3.

Δομές Δεδομένων και Αλγόριθμοι

Εισαγωγή

Εκτός από τους αλγορίθμους, σημαντική έννοια για την Πληροφορική είναι και η έννοια των "δεδομένων". Τα δεδομένα αποθηκεύονται στον υπολογιστή με τη βοήθεια των λεγόμενων "δομών δεδομένων". Θεωρώντας τους αλγόριθμους και τις δομές δεδομένων μία αδιάσπαστη ενότητα μπορεί να λεχθεί, ότι η ενότητα αυτή τελικά αποτελεί τη βάση ενός προγράμματος, που επιλύει ένα πρόβλημα. Στο κεφάλαιο αυτό γίνεται μία εισαγωγή στις σπουδαιότερες δομές δεδομένων και τις αντίστοιχες πράξεις που μπορούμε να κάνουμε με αυτές, όπως είναι η αναζήτηση, η εισαγωγή και η εξαγωγή στοιχείων, καθώς και η ταξινόμηση.

Διδακτικοί στόχοι

Στόχοι του κεφαλαίου αυτού είναι οι μαθητές:

- να αιτιολογούν τη σπουδαιότητα των δεδομένων για την επίλυση ενός προβλήματος,
- να επισημαίνουν την αδιάσπαστη ενότητα αλγόριθμων και δομών δεδομένων,
- να εκτελούν γενικές ασκήσεις και ασκήσεις αναζήτησης και ταξινόμησης με χρήση της δομής του πίνακα,
- να ορίζουν τις δομές της στοίβας και της ουράς με τις αντίστοιχες λειτουργίες,
- να ορίζουν την έννοια της αναδρομής και να εκτελούν απλές σχετικές ασκήσεις,
- να γνωρίζουν τις δομές της λίστας και του δένδρου.

Προερωτήσεις

- ✓ 'Εχεις ακούσει για τον όρο FIFO;
- ✓ Γνωρίζεις ότι μπορεί να εξομοιωθεί στον υπολογιστή μια ουρά ανθρώπων, τρένων ή προγραμμάτων;
- ✓ Υπάρχει δυνατότητα ταχύτερης αναζήτησης μιας πληροφορίας ανάμεσα σε πολλές και με ποιον τρόπο;
- ✓ Ξέρεις, αν υπάρχουν πολλές μέθοδοι για να ταξινομηθούν κάποια αντικείμενα;

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 κλπ., όχι όμως το βάρος, το ύψος κλπ. 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 κοκ. 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 κλπ. Η εγγραφή αποτελείται από τα πεδία (fields) που αποθηκεύουν χαρακτηριστικά (attributes) διαφορετικού τύπου, όπως για παράδειγμα ο κωδικός, η περιγραφή κλπ. 'Αλ-

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.

Οσον 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.

λη 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 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 ένα 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.

Ορισμός: 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 ένα κόμβο με σκοπό να εξετασθεί ή να τροποποιηθεί το περιεχόμενό του.
- ***Εισαγωγή*** (insertion), δηλαδή η προσθήκη νέων κόμβων σε μία υπάρχουσα δομή.
- ***Διαγραφή*** (deletion), που αποτελεί το αντίστροφο της εισαγωγής, δηλαδή ένας κόμβος αφαιρείται από μία δομή.
- ***Αναζήτηση*** (searching), κατά την οποία προσπελαύνονται οι κόμβοι μιας δομής, προκειμένου να εντοπιστούν ένας ή περισσότεροι που έχουν μια δεδομένη ιδιότητα.
- ***Ταξινόμηση*** (sorting), όπου οι κόμβοι μιας δομής διατάσσονται κατά αύξουσα ή φθίνουσα σειρά.

- ✓ ***Αντιγραφή*** (copying), κατά την οποία όλοι οι κόμβοι ή μερικοί από τους κόμβους μίας δομής αντιγράφονται σε μία άλλη δομή.
- ✓ ***Συγχώνευση*** (merging), κατά την οποία δύο ή περισσότερες δομές συνενώνονται σε μία ενιαία δομή.
- ✓ ***Διαχωρισμός*** (separation), που αποτελεί την αντίστροφη πράξη της συγχώνευσης.

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 εναλλακτικών δομών δεδομένων. Ωστόσο, στο σημείο αυτό τονίζεται ότι υπάρχει μεγάλη εξάρτηση μεταξύ της δομής δεδομένων και του αλγόριθμου 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)

$$\textit{Αλγόριθμοι} + \textit{Δομές Δεδομένων} = \textit{Προγράμματα}$$

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

Παράδειγμα

'Εστω 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 ακολυθία 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 διακρίνοναι 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

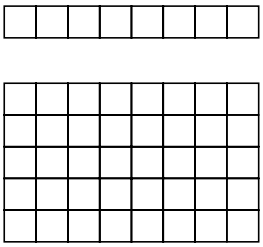
3.3 Πίνακες

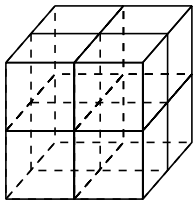
Με τον όρο *στατική δομή δεδομένων* εννοείται ότι το ακριβές μέγεθος της απαιτούμενης κύριας μνήμης καθορίζεται κατά τη στιγμή του προγραμματισμού τους, και κατά συνέπεια κατά τη στιγμή της μετάφρασής τους και όχι κατά τη στιγμή της εκτέλεσης τους προγράμματος. Μία άλλη σημαντική διαφορά σε σχέση με 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.

