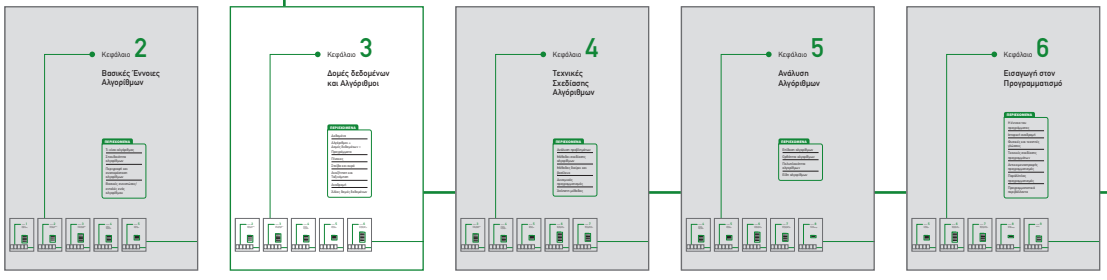
Κεφάλαιο 3

Δομές δεδομένων και Αλγόριθμοι

ΠΕΡΙΕΧΟΜΕΝΑ



3.1 | Δεδομένα

Τα *δεδομένα* (data) είναι η αφαιρετική αναπαράσταση της πραγματικότητας και συνεπώς μία απλοποιημένη όψη της. Για παράδειγμα, έστω ένα αρχείο μαθητών ενός σχολείου. Τα χρήσιμα δεδομένα που αποθηκεύονται είναι το ονοματεπώνυμο, η ηλικία, το φύλο, η τάξη, το τμήμα κ.λπ., όχι όμως το βάρος, το ύψος κ.λπ. Τα δεδομένα, λοιπόν, είναι ακατέργαστα γεγονότα, και κάθε φορά η επιλογή τους εξαρτάται από τον τύπο του προβλήματος. Η συλλογή των ακατέργαστων δεδομένων και ο συσχετισμός τους δίνει ως αποτέλεσμα την *πληροφορία* (information). Δεν είναι εύκολο να δοθεί επακριβής ορισμός της έννοιας της πληροφορίας, αλλά μπορεί να θεωρηθεί ότι ο αλγόριθμος είναι το μέσο για την παραγωγή πληροφορίας από τα δεδομένα. Με βάση τις δεδομένες πληροφορίες λαμβάνονται διάφορες αποφάσεις και γίνονται ενέργειες. Στη συνέχεια αυτές οι ενέργειες παράγουν νέα δεδομένα, νέες πληροφορίες, νέες αποφάσεις, νέες ενέργειες κ.ο.κ. Η μέτρηση, η κωδικοποίηση, η μετάδοση της πληροφορίας αποτελεί αντικείμενο μελέτης ενός ιδιαίτερου κλάδου, της *Θεωρίας Πληροφοριών* (Information Theory), που είναι ένα ιδιαίτερα σημαντικό πεδίο της Πληροφορικής.

Όπως η Πληροφορική ορίζεται ως επιστήμη σε συνάρτηση με την έννοια του αλγορίθμου, κατά τον ίδιο τρόπο η Πληροφορική ορίζεται και σε σχέση με την έννοια των δεδομένων. Έτσι, Πληροφορική θεωρείται η επιστήμη που μελετά τα δεδομένα από τις ακόλουθες σκοπιές:

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

- **Γλωσσών προγραμματισμού**. Οι γλώσσες προγραμματισμού υψηλού επιπέδου (high level programming languages) επιτρέπουν τη χρήση διάφορων τύπων (types) μεταβλητών (variables) 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 (data structure) 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 (attributes) 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 (Databases), της Μοντελοποίησης Δεδομένων (Data Modelling) και

Μια θέση μνήμης (byte) έχει ως περιεχόμενο 11110001. Η τιμή μπορεί να παριστάνει:

- Το χαρακτήρα _ στον κώδικα ASCII 437
- Το χαρακτήρα ρ στον κώδικα ΕΛΟΤ 928
- Το χαρακτήρα 1 στον κώδικα EBCDIC
- Την τιμή 241 στο δυαδικό σύστημα (ως μη προσημασμένο ακέραιο)
- Την τιμή -14 στο δυαδικό σύστημα (ως προσημασμένο ακέραιο στο συμπλήρωμα ως προς 1)
- Την τιμή -15 στο δυαδικό σύστημα (ως προσημασμένο ακέραιο στο συμπλήρωμα ως προς 2)

Ακόμη μπορεί να είναι τμήμα ενός ακέραιου σε 2 ή 4 bytes, καθώς και ενός αριθμού κινητής υποδιαστολής.

Όσον αφορά στη φυσική σημασία της, αν μεν είναι χαρακτήρας, τότε αποτελεί μέρος μιας αλφαριθμητικής σταθεράς, ενώ αν πρόκειται για αριθμητική τιμή, τότε μπορεί να είναι δεδομένο, διεύθυνση μνήμης ή κώδικας εντολής προγράμματος.

της Αναπαράστασης Γνώσης (Knowledge Representation) 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. Η έννοια της *δομής δεδομένων* (data structure) είναι σημαντική για την Πληροφορική και ορίζεται με τον ακόλουθο τυπικό ορισμό.

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 εξεταστεί 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.

Στην πράξη σπάνια χρησιμοποιούνται και οι οκτώ λειτουργίες για κάποια δομή. Συνηθέστατα παρατηρείται το φαινόμενο μία δομή δεδομένων να είναι αποδοτικότερη από μία άλλη δομή με κριτήριο κάποια λειτουργία, για παράδειγμα την αναζήτηση, αλλά λιγότερο αποδοτική για κάποια άλλη λειτουργία, για παράδειγμα την εισαγωγή. Αυτές οι παρατηρήσεις εξηγούν αφ' ενός την ύπαρξη διαφορετικών δομών, και αφ' ετέρου τη σπουδαιότητα της επιλογής της κατάλληλης δομής κάθε φορά.

Στη συνέχεια του βιβλίου αυτού θα γίνει πληρέστερη παρουσίαση εναλ-

λακτικών δομών δεδομένων. Ωστόσο, στο σημείο αυτό τονίζεται ότι υπάρχει μεγάλη εξάρτηση μεταξύ της δομής δεδομένων και του αλγορίθμου που επεξεργάζεται τη δομή. Μάλιστα, το πρόγραμμα πρέπει να θεωρεί τη δομή δεδομένων και τον αλγόριθμο ως μία αδιάσπαστη ενότητα. Η παρατήρηση αυτή δικαιολογεί την εξίσωση που διατυπώθηκε το 1976 από τον Wirth (που σχεδίασε και υλοποίησε τη γλώσσα Pascal)

Αλγόριθμοι + Δομές Δεδομένων = Προγράμματα

Ωστόσο για την πληρέστερη κατανόηση της σχέσης αυτής στη συνέχεια θα εξετασθεί ένα τέτοιο πρόβλημα.

ΠΑΡΑΔΕΙΓΜΑ

Έστω ότι πρέπει να γραφεί ένας αλγόριθμος που να δέχεται ως είσοδο το όνομα ενός συνδρομητή του ΟΤΕ και να δίνει ως έξοδο το τηλέφωνό του.

