ΤΕΙ ΠΑΤΡΩΝ
ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ
ΤΜΗΜΑ ΕΠΙΧΕΙΡΗΜΑΤΙΚΟΥ ΣΧΕΔΙΑΣΜΟΥ
ΚΑΙ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«ΑΝΑΠΤΥΞΗ ΕΚΠΑΙΔΕΥΤΙΚΟΥ ΥΛΙΚΟΥ ΓΙΑ ΕΞ'ΑΠΟΣΤΑΣΕΩΣ ΕΚΠΑΙΔΕΥΣΗ ΣΕ ΘΕΜΑΤΑ ΔΟΜΩΝ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΟΡΓΑΝΩΣΗΣ ΑΡΧΕΙΩΝ»

ΚΑΡΑΜΠΑΣΗ ΕΛΕΝΗ
ΚΟΛΛΗΤΗΡΗ ΒΑΣΙΛΙΚΗ

ΕΠΟΠΤΕΥΩΝ ΚΑΘΗΓΗΤΗΣ
ΚΟΥΤΣΟΝΙΚΟΣ ΓΙΑΝΝΗΣ

ΠΑΤΡΑ 2006

ΕΙΣΑΓΩΓΗ

ΕΞ ΑΠΟΣΤΑΣΕΩΣ ΕΚΠΑΙΔΕΥΣΗ ΜΕ ΧΡΗΣΗ ΤΩΝ ΝΕΩΝ ΤΕΧΝΟΛΟΓΙΩΝ

Ο όρος εξ αποστάσεως εκπαίδευση στην πρωτογενή του μορφή χρησιμοποιείται για να περιγράψει μία εκπαιδευτική διαδικασία στην οποία ένα σημαντικό κομμάτι της διδασκαλίας γίνεται από τον διδάσκοντα που βρίσκεται μακριά από τον διδασκόμενο τόσο σε απόσταση όσο και σε χρόνο (Perraton, 1988).

Η εξ αποστάσεως εκπαίδευση προσφέρει τεράστιες ευκαιρίες αλλά και προκλήσεις στην εκπαίδευση της σημερινής εποχής, καθώς η ανάγκη για τη δημιουργία ελαστικών εκπαιδευτικών προγραμμάτων είναι μεγάλη. Μέσα στο γενικό πλαίσιο της γρήγορης τεχνολογικής αλλαγής και της αλλαγής των συνθηκών της αγοράς, το εκπαιδευτικό σύστημα στις περισσότερες αναπτυσσόμενες χώρες καλείται να παράσχει αυξημένες εκπαιδευτικές ευκαιρίες χωρίς αύξηση του προϋπολογισμού. Πολλά εκπαιδευτικά ιδρύματα απαντούν σε αυτήν την πρόκληση στρέφοντας την προσοχή τους στην τηλεματική για να επιλύσουν προβλήματα που σχετίζονται με τις γεωγραφικές αποστάσεις και το αυξημένο κόστος της δημιουργίας πολλαπλών παράλληλων προγραμμάτων. Έτσι λοιπόν αναπτύσσουν προγράμματα εξ αποστάσεως εκπαίδευσης.

Στις περισσότερες περιπτώσεις, η εξ αποστάσεως εκπαίδευση υλοποιείται όταν ο καθηγητής και ένας ή περισσότεροι μαθητές είναι αποκομμένοι λόγω φυσικής απόστασης και η τεχνολογία (φωνή, εικόνα, δεδομένα και εκτύπωση), συχνά σε συνδυασμό με δια ζώσης επικοινωνία, χρησιμοποιείται για να γεφυρώσει το εκπαιδευτικό χάσμα. Αυτοί οι τύποι προγραμμάτων μπορούν να παράσχουν στους ενήλικες μια δεύτερη ευκαιρία στην τριτοβάθμια εκπαίδευση, να προσεγγίσουν αυτούς που έχουν το μειονέκτημα του περιορισμένου χρόνου, της απόστασης ή της σωματικής αναπηρίας και να εμπλουτίσει με σύγχρονα δεδομένα τη βάση γνώσης των εργαζομένων στους χώρους εργασίας τους.

Σύμφωνα με τον Rumble (1989), στην εξ αποστάσεως εκπαίδευση αναγνωρίζονται τέσσερα μέρη:

- Το πρώτο μέρος περιλαμβάνει διάφορα στοιχεία όπως τον διδάσκοντα, έναν ή περισσότερους σπουδαστές, το μάθημα που διδάσκεται και ένα είδος συμβολαίου μεταξύ του διδάσκοντα και των σπουδαστών.
- Στο δεύτερο μέρος, η εξ αποστάσεως εκπαίδευση περιγράφεται ως η μέθοδος διδασκαλίας κατά την οποία οι σπουδαστές βρίσκονται σε απόσταση από το διδάσκοντα.
- Στο τρίτο μέρος υπογραμμίζεται ότι οι σπουδαστές είναι μακριά από το χώρο που προσφέρει το μάθημα.
- Στο τέταρτο μέρος τονίζεται η διαδικασία της αξιολόγησης, η οποία ίσως και να διαφέρει από αυτή της συμβατικής διδασκαλίας.

Κατά την τελευταία δεκαετία του 20ού αιώνα, ο συνήθης τρόπος μάθησης υφίσταται μια αλλαγή με την εξ αποστάσεως εκπαίδευση (Brown and Brown, 1994). Η αλλαγή συντελείται με την είσοδο της τηλεματικής, με την βοήθεια της οποίας η διδασκαλία μπορεί να γίνει την ίδια χρονική στιγμή από το διδάσκοντα στους σπουδαστές σε διαφορετικές απομακρυσμένες αίθουσες διδασκαλίας (Keegan, 1995).

Ο όρος τηλεματική αναφέρεται πλέον στις δυνατότητες χρήσης που παρέχουν τα τελευταίας τεχνολογίας επικοινωνιακά μέσα όπως οι υπολογιστές, οι κάμερες, τα μικρόφωνα, οι προβολείς φωτεινών εικόνων, οι βιντεοκασέτες, τα CD-ROMs και η τηλεόραση. Τα μέσα αυτά εξασφαλίζουν τη μεταφορά διαφορετικών τύπων πληροφοριών (εικόνας, ήχου, κειμένου) και επιτρέπουν την άμεση, ζωντανή και αμφίδρομη σύνδεση (two-way interactive communication) μεταξύ του διδάσκοντα και των σπουδαστών, οι οποίοι βρίσκονται σε διαφορετικές αίθουσες διδασκαλίας.

Τα εκπαιδευτικά ιδρύματα προσπαθούν να εκμεταλλευτούν τη ραγδαία εξέλιξη των νέων μέσων και τεχνολογιών για να προσφέρουν εξ αποστάσεως εκπαίδευση σε ένα ευρύτερο κοινό από αυτό που είναι δυνατόν να παρακολουθήσει τις σπουδές με τις συμβατικές κλασσικές μεθόδους και αναζητούν τρόπους για να βελτιώσουν την αποτελεσματικότητα της μαθησιακής διαδικασίας (Berge and Collis, 1994, Rossman, 1992).

Η διδασκαλία με τηλεματική μπορεί να συμβάλλει πολλαπλώς στην εκπαίδευση των σπουδαστών ιδιαίτερα στην Τριτοβάθμια εκπαίδευση. Μεταξύ των κυριοτέρων πλεονεκτημάτων είναι τα ακόλουθα:

- Δημιουργούνται δυνατότητες συνεργασίας και ευκαιρίες ανταλλαγής γνώσεων και πληροφοριών μεταξύ των πανεπιστημιακών τμημάτων της Ελληνικής περιφέρειας.
- Παρουσιάζονται νέες ευκαιρίες πρόσβασης σε ένα μεγαλύτερο αριθμό διαπανεπιστημιακών προγραμμάτων για τους σπουδαστές ανεξάρτητα από τη γεωγραφική τους τοποθέτηση.
- Εξοικονομούνται χρόνος και χρήματα από δαπανηρές μετακινήσεις των διδασκόντων και των σπουδαστών.

Μορφές επικοινωνίας στην εξ αποστάσεως εκπαίδευση

Η επικοινωνία ανάμεσα στην ομάδα των σπουδαστών και σε κάποιον επιστήμονα μπορεί να έχει τις εξής μορφές:

- Ασύγχρονη
- Επικοινωνία σε πραγματικό χρόνο (Real - time chat)
- Επικοινωνία σε πραγματικό χρόνο με video
- Επικοινωνία μέσα από World Wide Web.

Αναλυτικότερα έχουμε:

Ασύγχρονη: Οι εκπαιδευόμενοι, κάτω από την επίβλεψη του εκπαιδευτικού, στέλνουν e-mail για διάφορα θέματα που τους απασχολούν. Ο επιστήμονας απαντά με e-mail στους μαθητές και ο εκπαιδευτικός χρησιμοποιεί την απάντηση ως βάση για συζήτηση στην τάξη.

