχείου που περιέχει εγγραφές μαθητών με πεδία `Αριθμός Μητρώου`, `Επώνυμο`, `Όνομα`, `Έτος γέννησης` και `Τμήμα`:

[11] Επίσης, καλείται και ως *βοηθητική μνήμη*.



Μερικές παρατηρήσεις:

- Συνήθως **ένα ή δύο πεδία μαζί ταυτοποιούν την εγγραφή**, **δηλαδή την κάνουν μοναδική στο σύνολο των εγγραφών**. Στην προκειμένη περίπτωση ο **Αριθμός Μητρώου** είναι ένα πεδίο που ταυτοποιεί μοναδικά μία εγγραφή μαθητή (δηλαδή, δεν πρέπει δύο εγγραφές να έχουν τον ίδιο αριθμό μητρώου). Έτσι, το πεδίο αυτό ονομάζεται **πρωτεύον κλειδί (primary key[12])**.

- **Και άλλα πεδία μπορούν να χρησιμοποιηθούν για να ταυτοποιήσουν μία εγγραφή, αλλά όχι μοναδικά**. Για παράδειγμα, το πεδίο `Επώνυμο` μπορεί να χρησιμοποιηθεί για ταυτοποίηση αλλά δύο μαθητές ενδέχεται να έχουν τον ίδιο επώνυμο. Στην περίπτωση αυτή, το πεδίο `Επώνυμο` ονομάζεται δευτερεύον κλειδί (secondary key).[13]

- **Τα κλειδιά, πρωτεύοντα ή δευτερεύοντα, σε διάταξη αύξουσα ή φθίνουσα, διευκολύνουν στην ταχύτατη αναζήτηση μίας εγγραφής.** Βάσει αυτών, δημιουργούνται ειδικές δομές που λέγονται ***ευρετήρια***, τα οποία χρησιμεύουν όπως το ευρετήριο στο τέλος ενός βιβλίου: η λέξη δίπλα έχει τη σελίδα/σελίδες στις οποίες υπάρχει και βοηθάει να μεταβούμε κατευθείαν εκεί.

[12] Το πρωτεύον κλειδί μπορεί να αποτελεί *συνδυασμό δύο ή περισσοτέρων πεδίων*, αν χρειάζεται, προκειμένου να ταυτοποιήσει μοναδικά μία εγγραφή.
[13] Προφανώς, υπονοείται ότι έχει οριστεί πρωτεύον κλειδί, κάτι που θεωρείται απαραίτητο στο σχεδιασμό των αρχείων και ιδιαίτερα των Βάσεων Δεδομένων.

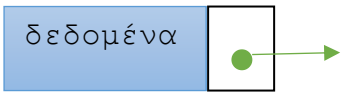
Άλλες δομές δεδομένων

Στις προηγούμενες δομές που εξετάσαμε (Πίνακας, Στοίβα και Ουρά), οι κόμβοι της δομής αποθηκεύονται σε συνεχόμενες θέσεις στην κύρια μνήμη. Στις επόμενες δομές που θα δούμε, οι κόμβοι δεν είναι απαραίτητο να βρίσκονται σε συνεχόμενες θέσεις κι επιπλέον το μέγεθος της δομής μπορεί να αυξομειώνεται δυναμικά (δεν είναι προκαθορισμένο).

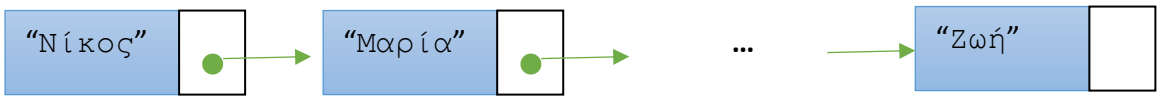
Λίστες[14]

Στη λίστα, οι κόμβοι μπορεί να βρίσκονται σε απομακρυσμένες θέσεις στην κύρια μνήμη και να συνδέονται μεταξύ τους με τη χρήση **δεικτών (pointers)**[15]. **Κάθε κόμβος, περιλαμβάνει δύο μέρη: τα δεδομένα και τον pointer.** Ο τελευταίος **περιέχει τη διεύθυνση μνήμης του επόμενου κόμβου** της λίστας, δηλαδή «δείχνει» προς τον επόμενο κόμβο.