Πρώτη Λύση

Δομή Δεδομένων: Δημιουργείται μία ακολουθία (O_1,T_1), (O_2,T_2), ..., (O_n,T_n), όπου οι μεταβλητές O_i και T_i αναφέρονται στο όνομα και στο τηλέφωνο του i-οστού συνδρομητή, για i = 1,2,...,n.

Αλγόριθμος: Η ακολουθία ανιχνεύεται μέχρι να βρεθεί το ζητούμενο όνομα του συνδρομητή O_k και εκτυπώνεται το τηλέφωνο T_k. Ο αλγόριθμος αυτός είναι αποδοτικός για συνδρομητές ενός χωριού ή μίας κωμόπολης, αλλά για συνδρομητές μίας μεγάλης πόλης είναι χρονοβόρος.

Δεύτερη Λύση

Δομή Δεδομένων: Χρησιμοποιείται και πάλι η ακολουθία της πρώτης λύσης, αλλά αυτή τη φορά οι συνδρομητές είναι ταξινομημένοι λεξικογραφικά. Επιπλέον δημιουργείται μία δεύτερη ακολουθία με τα στοιχεία (A,n_1), (B,n_2), ..., (Ω,n_{24}). Κάθε στοιχείο της δεύτερης αυτής ακολουθίας δίνει για κάθε γράμμα του αλφαβήτου τη θέση n_i (για i = 1, 2, ..., 24) στην πρώτη ακολουθία με το πρώτο όνομα συνδρομητή που αρχίζει από το γράμμα αυτό.

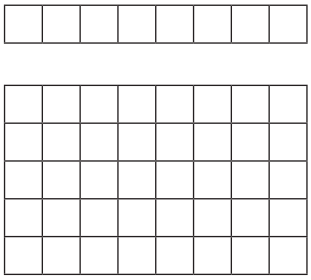
Αλγόριθμος: Αφήνεται για άσκηση στο μαθητή.

Οι δομές δεδομένων διακρίνονται σε δύο μεγάλες κατηγορίες: τις *στατικές* (static) και τις *δυναμικές* (dynamic). Οι δυναμικές δομές δεν αποθηκεύονται σε συνεχόμενες θέσεις μνήμης αλλά στηρίζονται στην τεχνική της λεγόμενης *δυναμικής παραχώρησης μνήμης* (dynamic memory allocation). Με άλλα λόγια, οι δομές αυτές δεν έχουν σταθερό μέγεθος, αλλά ο αριθμός των κόμβων τους μεγαλώνει και μικραίνει καθώς στη δομή εισάγονται νέα δεδομένα ή διαγράφονται κάποια δεδομένα αντίστοιχα. Όλες οι σύγχρονες γλώσσες προγραμματισμού προσφέρουν τη δυνατότητα δυναμικής παραχώρησης μνήμης. Ωστόσο, εμείς στη συνέχεια θα εξετάσουμε μόνο τις στατικές δομές που είναι ευκολότερες στην κατανόηση και την υλοποίησή τους.

3.3 | Πίνακες

Με τον όρο *στατική δομή δεδομένων* εννοείται ότι 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. Μία άλλη σημαντική διαφορά σε σχέση με τις δυναμικές δομές είναι ότι τα στοιχεία των στατικών δομών αποθηκεύονται σε συνεχόμενες θέσεις μνήμης.

Στην πράξη, οι στατικές δομές υλοποιούνται με πίνακες που μας είναι γνωστοί από άλλα μαθήματα και υποστηρίζονται από κάθε γλώσσα προγραμματισμού. 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 και η μέθοδος αναφοράς τους εξαρτάται από τη συγκεκριμένη γλώσσα υψηλού επιπέδου που χρησιμοποιείται. Όμως, γενικά η αναφορά στα στοιχεία ενός πίνακα γίνεται με τη χρήση του συμβολικού ονόματος του πίνακα ακολουθούμενου από την τιμή ενός ή περισσότερων *δεικτών* (indexes) σε παρένθεση ή αγκύλη.



Σχ. 3.1. *Παραδείγματα πινάκων (μονοδιάστατος, δισδιάστατος, τρισδιάστατος)*

Ένας πίνακας μπορεί να είναι μονοδιάστατος, αλλά στη γενικότερη περίπτωση μπορεί να είναι δισδιάστατος, τρισδιάστατος και γενικά ν-διάστατος πίνακας. Όσον αφορά στους δισδιάστατους πίνακες σημειώνεται ότι, αν το μέγεθος των δύο διαστάσεων είναι ίσο, τότε ο πίνακας λέγεται *τετραγωνικός* (square) 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. Στη συνέχεια δίνουμε δύο απλά παραδείγματα χρήσης πινάκων, τα οποία στηρίζονται σε αλγορίθμους του προηγουμένου κεφαλαίου.

ΠΑΡΑΔΕΙΓΜΑ 1. Εύρεση του μικρότερου στοιχείου ενός μονοδιάστατου πίνακα

Δίνεται ένας μονοδιάστατος πίνακας table 100 στοιχείων. Να σχεδιασθεί αλγόριθμος που να βρίσκει το μικρότερο στοιχείο του.

```
Αλγόριθμος Ελάχ_Πίνακα
Δεδομένα // table //
Min ← table[1]
Για i από 2 μέχρι 100
   Αν table[i] < Min τότε Min ← table[i]
Τέλος_επανάληψης
Αποτελέσματα //Min//
Τέλος Ελάχ_Πίνακα
```

Στον αλγόριθμο αυτό αρχικά το πρώτο στοιχείο του πίνακα εκχωρείται στη μεταβλητή Min. Στη συνέχεια κάθε επόμενο στοιχείο του πίνακα εξετάζεται, αν είναι μικρότερο της Min και αν ναι, τότε αντικαθιστά το προηγούμενο. Έτσι στο τέλος θα υπάρχει στη μεταβλητή Min το μικρότερο στοιχείο όλου του πίνακα table.

ΠΑΡΑΔΕΙΓΜΑ 2. Εύρεση αθροίσματος στοιχείων δισδιάστατου πίνακα

Δίδεται ο δισδιάστατος πίνακας table με m γραμμές n στήλες. Να βρεθεί το άθροισμα κατά γραμμή, κατά στήλη και συνολικά.