Επικοινωνία σε πραγματικό χρόνο (Real - Time chat) : Οι εκπαιδευόμενοι θέτουν ερωτήματα στον επιστήμονα και αυτός απαντά ενώ η εικόνα του φτάνει στις οθόνες του υπολογιστή με video μέσω του Internet.

Επικοινωνία σε πραγματικό χρόνο με video: Πραγματική επικοινωνία μέσω Internet, όπου κάθε πλευρά διαθέτει video, υπολογιστές, κάμερα και μικρόφωνα. Οι μαθητές μπορούν να επικοινωνήσουν έχοντας και εικόνα με επιστήμονες από όλο τον κόσμο.

Επικοινωνία μέσα από το World Wide Web: Οι εκπαιδευόμενοι θέτουν ερωτήματα σε κάποιον επιστήμονα απευθείας μέσα από σελίδες WWW. Οι σελίδες εμπλουτίζονται με στοιχεία που αφορούν τις ερωτήσεις των μαθητών.

Πολλοί εκπαιδευτικοί αναρωτιούνται εάν στην εξ αποστάσεως εκπαίδευση οι εκπαιδευόμενοι μαθαίνουν τα ίδια πράγματα με αυτούς που ακολουθούν την

παραδοσιακή διδασκαλία. Σε έρευνες που έχουν γίνει διαπιστώθηκε ότι συγκρίνοντας την εξ αποστάσεως εκπαίδευση με αυτήν της παραδοσιακής μπορούμε να πούμε ότι και η διδασκαλία αλλά και η μάθηση στην εξ αποστάσεως εκπαίδευση είναι το ίδιο αποτελεσματική με αυτήν της παραδοσιακής εκπαίδευσης.

Βέβαια αυτό συμβαίνει μόνο όταν οι μέθοδοι και οι τεχνολογίες χρησιμοποιούνται κατάλληλα κατά τη διδασκαλία, όταν υπάρχει διάδραση μεταξύ εκπαιδευομένων και, τέλος, όταν υπάρχει έγκαιρη ανατροφοδότηση πληροφοριών από διδάσκοντα σε διδασκόμενο.

Η παραδοσιακή διδασκαλία "πρόσωπο με πρόσωπο" έχει το πλεονέκτημα ότι οι σπουδαστές και οι διδάσκοντες είναι εξοικειωμένοι με το περιβάλλον της διδασκαλίας και η επικοινωνία είναι άμεση με αποτέλεσμα οι σπουδαστές να νιώθουν άνετα. Από την άλλη πλευρά, η διδασκαλία με τηλεματική βασίζεται σε συστήματα τηλεπικοινωνίας με υπολογιστές και οθόνες και ο τρόπος διεξαγωγής της διαφέρει από τη διδασκαλία στην παραδοσιακή τάξη.

Τεχνολογικά μέσα στην εξ αποστάσεως εκπαίδευση

Τα τεχνολογικά μέσα που χρησιμοποιούνται στη συμβατική διδασκαλία χρησιμοποιούνται και στην τηλεματική. Η διαφορά έγκειται στο γεγονός ότι τα τεχνολογικά μέσα στην τηλεματική αποτελούν απαραίτητο στοιχείο της επικοινωνίας του διδάσκοντα με τους σπουδαστές και ειδικότερα των απομακρυσμένων τάξεων.

Τα τεχνολογικά μέσα που χρησιμοποιούνται στη διδασκαλία με τηλεματική είναι ίδια για τις επιστήμες τόσο της θεωρητικής όσο και της πρακτικής κατεύθυνσης. Διαφοροποίηση υπάρχει μόνο στην περίπτωση που το μάθημα περιλαμβάνει την εξάσκηση των σπουδαστών μέσα από εργαστηριακές δραστηριότητες.

Μια μεγάλη κλίμακα τεχνολογικών επιλογών είναι διαθέσιμες σε κάθε εξ αποστάσεως εκπαιδευόμενο. Αυτές χωρίζονται σε 4 κατηγορίες:

1. Φωνή (Voice). Τα εργαλεία ήχου που χρησιμοποιούνται για την εξ αποστάσεως εκπαίδευση συμπεριλαμβάνουν τις διαδραστικές τεχνολογίες, όπως αυτές του τηλεφώνου, του audio conferencing και short wave radio. Στα παθητικά εργαλεία ήχου συμπεριλαμβάνονται οι κασέτες και το ραδιόφωνο.

2. Εικόνα (Video). Τα εργαλεία εικόνας που χρησιμοποιούνται για την εξ αποστάσεως εκπαίδευση περιλαμβάνουν slides, προπαρασκευασμένες κινούμενες εικόνες (film και video κασέτες) καθώς και κινούμενες εικόνες πραγματικού

χρόνου συνδυασμένες με ήχο (μονής ή διπλής κατεύθυνσης video με ήχο διπλής κατεύθυνσης).

- ***Τα Videotapes*** χρησιμοποιούνται για την επισήμανση εννοιών και ειδικότερα είναι χρήσιμη η προβολή τους στην περίπτωση των case studies όπου χρειάζεται ανάλυση, επεξήγηση και συζήτηση.
- ***Το Slide-projector*** χρησιμοποιείται για προβολές οπτικού υλικού. Το οπτικό υλικό (σε μορφή slides) χρησιμεύει στον εμπλουτισμό και την επεξήγηση εννοιών.
- ***Το Elmo***, ένα είδος προβολέα, επιτρέπει την προβολή στοιχείων, πληροφοριών και εικόνων από οποιοδήποτε έντυπο υλικό όπως βιβλία και περιοδικά.

3. Δεδομένα (Data). Οι υπολογιστές στέλνουν και λαμβάνουν πληροφορίες σε ηλεκτρονική μορφή. Γι' αυτό το λόγο, ο όρος "δεδομένα (data) " χρησιμοποιείται για να περιγράψει αυτήν την ευρεία κατηγορία εργαλείων διδασκαλίας. Οι εφαρμογές των υπολογιστών για την εξ αποστάσεως εκπαίδευση ποικίλουν και περιλαμβάνουν τα εξής:

- ***Υπολογιστές υποστήριξης διδασκαλίας (CAI Computer-Assisted Instruction).*** Σε αυτήν την κατηγορία ο υπολογιστής έχει το ρόλο της αυτόνομης μηχανής διδασκαλίας που παρουσιάζει τα διάφορα μαθήματα.
- ***Υπολογιστές οργάνωσης διδασκαλίας (CMI Computer-Managed Instruction).*** Σε αυτήν την κατηγορία ο υπολογιστής έχει το ρόλο να οργανώσει την διδασκαλία και να κατευθύνει την πρόοδο του εκπαιδευόμενου. Η διδασκαλία από μόνη της δεν χρειάζεται να λαμβάνεται μέσω υπολογιστή αν και η μέθοδος CAI χρησιμοποιείται συνήθως σε συνδυασμό με την μέθοδο CMI.
- ***Υπολογιστές εκπαιδευτικής μεσολάβησης (CME Computer-Mediated Education).*** Περιγράφει εφαρμογές υπολογιστών που διευκολύνουν την παράδοση της διδασκαλίας. Για παράδειγμα εφαρμογές όπως: ηλεκτρονικό ταχυδρομείο, fax, τηλεδιάσκεψη πραγματικού χρόνου μέσω ηλεκτρονικού υπολογιστή και εφαρμογές διαδικτύου.

Τεχνολογικά μέσα που χρησιμοποιούν τον όρο "data" μπορούμε να πούμε ότι θεωρούνται τα παρακάτω:

- ***Το PowerPoint της Microsoft ή το Director από την Macromedia*** αποτελούν προγράμματα κατάλληλα διαμορφωμένα για την παρουσίαση και ενίσχυση θεμάτων διδασκαλίας.

- ***Το CD-ROM*** προσφέρεται για την αποθήκευση μεγάλου όγκου πληροφοριών και η χρήση τους διευκολύνει τον διδάσκοντα καθώς μπορεί να επιλέγει και να προβάλει το συγκεκριμένο υλικό που τον ενδιαφέρει κατά περίπτωση.
- Κατάλληλα διαμορφωμένα ***Web Sites*** από τα οποία ο διδασκόμενος μπορεί να αντλήσει πληροφορίες για διάφορα θέματα που τον αφορούν.

4. Εκτύπωση (Print). Είναι ένα ιδρυτικό στοιχείο των προγραμμάτων της εξ αποστάσεως εκπαίδευσης και αποτελεί τη βάση εξέλιξης όλων των μεταγενέστερων συστημάτων. Διάφορες μορφές εκτυπώσεων μπορούν να υποστηριχθούν όπως βιβλία κειμένων, βιβλία εργασίας, οδηγοί σπουδών, αναλυτικά προγράμματα και μελέτες περιπτώσεων (Case studies).