Η γενική μορφή ενός κόμβου είναι η εξής:



Το γενικό σχήμα μίας λίστας που περιέχει, για παράδειγμα N κόμβους με αλφαριθμητικά δεδομένα είναι το εξής:

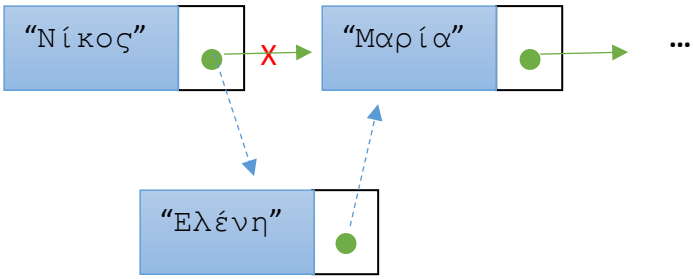


Παρατηρούμε, ότι στον τελευταίο κόμβο ο **pointer** δεν δείχνει πουθενά, που σημαίνει το τέλος της λίστας.

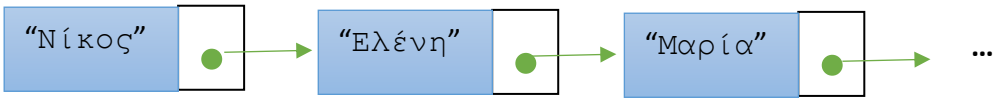
Παρακάτω, θα δούμε σχηματικά πώς γίνεται εισαγωγή και διαγραφή ενός κόμβου.

[14] Ονομάζονται και ***συνδεδεμένες λίστες (Linked Lists)***.
[15] Στην ορολογία των δομών δεδομένων, ο δείκτης θέσης στις λίστες ονομάζεται pointer και περιέχει τη διεύθυνση του επόμενου κόμβου ενώ στους πίνακες ονομάζεται index.

Παράδειγμα: **Εισαγωγή** ενός κόμβου με περιεχόμενο "Ελένη" μεταξύ του 1ου και 2ου.

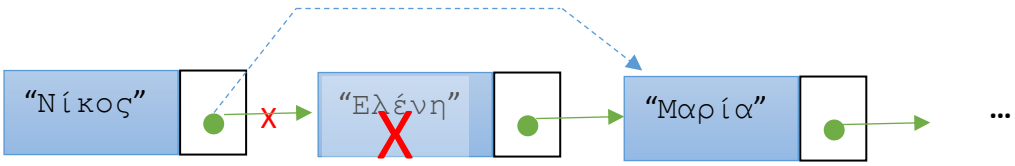


Και τελικά:

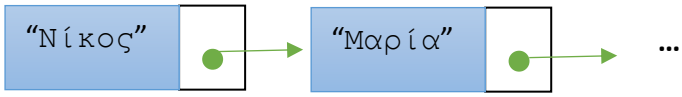


Ο **pointer** του "Νίκου" πλέον δεν δείχνει στη "Μαρία αλλά στην "Ελένη". Και ο **pointer** της "Ελένης" δείχνει στη "Μαρία".

Παράδειγμα: **Διαγραφή** του κόμβου με περιεχόμενο "Ελένη".



Και τελικά:



Μετά τη διαγραφή της "Ελένης" πρέπει να διορθώσουμε τον **pointer** του "Νίκου". Έτσι, ο **pointer** του "Νίκου" πλέον δεν δείχνει στη "Ελένη" αλλά στην "Μαρία".

Τι γίνεται με τον κόμβο της "Ελένης"; Αυτός πλέον, θεωρείται ένα *«άχρηστο δεδομένο»* και η μνήμη που καταλάμβανε πρέπει να απελευθερωθεί. Οι γλώσσες προγραμματισμού έχουν