Στη συνέχεια ακολουθεί ο αλγόριθμος που επιλύει το πρόβλημα. Για καλύτερη κατανόηση σημειώνεται, ότι οι δύο πρώτοι βρόχοι μηδενίζουν τις αντίστοιχες μεταβλητές που θα υποδεχθούν τα αθροίσματα. Αυτό είναι μία τακτική που πρέπει να εφαρμόζεται οποτεδήποτε στα προβλήματά μας έχουμε να υπολογίσουμε αθροίσματα.

```
Αλγόριθμος Αθρ_Πίνακα
Δεδομένα // m, n, table //
sum ← 0
Για i από 1 μέχρι m
   row[i] ← 0
Τέλος_επανάληψης
Για j από 1 μέχρι n
   col[j] ← 0
Τέλος_επανάληψης
Για i από 1 μέχρι m
   Για j από 1 μέχρι n
      sum ← sum + table[i,j]
      row[i] ← row[i] + table[i,j]
      col[j] ← col[j] + table[i,j]
   Τέλος_επανάληψης
Τέλος_επανάληψης
Αποτελέσματα // row, col, sum //
Τέλος Αθρ_Πίνακα
```

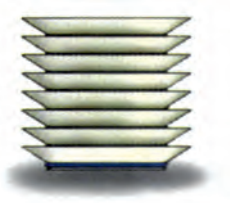
Ο διπλός εμφωλευμένος βρόχος που ακολουθεί τους δύο πρώτους απλούς βρόχους είναι η καρδιά του αλγορίθμου, όπου γίνονται οι υπολογισμοί που ζητά η εκφώνηση του παραδείγματος. Γενικά σε εμφωλευμένους βρόχους, μία τιμή μεταβλητής του εξωτερικού βρόχου παραμένει σταθερή, όσο μεταβάλλεται η τιμή της μεταβλητής του εσωτερικού βρόχου. Πιο συγκεκριμένα, στον αλγόριθμό μας αρχικά το i λαμβάνει την τιμή 1 και το j διαδοχικά τις τιμές 1,2,...,n. Κατόπιν, το i λαμβάνει την τιμή 2, ενώ το j και πάλι λαμβάνει διαδοχικά τις τιμές 1,2,...,n. Η διαδικασία αυτή επαναλαμβάνεται μέχρι το i να λάβει την τιμή m.

Ο επόμενος πίνακας είναι ένας δισδιάστατος πίνακας 5x5. Αν ο προηγούμενος αλγόριθμος εφαρμοσθεί στον πίνακα αυτό, τότε οι τιμές των στοιχείων του πίνακα row παρουσιάζονται στην τελευταία κατακόρυφη στήλη, ενώ οι τιμές των στοιχείων του πίνακα col παρουσιάζονται στην τελευταία γραμμή του πίνακα. Τέλος το συνολικό άθροισμα sum παρουσιάζεται στην κάτω-δεξιά γωνία.

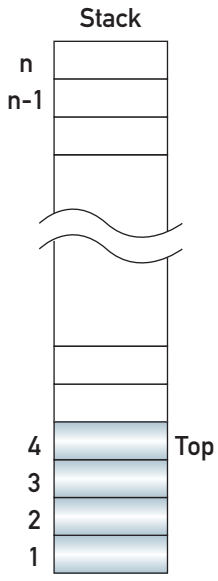
	Πίνακας table					Πίνακας row
	4	16	5	21	7	53
	28	9	38	13	51	139
	17	67	22	40	30	176
	20	40	10	3	13	86
	21	34	48	29	26	158
Πίνακας col	90	166	123	106	127	612 Sum

Οι πίνακες χρησιμεύουν για την αποθήκευση και διαχείριση δύο σπουδαίων δομών, της στοίβας (stack) και της ουράς (queue), που θα εξεταστούν λεπτομερέστερα στη συνέχεια, επειδή χρησιμοποιούνται σε πληθώρα πρακτικών εφαρμογών.

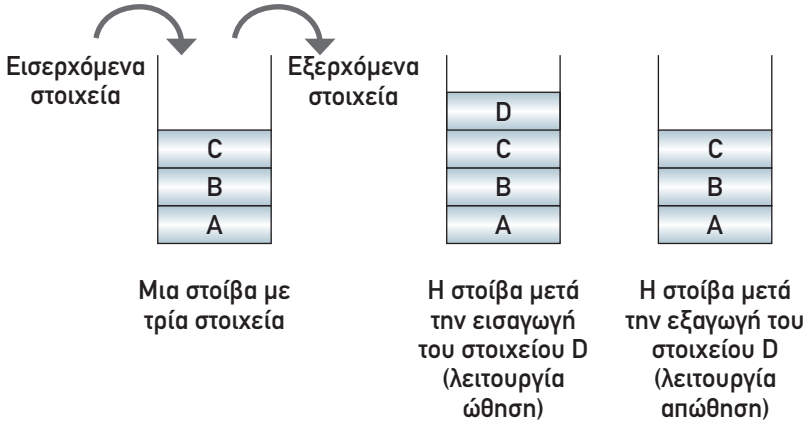
3.4 | Στοίβα



Μία στοίβα δεδομένων μοιάζει με μία στοίβα από πιάτα. Για παράδειγμα, κάθε πιάτο που πλένεται τοποθετείται στην κορυφή (top) της στοίβας των πιάτων, ενώ για σκούπισμα λαμβάνεται και πάλι το πιάτο της κορυφής. Αντίστοιχα, τα δεδομένα που βρίσκονται στην κορυφή της στοίβας λαμβάνονται πρώτα, ενώ αυτά που βρίσκονται στο βάθος της στοίβας λαμβάνονται τελευταία. Αυτή η μέθοδος επεξεργασίας ονομάζεται *Τελευταίο μέσα, πρώτο έξω* ή απλούστερα με την αγγλική συντομογραφία LIFO



Σχ. 3.3 *Υλοποίηση στοίβας με χρήση πίνακα*



Σχ. 3.2. *Λειτουργίες στοίβας*

(Last-In-First-Out). Στα αριστερά του επόμενου σχήματος δίνεται μία στοίβα με τρία στοιχεία, ενώ στο κέντρο παρουσιάζεται η ίδια στοίβα με ένα πρόσθετο στοιχείο στην κορυφή της.

Δύο είναι οι κύριες λειτουργίες σε μία στοίβα:

η *ώθηση* (push) στοιχείου στην κορυφή της στοίβας, και

η *απώθηση* (pop) στοιχείου από τη στοίβα.

Η διαδικασία της ώθησης πρέπει οπωσδήποτε να ελέγχει, αν η στοίβα είναι γεμάτη, οπότε λέγεται ότι συμβαίνει *υπερχείλιση* (overflow) της στοίβας. Αντίστοιχα, η διαδικασία απώθησης ελέγχει, αν υπάρχει ένα τουλάχιστον στοιχείο στη στοίβα, δηλαδή ελέγχει αν γίνεται *υποχείλιση* (underflow) της στοίβας.

Μια στοίβα μπορεί να υλοποιηθεί πολύ εύκολα με τη βοήθεια ενός μονοδιάστατου πίνακα, όπως φαίνεται στο σχήμα 3.3. Μια βοηθητική μεταβλητή (με όνομα συνήθως top) χρησιμοποιείται για να δείχνει το στοιχείο που τοποθετήθηκε τελευταίο στην κορυφή της στοίβας. Για την εισαγωγή ενός νέου στοιχείου στη στοίβα (ώθηση) αρκεί να αυξηθεί η μεταβλητή top κατά ένα και στη θέση αυτή να εισέλθει το στοιχείο. Αντίθετα για την εξαγωγή ενός στοιχείου από τη στοίβα (απώθηση) εξέρχεται πρώτα το στοιχείο που δείχνει η μεταβλητή top και στη συνέχεια η top μειώνεται κατά ένα για να δείχνει τη νέα κορυφή.