Αν και η τεχνολογία παίζει ρόλο - κλειδί στην εξ αποστάσεως εκπαίδευση, οι εκπαιδευτικοί θα πρέπει να παραμένουν επικεντρωμένοι στο αποτέλεσμα της διδασκαλίας και όχι τόσο στην τεχνολογία που θα χρησιμοποιηθεί για τους εκπαιδευτικούς σκοπούς. Το κλειδί για τη σωστή διδασκαλία κατά την εξ αποστάσεως εκπαίδευση είναι ο εκπαιδευτικός να επικεντρωθεί στις ανάγκες του εκπαιδευομένου, στο απαιτούμενο περιεχόμενο της μάθησης, στις ανάγκες που έχει ο εκπαιδευόμενος και όλα αυτά πριν επιλέξει το τεχνολογικό ή μη μέσο διδασκαλίας. Τυπικά αυτή η συστηματική προσέγγιση θα έχει ως αποτέλεσμα την μίξη μέσων διδασκαλίας, καθένα από τα οποία θα εξυπηρετήσει ένα συγκεκριμένο σκοπό. Για παράδειγμα:

- Ένα δυνατό μέσο εκτύπωσης μπορεί να παρέχει πολλά από τα βασικά περιεχόμενα εκπαίδευσης σε μορφή κειμένου κατεύθυνσης (course text), πρόγραμμα μαθημάτων ή περίληψη σπουδών (syllabus) καθώς και ημερήσιο πρόγραμμα.
- Ο διαδραστικός ήχος (interactive audio) ή η εικονική διάσκεψη (video conferencing) μπορούν να παρέχουν διάδραση σε πραγματικό χρόνο.
- Η διάσκεψη μέσω ηλεκτρονικού υπολογιστή ή η χρήση ηλεκτρονικού ταχυδρομείου μπορούν να χρησιμοποιηθούν για:

  1) Να σταλούν μηνύματα

  2) Να γίνει παράθεση έργου

  3) Να στοχεύσουν στην επικοινωνία προς ένα ή περισσότερα μέλη της τάξης

  4) Αύξηση της διαδραστικότητας μεταξύ των εκπαιδευομένων

- Προ-ηχογραφημένες κασέτες video: μπορούν να χρησιμοποιηθούν με σκοπό την παράδοση μαθήματος και τον εικονικό καθορισμό των απαιτήσεων.
- Το FAX μπορεί να χρησιμοποιηθεί για τα εξής:

  1) Ανάθεση και διαμοιρασμός έργου

  2) Τελευταίας στιγμής ανακοινώσεις

  3) Λήψη έργου από μαθητές

  4) Παροχή μιας σωστής χρονικά ανατροφοδότησης πληροφοριών

Χρησιμοποιώντας αυτή την ολοκληρωμένη προσέγγιση, ο ρόλος του εκπαιδευτικού είναι έτοιμος να καθοριστεί πέρα από τις τεχνολογικές επιλογές. Στόχος είναι η δημιουργία ενός μείγματος από εκπαιδευτικά μέσα, ο εντοπισμός των αναγκών των εκπαιδευομένων στο σημείο της εκπαιδευτικής επίδρασης, αλλά και κατά πόσο οικονομικός είναι για τον εκπαιδευόμενο ο συγκεκριμένος τρόπος εκπαίδευσής του.

Χωρίς καμία εξαίρεση τα προγράμματα εξ αποστάσεως εκπαίδευσης ξεκίνησαν με έναν προσεκτικό σχεδιασμό, με μια εστίαση στην κατανόηση των απαιτήσεων των μαθημάτων και των αναγκών των εκπαιδευομένων.

Η κατάλληλη τεχνολογία μπορεί να επιλεγεί μόνον όταν όλα τα παραπάνω στοιχεία που αναφέραμε έχουν κατανοηθεί λεπτομερώς. Δεν υπάρχει κανένα μυστήριο στον τρόπο ανάπτυξης των αποτελεσματικών προγραμμάτων της εξ αποστάσεως εκπαίδευσης. Δεν συμβαίνουν τυχαία. Εξελίσσονται μέσα από σκληρή δουλειά αλλά και μέσα από σημαντικές προσπάθειες από κάθε άτομο ή από έναν ολόκληρο οργανισμό. Η επιτυχία των προγραμμάτων της εξ αποστάσεως εκπαίδευσης βασίζεται στις αξιόλογες προσπάθειες των μαθητών, στις προσπάθειες των υπαλλήλων υποστήριξης, στις διευκολύνσεις που παρέχουν τα τεχνολογικά μέσα, αλλά και στην υποστήριξη και στις υπηρεσίες που παρέχουν οι διαχειριστές των συστημάτων.

Κρίσιμοι παράγοντες στην εξ αποστάσεως εκπαίδευση

Οι λέξεις κλειδιά στην εξ αποστάσεως εκπαίδευση είναι οι εξής:

- Μαθητές (Students)
- Ικανότητα (Faculty)
- Διευκολύνσεις (Facilitations)
- Προσωπικό υποστήριξης (Support Staff)

- Διαχειριστές (Administrators)

Μαθητές (Students)

Ο εντοπισμός των διδακτικών αναγκών των μαθημάτων είναι ο θεμέλιος λίθος για κάθε αποτελεσματικό πρόγραμμα εξ αποστάσεως εκπαίδευσης, καθώς επίσης και το πείραμα από το οποίο κρίνονται όλες οι προσπάθειες στον τομέα αυτό. Χωρίς να αναφερθούμε στο γενικό πλαίσιο της εκπαίδευσης, ο πρωταρχικός σκοπός του μαθητή είναι να μάθει. Αυτό είναι ένα αυτονόητο έργο - αυτονόητος σκοπός - και αυτό πρέπει να γίνεται κάτω από τις καλύτερες συνθήκες απαιτώντας για την επίτευξή του σχεδιασμό και ικανότητα για ανάλυση και εφαρμογή του περιεχομένου που πρέπει ο μαθητής να διδαχθεί.

Όταν η διδασκαλία διεξάγεται εξ αποστάσεως, διαφορετικές προκλήσεις επιδρούν και αυτό γιατί οι μαθητές είναι συχνά χωρισμένοι και έτσι δεν μοιράζονται τις εμπειρίες και τα ενδιαφέροντά τους. Έχουν λίγες ευκαιρίες στο να έρθουν σε επαφή με τους καθηγητές. Έτσι λοιπόν, η διδασκαλία τους πρέπει να βασίζεται σε τεχνικές σύνδεσης έτσι ώστε να γεφυρώσουν το χάσμα που τους χωρίζει.

Ικανότητα (Faculty)

Η επιτυχία κάθε προσπάθειας για την εξ αποστάσεως εκπαίδευση βασίζεται στην ικανότητα. Σε μια παραδοσιακή αίθουσα η ευθύνη του δάσκαλου περιλαμβάνει την οργάνωση του περιεχομένου του μαθήματος, την ανάπτυξη και κατανόηση των αναγκών των μαθητών. Ιδιαίτερες προκλήσεις αντιμετωπίζουν εκείνοι που διδάσκουν εξ αποστάσεως. Για παράδειγμα, ο δάσκαλος θα πρέπει:

- Να αναπτύξει και να κατανοήσει τα χαρακτηριστικά αλλά και τις ανάγκες των εξ αποστάσεως εκπαιδευομένων.
- Να προσαρμόσει το στυλ διδασκαλίας έχοντας υπ' όψη τις ανάγκες και τις προσδοκίες των μαθητών.
- Να κατανοήσει και να αναπτύξει έργο σχετικό με τις μεθόδους διδασκαλίας που βασίζονται στην τεχνολογία, παράλληλα όμως να επικεντρωθεί στον διδασκαλικό του ρόλο.

Διευκολύνσεις (Facilitations)

Ο δάσκαλος συχνά βρίσκει ωφέλιμο να βασίζεται σε μια ιστοσελίδα που ουσιαστικά θα παίζει το ρόλο της διευκόλυνσης, γεφυρώνοντας το χάσμα μεταξύ μαθητών και δασκάλων. Για να επιδράσει σωστά ένα "διευκολυντής" θα πρέπει να καθοριστούν οι ανάγκες των μαθητών και οι προσδοκίες των δασκάλων. Το πιο σημαντικό είναι να ακολουθηθούν οι οδηγίες / ντιρεκτίβα που έχει θέσει ο καθηγητής.