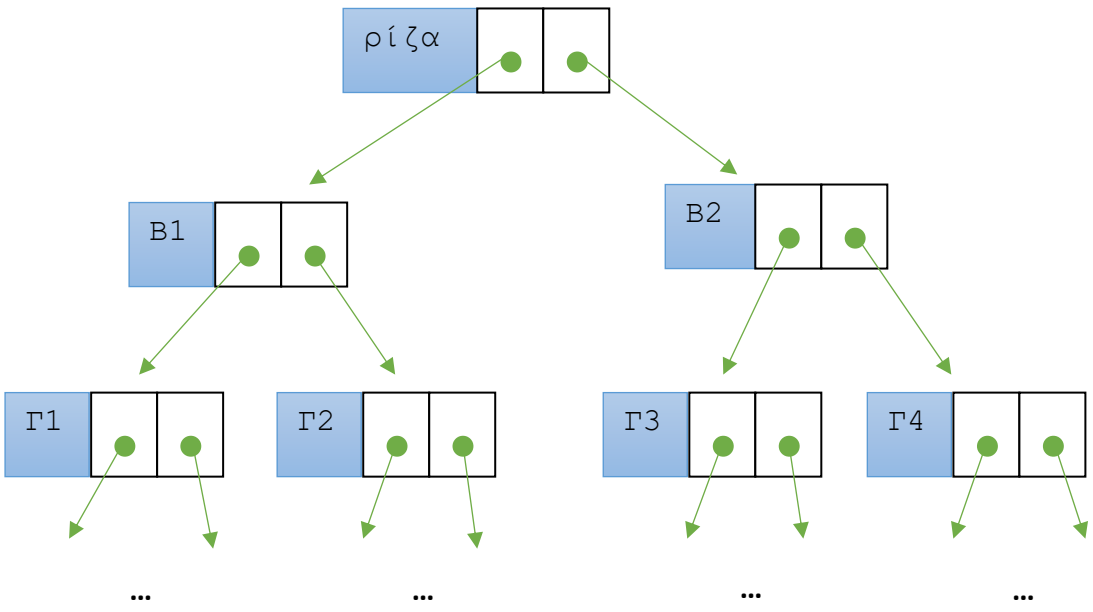
ειδικούς μηχανισμούς για την αυτόματη απελευθέρωση της μνήμης από ανάλογα *«σκουπίδια»*[16].

Δένδρα

Τα **δένδρα** είναι περισσότερο πολύπλοκες δομές από τις λίστες που έχουν τα εξής **χαρακτηριστικά**:

- **Κάθε κόμβος μπορεί να έχει περισσότερους από έναν pointers**. Με άλλα λόγια, δεν υπάρχει μόνο ένας επόμενος αλλά δύο (στην πιο απλή περίπτωση) ή περισσότεροι.
- **Υπάρχει μόνο ένας κόμβος από τον οποίο ξεκινούν όλοι οι άλλοι κόμβοι, γι' αυτό και ονομάζεται ρίζα (root).**
- **Οι κόμβοι που ξεκινούν από έναν προηγούμενο ανωτέρου επιπέδου, ονομάζονται παιδιά**. Με τη σειρά τους αυτά τα παιδιά-κόμβοι μπορεί να έχουν δικά τους παιδιά κ.ο.κ. Έτσι, υπάρχει μία **ιεραρχική δόμηση**.

Στο παρακάτω σχήμα, φαίνεται ένα δέντρο, όπου από κάθε κόμβο ξεκινούν **δύο pointers,** δηλαδή κάθε κόμβος έχει δύο επόμενους-παιδιά.



[16] Τέτοιοι μηχανισμοί ονομάζονται "*garbage collectors*".

Η εισαγωγή και διαγραφή κόμβων γίνεται όπως και στις λίστες, δηλαδή με κατάλληλη αναπροσαρμογή των **pointers** ώστε να δείχνουν σωστά στη διεύθυνση μνήμης του επόμενου κόμβου.

Αντίθετα με τους πίνακες και τις λίστες, τα δέντρα δεν είναι γραμμικές δομές δεδομένων αλλά <u>ιεραρχικές</u>. Δηλαδή, περιλαμβάνουν επίπεδα ιεραρχίας και χρησιμοποιούμε έννοιες όπως πατρικός-κόμβος (parent), κόμβοι-παιδιά (children), κόμβοι-αδέλφια (siblings) κ.α.