Ένας πίνακας μπορεί να είναι μονοδιάστατος, αλλά στη γενικότερη πε-

ρίπτωση μπορεί να είναι δισδιάστατος, τρισδιάστος και γενικά ν-διάστατος πίνακας. Όσον αφορά στους δισδιάστους 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 δισδιάστο 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.





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

Παράδειγμα 1. Εύρεση του μικρότερου στοιχείου ενός μονοδιάστατου πίνακα

Δίνεται ένας μονοδιάστατος πίνακα 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. Ετσι στο τέλος θα υπάρχει στη μεταβλητή Min το μικρότερο στοιχείο όλου του πίνακα table.

Παράδειγμα 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 επαναμβάνεται μέχρι το i να λάβει την τιμή m.

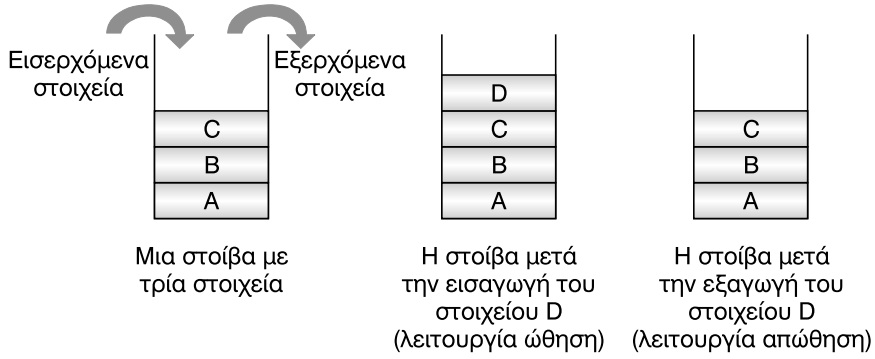
Ο επόμενος πίνακας είναι ένας δισδιάστατος πίνακας 5x5. Αν ο προηγούμενος αλγόριθμος εφαρμοσθεί στον πίνακα αυτό, τότε οι τιμές των στοιχείων του πίνακα row παρουσιάζονται στην τελευταία κατακόρυφη στήλη, ενώ οι τιμές των στοιχείων του πίνακα col παρουσιάζονται στην τελευταία γραμμή του πίνακα. Τέλος 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

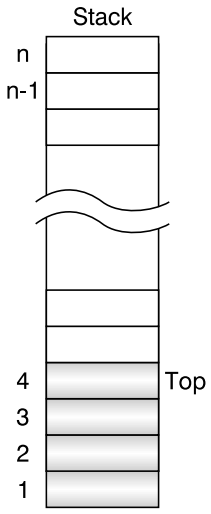
Οι πίνακες χρησιμεύουν για την αποθήκευση και διαχείριση δύο σπουδαίων δομών, της στοίβας (stack) και της ουράς (queue), που θα εξετασθούν λεπτομερέστερα στη συνέχεια, επειδή χρησιμοποιούνται σε πληθώρα πρακτικών εφαρμογών.

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. 3.2.** Λειτουργίες στοίβας.*



***Σχ. 3.3** Υλοποίηση στοίβας με χρήση πίνακα*

(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

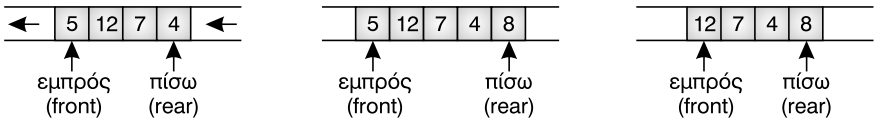
Οι ουρές είναι καθημερινό φαινόμενο. Για παράδειγμα, ουρές δημιουργούνται όταν άνθρωποι, αυτοκίνητα, εργασίες, προγράμματα κ.λπ. περιμένουν για να εξυπηρετηθούν. Το θέμα είναι τόσο σημαντικό και με τέτοιες πρακτικές επιπτώσεις, ώστε ένας ιδιαίτερος κλάδος των Μαθηματικών, η Επιχειρησιακή Έρευνα (Operations 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.

'Αρα, 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.