Προσωπικό υποστήριξης (Support Stuff)

Αυτοί είναι και οι σιωπηλοί ήρωες της "επιχείρησης" εξ αποστάσεως εκπαίδευσης που εξασφαλίζουν ότι οι αναρίθμητες λεπτομέρειες που απαιτούνται για την επιτυχία του προγράμματος θα διεκπεραιωθούν αποτελεσματικά. Τα περισσότερα προγράμματα εξ αποστάσεως εκπαίδευσης συγκεντρώνουν λειτουργίες υποστήριξης που περιλαμβάνουν καταχώρηση στοιχείων των εκπαιδευομένων, αναπαραγωγή εκπαιδευτικού υλικού και διανομή του, παραγγελία εγχειριδίων, εξασφάλιση των δικαιωμάτων του δημιουργού και άδεια χειρισμού απορρήτων, χρονοδιάγραμμα διευκολύνσεων, κλπ. Τα πρόσωπα υποστήριξης είναι ο συνδετικός κρίκος που κρατά την προσπάθεια της εξ αποστάσεως εκπαίδευσης ενωμένη και σε μια συγκεκριμένη τροχιά.

Διαχειριστές (Administrators)

Αν και οι διαχειριστές είναι τυπικά σημαντικοί στο σχεδιασμό των προγραμμάτων εξ αποστάσεως εκπαίδευσης των εκπαιδευτικών ιδρυμάτων, συχνά χάνουν την επαφή ή παραδίδουν τον έλεγχο στους τεχνικούς διευθυντές εφόσον το πρόγραμμα διαχείρισης είναι λειτουργικό. Οι διαχειριστές της αποτελεσματικής εξ αποστάσεως εκπαίδευσης είναι ομόφωνα οι δημιουργοί, αυτοί που παίρνουν τις αποφάσεις, αυτοί που έχουν τον ρόλο του παρατηρητή. Δουλεύουν από κοντά με το τεχνικό προσωπικό και το προσωπικό υποστήριξης, εξασφαλίζοντας την αποτελεσματική ανάπτυξη των τεχνολογικών πηγών. Το πιο σημαντικό είναι ότι οι διαχειριστές, αντιλαμβανόμενοι ότι ο εντοπισμός των αναγκών των εξ αποστάσεως εκπαιδευομένων είναι δική τους απόλυτα ευθύνη, αναλαμβάνουν σε μεγάλο μέρος την ακαδημαϊκή εστίαση των αναγκών αυτών.

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΑΛΓΟΡΙΘΜΟΙ

Σκοπός κεφαλαίου

Σε αυτό το κεφάλαιο γίνεται μια γενική αναφορά στους αλγορίθμους. Ειδικότερα ποια είναι η χρήση τους, οι τρόποι παράστασής τους, οι διάφορες κατηγορίες τους. Παρουσιάζονται οι βασικές έννοιες των αλγορίθμων έτσι ώστε να μπορέσετε να προχωρήσετε στη μελέτη των επόμενων κεφαλαίων.

Λέξεις - κλειδιά

Αλγόριθμος

Διάγραμμα ροής

Ψευδοκώδικας

Προγραμματιστικές δομές

1.1. Ορισμός αλγόριθμου

Ο όρος αλγόριθμος χρησιμοποιείται για να δηλώσει μεθόδους που εφαρμόζονται για την επίλυση προβλημάτων. Ένας πιο αυστηρός ορισμός της έννοιας αυτής είναι:

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

Κάθε αλγόριθμος απαραίτητα ικανοποιεί τα επόμενα κριτήρια:

- **Είσοδος (input).** Καμία, μία ή και περισσότερες τιμές δεδομένων πρέπει να δίνονται ως είσοδοι στον αλγόριθμο. Η περίπτωση που δεν δίνονται τιμές δεδομένων εμφανίζεται όταν ο αλγόριθμος δημιουργεί και επεξεργάζεται κάποιες πρωτογενείς τιμές με τη βοήθεια των συναρτήσεων παραγωγής τυχαίων αριθμών ή με τη βοήθεια άλλων απλών εντολών.
- **Έξοδος (output).** Ο αλγόριθμος πρέπει να δημιουργεί τουλάχιστον μία τιμή δεδομένων ως αποτέλεσμα προς το χρήστη ή προς έναν άλλο αλγόριθμο.

- **Καθοριστικότητα (definiteness).** Κάθε εντολή πρέπει να καθορίζεται χωρίς καμία αμφιβολία για τον τρόπο εκτέλεσής της. Λόγου χάριν, μία εντολή διαίρεσης πρέπει να θεωρεί και την περίπτωση που ο διαιρέτης λαμβάνει τη μηδενική τιμή.
- **Περατότητα (finiteness).** Ο αλγόριθμος να τελειώνει μετά από πεπερασμένα βήματα εκτέλεσης των εντολών του. Μία διαδικασία που δεν τελειώνει μετά από ένα συγκεκριμένο αριθμό βημάτων δεν αποτελεί αλγόριθμο, αλλά λέγεται απλά υπολογιστική διαδικασία (computational procedure).
- **Αποτελεσματικότητα (effectiveness).** Κάθε μεμονωμένη εντολή του αλγορίθμου να είναι απλή. Αυτό σημαίνει ότι μια εντολή δεν αρκεί να έχει ορισθεί, αλλά πρέπει να είναι και εκτελέσιμη.

1.2. Δομή ενός αλγορίθμου

Ένας αλγόριθμος περιλαμβάνει τρία τμήματα:

Κεφαλή: Περιλαμβάνει το όνομα και τον κατάλογο των δεδομένων (εισόδου / εξόδου)

Τμήμα δηλώσεων των μεταβλητών: Γίνεται περιγραφή των εσωτερικών μεταβλητών (δεδομένων) του αλγόριθμου

Κυρίως τμήμα: Γίνεται περιγραφή των πράξεων (υπολογισμών) που εκτελούνται πάνω στα δεδομένα

Ένας αλγόριθμος περιγράφεται:

Με λεκτική περιγραφή

- Ψευδοκώδικας: Πλησιάζει στη φυσική γλώσσα και είναι κατανοητός και από μη προγραμματιστές.
- Γλώσσα προγραμματισμού: Είναι τυποποιημένη γλώσσα και είναι κατανοητή από προγραμματιστές και Η/Υ.

Διαγραμματικά

- Διάγραμμα ροής: Εφαρμόζει γραφικά δομικά στοιχεία και γίνεται εύκολη οπτικοποίηση της ροής του αλγορίθμου.

Υπάρχουν διάφοροι τρόποι για την αναπαράσταση ενός αλγορίθμου:

- με **ελεύθερο κείμενο (free text),** που αποτελεί τον πιο ανεπεξέργαστο και αδόμητο τρόπο παρουσίασης αλγορίθμου. Έτσι μπορεί εύκολα να οδηγήσει σε μη εκτελέσιμη παρουσίαση παραβιάζοντας το τελευταίο χαρακτηριστικό των αλγορίθμων, δηλαδή την αποτελεσματικότητα.
- με **διαγραμματικές τεχνικές (diagramming techniques),** που συνιστούν ένα γραφικό τρόπο παρουσίασης αλγορίθμου. Η πιο γνωστή από αυτές είναι το διάγραμμα ροής (**flowchart**). Ωστόσο η χρήση διαγραμμάτων ροής για την παρουσίαση αλγορίθμων δεν αποτελεί την καλύτερη λύση γι' αυτό και εμφανίζονται όλο και σπανιότερα στη βιβλιογραφία και πράξη.
- με **φυσική γλώσσα (natural language)** κατά βήματα. Στην περίπτωση αυτή χρειάζεται προσοχή, γιατί μπορεί να παραβιασθεί το τρίτο βασικό χαρακτηριστικό ενός αλγορίθμου, όπως προσδιορίστηκε προηγουμένως δηλαδή το κριτήριο του καθορισμού.
- με **κωδικοποίηση (coding),** δηλαδή με ένα πρόγραμμα που όταν εκτελεστεί θα δώσει τα ίδια αποτελέσματα με τον αλγόριθμο.

Οι αλγόριθμοι που παρουσιάζονται στο μάθημα είναι κωδικοποιημένοι σε γλώσσα προγραμματισμού PASCAL, η οποία θεωρείται η καταλληλότερη γλώσσα προγραμματισμού για την κατανόηση του αλγορίθμου. Σε ορισμένες περιπτώσεις, χρησιμοποιείται και η C, μια ευρύτατα χρησιμοποιούμενη γλώσσα προγραμματισμού.