3.5 | Ουρά

Οι ουρές είναι καθημερινό φαινόμενο. Για παράδειγμα, ουρές δημιουργούνται όταν άνθρωποι, αυτοκίνητα, εργασίες, προγράμματα κ.λπ. περιμένουν για να εξυπηρετηθούν. Το θέμα είναι τόσο σημαντικό και με τέτοιες πρακτικές επιπτώσεις, ώστε ένας ιδιαίτερος κλάδος των Μαθηματικών, η Επιχειρησιακή Έρευνα (Operations Research), και ιδιαίτερα η Θεωρία Ουρών (Queueing Theory), μελετά τη συμπεριφορά και την επίδοση των ουρών. Σε μία ουρά αναμονής με ανθρώπους, συμβαίνει να εξυπηρετείται εκείνος που στάθηκε στην ουρά πρώτος από όλους τους άλλους (αν και υπάρχουν εξαιρέσεις που όμως δεν θα εξετασθούν στο βιβλίο αυτό). Η μέθοδος αυτή επεξεργασίας ονομάζεται *Πρώτο μέσα, πρώτο έξω* ή απλούστερα ακολουθώντας την αγγλική συντομογραφία FIFO (First-In-First-Out).

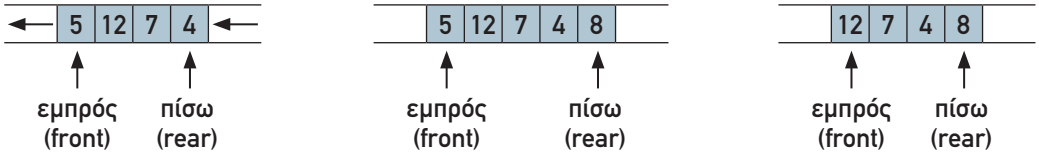
Δύο είναι οι κύριες λειτουργίες που εκτελούνται σε μία ουρά:

η *εισαγωγή* (enqueue) στοιχείου στο πίσω άκρο της ουράς, και

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

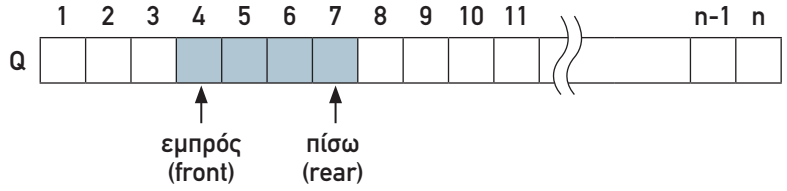
Άρα, σε αντίθεση με τη δομή της στοίβας, στην περίπτωση της ουράς απαιτούνται δύο δείκτες: ο *εμπρός* (front) και ο *πίσω* (rear) δείκτης, που μας δίνουν τη θέση του στοιχείου που σε πρώτη ευκαιρία θα εξαχθεί και τη θέση του στοιχείου που μόλις εισήλθε.

Στο σχήμα 3.4 φαίνεται μια ουρά με τέσσερα στοιχεία (α), στην οποία εισάγεται ένα νέο στοιχείο (β) και ακολούθως εξάγεται ένα στοιχείο.



Σχ. 3.4. *Εισαγωγή και εξαγωγή από ουρά*

Μια ουρά μπορεί να υλοποιηθεί με τη βοήθεια ενός μονοδιάστατου πίνακα, όπως φαίνεται στο σχήμα 3.5. Για την εισαγωγή ενός νέου στοιχείου στην ουρά αυξάνεται ο δείκτης rear κατά ένα και στη θέση αυτή αποθηκεύεται το στοιχείο. Αντίστοιχα για τη λειτουργία της εξαγωγής, εξέρχεται το στοιχείο που δείχνει ο δείκτης front, ο οποίος στη συνέχεια αυξάνεται κατά ένα, για να δείχνει το επόμενο στοιχείο που πρόκειται να εξαχθεί. Σε κάθε περίπτωση όμως, πρέπει να ελέγχεται πριν από οποιαδήποτε ενέργεια, αν υπάρχει ελεύθερος χώρος στον πίνακα για την εισαγωγή και αν υπάρχει ένα τουλάχιστον στοιχείο για την εξαγωγή.



Σχ. 3.5 *Υλοποίηση ουράς με χρήση πίνακα*

FIFO ΚΑΙ LIFO

Όπως είδαμε η δομή της στοίβας λειτουργεί με τη μέθοδο FIFO. Οι δύο αυτές μέθοδοι έχουν αρκετές χρήσεις σε πραγματικά προβλήματα. Ας θεωρήσουμε για παράδειγμα την περίπτωση ενός αποθηκευτικού χώρου μιας επιχείρησης. Σε κάθε αποθήκη γίνονται εισαγωγές ειδών που προέρχονται από αγορές από προμηθευτές, αν η επιχείρηση είναι εμπορική ή από την παραγωγή, αν πρόκειται για βιομηχανική επιχείρηση. Τα εμπορεύματα ή προϊόντα τοποθετούνται σε κάποιους χώρους, αποθήκες, ράφια κ.λπ. Όταν γίνονται πωλήσεις κάποιων ειδών, τα είδη αυτά βγαίνουν από την αποθήκη και αποστέλλονται στους πελάτες. Έτσι εισαγωγές και εξαγωγές ειδών γίνονται συνεχώς στην αποθήκη ανάλογα με τη διαδικασία προμηθειών και τη ροή των πωλήσεων.

Σε μια δεδομένη στιγμή για κάποιο είδος μπορεί να υπάρχουν αποθηκευμένα κάποια τεμάχια που προέρχονται από μια παραλαβή και κάποια άλλα που υπήρχαν πιο πριν. Όταν πρέπει να εξαχθεί λοιπόν ένα τεμάχιο από αυτό το είδος, προκύπτει το πρόβλημα, από ποια παρτίδα πρέπει να είναι;

Η απάντηση στο ερώτημα αυτό έχει φυσική και λογιστική αξία. Αν το είδος αυτό δεν επηρεάζεται από το χρόνο, τότε ίσως δεν έχει μεγάλη σημασία η επιλογή. Αν όμως πρόκειται για είδος που μπορεί να αλλοιωθεί ή έχει ημερομηνία λήξης (π.χ. φάρμακα), τότε είναι φανερό ότι πρέπει να επιλεγεί το παλαιότερο. Στην περίπτωση αυτή λοιπόν πρέπει η εξαγωγή των ειδών να γίνεται με τη μέθοδο FIFO και συνήθως επαφίεται στον αποθηκάριο να κάνει τη σωστή επιλογή.

Εξίσου δύσκολο είναι το πρόβλημα αυτό από την οικονομική και λογιστική σκοπιά, που μάλιστα αφορά όλα τα είδη με ή χωρίς ημερομηνία λήξης. Ας υποθέσουμε ότι μια επιχείρηση έχει πραγματοποιήσει τις επόμενες αγορές και πωλήσεις για ένα είδος.

Αγορές

Ημ/νία	Ποσότητα	Τιμή μονάδας	Αξία
1/1/02	4	10	40
15/1/02	6	12	72
ΣΥΝΟΛΟ	10		112

Πωλήσεις

Ημ/νία	Ποσότητα	Τιμή μονάδας	Αξία
30/1/02	5	20	100

Από τα παραπάνω στοιχεία αγορών και πωλήσεων δημιουργείται η επόμενη καρτέλα είδους.