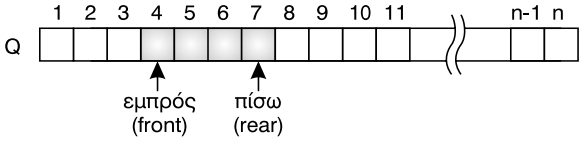
α) Μια ουρά με 4 στοιχεία

β) Η ουρά μετά την εισαγωγή του στοιχείου 8

γ) Η ουρά μετά την εξαγωγή του στοιχείου 5

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 LIFO, ενώ η δομή της ουράς με τη μέθοδο FIFO. Οι δύο αυτές μέθοδοι έχουν αρκετές χρήσεις σε ρεαλιστικά 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. Οταν γίνονται πωλήσεις κάποιων ειδών, τα είδη αυτά βγαίνουν από την αποθήκη και αποστέλονται 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.

Εξ ίσου 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
1/1/99	4	100	400
15/1/99	6	120	720
ΣΥΝΟΛΟ	10		1120

Πωλήσεις

Ημ/νία	Ποσότητα	Τιμή μονάδας	Αξία
30/1/99	5	200	1000

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
1/1/99	Αγορά	4		4	400		400
15/1/99	Αγορά	6		10	720		1120
30/1/99	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 30/1/99 τα 5 τεμάχια που πουλήθηκαν θεωρούνται ότι ανήκουν στα 6 τεμάχια της τελευταίας αγοράς, δηλαδή με τιμή μονάδας 120 δρχ. Αρα X=5x120 = 600 δρχ. και Y=1120-600=520. Κατ' επέκταση το καθαρό κέρδος από την πώληση είναι 1000-600=400.

β) Λειτουργία FIFO

Στις 30/1/99 από τα 5 τεμάχια που πουλήθηκαν, τα 4 είναι από την αγορά της 1/1/99 και το 1 από την αγορά της 15/1/99. Άρα το κόστος τους είναι X=4x100+1x120=520 και Y=1120-520=600. Τώρα, το καθαρό κέρδος της πώλησης γίνεται 1000-520=480.

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 1120/10=112. Αρα X=5x112=560 και Y=1120-560=560. Στην περίπτωση αυτή το καθαρό κέρδος γίνεται 1000-560=440 δρχ.

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 $i \leq n$. Εξ άλλου, αν τα στοιχεία του πίνακα είναι ταξινομημένα, τότε ο αλγόριθμος πρέπει να σταματήσει, μόλις συναντήσει κάποιο στοιχείο που είναι μεγαλύτερο από το αναζητούμενο.

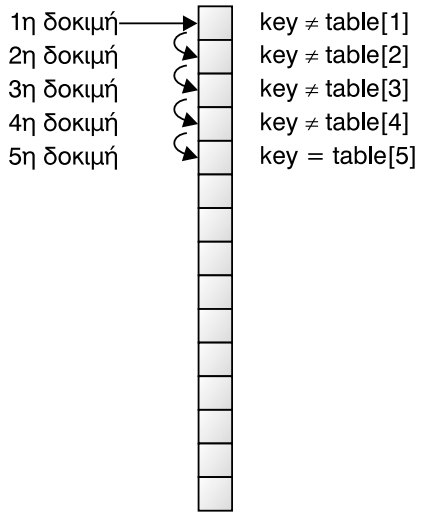
1	2	3	4	5	6	7	8	9
52	12	71	56	5	10	19	90	45

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. 'Ετσι, δικαιολογείται η χρήση της μόνο σε περιπτώσεις όπου:

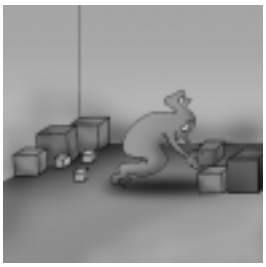
- ✓ ο πίνακας είναι μη ταξινομημένος,
- ✓ ο πίνακας είναι μικρού μεγέθους (για παράδειγμα, $n \leq 20$),
- ✓ η αναζήτηση σε ένα συγκεκριμένο πίνακα γίνεται σπάνια,



***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.

Ορισμός. 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 φυσσαλίδες (bubbles) σε μία δεξαμενή νερού με βάρη σύμφωνα με την τιμή τους, τότε κάθε προσπέλαση στον πίνακα έχει ως αποτέλεσμα την άνοδο της φυσσα-

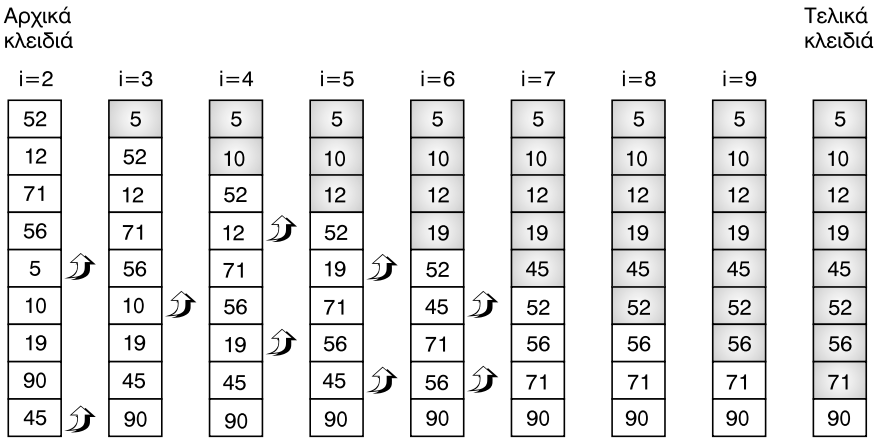
Δομές Δεδομένων δευτερεύουσας μνήμης

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, 'Ετος 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 κλπ.).