Παράδειγμα

Εύρεση του μέσου όρου των θετικών αριθμών ενός πίνακα N ακεραίων

-4	8	-2	5	4	-6	2

N = 7 (μέγεθος πίνακα)

K = 4 (πλήθος θετικών)

MO = (8 + 5 + 4 + 2) / 4 = 4.75

Με τη χρήση ψευδοκώδικα

```
ΑΛΓΟΡΙΘΜΟΣ ΜΕΣΟΣ ΟΡΟΣ ΘΕΤΙΚΩΝ ΑΡΙΘΜΩΝ ΕΝΟΣ ΠΙΝΑΚΑ Ν ΑΚΕΡΑΙΩΝ
ΔΕΔΟΜΕΝΑ
    Ρ: ΠΙΝΑΚΑΣ [1..Ν] ΑΚΕΡΑΙΩΝ
    Ν, Ι, C, S: ΑΚΕΡΑΙΟΙ
    ΜΟ: ΠΡΑΓΜΑΤΙΚΟΣ
ΑΡΧΗ
    S:=0                          /*ΑΡΧΙΚΟΠΟΙΗΣΗ ΜΕΤΑΒΛΗΤΩΝ */
    C:=0
    ΓΙΑ Ι:=1 ΕΩΣ Ν ΕΠΑΝΑΛΑΒΕ
        ΕΑΝ Ρ[Ι]>0 ΤΟΤΕ           /* ΕΛΕΓΧΟΣ ΑΝ ΕΙΝΑΙ ΘΕΤΙΚΟΣ */
            S:= S + Ρ[Ι]
            C:= C + 1
        ΕΑΝ-ΤΕΛΟΣ
    ΓΙΑ-ΤΕΛΟΣ
    ΜΟ:= S / C                    /* ΥΠΟΛΟΓΙΣΜΟΣ ΜΕΣΟΥ ΟΡΟΥ */
    ΤΥΠΩΣΕ (ΜΟ)
ΤΕΛΟΣ
```

Με χρήση της γλώσσας προγραμματισμού PASCAL

```
Program Mean;
Const
    N = 100;
Var
    P: Array[1..N] Of Integer;
    I, C, S: Integer;
```

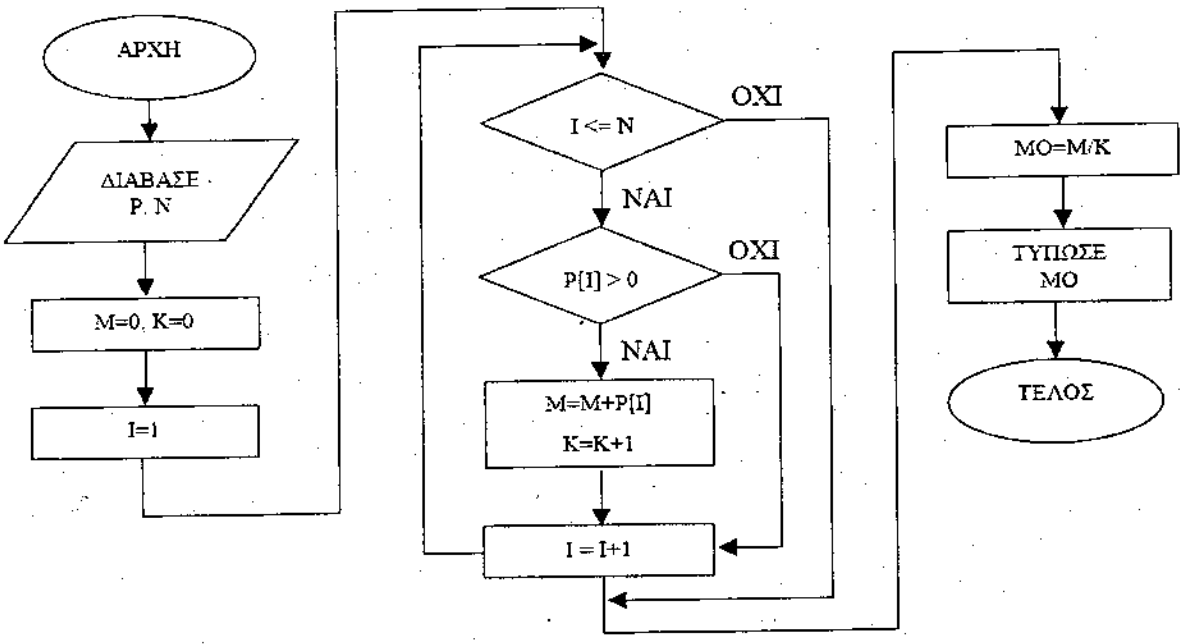
```
    MO: Real;
Begin
    {* Αρχικοποίηση μεταβλητών *}
    S:= 0;
    C:= 0;
    {* Έναρξη επανάληψης *}
    For I:= 1 To N Do
        If (P[I] > 0) Then        {* Έλεγχος αν είναι θετικός *}
        Begin
            S:= S + P[I];
            C:= C + 1;
        End;
    MO:= S / C;                   {* Υπολογισμός μέσου όρου *}
    Writeln('Mesos Oros = ', MO);
End.
```

Με χρήση της γλώσσας προγραμματισμού C

```
#include <stdio. h>
void find_mean(int P[N], int N)
{
    int i, c, s;                  /* Δήλωση μεταβλητών */
    float mo;
    s = 0; c = 0;                 /*Αρχικοποίηση μεταβλητών */
    for (i = 0; i < N; i++)
    {
        if (P[i] > 0)             /* Έλεγχος αν είναι θετικός */
```

```
        {
            s = s + P[i];
            c = c + 1;
        }
    }
    mo = s / c;                    /* Υπολογισμός μέσου όρου */
    printf("Mesos Oros = %f\n", mo);
}
```

Με Διάγραμμα Ροής Προγράμματος



1.2.1. Περιγραφή ενός αλγορίθμου

Ένας αλγόριθμος περιλαμβάνει τα ακόλουθα στοιχεία:

1. Μεταβλητές και Δομές Δεδομένων

2. Τελεστές

Αριθμητικοί τελεστές: + (πρόσθεση), – (αφαίρεση), * (πολλαπλασιασμός), / (διαίρεση)

Επιπλέον, στη γλώσσα C, στους αριθμητικούς τελεστές συμπεριλαμβάνονται και οι τελεστές % (υπόλοιπο ακέραιας διαίρεσης), ++ (αύξηση κατά 1), -- (μείωση κατά 1), ενώ στη γλώσσα PASCAL συμπεριλαμβάνονται οι τελεστές div (πηλίκο ακέραιας διαίρεσης) και mod (υπόλοιπο ακέραιας διαίρεσης).

Συγκριτικοί τελεστές: < (μικρότερο), > (μεγαλύτερο), = (ίσο, σε PASCAL), = = (ίσο, σε C), <= (μικρότερο ή ίσο), >= (μεγαλύτερο ή ίσο), <> (διάφορο, PASCAL), != (διάφορο, C)

Λογικοί τελεστές: Λογικό ΚΑΙ (&& στη γλώσσα C, AND στη γλώσσα PASCAL), Λογικό Ή (||στη γλώσσα C, OR στη γλώσσα PASCAL)

3. Εντολές απόφασης

Η εντολή IF χρησιμοποιείται για την έκφραση δομών απόφασης και στη γλώσσα PASCAL έχει δύο μορφές:

- **IF <λογική έκφραση> THEN <εντολή>;**

  Αν η λογική έκφραση είναι αληθής (TRUE) τότε εκτελείται η εντολή, αλλιώς δεν εκτελείται. Για παράδειγμα, στην εντολή:

  IF (a<0) THEN a:= -a;

  Η εντολή a:= -a θα εκτελεστεί μόνο αν το a είναι αρνητικό (ως αποτέλεσμα αυτής της εντολής, η τιμή του a θα γίνει θετική)

- **IF <λογική έκφραση> THEN <εντολή 1> ELSE <εντολή 2>;**

  Αν η λογική έκφραση είναι αληθής (TRUE) εκτελείται η εντολή 1, αλλιώς εκτελείται η εντολή 2. Για παράδειγμα, στην εντολή:

  IF (a<b) THEN write(a) ELSE write(b);

  Η εντολή αυτή εμφανίζει στην οθόνη τη μεγαλύτερη από τις τιμές των a και b.

Σε πολλές περιπτώσεις, υπάρχουν περισσότερες εναλλακτικές επιλογές, που εξαρτώνται από την τιμή μιας μεταβλητής. Στις περιπτώσεις αυτές, στη γλώσσα PASCAL χρησιμοποιείται η εντολή CASE, που έχει την παρακάτω μορφή:

CASE <έκφραση> OF

<Τιμή 1>: <εντολή 1>;

<Τιμή 2>: <εντολή 1>;

...

<Τιμή k>: <εντολή k>;

END;

Υπολογίζεται η *έκφραση* και συγκρίνεται με κάθε *τιμή* της λίστας των πιθανών τιμών της έκφρασης. Εκτελείται η εντολή εκείνη, στην οποία η τιμή συμπίπτει με την τιμή της έκφρασης και ο έλεγχος του προγράμματος μεταφέρεται στην εντολή που ακολουθεί το end. Η έκφραση και οι πιθανές τιμές της πρέπει απαραίτητα να είναι τακτικού τύπου.

Στην εντολή case μπορεί να υπάρχει πριν το end μία εναλλακτική εντολή που θα εκτελεστεί όταν καμία τιμή δεν συμπίπτει με την τιμή της έκφρασης. Η εντολή αυτή συντάσσεται με else, δηλαδή:

CASE <έκφραση> OF

<Τιμή 1>: <εντολή 1>;

...