Καρτέλα είδους

Ημ/νία	Αιτιολογία	Ποσότητα			Αξία κόστους		
		Εισαγωγή	Εξαγωγή	Υπόλοιπο	Εισαγωγή	Εξαγωγή	Υπόλοιπο
1/1/02	Αγορά	4		4	40		40
15/1/02	Αγορά	6		10	72		112
30/1/02	Πώληση		5	5		X	Y

Το πρόβλημα που ανακύπτει στις εφαρμογές αυτές είναι ο καθορισμός των τιμών X και Y. Από τις τιμές αυτές εξάγεται στη συνέχεια το καθαρό κέρδος, με το οποίο η επιχείρηση θα φορολογηθεί.

α) Λειτουργία LIFO

Στις 30/1/02 τα 5 τεμάχια που πουλήθηκαν θεωρούνται ότι ανήκουν στα 6 τεμάχια της τελευταίας αγοράς, δηλαδή με τιμή μονάδας 12€. Άρα X = 5 x 12 = 60€ και Y = 112 – 52 = 60. Τώρα, το καθαρό κέρδος της πώλησης γίνεται 100 – 60 = 40.

β) Λειτουργία FIFO

Στις 30/1/02 από τα 5 τεμάχια που πουλήθηκαν, τα 4 είναι από την αγορά της 1/1/02 και το 1 από την αγορά της 15/1/02. Άρα το κόστος τους είναι X = 4 x 10 + 1 x 12 = 52 και Y = 112 – 52 = 60. Τώρα, το καθαρό κέρδος της πώλησης γίνεται 100 – 52 = 48.

γ) Λειτουργία με τη σταθμική μέση τιμή

Λόγω της αυξημένης πολυπλοκότητας των αντίστοιχων προγραμμάτων, αλλά και των απαιτούμενων διαδικασιών οι περισσότερες επιχειρήσεις εφαρμόζουν τη μέθοδο της **σταθμικής μέσης τιμής**. Η τελευταία για το προηγούμενο παράδειγμα είναι 112/10 = 11,2. Άρα X = 5 x 11,2 = 56 και Y = 112 – 56 = 56. Στην περίπτωση αυτή το καθαρό κέρδος γίνεται 100 – 56 = 44€.

3.6 | Αναζήτηση

Το πρόβλημα της *αναζήτησης* (searching) ενός στοιχείου σε πίνακα είναι ιδιαίτερα ενδιαφέρον λόγω της χρησιμότητάς του σε πλήθος εφαρμογών. Υπάρχουν αρκετές μέθοδοι αναζήτησης σε πίνακα που εξαρτώνται κυρίως από το αν ο πίνακας είναι ταξινομημένος ή όχι. Μια άλλη παράμετρος είναι αν ο πίνακας περιέχει στοιχεία που είναι όλα διάφορα μεταξύ τους ή όχι. Τα στοιχεία του πίνακα μπορεί να είναι αριθμητικά ή αλφαριθμητικά.

Η πιο απλή μορφή αναζήτησης στοιχείου σε πίνακα είναι η *σειριακή* (sequential) ή *γραμμική* (linear) μέθοδος. Έτσι για τον επόμενο αλγόριθμο Sequential Search υποτίθεται ότι αναζητείται η τιμή key στο μη ταξινομημένο πίνακα table. Μετά την εκτέλεση του αλγορίθμου η μεταβλητή position επιστρέφει την τιμή 0, αν η αναζήτηση είναι ανεπιτυχής, ενώ αν

η αναζήτηση είναι επιτυχής, τότε επιστρέφει τη θέση του στοιχείου στον πίνακα (δηλαδή, έναν αριθμό από 1 ως n).

```
Αλγόριθμος Sequential_Search
Δεδομένα // n, table, key //
done ← ψευδής
position ← 0
i ← 1
Όσο (done=ψευδής) και (i<=n) επανάλαβε
Αν table[i]=key τότε
   done ← αληθής
   position ← i
αλλιώς
   i ← i+1
Τέλος_αν
Τέλος_επανάληψης
Αποτελέσματα //done, position //
Τέλος Sequential_Search
```

Όπως αναφέρθηκε, τα στοιχεία που περιέχονται στον πίνακα table δεν είναι ταξινομημένα. Επίσης, ο προηγούμενος αλγόριθμος ισχύει για την περίπτωση όπου κάθε στοιχείο υπάρχει μία μόνο φορά στον πίνακα. Αν κάποιο στοιχείο εμφανίζεται στον πίνακα περισσότερο από μία φορές, τότε ο αλγόριθμος πρέπει να τροποποιηθεί κατά το εξής: η μεταβλητή done είναι περιττή και η αναζήτηση συνεχίζεται μέχρι το τέλος του πίνακα ελέγχοντας με τη συνθήκη $i \leq n$. Εξάλλου, αν τα στοιχεία του πίνακα είναι ταξινομημένα, τότε ο αλγόριθμος πρέπει να σταματήσει, μόλις συναντήσει κάποιο στοιχείο που είναι μεγαλύτερο από το αναζητούμενο.

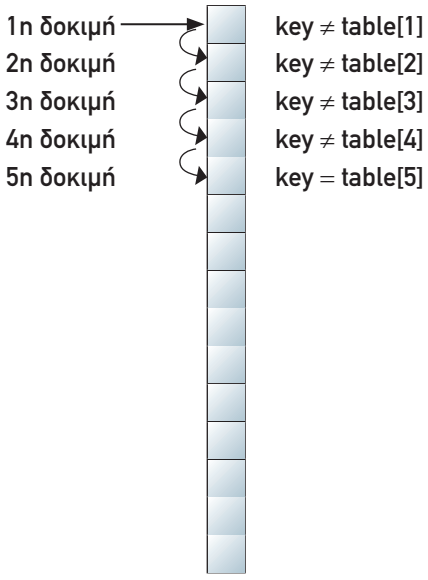
1	2	3	4	5	6	7	8	9
52	12	71	56	5	10	19	90	45

Για παράδειγμα, στο προηγούμενο σχήμα παρουσιάζεται ένας πίνακας που περιέχει εννέα αταξινόμητους ακεραίους. Έτσι, για την επιτυχή αναζήτηση της τιμής 56 απαιτούνται 4 προσπελάσεις. Αντίθετα, για την αναζήτηση της (ανύπαρκτης) τιμής 11 απαιτούνται 9 προσπελάσεις στον πίνακα, δηλαδή σάρωση ολόκληρου του πίνακα. Στο επόμενο σχήμα παρουσιάζεται ένας πίνακας που περιέχει τα ίδια στοιχεία αλλά σε ταξινομημένη μορφή. Στον πίνακα αυτό η ανεπιτυχής αναζήτηση για την τιμή 11 τερματίζει μετά την τρίτη προσπάθεια και την ανάγνωση του αριθμού 12.

1	2	3	4	5	6	7	8	9
5	10	12	19	45	52	56	71	90

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

- ο πίνακας είναι μη ταξινομημένος,
- ο πίνακας είναι μικρού μεγέθους (για παράδειγμα, $n \leq 20$),
- η αναζήτηση σε ένα συγκεκριμένο πίνακα γίνεται σπάνια,



Σχ. 3.6. *Σειριακή αναζήτηση*

Σε επόμενο κεφάλαιο θα εξετασθεί μία αποτελεσματικότερη μέθοδος αναζήτησης, η δυαδική αναζήτηση.