λίδας στο κατάλληλο επίπεδο βάρους. Η μέθοδος είναι γνωστή ως *ταξινόμηση φυσσαλίδας* (bubblesort).

Παράδειγμα. 'Εστω ότι ο αρχικός πίνακας αποτελείται από εννέα κλειδιά τα εξής: 52, 12, 71, 56, 5, 10, 19, 90 και 45. Η μέθοδος εφαρμοζόμενη στε 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 φυσσαλίδες ανέρχονται μέσα στον πίνακα εντοπίζο-

νται με το αντίστοιχο βέλος στα δεξιά τους. Κάθε φορά εμφανίζεται η τάξη της επανάληψης (i).



***Σχ. 3.7.** Ταξινόμηση φυσσαλίδας.*

Η ταξινόμηση φυσσαλίδας υλοποιείται με τον επόμενο αλγόριθμο.

```
Αλγόριθμος Φυσσαλίδα
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 Φυσσαλίδα
```

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 φυσσαλίδας είναι ο πιο απλός και ταυτόχρονα ο πιο αργός αλγόριθμος ταξινόμησης.

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 αυτό μπορεί να γίνει με εξής τρεις εντολές.

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

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 συγκεκριμένα, η αναδρομή στηρίζεται στη δυνατότητα που προσφέρεται από όλες τις σύγχρονες γλώσσες προγραμματισμού, μία διαδικασία ή συνάρτηση να καλεί τον εαυτό της.

3.8.1 Υπολογισμός του παραγοντικού

Η έννοια του παραγοντικού είναι γνωστή από τα μαθηματικά. Πιο συγκεκριμένα, για ένα ακέραιο n, το n παραγοντικό, που συμβολίζεται με n!, ορίζεται από τη σχέση:

$$n! = 1 \times 2 \times 3 \times \ldots \times (n-1) \times n$$

Ισοδύναμος είναι και ο εξής ορισμός

$$n! = \begin{cases} n \cdot (n-1)! & \alpha\nu\ n > 0 \\ 1 & \alpha\nu\ n = 0 \end{cases}$$

Ο ορισμός αυτός διακρίνεται από το βασικό χαρακτηριστικό ότι το παραγοντικό εκφράζεται με βάση μία απλούστερη περίπτωση του εαυτού του. Δηλαδή εξ ορισμού το παραγοντικό είναι μία αναδρομική αλγεβρική συνάρτηση. Με τον επόμενο αλγόριθμο υπολογίζεται με πολύ απλό αναδρομικό τρόπο το n παραγοντικό.

```
Αλγόριθμος Παραγοντικό
Δεδομένα // n //
Αν n = 0 τότε
    product ← 1
αλλιώς
    product ← n * Παραγοντικό(n-1)
Τέλος_αν
Αποτελέσματα // product //
Τέλος Παραγοντικό
```

Ωστόσο, το n! μπορεί να υπολογισθεί και με επαναληπτική μέθοδο, όπως παρουσιάζεται στον επόμενο αλγόριθμο Παραγοντικό2.

```
Αλγόριθμος Παραγοντικό2
Δεδομένα  // n //
product ← 1
Για i από 2 μέχρι n
    product ← product * i
Τέλος_επανάληψης
Αποτελέσματα  // product //
Τέλος Παραγοντικό2
```

Επομένως, δοθέντων αυτών των δύο λύσεων στο πρόβλημα υπολογισμού του n παραγοντικού προκύπτει το εύλογο ερώτημα, ποια μέθοδος είναι καλύτερη. Αφού εξετάσουμε μερικά ακόμη παραδείγματα, στη συνέχεια θα δοθεί απάντηση στην ερώτηση αυτή.