<Τιμή k>: <εντολή k>;

ELSE <εντολή n>;

END;

4. Εντολές επανάληψης - Η εντολή FOR

Η εντολή FOR χρησιμοποιείται σε όλες τις γλώσσες προγραμματισμού για επαναληπτική εκτέλεση κάποιων εντολών, όταν ο αριθμός των επαναλήψεων είναι σταθερός. Στην PASCAL υπάρχουν δύο μορφές σύνταξης της εντολής, οι ακόλουθες:

FOR <μετρητής>:=<αρχική τιμή> TO <τελική τιμή> DO <εντολή>;

FOR <μετρητής>:=<αρχική τιμή> DOWNTO <τελική τιμή> DO <εντολή>;

Η εντολή που ακολουθεί μετά το DO εκτελείται μια φορά για κάθε τιμή του μετρητή μεταξύ των τιμών «αρχική τιμή» και «τελική τιμή», συμπεριλαμβανομένων και αυτών. Στην πρώτη περίπτωση το βήμα αύξησης του μετρητή είναι 1 και στη δεύτερη -1. Αν, στην πρώτη μορφή της FOR, η αρχική τιμή είναι μεγαλύτερη της τελικής (ή μικρότερη της τελικής, στη δεύτερη περίπτωση), η εντολή που ακολουθεί μετά το DO δεν θα εκτελεστεί καμία φορά.

Αν θέλουμε να εκτελούνται περισσότερες εντολές επαναληπτικά, χρησιμοποιούμε σύνθετη εντολή, δηλαδή ομάδα εντολών που περικλείονται μεταξύ BEGIN - END.

Ο μετρητής πρέπει να είναι **τακτικού** τύπου, δηλαδή integer, char ή boolean, ενώ η αρχική και η τελική τιμή πρέπει να είναι του ίδιου τύπου με το μετρητή.

5. Εντολές επανάληψης - Οι εντολές REPEAT-UNTIL και WHILE-DO

Οι εντολές αυτές χρησιμοποιούνται στη γλώσσα PASCAL (και σε άλλες γλώσσες προγραμματισμού, με κάποιες παραλλαγές) για επαναληπτική εκτέλεση τμήματος του προγράμματος, με βάση κάποια συνθήκη. Η σύνταξη της εντολής **REPEAT - UNTIL** είναι:

REPEAT

<εντολή 1>;

<εντολή 2>;

...

<εντολή k>;

UNTIL <συνθήκη>;

Οι εντολές 1, 2, ..., k (καλούνται και **βρόχος** ή **loop**) εκτελούνται **συνεχώς, μέχρι να ικανοποιηθεί η συνθήκη**. Κάθε φορά προηγείται η εκτέλεση των εντολών με τη σειρά που αναγράφονται και ακολουθεί ο έλεγχος της συνθήκης. Αν η συνθήκη είναι ψευδής επαναλαμβάνεται η ίδια διαδικασία, ενώ αν είναι αληθής η ροή του προγράμματος συνεχίζεται με την επόμενη μετά το UNTIL εντολή.

Η σύνταξη της εντολής **WHILE - DO** είναι:

WHILE <συνθήκη> DO <εντολή k>;

Η εντολή k εκτελείται **συνεχώς, όσο είναι αληθής η συνθήκη**. Κάθε φορά προηγείται ο έλεγχος της συνθήκης και εφόσον είναι αληθής ακολουθεί η εκτέλεση της εντολής k, αλλιώς συνεχίζεται η ροή του προγράμματος με την εντολή που ακολουθεί μετά την εντολή k. Αν

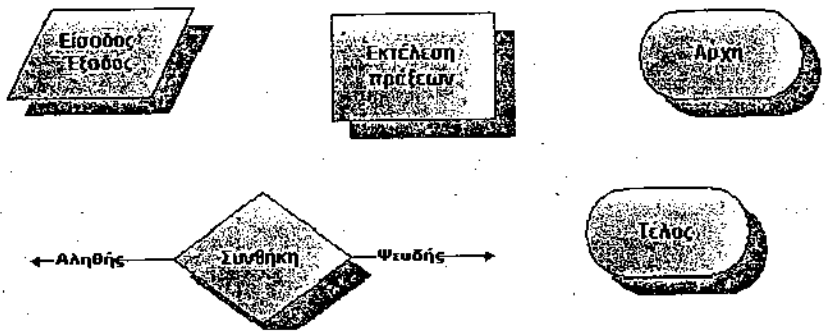
θέλουμε να εκτελούνται περισσότερες εντολές επαναληπτικά, τότε χρησιμοποιούμε σύνθετη εντολή, δηλαδή ομάδα εντολών που περικλείονται μεταξύ BEGIN - END.

1.3. Σύμβολα ροής διαγράμματος

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

- **έλλειψη,** που δηλώνει την αρχή και το τέλος του κάθε αλγορίθμου,
- **ρόμβος,** που ελέγχει μια λογική συνθήκη και έχει δύο εξόδους, μία για αληθή τιμή της συνθήκης και μία για ψευδή τιμή της συνθήκης,
- **ορθογώνιο,** που δηλώνει την εκτέλεση μίας ή περισσοτέρων πράξεων, και
- **πλάγιο παραλληλόγραμμο,** που δηλώνει είσοδο ή έξοδο στοιχείων.

Το επόμενο σχήμα αποτυπώνει όλα αυτά τα σύμβολα:



2. ΠΙΝΑΚΕΣ

Σκοπός κεφαλαίου

Οι πίνακες είναι η σημαντικότερη και η πλέον χρησιμοποιούμενη δομή δεδομένων. Στο κεφάλαιο αυτό παρουσιάζεται ο ορισμός και τα βασικά χαρακτηριστικά του πίνακα, καθώς και οι βασικότερες λειτουργίες σε πίνακες (δήλωση, ανάγνωση, εκτύπωση, διαπέραση και υπολογισμοί με βάση τα στοιχεία του πίνακα).

Λέξεις - κλειδιά

Πίνακας

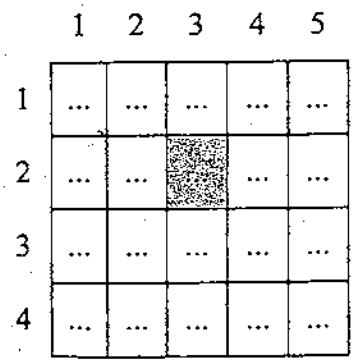
Διαστάσεις πίνακα

Ειδικοί πίνακες

Συμβολοσειρά

Εγγραφή

2.1. Ορισμός πίνακα



Ο πίνακας είναι η πιο απλή δομή δεδομένων. Περιλαμβάνει ένα σύνολο συγκεκριμένων, σταθερών θέσεων, στις οποίες τοποθετούνται τα δεδομένα.

Πλεονεκτήματα αυτής της δομής δεδομένων είναι η απλότητά της και η άμεση υποστήριξή της από όλες τις γλώσσες προγραμματισμού.

Η στατική φύση των πινάκων είναι το σημαντικότερο μειονέκτημά τους, που τους καθιστά άχρηστους σε αλγόριθμους που απαιτείται ευελιξία (δηλ. διαφοροποίηση του μεγέθους του πίνακα).

Οι διαστάσεις και οι θέσεις του πίνακα παραμένουν πάντα σταθερά. Αυτό που μπορεί να τροποποιείται είναι μόνο το περιεχόμενο τους. Έτσι, δεν μπορεί να εφαρμοστεί προσθήκη νέων στοιχείων στον πίνακα ούτε αφαίρεση στοιχείων από πίνακα.

Επειδή δεν υπάρχουν προκαθορισμένοι τύποι πινάκων τους οποίους μπορούμε να χρησιμοποιήσουμε, κάθε φορά που θα πρέπει να χρησιμοποιήσουμε ένα πίνακα θα πρέπει πρώτα να κατασκευάσουμε (ορίσουμε) τον τύπο του πίνακα και στη συνέχεια να δηλώσουμε μία μεταβλητή αυτού του τύπου.

Για τη δημιουργία ενός τύπου πίνακα, δύο είναι τα βασικά χαρακτηριστικά που πρέπει να προσδιορίσουμε:

- το μέγεθος (πόσες θέσεις έχει ο πίνακας, καθώς και να προσδιοριστεί ποια είναι η πρώτη και ποια η τελευταία θέση του), και
- ο τύπος των περιεχομένων (π.χ. ακέραιοι ή πραγματικοί αριθμοί, χαρακτήρες κλπ.).