3.7 | Ταξινόμηση

Η τακτοποίηση των κόμβων μίας δομής με μία ιδιαίτερη σειρά είναι μία πολύ σημαντική λειτουργία που ονομάζεται *ταξινόμηση* (sorting) ή *διάταξη* (ordering). Συνήθως η σειρά αυτή είναι η *αύξουσα τάξη* (ascending sequence) της τιμής των μεγεθών προς ταξινόμηση. Από το προηγούμενο παράδειγμα έγινε σαφές ότι σκοπός της ταξινόμησης είναι να διευκολυνθεί στη συνέχεια η αναζήτηση των στοιχείων του ταξινομημένου πίνακα. Η χρησιμότητα της ταξινόμησης αποδεικνύεται στην πράξη σε αναρίθμητες περιπτώσεις αναζήτησης αριθμητικών ή αλφαβητικών δεδομένων, όπως σε βιβλιοθηκονομικά συστήματα, λεξικά, τηλεφωνικούς καταλόγους, καταλόγους φόρου εισοδήματος και γενικά παντού όπου γίνεται αναζήτηση αποθηκευμένων αντικειμένων. Στη συνέχεια δίνεται ένας τυπικός ορισμός της ταξινόμησης.

ΟΡΙΣΜΟΣ

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

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

Αξίζει να σημειωθεί ότι η προηγούμενη συνάρτηση διάταξης μπορεί να τροποποιηθεί, ώστε να καλύπτει και την περίπτωση που η ταξινόμηση γίνεται με *φθίνουσα τάξη* (descending sequence) μεγέθους.

Ταξινόμηση ευθείας ανταλλαγής

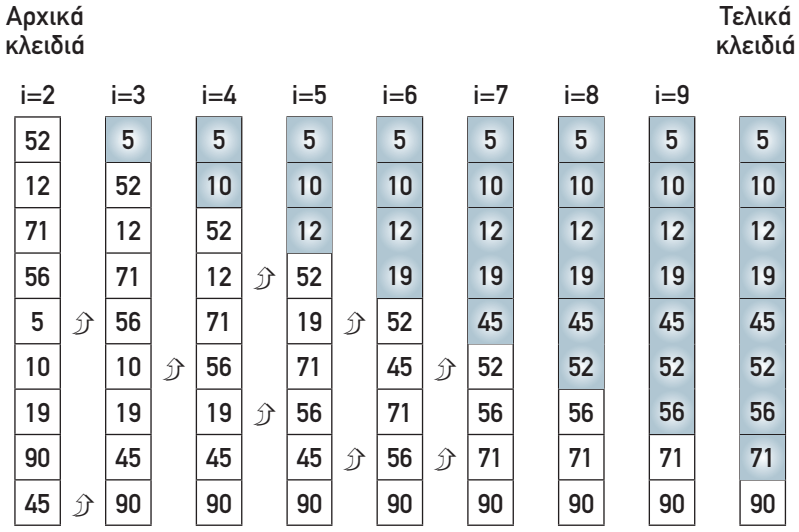
Η μέθοδος της *ταξινόμησης ευθείας ανταλλαγής* (straight exchange sort) βασίζεται στην αρχή της σύγκρισης και ανταλλαγής ζευγών γειτονικών στοιχείων, μέχρις ότου διαταχθούν όλα τα στοιχεία. Σύμφωνα με τη μέθοδο αυτή κάθε φορά γίνονται διαδοχικές προσπελάσεις στον πίνακα και μετακινείται το μικρότερο κλειδί της ακολουθίας προς το αριστερό άκρο του πίνακα. Αν ο πίνακας θεωρηθεί σε κατακόρυφη θέση αντί σε οριζόντια και οι ακέραιοι θεωρηθούν –επιστρατεύοντας αρκετή φαντασία– ως φυσαλίδες (bubbles) σε μία δεξαμενή νερού με βάρη σύμφωνα με την τιμή τους, τότε κάθε προσπέλαση στον πίνακα έχει ως αποτέλεσμα την άνοδο της φυσαλίδας στο κατάλληλο επίπεδο βάρους. Η μέθοδος είναι γνωστή ως ταξινόμηση φυσαλίδας (bubblesort).

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΔΕΥΤΕΡΕΥΟΥΣΑΣ ΜΝΗΜΗΣ

Σε μεγάλες πρακτικές εμπορικές/επιστημονικές εφαρμογές, το μέγεθος της κύριας μνήμης δεν επαρκεί για την αποθήκευση των δεδομένων. Στην περίπτωση αυτή χρησιμοποιούνται ειδικές δομές για την αποθήκευση των δεδομένων στη δευτερεύουσα μνήμη, δηλαδή κυρίως στο μαγνητικό δίσκο. Οι ειδικές αυτές δομές ονομάζονται *αρχεία* (files). Είναι γνωστό ότι μία σημαντική διαφορά μεταξύ κύριας μνήμης και μαγνητικού δίσκου είναι ότι στην περίπτωση του δίσκου, τα δεδομένα δεν χάνονται, αν διακοπεί η ηλεκτρική παροχή. Έτσι, τα δεδομένα των αρχείων διατηρούνται ακόμη και μετά τον τερματισμό ενός προγράμματος, κάτι που δεν συμβαίνει στην περίπτωση των δομών της κύριας μνήμης, όπως είναι οι πίνακες, όπου τα δεδομένα χάνονται όταν τελειώσει το πρόγραμμα. Τα στοιχεία ενός αρχείου ονομάζονται *εγγραφές* (records), όπου κάθε εγγραφή αποτελείται από ένα ή περισσότερα *πεδία* (fields), που ταυτοποιούν την εγγραφή, και από άλλα πεδία που περιγράφουν διάφορα χαρακτηριστικά της εγγραφής. Για παράδειγμα, έστω η εγγραφή ενός μαθητή με πεδία: Αριθμός Μητρώου, Ονοματεπώνυμο, Έτος Γέννησης, Τάξη, Τμήμα. Το πεδίο Αριθμός Μητρώου ταυτοποιεί την εγγραφή και ονομάζεται *πρωτεύον κλειδί* (primary key) ή απλά κλειδί. Το πεδίο Ονοματεπώνυμο επίσης ταυτοποιεί την εγγραφή και γι' αυτό αποκαλείται *δευτερεύον κλειδί* (secondary keys), αν υπάρχει πρωτεύον κλειδί. Το πρόβλημα της *αναζήτησης* (searching) μίας εγγραφής με βάση την τιμή του πρωτεύοντος ή ενός δευτερεύοντος κλειδιού σε αρχεία είναι ιδιαίτερα ενδιαφέρον, αν ληφθεί υπ' όψη η μεγάλη ποικιλία των χαρακτηριστικών τόσο της δομής (για παράδειγμα, στατική ή δυναμική, τρόπος οργάνωσης, μέσο αποθήκευσης κ.λπ.), του τύπου των δεδομένων (για παράδειγμα, ακέραιοι, κείμενο, χαρτογραφικά δεδομένα, χρονοσειρές κ.λπ.), όσο και της αναζήτησης (δηλαδή, με βάση το πρωτεύον ή το δευτερεύον κλειδί κ.λπ.).