3.8.2 Υπολογισμός του μέγιστου κοινού διαιρέτη

Ένα ιστορικό πρόβλημα είναι η εύρεση του μέγιστου κοινού διαιρέτη (μκδ) δύο ακεραίων αριθμών. Ο αλγόριθμος εύρεσης του μκδ ανήκει στον Ευκλείδη. Ωστόσο η υλοποίηση και αυτού του αλγορίθμου μπορεί να γίνει κατά πολλούς τρόπους. Στη συνέχεια δίνεται ένας πρώτος αλγόριθμος για τον υπολογισμό του μκδ δύο ακεραίων x και y. Η μέθοδος αυτή είναι αρκετά αργή (και δεν στηρίζεται στον αλγόριθμο του Ευκλείδη), αλλά απλώς δίνεται για να διαφανεί η βελτίωση στην επίδοση από τους επόμενους αλγορίθμους. Ουσιαστικά λαμβάνει το μικρότερο από τους δύο ακεραίους, τον z, και εξετάζει με τη σειρά όλους τους ακεραίους ξεκινώντας από τον z και μειώνοντας συνεχώς κατά μία μονάδα μέχρι και οι δύο αριθμοί, x και y, να διαιρούνται από τη νέα τιμή του z.

```
Αλγόριθμος Μέγιστος_Κοινός_Διαιρέτης
Δεδομένα // x,y //
Αν x < y τότε
    z ← x
αλλιώς
    z ← y
Τέλος_αν
Όσο (x mod z ≠ 0) ή (y mod z ≠ 0) επανάλαβε
    z ← z-1
Τέλος_επανάληψης
Αποτελέσματα // z //
Τέλος Μέγιστος_Κοινός_Διαιρέτης
```

Η επόμενη μέθοδος για την εύρεση του μκδ αποδίδεται στον Ευκλείδη, και προφανώς βελτιώνει την προηγούμενη εκδοχή, επειδή δεν εξετάζει με τη σειρά όλους τους ακεραίους.

```
Αλγόριθμος Ευκλείδης
Δεδομένα  // x,y //
z ← y
Όσο z ≠ 0 επανάλαβε
    z ← x mod y
    x ← y
    y ← z
Τέλος_επανάληψης
Αποτελέσματα //  x //
Τέλος Ευκλείδης
```

Παράδειγμα. Για καλύτερη κατανόηση της μεθόδου, ας παρακολουθήσουμε πως ο αλγόριθμος αυτός βρίσκει το μκδ των αριθμών 150 και 35. Ο επόμενος πίνακας δείχνει τις τιμές των μεταβλητών z, x και y, κατά τη διάρκεια των επαναλήψεων. Δηλαδή, πριν την έναρξη της επαναληπτικής δομής οι αρχικές τιμές των x και y είναι 150 και 35 (όπως φαίνεται στη δεύτερη γραμμή του πίνακα). Ο αλγόριθμος σταματά όταν γίνει 0 η τιμή του z, οπότε ο μκδ είναι η τιμή του x, δηλαδή το 5.

z	x	y
35	150	35
10	35	10
5	10	5
0	5	0

Ωστόσο η μέθοδος του Ευκλείδη μπορεί να υλοποιηθεί και με έναν εναλλακτικό αναδρομικό τρόπο, που δίνεται στη συνέχεια. Η τρίτη αυτή εκδοχή είναι πολύ απλή στον προγραμματισμό και την κατανόησή της.

```
Αλγόριθμος Ευκλείδης
Δεδομένα // x,y //
Αν y = 0 τότε
     z ← x
αλλιώς
    z ← Ευκλείδης(y, x mod y)
Τέλος_αν
Αποτελέσματα // z //
Τέλος Ευκλείδης
```

3.8.3 Υπολογισμός αριθμών ακολουθίας Fibonacci

Για καλύτερη κατανόηση της διαφοράς μεταξύ επαναληπτικών και αναδρομικών μεθόδων, στη συνέχεια θα εξετασθεί ένα τελευταίο παράδειγμα, όπου υπολογίζεται η ακολουθία αριθμών Fibonacci πρώτης τάξης, που ορίζεται ως εξής:

$$F_i = \begin{cases} 0 & \alpha\nu\, i = 0 \\ 1 & \alpha\nu\, i = 1 \\ F_i = F_{i-1} + F_{i-2} & \alpha\nu\, i > 1 \end{cases}$$

ενώ οι πρώτοι όροι της ακολουθίας Fibonacci πρώτης τάξης είναι:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 κλπ.

Από τον ορισμό φαίνεται ανάγλυφα η αναδρομική φύση της συνάρτησης. Οι δύο επόμενοι αλγόριθμοι υπολογίζουν τον αριθμό Fibonacci πρώτης τάξης F_n με επαναληπτική και αναδρομική μέθοδο. Υποτίθεται ότι κατά την κλήση του αλγορίθμου μία μη αρνητική τιμή περνά ως όρισμα στη μεταβλητή n.

```
Αλγόριθμος Fibonacci1
Δεδομένα   // n //
Αν n ≤ 1 τότε Fib ← n
f0 ← 0
f1 ← 1
Για i από 2 μέχρι n
    fib ← f0+f1
    f0 ← f1
    f1 ← fib
Τέλος_επανάληψης
Αποτελέσματα // Fib //
Τέλος Fibonacci1

Αλγόριθμος Fibonacci2
Δεδομένα   // n //
Αν n ≤ 1 τότε
    Fib ← n
αλλιώς
    Fib ← Fib(n-1) + Fib(n-2)
Τέλος_Αν
Αποτελέσματα // Fib //
Τέλος Fibonacci2
```