Όπως στους πίνακες και στις λίστες, και στα δέντρα **οι κόμβοι μπορεί να είναι ταξινομημένοι αύξουσα ή φθίνουσα με βάση το περιεχόμενό τους**, π.χ αλφαβητικά κατά Επώνυμο. Σε αυτήν την περίπτωση, έχουμε *ταξινομημένα δέντρα* (*ordered trees*). Τέτοιες δομές επιταχύνουν σημαντικά την αναζήτηση της πληροφορίας.

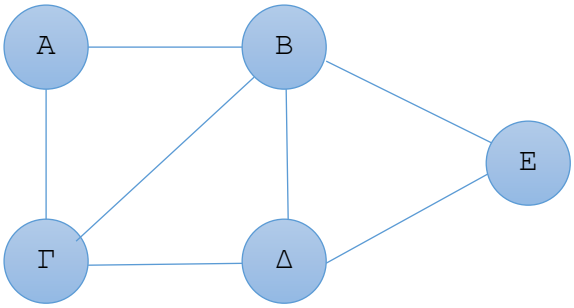
Τα δέντρα μπορούν να υλοποιηθούν και με στατικές δομές (π.χ. πίνακες).

Η δομή των δέντρων χρησιμοποιείται για την υλοποίηση του συστήματος αρχείων (file system) ενός υπολογιστή, την δημιουργία ευρετηρίων (indexes) στα συστήματα διαχείρισης Βάσεων Δεδομένων, στη δημιουργία ουρών προτεραιότητας (priority queues), στον συντακτικό έλεγχο ενός προγράμματος κατά τη μεταγλώττισή του (compiling) κ.α.

Γράφοι

Οι γράφοι αποτελούν τις πιο πολύπλοκες *μη-γραμμικές* δομές δεδομένων. Ένας **γράφος** είναι μία δομή δεδομένων που **αποτελείται από ένα σύνολο κόμβων κι ένα σύνολο γραμμών που συνδέουν μερικούς ή όλους τους κόμβους**.

Στο παρακάτω σχήμα, βλέπουμε ένα γενικό παράδειγμα:



Μερικές παρατηρήσεις:

- Οι **κόμβοι** του γράφου ονομάζονται επίσης και **κορυφές** (vertices) ή **σημεία** (points).
- Οι **γραμμές** που ενώνουν τους κόμβους ονομάζονται επίσης και **ακμές** (edges).
- Όπως και στα δέντρα, **κάθε κόμβος**, εκτός από τα δεδομένα, **μπορεί να έχει δύο ή περισσότερους pointers**. Για παράδειγμα, στο παραπάνω σχήμα, ο κόμβος "Γ" έχει 3 pointers.

Οι γράφοι, από πρακτικής άποψης, μπορούν να περιγράψουν πολλά προβλήματα και καταστάσεις της καθημερινής ζωής. **Μία συνήθης περίπτωση, είναι η αναπαράσταση ενός δικτύου (οδικού, τηλεφωνικού κ.λπ.)**. Για παράδειγμα, σε ένα οδικό δίκτυο, οι κόμβοι αναπαριστούν τις πόλεις και οι γραμμές τους δρόμους που τις ενώνουν. Με την χρήση κατάλληλων αλγορίθμων μπορούμε να βελτιστοποιήσουμε την κίνηση στο οδικό δίκτυο.

Ερωτήσεις κατανόησης

1. Ποιος είναι ο ορισμός για τα δεδομένα;
2. Από ποιές οπτικές γωνίες η Πληροφορική μελετά τα δεδομένα;
3. Ποιός είναι ο ορισμός για τη δομή δεδομένων; Ποιές οι βασικές λειτουργίες (πράξεις) που εφαρμόζονται επί των δομών δεδομένων;
4. Περιγράψτε τις δύο κυριότερες κατηγορίες των δομών δεδομένων. Δώστε από ένα παράδειγμα.

5. Συμπληρώστε τη βασική ισότητα: Προγράμματα = …… + …….

6. Περιγράψτε τη δομή του μονοδιάστατου και του δισδιάστατου πίνακα.