ΠΑΡΑΔΕΙΓΜΑ

Έστω ότι ο αρχικός πίνακας αποτελείται από εννέα κλειδιά, τα εξής: 52, 12, 71, 56, 5, 10, 19, 90 και 45. Η μέθοδος εφαρμοζόμενη σε αυτά τα εννέα κλειδιά εξελίσσεται όπως φαίνεται στο επόμενο σχήμα. Κάθε φορά το ταξινομημένο τμήμα του πίνακα εμφανίζεται με χρώμα, ενώ τα στοιχεία που σαν φυσαλίδες ανέρχονται μέσα στον πίνακα εντοπίζονται με το αντίστοιχο βέλος στα δεξιά τους. Κάθε φορά εμφανίζεται η τάξη της επανάληψης (i).



Σχ. 3.7. *Ταξινόμηση φυσαλίδας*

Η ταξινόμηση φυσαλίδας υλοποιείται με τον επόμενο αλγόριθμο.

```
Αλγόριθμος Φυσαλίδα
Δεδομένα // table, n //
Για i από 2 μέχρι n
   Για j από n μέχρι i με_βήμα -1
      Αν table[j-1] > table[j] τότε
         αντιμετάθεσε table[j−1], table[j]
      Τέλος_αν
   Τέλος_επανάληψης
Τέλος_επανάληψης
Αποτελέσματα // table //
Τέλος Φυσαλίδα
```

Στον αλγόριθμο αυτό ως είσοδος δίνεται η μεταβλητή table με n ακεραίους που πρέπει να ταξινομηθούν. Φυσικά η επιλογή του ακέραιου τύπου για το κλειδί είναι αυθαίρετη, αφού μπορεί να χρησιμοποιηθεί οποιοσδήποτε άλλος τύπος, όπου ορίζεται μία συνάρτηση διάταξης, όπως για παράδειγμα ο τύπος του χαρακτήρα.

Σημειώνεται ότι η εντολή "αντιμετάθεσε table[j-1], table[j]" ανταλλάσσει το περιεχόμενο δύο θέσεων με τη βοήθεια μίας βοηθητικής θέσης. Εναλλακτικά αυτό μπορεί να γίνει με τις εξής τρεις εντολές:

```
temp ← table[j−1]
table[j-1] ← table[j]
table[j] ← temp
```

Για την ταξινόμηση δεδομένων έχουν εκπονηθεί πάρα πολλοί αλγόριθμοι. Άλλοι σχετικά απλοί αλγόριθμοι είναι η ταξινόμηση με επιλογή και η ταξινόμηση με παρεμβολή. Ο πιο γρήγορος αλγόριθμος ταξινόμησης είναι η "γρήγορη ταξινόμηση" (quicksort). Η ταξινόμηση φυσαλίδας είναι ο πιο απλός και ταυτόχρονα ο πιο αργός αλγόριθμος ταξινόμησης.

3.8 | Αναδρομή

Στο κεφάλαιο αυτό θα εξετάσουμε την έννοια της *αναδρομής* (recursion), που είναι μία σπουδαία εφαρμογή των στοιβών που εξετάσθηκαν προηγουμένως. Η τεχνική της αναδρομής χρησιμοποιείται ευρύτατα τόσο από το λογισμικό συστήματος όσο και στο λογισμικό εφαρμογών. Πιο

1. Τι είναι (τι θεωρούνται) τα δεδομένα; (για Σ/Λ)

Τα δεδομένα είναι μια αφαιρετική αναπαράσταση της πραγματικότητας, δηλ. μια απλοποιημένη όψη της.

Είναι ακατέργαστα γεγονότα, στοιχεία που τα αντιλαμβανόμαστε με τουλάχιστον μία από τις αισθήσεις μας.

2. Πως παράγονται οι πληροφορίες; (συσχέτιση με 1ο κεφ)

Η συλλογή των δεδομένων και ο συσχετισμός τους δίνει ως αποτέλεσμα την πληροφορία.

Από 1ο κεφ: Η πληροφορία είναι γνωσιακό στοιχείο που προκύπτει από την επεξεργασία δεδομένων.

3. Ποιο είναι το αντικείμενο της *Θεωρίας Πληροφοριών*;

Η μέτρηση, η κωδικοποίηση και η μετάδοση της πληροφορίας αποτελεί το αντικείμενο της *Θεωρίας Πληροφοριών* (που είναι κλάδος της Πληροφορικής).

4. Από ποιες σκοπιές μελετά η Πληροφορική τα δεδομένα;

Υλικού: Από τη σκοπιά του υλικού, μελετάμε με ποιες αναπαραστάσεις μπορούν να αποθηκευτούν τα δεδομένα στην κύρια μνήμη και στις περιφερειακές συσκευές.

Γλωσσών Προγραμματισμού: Από τη σκοπιά των γλωσσών προγραμματισμού, μελετάμε τους διάφορους τύπους δεδομένων που μπορούμε να χρησιμοποιήσουμε για τις μεταβλητές.

Δομών Δεδομένων: Από τη σκοπιά των δομών δεδομένων, μελετάμε τα διάφορα είδη δομών που μπορούν να δημιουργηθούν καθώς και τις λειτουργίες που γίνονται σε αυτές τις δομές.

Ανάλυσης Δεδομένων: Απο τη σκοπιά της ανάλυσης δεδομένων, μελεταμε με ποιούς τρόπους μπορούν να καταγραφούν και να συσχετιστούν τα δεδομένα ώστε να παραχθεί γνώση.

5. Τι είναι Δομή Δεδομένων (ορισμός)

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

6. Ποιες είναι οι βασικές λειτουργίες ή πράξεις των Δομών Δεδομένων;

Προσπέλαση: πρόσβαση (access) σε κάποιο κόμβο για να εξετάσουμε ή να τροποποιήσουμε το περιεχόμενό του

Εισαγωγή: προσθήκη νέου κόμβου σε μία δομή

Διαγραφή: αφαίρεση ενός κόμβου από μία δομή

Αναζήτηση: προσπέλαση των κόμβων μιας δομής για να εντοπιστούν ένας ή περισσότεροι που έχουν κάποια συγκεκριμένη ιδιότητα

Ταξινόμηση: η τακτοποίηση των κόμβων μίας δομής σε αύξουσα ή φθίνουσα σειρά

Αντιγραφή: όλοι ή μερικοί κόμβοι μίας δομής αντιγράφονται σε μία άλλη

Συγχώνευση: δύο ή περισσότερες δομές συνενώνονται σε μία εναία δομή

Διαχωρισμός: αντίστροφη πράξη της συγχώνευσης

7. (κ) Ποιες από αυτές μπορούν να υλοποιηθούν σε πίνακες;

Όλες εκτός από την εισαγωγή και τη διαγραφή, οι οποίες γίνονται μόνο σε δυναμικές δομές δεδομένων.

8. Όλες οι λειτουργίες είναι κατάλληλες για όλες τις Δομές Δεδομένων;

Όχι. Στην πράξη σπάνια χρησιμοποιούνται και οι 8 λειτουργίες για κάποια δομή δεδομένων. Συνήθως οι διάφορες δομές είναι πιο αποδοτικές για κάποιες λειτουργίες και όχι για όλες.