Συχνά η χρήση αναδρομής διευκολύνει τον προγραμματιστή στην υλοποίηση και τον έλεγχο του προγράμματος. Αν και πολλές φορές η αναδρομή φαίνεται ως πιο φυσικός τρόπος προγραμματισμού, ωστόσο πρέπει να χρησιμοποιείται με μέτρο. Μεταξύ ενός απλού επαναληπτικού προγράμματος και ενός αναδρομικού προγράμματος προτιμάται το πρώτο. Ο λόγος είναι ότι κάθε κλήση μίας συνάρτησης ή μίας διαδικασίας έχει χρονικό κόστος μη αμελητέο. Έτσι το κέρδος σε χρόνο προγραμματισμού δημιουργεί απώλεια σε χρόνο εκτέλεσης. Επομένως, αν ο χρόνος απόκρισης είναι κρίσιμος για την εφαρμογή μας, τότε είναι βέβαιο ότι πρέπει να προτιμηθεί η επαναληπτική μέθοδος.

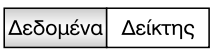
Οι προηγούμενοι αλγόριθμοι για τον υπολογισμό των αριθμών Fibonacci πρώτης τάξης εκτός της διαφοράς τους στην επίδοση λόγω του ότι, η πρώτη είναι επαναληπτική ενώ η δεύτερη είναι αναδρομική, έχουν και μία ακόμη σημαντική διαφορά. Η διαφορά αυτή έγκειται στο γεγονός ότι, η δεύτερη καλεί περισσότερο από μία φορά τον εαυτό της για τις ίδιες τιμές. Αυτό θα γίνει κατανοητό δοκιμάζοντας για παράδειγμα τον υπολογισμό του F_5. Σε επόμενο κεφάλαιο θα επανέλθουμε στη μέλετη των αλγορίθμων αυτών για τον υπολογισμό των αριθμών Fibonacci πρώτης τάξης.

3.9 Άλλες δομές δεδομένων

Κοινό γνώρισμα των δομών που εξετάσθηκαν προηγουμένως είναι ότι οι διαδοχικοί κόμβοι αποθηκεύονται σε συνεχόμενες θέσεις της κύριας μνήμης. Στην παράγραφο αυτή γίνεται μια παρουσίαση τριών πολύ σπουδαίων δομών δεδομένων, στις οποίες οι κόμβοι δεν είναι απαραίτητο να κατέχουν συνεχόμενες θέσεις μνήμης. Πρόκειται για τις *λίστες*, τα *δένδρα* και τους *γράφους*.

3.9.1 Λίστες

Στις λίστες το κύριο χαρακτηριστικό είναι ότι οι κόμβοι τους συνήθως βρίσκονται σε απομακρυσμένες θέσεις μνήμης και η σύνδεσή τους γίνεται με δείκτες. Ο *δείκτης* (pointer) είναι ένας ιδιαίτερος τύπος που προσφέρεται από τις περισσότερες σύγχρονες γλώσσες προγραμματισμού. Ο δείκτης δεν λαμβάνει αριθμητικές τιμές όπως ακέραιες, πραγματικές κ.α., αλλά οι τιμές του είναι διευθύνσεις στην κύρια μνήμη και χρησιμοποιείται ακριβώς για τη σύνδεση των διαφόρων στοιχείων μιας δομής, που είναι αποθηκευμένα σε μη συνεχόμενες θέσεις μνήμης. Συνήθως ο δείκτης είναι ένα πεδίο κάθε κόμβου της δομής, όπως φαίνεται στο σχήμα 3.8. Το πεδίο Δεδομέ-

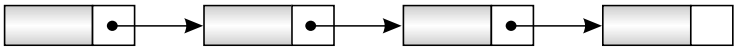


***Σχ. 3.8** Δομή κόμβου λίστας*

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

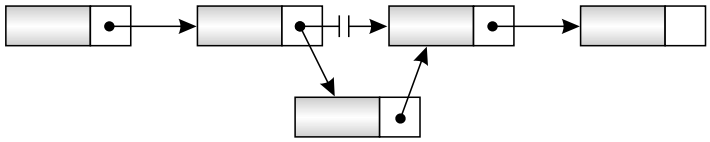
να μπορεί να περιέχει μία ή περισσότερες αλφαριθμητικές ή αριθμητικές πληροφορίες.

Στο σχήμα 3.9 παρουσιάζεται μια λίστα με τέσσερις κόμβους, όπου οι δείκτες έχουν τη μορφή βέλους, προκειμένου να φαίνεται ο κόμβος που παραπέμπουν.



***Σχ. 3.9.** Μία λίστα με τέσσερις κόμβους*

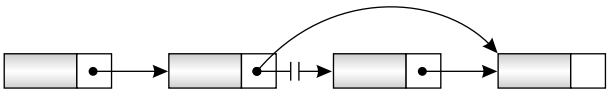
Με τη χρήση δεικτών διευκολύνονται οι λειτουργίες της εισαγωγής και της διαγραφής δεδομένων στις λίστες. Στο σχήμα 3.10 φαίνεται η εισαγωγή ενός νέου κόμβου μεταξύ του δεύτερου και τρίτου κόμβου της προηγούμενης λίστας.