7. Περιγράψτε τη σειριακή αναζήτηση στοιχείου σε μονοδιάστατο πίνακα. Στη συνέχεια, προσπαθήστε να εκτελέσετε βήμα-βήμα πώς γίνεται η αναζήτηση του στοιχείου 3 στον εξής πίνακα:

```
Π = [12, 4, 6, 3, 24]
```

8. Περιγράψτε την ταξινόμηση στοιχείων σε έναν μονοδιάστατο πίνακα με τη μέθοδο της ευθείας ανταλλαγής (φυσαλίδας). Στη συνέχεια, προσπαθήστε να εκτελέσετε βήμα-βήμα πώς γίνεται η αύξουσα ταξινόμηση στον εξής πίνακα:

```
Π = [4, 18, 3, 2, 15]
```

9. Ποιά είναι τα πλεονεκτήματα/μειονεκτήματα της δομής των πινάκων σε σχέση με τη χρήση τους σε μία γλώσσα προγραμματισμού.

10. Περιγράψτε τη δομή της στοίβας και τις λειτουργίες της.

11. Περιγράψτε τη δομή της ουράς και τις λειτουργίες της.

12. Ποιές είναι οι δομές δεδομένων δευτερεύουσας μνήμης. Δώστε ένα παράδειγμα.

13. Περιγράψτε τη δομή μίας λίστας και δώστε ένα σχηματικό παράδειγμα του πώς γίνεται προσθήκη/διαγραφή ενός κόμβου.

14. Ποιά είναι τα χαρακτηριστικά της δομής ενός δένδρου. Δώστε ένα σχηματικό παράδειγμα.

15. Συμπληρώστε:
    - Τα δένδρα δεν είναι γραμμικό είδος δομής αλλά ……
    - Ο αρχικός κόμβος ενός δένδρου ονομάζεται …….. και κάθε κόμβος μπορεί να έχει περισσότερα από δύο …….

16. Σωστό ή Λάθος:
    - Τα δένδρα, αν και δυναμική δομές, μπορούν να υλοποιηθούν και με στατικές δομές (π.χ. πίνακες).

17. Δώστε ένα σχηματικό παράδειγμα ενός γράφου.

18. Συμπληρώστε:
    - Οι κόμβου ενός γράφου ονομάζονται ……
    - Μία γραμμή που συνδέει δύο κόμβους ενός γράφου ονομάζεται …….

19. Ποια είναι η χρησιμότητα της δομής ενός γράφου;

Παράρτημα

Ο επιστημονικός ορισμός της ταξινόμησης

Δοθέντων των στοιχείων $\alpha_1, \alpha_2, \ldots, \alpha_\nu$ η ταξινόμηση συνίσταται στη μετάθεση της θέσης των στοιχείων, ώστε να τοποθετηθούν σε μία σειρά $\alpha_{\kappa 1}, \alpha_{\kappa 2}, \ldots, \alpha_{\kappa \nu}$ έτσι ώστε, δοθείσης μίας συνάρτησης διάταξης (ordering function) f να ισχύει:

$$f(\alpha_{\kappa 1}) <= f(\alpha_{\kappa 2}) <= \ldots <= f(\alpha_{\kappa \nu}) \quad (\textbf{αύξουσα διάταξη})$$

ή

$$f(\alpha_{\kappa 1}) >= f(\alpha_{\kappa 2}) >= \ldots >= f(\alpha_{\kappa \nu}) \quad (\textbf{φθίνουσα διάταξη})$$

(από το σχολικό βιβλίο)

Ένας αλγόριθμος συγχώνευσης (merge) δύο πινάκων

Μία ενδιαφέρουσα και χρήσιμη λειτουργία είναι η συγχώνευση δύο ταξινομημένων πινάκων. Για παράδειγμα, έχουμε δύο ταξινομημένους, σε αύξουσα διάταξη, πίνακες:

A =	2	8	15	24	27

B =	4	15	17	25	29	32	33

οι οποίοι έχουν διαφορετικές διαστάσεις `A[1:5]` και `B[1:7]`. Κατά τη συγχώνευσή τους θα δημιουργηθεί ένας νέος ταξινομημένος πίνακας `Γ[1:12]`, ο οποίος θα περιλαμβάνει τα

στοιχεία κι από τους δύο πίνακες, επίσης σε αύξουσα διάταξη:

Γ =	2	4	8	15	15	17	24	25	27	29	32	33

Ας υποθέσουμε ότι θα χρειαστούμε δύο δείκτες για τη «σάρωση» των Α και Β. Ονομάζουμε i τον δείκτη του Α και j τον δείκτη του Β. Προφανώς, θα χρειαστούμε κι έναν δείκτη θέσης για τον Γ, ας τον ονομάσουμε k.