2.1.1. Κύρια Χαρακτηριστικά Πινάκων

- Οι πίνακες μπορεί να είναι μονοδιάστατοι, διδιάστατοι, και γενικά ν-διάστατοι.
- Το μέγεθος των διαστάσεων μπορεί να διαφέρει. Σε διδιάστατο πίνακα αν το μέγεθος των δυο διαστάσεων είναι ίδιο έχουμε ένα τετραγωνικό πίνακα.
- Η αναφορά σε ένα πίνακα γίνεται με τη χρήση ενός συμβολικού ονόματος και ενός δείκτη.
- Οι πίνακες (arrays) επιτρέπουν τη χρήση πολλών στοιχείων ομοίου τύπου.
- Συνήθως το στοιχείο i ενός πίνακα A προσδιορίζεται ως A[i] ή A(i).
- Μπορούν να οριστούν και πίνακες πολλαπλών διαστάσεων με στοιχεία αντίστοιχα προσδιορισμένα ως A[i, j] κλπ.
- Οι πίνακες υλοποιούνται με τη διαδοχική φύλαξη των στοιχείων τους στη μνήμη.

Παράδειγμα

Το πρόγραμμα σε γλώσσα PASCAL που ακολουθεί υπολογίζει το άθροισμα των στοιχείων ενός μονοδιάστατου πίνακα a με N στοιχεία.

```
Program Sum_Array;
Const N = 100;
Var
    A: Array[1..N] Of Integer;
    I, SUM: Integer;
Begin
    SUM:= 0;
    For I:= 1 to N do
        SUM:= SUM + A[I];
    Writeln ('Άθροισμα = ', SUM);
End.
```

2.2. Αναγκαιότητα πινάκων

Σε ένα πρόγραμμα διαχειριζόμαστε τα δεδομένα του προβλήματος μέσω των μεταβλητών του προγράμματος, όπως στο παράδειγμα που ακολουθεί (υπολογίζει το άθροισμα 1+...+N).

```
Program I_SUM;
Var
    I, N, SUM: integer;
Begin
    Readln(N);
    SUM:= 0;
    For I:= 1 to N do
```

```
        SUM:= SUM + I;
    Writeln(SUM);
End.
```

Η χρησιμότητα των εφαρμογών Πληροφορικής όμως βασίζεται σε μεγάλο βαθμό στη δυνατότητα των Ηλεκτρονικών Υπολογιστών να επεξεργάζονται μεγάλο πλήθος δεδομένων σε λίγο χρόνο και όχι λίγα (δύο), όπως παραπάνω.

Προβλήματα των εφαρμοσμένων μαθηματικών, της γεωγραφίας, της ιατρικής, της αστρονομίας κλπ. απαιτούν την επεξεργασία χιλιάδων δεδομένων.

Ένα τέτοιο πρόγραμμα θα απαιτούσε τη δήλωση χιλιάδων μεταβλητών και τη διαχείριση των (διαφορετικών) ονομάτων τους.

```
Var
    a, b, c, …, y, z, aa, ba, …: Integer;
Begin
    a:= b * c;
    b:= c * d;
    …
End.
```

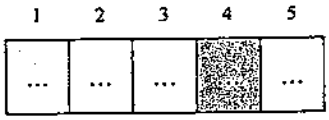
Κάτι τέτοιο οδηγεί σε προγράμματα που είναι:

- Τεράστια σε μέγεθος, γιατί όταν δεν χρησιμοποιούνται επαναληπτικές δομές, πρέπει οι ίδιες πράξεις να πραγματοποιηθούν για κάθε μία από τις πολλές μεταβλητές που χρησιμοποιούνται.
- Δύσκολα στη συγγραφή, γιατί ο προγραμματιστής είναι υποχρεωμένος να διαχειρίζεται χιλιάδες μεταβλητές. Αυτό οδηγεί σε σύγχυση και λάθη.
- Δύσκολα συντηρήσιμα, καθώς η αναγνωσιμότητα και δυνατότητα ενημέρωσης ενός προγράμματος είναι ανάλογα του συνολικού μεγέθους του προγράμματος και του πλήθους των μεταβλητών.

Επομένως, η ύπαρξη τρόπου αναπαράστασης ενός μεγάλου πλήθους ομοειδών δεδομένων με έναν απλό τρόπο είναι απαραίτητη. Αυτός ο τρόπος είναι οι ΠΙΝΑΚΕΣ.

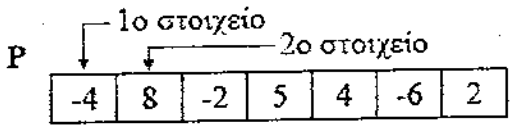
2.3. Είδη πινάκων

Α) Μονοδιάστατος Πίνακας



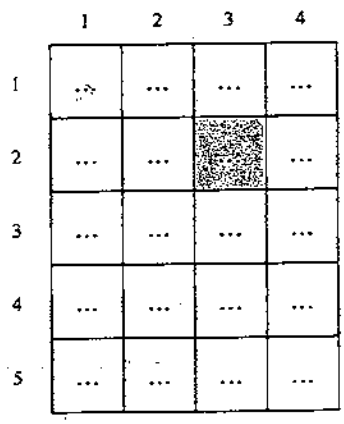
- Ένας καθορισμένος αριθμός από στοιχεία του ίδιου τύπου (ακέραιοι, χαρακτήρες, κλπ.) που αποθηκεύονται στη σειρά.
- Η πρόσβαση σε κάθε στοιχείο επιτυγχάνεται με έναν δείκτη.

Για παράδειγμα, ένας πίνακας ακεραίων 7 θέσεων



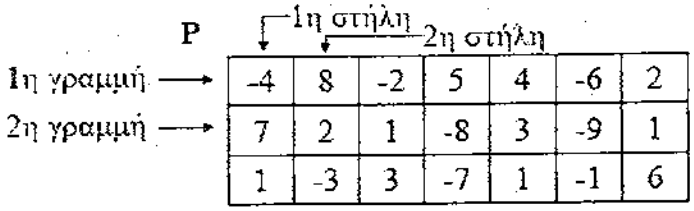
- Η ανάκτηση του i στοιχείου του πίνακα επιτυγχάνεται μέσω της αναφοράς P[i] (π.χ. P[4] = 5)
- Τα στοιχεία του πίνακα αποθηκεύονται σε γειτονικές θέσεις μνήμης (αυτό εξασφαλίζει την άμεση πρόσβαση)

Β) Πίνακας δύο Διαστάσεων



- Ένας καθορισμένος αριθμός από στοιχεία του ίδιου τύπου που οργανώνονται σε δύο διαστάσεις.
- Η πρόσβαση σε κάθε στοιχείο επιτυγχάνεται με ένα ζεύγος δεικτών (γραμμή, στήλη του πίνακα).

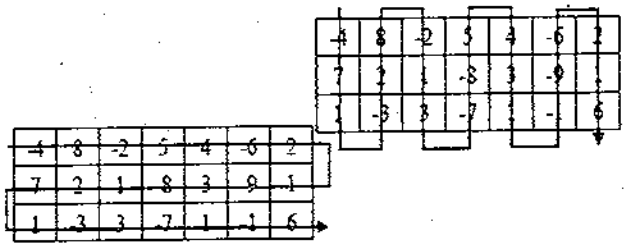
Για παράδειγμα, ένας πίνακας ακεραίων 3 γραμμών και 7 στηλών:



-4	8	-2	5	4	-6	2
7	2	1	-8	3	-9	1
1	-3	3	-7	1	-1	6

- Η ανάκτηση του i, j στοιχείου του πίνακα επιτυγχάνεται μέσω της αναφοράς P[i][j], π.χ. P[2][3]:= 1
- Η αναπαράσταση του πίνακα στη μνήμη επιτυγχάνεται με αποθήκευσή του είτε κατά γραμμές είτε κατά στήλες.

Για παράδειγμα:



2.4. Εγγραφές

Εγγραφή είναι μια συλλογή ενός καθορισμένου πλήθους αλληλοσχετιζόμενων στοιχείων (πεδίων διαφορετικού τύπου).

Για παράδειγμα, η εγγραφή ενός συνδρομητή του ΟΤΕ:

Επώνυμο	**Όνομα**	**Διεύθυνση**	**Τηλέφωνο**
Παπαδοπούλου	Μαρία	Δεινοκράτους 4	2610-585883

Πίνακας εγγραφών

Ο πίνακας εγγραφών αποτελεί ένα συνδυασμό των δομών πίνακα και εγγραφής.

Για παράδειγμα, ένας πίνακας εγγραφών του συνδρομητή του ΟΤΕ:

Επώνυμο	Όνομα	Διεύθυνση	Τηλέφωνο
Παπαδοπούλου	Μαρία	Δεινοκράτους 4	2610-485883
Φραγκάκης	Απόστολος	Ερμού 41	2610-325896
Κιούσης	Αλέξανδρος	Κορίνθου 55	2610-622544

Η πρόσβαση σε ένα στοιχείο του πίνακα εγγραφών επιτυγχάνεται με τη χρήση του τελεστή . (τελεία) μεταξύ του στοιχείου του πίνακα και του πεδίου της εγγραφής:

Εγγραφή[i].Στοιχείο_εγγραφής

Για παράδειγμα, Εγγραφή[2].Διεύθυνση = 'Ερμού 41'

2.5. Ειδικές μορφές πινάκων

α) Συμμετρικός πίνακας

Λέγεται ένας πίνακας δύο διαστάσεων (π.χ. pin), αν είναι τετραγωνικός και ισχύει:

pin[i][j] = pin[j][i]

για κάθε i και j.