***Σχ. 3.10.** Εισαγωγή σε λίστα*

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

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

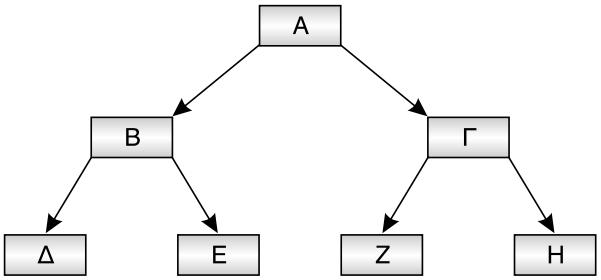


***Σχ. 3.11.** Διαγραφή κόμβου λίστας*

Αντίστοιχα για τη διαγραφή ενός κόμβου αρκεί ν' αλλάξει τιμή ο δείκτης του προηγούμενου κόμβου και να δείχνει πλέον τον επόμενου αυτού που διαγράφεται, όπως φαίνεται στο σχήμα 3.11. Ο κόμβος που διαγράφηκε (ο τρίτος) αποτελεί "άχρηστο δεδομένο" και ο χώρος μνήμης που καταλάμβανε, παραχωρείται για άλλη χρήση.

3.9.2 Δένδρα

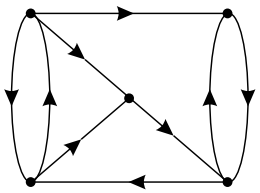
Τα *δένδρα* (trees) είναι δομές που στις σύγχρονες γλώσσες προγραμματισμού υλοποιούνται με τη βοήθεια των δεικτών, όπως εξηγήθηκε στην αρχή αυτής της παραγράφου. Βέβαια, μπορούν να υλοποιηθούν και με στατικές δομές (με πίνακες). Το κύριο χαρακτηριστικό των δένδρων είναι, ότι από ένα κόμβο δεν υπάρχει ένας μόνο επόμενος κόμβος, αλλά περισσότεροι. Υπάρχει ένας μόνο κόμβος, που λέγεται *ρίζα*, από τον οποίο ξεκινούν όλοι οι άλλοι κόμβοι. Στο σχήμα 3.12. παρατηρούμε ότι από τη ρίζα ξεκινούν δύο κόμβοι. Οι κόμβοι αυτοί λέγονται *παιδιά* της ρίζας. Με την ίδια λογική, από κάθε παιδί της ρίζας ξεκινούν άλλα παιδιά κ.ο.κ. Στη βιβλιογραφία αναφέρεται μία τεράστια ποικιλία δομών δένδρων, που η αναφορά σε αυτές βρίσκεται εκτός των ορίων του βιβλίου αυτού.



***Σχ. 3.12.** Δομή δένδρου*

3.9.3 Γράφοι

Ενας γράφος (graph) αποτελείται από ένα σύνολο κόμβων (ή σημείων ή κορυφών) και ένα σύνολο γραμμών (ή ακμών ή τόξων) που ενώνουν μερικούς ή όλους τους κόμβους. Ο γράφος αποτελεί την πιο γενική δομή δεδομένων, με την έννοια ότι όλες οι προηγούμενες δομές που παρουσιάστηκαν μπορούν να θεωρηθούν περιπτώσεις γράφων.



***Σχ. 3.13.** Ένας γράφος*

Πολλά προβλήματα και καταστάσεις της καθημερινής μας ζωής μπορούν να περιγραφούν με τη βοήθεια γράφων. Για παράδειγμα τα σημεία ενός γράφου μπορούν να παριστούν πόλεις και οι γραμμές τις οδικές συνδέσεις μεταξύ τους. Λόγω της μεγάλης πληθώρας και ποικιλίας των προβλημάτων που σχετίζονται με γράφους, έχει αναπτυχθεί ομώνυμη θεωρία, η Θεωρία Γράφων, η οποίο συχνά αποτελεί αυτοδύναμο μάθημα σε πανεπιστημιακά τμήματα.