9. Αλγόριθμος + Δομές Δεδομένων = Προγράμμα

Η εξίσωση αυτή δείχνει την στενή σχέση (αλληλεξάρτηση) που υπάρχει μεταξύ των αλγορίθμων και των δομών δεδομένων.

Αυτό σημαίνει ότι αν σε κάποιο πρόγραμμα αλλάξουμε τη δομή δεδομένων, τότε θα αλλάξει και η λογική του προγράμματος (δηλ. ο αλγόριθμος). Ισχύει όμως και το αντίστροφο: αν αλλάξουμε τη λογική προγράμματος (δηλ. τον αλγόριθμο) τότε πολύ πιθανόν να είναι καταλληλότερη κάποια άλλη δομή δεδομένων.

10. Σε ποιες μεγάλες κατηγορίες διακρίνονται οι Δομές Δεδομένων;

Οι Δομές Δεδομένων διακρίνονται σε δύο μεγάλες κατηγορίες: στατικές και δυναμικές. Στις στατικές δομές το μέγεθος δεν αλλάζει, ενώ στις δυναμικές μπορεί να αυξομειωθεί κατά τη διάρκεια εκτέλεσης του προγράμματος.

11. Πως αποθηκεύονται στην μνήμη οι δυναμικές Δομές Δεδομένων;

Η τεχνική που χρησιμοποιεί ο υπολογιστής για να αποθηκεύει στη μνήμη τις δυναμικές δομές δεδομένων λέγεται ***Δυναμική Παραχώρηση Μνήμης***. Οι κόμβοι αυτών των δομών δεν αποθηκεύονται σε συνεχόμενες, αλλά σε τυχαίες θέσεις μνήμης. Και πρέπει ο προγραμματιστής να διαχειρίζεται κατάλληλους δείκτες ώστε να γνωρίζει που βρίσκεται ο προηγούμενος ή/και ο επόμενος κόμβος της δομής.

12. Πως αποθηκεύονται στην μνήμη οι στατικές Δομές Δεδομένων;

Οι στατικές δομές δεδομένων αποθηκεύονται σε **συνεχόμενες** θέσεις μνήμης.

13. Πότε καθορίζεται το μέγεθος μιας στατικής Δομής Δεδομένων;

Το μέγεθος μιας στατικής δομής δεδομένων πρέπει να δηλωθεί κατά την φάση της συγγραφής του προγράμματος. Μένει υποχρεωτικά σταθερό κατά τις φάσεις τις μεταγλώττισης και εκτέλεσης του προγράμματος.

14. Πως υλοποιούνται οι στατικές Δομές Δεδομένων;

Οι στατικές δομές δεδομένων υλοποιούνται συνήθως με μονοδιάστατους ή πολυδιάστατους πίνακες.

15. Τι είναι η στοίβα; Περιγράψτε τη λειτουργία της.

Από το μπλε βιβλίο.

16. Τι είναι η ουρά; Περιγράψτε τη λειτουργία της.

Από το μπλε βιβλίο.

17. Από ποιους παράγοντες εξαρτάται η μέθοδος που θα ακολουθήσουμε κατά την διαδικασία της αναζήτησης σε στατική δομή δεδομένων;

Ο τρόπος που θα υλοποιηθεί η αναζήτηση εξαρτάται από το αν τα στοιχεία του πίνακα είναι μοναδικά ή όχι, καθώς και από το αν τα στοιχεία είναι ταξινομημένα ή όχι.

18.Σχολιάστε την αποδοτικότητα της σειριακής αναζήτησης. Πότε δικαιολογείται η χρήση της;

Η σειριακή αναζήτηση είναι ο πιο απλός αλλά και λιγότερο αποδοτικός αλγόριθμος αναζήτησης. Δικαιολογείται η χρήση της όταν:

(α) ο πίνακας είναι μικρός (μέχρι 20 στοιχεία)

(β) ο πίνακας δεν είναι ταξινομημένος

(γ) η αναζήτηση γίνεται σπάνια

19. Αναφέρατε έναν πιο αποτελεσματικό αλγόριθμο αναζήτησης.

Πιο αποτελεσματική είναι η δυαδική αναζήτηση, η οποία είναι ένας αλγόριθμος που ανήκει στην κατηγορία “Διαίρει κ Βασίλευε”. Η δυαδική αναζήτηση, όμως, προϋποθέτει να είναι ταξινομημένος ο πίνακας.

20. (κ) Η δυαδική αναζήτηση μπορεί να εφαρμοστεί σε οποιαδήποτε δομή δεδομένων;

Μπορεί να εφαρμοστεί μόνο σε ταξινομημένους πίνακες.

21. Τι είναι ταξινόμηση δομής δεδομένων;

Είναι η τακτοποίηση των κόμβων της δομής με μια συγκεκριμένη σειρά, συνήθως αύξουσα ή φθίνουσα.

22. Αναφέρατε έναν λόγο (παράδειγμα) για τον οποίο απαιτείται η ταξινόμηση μιας δομής δεδομένων.

Ένας βασικός λόγος για τον οποίο ταξινομούμε τις δομές δεδομένων είναι για να διευκολύνεται η αναζήτηση σε αυτές. Γενικά, αν μία δομή δεδομένων είναι ταξινομημένη τότε μπορούμε να γράψουμε πιο αποδοτικούς-γρήγορους αλγορίθμους αναζήτησης.

23. Περιγράψτε τον αλγόριθμο ταξινόμησης ευθείας ανταλλαγής (φυσαλίδα).

Στην ταξινόμηση ευθείας ανταλλαγής, συγκρίνονται διπλανά στοιχεία και αν είναι με λάθος σειρά τότε τα αντιμεταθέτουμε. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να ταξινομηθεί όλος ο πίνακας.

24. Αναφέρατε τους αλγορίθμους ταξινόμησης που γνωρίζετε.

Ταξινόμηση με επιλογή

Ταξινόμηση με παρεμβολή

Γρήγορη ταξινόμηση (quicksort)

Ταξινόμηση ευθείας ανταλλαγής

Μεταξύ αυτών, ο ταχύτερος αλγόριθμος ταξινόμησης είναι η *Γρήγορη Ταξινόμηση*.

25. Ποιες δομές δεδομένων χρησιμοποιούνται στις δευτερεύουσες μνήμες;

Δευτερεύουσες μνήμες είναι ο σκληρός δίσκος, ένα usb στικάκι, ένα cd, γενικά δηλ. είναι αποθηκευτικά μέσα τα οποία διατηρούν το περιεχόμενό τους όταν σβήνουμε τον υπολογιστή.

Στις δευτερεύουσες μνήμες δεν υπάρχουν μεταβλητές και στατικές ή δυναμικές δομές, αλλά αρχεία.

Αρχείο είναι μία συλλογή εγγραφών.

Εγγραφή είναι μία συλλογή από πεδία, τα οποία περιγράφουν διάφορα χαρακτηριστικά της.

Κάθε εγγραφή πρέπει να έχει ένα τουλάχιστον πεδίο το οποίο να έχει μοναδική τιμή, ώστε να ταυτοποιεί την εγγραφή. Αυτό το πεδίο λέγεται πρωτεύον κλειδί. Αν υπάρχουν περισσότερα από ένα τέτοια πεδία, λέγονται δευτερεύοντα κλειδιά.