Η περιγραφή των βημάτων σε φυσική γλώσσα είναι η εξής:

ΒΗΜΑ 1: Ξεκίνα τους δείκτες θέσης των πινάκων από τη θέση 1.

ΒΗΜΑ 2: Έλεγξε ποιό είναι το μικρότερο στοιχείο από τους δύο πίνακες. Δηλαδή, ελέγχουμε τα στοιχεία A[i] και B[j].

ΒΗΜΑ 3: Έλεγχος: Αν A[i] < B[j] τότε βάλε στην τρέχουσα θέση του Γ, δηλαδή το Γ[k] το στοιχείο από τον Α και προχώρα τους δείκτες των Α και Γ κατά 1.
Αλλιώς, αν A[i] > B[j] τότε βάλε στην τρέχουσα θέση του Γ, δηλαδή το Γ[k] το στοιχείο από τον Β και προχώρα τους δείκτες των Β και Γ κατά 1.
Αλλιώς, αν A[i] = B[j] τότε βάλε στην τρέχουσα θέση του Γ και στην επόμενη, δηλαδή το Γ[k] και Γ[k+1] το στοιχείο από τον Α και τον Β και προχώρα τους δείκτες των Α και Β κατά 1 και του Γ κατά 2.

ΒΗΜΑ 4: Αν δεν έχει τελειώσει κάποιος από τους Α και Β τότε επανάλαβε από το ΒΗΜΑ 2 αλλιώς πήγαινε στο ΒΗΜΑ 5.

ΒΗΜΑ 5: Αν έχει τελειώσει ο Α τότε βάλε στον Γ όλα τα υπόλοιπα στοιχεία του Β. Αν έχει τελειώσει ο Β τότε βάλε στον Γ όλα τα υπόλοιπα στοιχεία του Α.