Ανακεφαλαίωση

Στο κεφάλαιο αυτό αρχικά ορίσθηκε η Πληροφορική ως η επιστήμη που μελετά τα δεδομένα από τις σκοπιές του υλικού, των γλωσσών προγραμματισμού, των δομών δεδομένων και της ανάλυσης δεδομένων. Δόθηκε ο ορισμός της δομής δεδομένων και ένας κατάλογος με τις λειτουργίες που μπορούν να γίνουν με μία δομή δεδομένων. Η πρώτη δομή που εξετάσθηκε ήταν η δομή του πίνακα (μονοδιάστατου και δισδιάστατου), που είναι μία στατική δομή, με μέγεθος που δεν μεταβάλλεται χρονικά. Στη συνέχεια παρουσιάσθηκε η δομή της στοίβας, καθώς και των δύο βασικών πράξεων της ώθησης και της απώθησης των στοιχείων της. Επίσης περιγράφηκε η δομή της ουράς με αναφορά στις πράξεις της εισαγωγής και της εξαγωγής στοιχείων από αυτήν. Στη συνέχεια παρουσιάσθηκαν προβλήματα η λύση των οποίων εντάσσεται στις κατηγορίες της αναζήτησης και της ταξινόμησης. Η τεχνική της σειριακής/γραμμικής αναζήτησης στοιχείων από πίνακα δόθηκε με χρήση σχετικών αλγορίθμων και προσδιορίσθηκαν οι περιπτώσεις όπου η μέθοδος αυτή είναι αποτελεσματική. Έγινε σημαντική εμβάθυνση στη μέθοδο της αναδρομής μέσω διαφόρων παραδειγμάτων επίλυσης γνωστών προβλημάτων. Τέλος, επιγραμματικά παρουσιάζονται οι δομές της λίστας, του δένδρου και του γράφου.

Λέξεις κλειδιά

Δεδομένα, Πληροφορία, Δομή δεδομένων, Στατικές και δυναμικές δομές, Πίνακες, Στοίβα, Ουρά, FIFO και LIFO, Γραμμική αναζήτηση, Ταξινόμηση, Αναδρομή, Λίστες, Δένδρα, Γράφοι.

Ερωτήσεις - Θέματα για συζήτηση

1. Τι είναι δεδομένα και τι είναι πληροφορία ; Να δοθεί σύντομος ορισμός των όρων αυτών.
2. Ποιές είναι οι απόψεις από τις οποίες η επιστήμη της Πληροφορικής μελετά τα δεδομένα;
3. Να δοθεί ο ορισμός της δομής δεδομένων.
4. Ποιές είναι οι βασικές πράξεις επί των δομών δεδομένων;
5. Ποιά είναι η εξάρτηση μεταξύ της δομής δεδομένων και του αλγορίθμου που επεξεργάζεται τη δομή;
6. Να περιγραφούν οι δύο κυριότερες κατηγορίες των δομών δεδομένων.
7. Να περιγραφεί η δομή του πίνακα και να δοθεί παράδειγμα χρήσης του.
8. Να δοθεί ο ορισμός της στοίβας.
9. Ποιές είναι οι βασικές λειτουργίες που γίνονται σε μία στοίβα;
10. Να δοθεί ο ορισμός της ουράς.
11. Ποιές είναι οι βασικές λειτουργίες που γίνονται σε μία ουρά;
12. Να περιγραφεί η λειτουργία της αναδρομής και να σχολιασθεί η χρησιμότητα της.
13. Να δοθεί αναδρομικός αλγόριθμος υπολογισμού της δύναμης παραγματικού αριθμού υψωμένου σε ακέραια δύναμη.
14. Να περιγραφεί η λειτουργία της αναζήτησης.
15. Να δοθεί ένα παράδειγμα για τη σειριακή αναζήτηση στοιχείου σε έναν πίνακα.
16. Να δοθεί ο ορισμός της έννοιας της ταξινόμησης.
17. Να περιγραφεί η ταξινόμηση ευθείας ανταλλαγής και να δοθεί ένα παράδειγμα.

Βιβλιογραφία

1. Νικόλαος Γλυνός, *Δομές Δεδομένων, Πανεπιστήμιο Ιωαννίνων*, 1996.
2. Χρήστος Κοίλιας, *Δομές Δεδομένων και Οργάνωση Αρχείων*. Εκδόσεις Νέων Τεχνολογιών, Αθήνα, 1993.

3. Ιωάννης Μανωλόπουλος, *Δομές Δεδομένων – μία Προσέγγιση με Pascal*, Εκδόσεις Art of Text, Θεσσαλονίκη, 1998.

4. Νικόλαος Μισυρλής, *Δομές Δεδομένων*, Αθήνα, 1993.

5. D. Brunskill and J. Turner: "*Understanding Algorithms and Data Structures*", McGraw-Hill, 1996.

6. D. E. Knuth: "*The Art of Computer Programming : Fundamental Algorithms*", Vol.1, 3rd edition, Addison Wesley, 1997.

7. M.A. Weiss: "Data Structures and Algorithm Analysis", 2nd edition, Benjamin/Cummings, 1995.

Διευθύνσεις Διαδικτύου

➨ http://hissa.ncsl.nist.gov/~black/CRCDict/

Κόμβος με ευρετήριο όρων για αλγορίθμους, Δομές Δεδομένων και Προβλήματα (Algorithms, Data Structures, and Problems Terms and Definitions for the CRC Dictionary of Computer Science, Engineering and Technology).

➨ http://www.ee.uwa.edu.au/~plsd210/ds/ds_ToC.html

Κόμβος ενός πρότυπου μαθήματος ακαδημαϊκού επιπέδου για Δομές Δεδομένων και Αλγορίθμους με παρουσίαση, εξηγήσεις και κώδικα προγραμμάτων για τις κυριότερες κατηγορίες προβλημάτων.