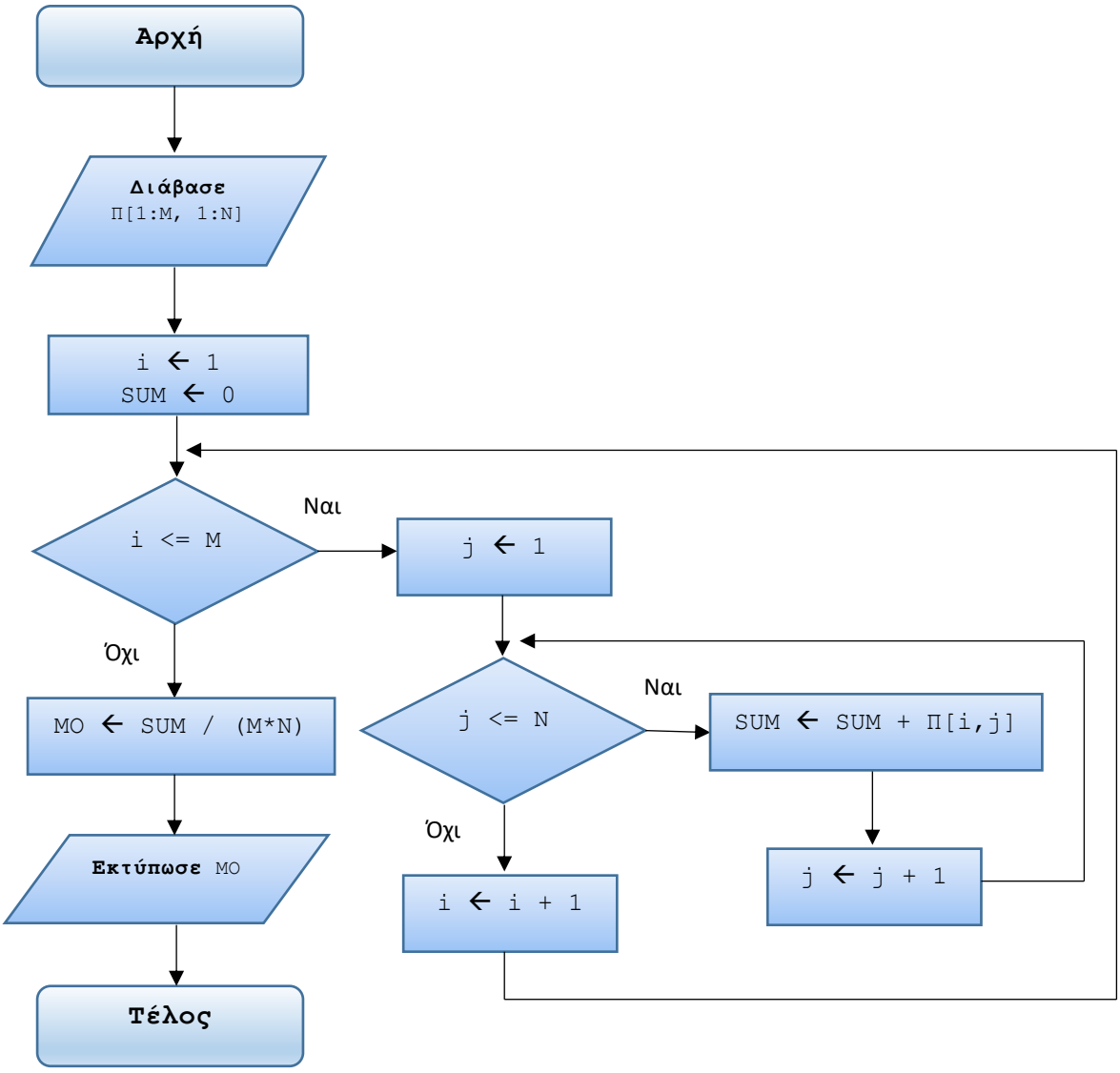
```
Αλγόριθμος Συγχώνευση_πινάκων
Δεδομένα // A[1:N], B[1:M] //
     i ← 1
     j ← 1
     k ← 1
     !όσο δεν έχει τελειώσει κάποιος από τους Α και Β
     Όσο  i <= N ΚΑΙ j <= M επανάλαβε
          Αν A[i] < B[j] τότε
               Γ[k] ← A[i]
```

```
            i ← i + 1
            k ← k + 1
        αλλιώς_αν A[i] > B[j] τότε
            Γ[k] ← B[j]
            j ← j + 1
            k ← k + 1
        αλλιώς  !είναι ίσα.
            Γ[k] ← A[i]
            Γ[k+1] ← B[j]
            i ← i + 1
            j ← j + 1
            k ← k + 2
        Τέλος_αν
    Τέλος_επανάληψης

    !όταν βγούμε από την επανάληψη σημαίνει ότι κάποιος ή και οι δύο
    !πίνακες Α και Β έχουν τελειώσει.

    !ελέγχουμε πρώτα αν έχει τελειώσει ο Α. Αν όχι, τότε αντίγραψε όλα τα
    !υπόλοιπα στοιχεία του στον Γ.

    Αν i <= N τότε
        Όσο i <= N επανάλαβε
            Γ[k]← A[i]
            i ← i + 1
            k ← k + 1
        Τέλος_επανάληψης
    Τέλος_αν

    !ελέγχουμε τώρα αν έχει τελειώσει ο Β. Αν όχι, τότε αντίγραψε όλα τα
    !υπόλοιπα στοιχεία του στον Γ.

    Αν j <= M τότε
        Όσο j <= M επανάλαβε
            Γ[k]← B[j]
            j ← j + 1
            k ← k + 1
```

```
            Τέλος_επανάληψης
        Τέλος_αν
Αποτελέσματα // Γ[1:N+M] //
Τέλος Συγχώνευση_πινάκων
```

Υπολογισμός ΜΟ δισδιάστατου πίνακα με διάγραμμα ροής

Έχουμε δει τον αλγόριθμο για τον υπολογισμό του μέσου όρου των στοιχείων ενός δισδιάστατου πίνακα με ψευδογλώσσα. Παρακάτω, θα περιγράψουμε τον ίδιο αλγόριθμο με **διάγραμμα ροής**.



ΤΕΛΟΣ ΣΗΜΕΙΩΣΕΩΝ ΚΕΦΑΛΑΙΟΥ 3