Ο παρακάτω πίνακας είναι συμμετρικός:

```
6 3 7 5
3 8 2 9
7 2 4 1
5 9 1 0
```

β) Τριγωνικός πίνακας

Λέγεται ένας πίνακας δύο διαστάσεων (π.χ. mat), αν είναι τετραγωνικός και ισχύει:

mat[i][j] = 0

για κάθε i > j.

Ο παρακάτω πίνακας είναι τριγωνικός:

6	0	0	0
3	8	0	0
7	2	4	0
5	9	1	5

γ) Αραιός πίνακας

Λέγεται ένας πίνακας αν ένα πολύ μεγάλο ποσοστό των στοιχείων του είναι μηδέν. Ο παρακάτω πίνακας είναι αραιός:

0	0	2	0
3	0	0	0
0	0	0	0
0	9	0	5

Οι παραπάνω μορφές των πινάκων μπορούν να αποθηκεύονται με διαφορετικό τρόπο από ότι ένας συμβατικός πίνακας. Στους συμμετρικούς και τριγωνικούς χρειάζεται να αποθηκευτεί μόνο το ένα μισό του πίνακα. Στους αραιούς αποθηκεύεται μόνο η θέση του πίνακα (δείκτης ή αριθμοί γραμμής-στήλης), η οποία έχει μη μηδενική τιμή, καθώς και η τιμή αυτή.

2.6. Δήλωση μεταβλητής πίνακα

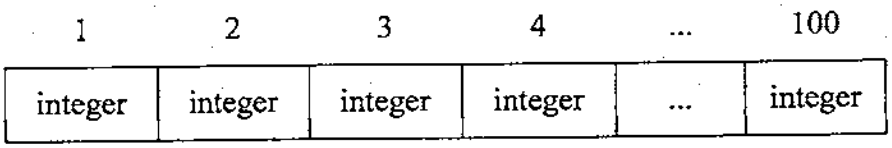
Για να χρησιμοποιήσουμε πίνακες θα πρέπει πρώτα να δηλώσουμε μεταβλητές τύπου πίνακα. Στις δηλώσεις αυτές προσδιορίζουμε τις διαστάσεις του πίνακα και τον τύπο των στοιχείων του.

Για παράδειγμα, με τις ακόλουθες εντολές δηλώνουμε τις ακόλουθες μεταβλητές πίνακα: (α) έναν πίνακα, 100 θέσεων, με περιεχόμενα ακεραίους αριθμούς, (β) δύο πίνακες, 45 θέσεων ο καθένας, με περιεχόμενα πραγματικούς αριθμούς και (γ) έναν πίνακα, 80 θέσεων, με περιεχόμενα χαρακτήρες:

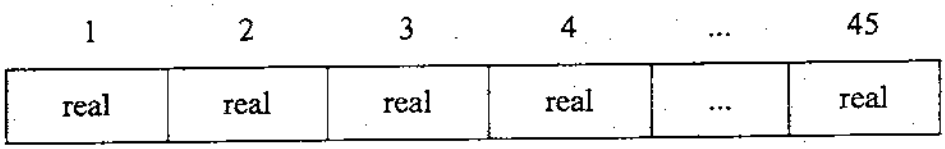
```
Var
    A: Array[1..100] Of Integer;
    B, C: Array[1..45] Of Real;
```

```
D: Array[1..80] Of Char;
```

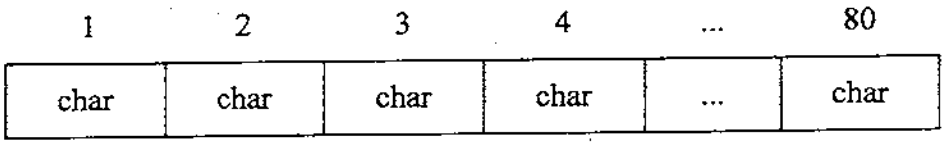
Με το παραπάνω τμήμα κώδικα δηλώνονται τέσσερις μεταβλητές (A, B, C και D). Η μεταβλητή A είναι ένας πίνακας με 100 θέσεις για ακέραιους. Οι μεταβλητές B και C είναι πίνακες με 45 θέσεις για πραγματικούς αριθμούς. Η μεταβλητή D είναι ένας πίνακας με 80 θέσεις για χαρακτήρες. Τα παραπάνω παρουσιάζονται και σχηματικά:



Σχηματική αναπαράσταση της μεταβλητής A



Σχηματική αναπαράσταση των μεταβλητών B και C



Σχηματική αναπαράσταση της μεταβλητής D

Κατά τον ορισμό των πινάκων, το τμήμα εντός των αγκυλών ([min..max]) προσδιορίζει το πλήθος των θέσεων του πίνακα, καθώς και τους δείκτες για τη διαχείριση των περιεχομένων του πίνακα, όπως θα δούμε στη συνέχεια.

2.7. Πίνακες σε γλώσσα PASCAL

Η δήλωση ενός μονοδιάστατου πίνακα στην PASCAL μπορεί να γίνει ως εξής:

VAR array_name: ARRAY[value1..value2] OF DATA_TYPE;

Οι τιμές *value1* και *value2* δηλώνουν το εύρος τιμών που μπορεί να πάρει ο δείκτης του πίνακα. Παράλληλα οι τιμές *value1* και *value2* δηλώνουν και το πλήθος των στοιχείων του πίνακα. Είναι απαραίτητο να έχει καθοριστεί επακριβώς το πλήθος των στοιχείων ενός πίνακα (δηλαδή οι *value1* και *value2* πρέπει να έχουν καθορισμένες τιμές και να μην είναι μεταβλητές).

Η προσπέλαση κάποιου στοιχείου γίνεται με χρήση του ονόματος του πίνακα (στο παραπάνω παράδειγμα, array_name) και ενός δείκτη, π.χ. array_name[i].

Η δήλωση ενός πίνακα δύο διαστάσεων στην PASCAL μπορεί να γίνει ως εξής:

VAR d2_array_name: ARRAY[Value1..Value2, Value3..Value4] OF DATA_TYPE;

Η προσπέλαση κάποιου στοιχείου γίνεται με χρήση του ονόματος του πίνακα (στο παραπάνω παράδειγμα, d2_array_name) και δύο δεικτών, π.χ. d2_array_name[i, j].

Παράδειγμα

Ο ορισμός ενός μονοδιάστατου πίνακα που περιέχει 100 ακεραίους μπορεί να γίνει ως εξής:

```
VAR array_of_integers: Array[1..100] Of Integer;
```

Εναλλακτικά:

```
Const MAX_NO_INTEGERS = 100
VAR array_of_integers: Array[1..MAX_NO_INTEGERS] Of Integer;
```

Η προσπέλαση του 50ού στοιχείου του παραπάνω πίνακα μπορεί να γίνει με τη χρήση του array_of_integers[50].

Παράδειγμα

Ο ορισμός ενός μονοδιάστατου πίνακα που περιέχει 50 χαρακτήρες μπορεί να γίνει ως εξής:

```
VAR array_of_chars: Array[1..50] Of Char;
```

Εναλλακτικά:

```
Const MAX_NO_CHARS = 50
VAR array_of_chars: Array[1..MAX_NO_CHARS] Of Char;
```

Η προσπέλαση του 30ού στοιχείου του παραπάνω πίνακα μπορεί να γίνει με τη χρήση του array_of_chars[30].

3. ΣΤΟΙΒΕΣ

Σκοπός κεφαλαίου

Σε αυτό το κεφάλαιο αναλύεται η δομή της στοίβας, παρουσιάζονται οι αλγόριθμοι διαχείρισης της στοίβας και δίνονται κάποια χαρακτηριστικά παραδείγματα χρήσης στοιβών.

Λέξεις - κλειδιά

Στοίβα

Last-In-First-Out (LIFO)

Εισαγωγή - Ώθηση

Εξαγωγή - Απώθηση

Δείκτης κορυφής - Head

3.1. Ορισμός

Στοίβα είναι η δομή που εκφράζει μία συλλογή δεδομένων με γραμμική διάταξη στην οποία η εισαγωγή στοιχείων ή ώθηση (Push) και η αφαίρεση στοιχείων ή απώθηση (Pop) πραγματοποιείται μόνο στο ένα άκρο της (κορυφή ή Head).

Ένα παράδειγμα στοίβας από την καθημερινή ζωή είναι μία στοίβα με χαρτιά για έλεγχο που έχει πάνω στο γραφείο του ένας υπάλληλος. Άλλοι υπάλληλοι μπορούν να αφήνουν χαρτιά πάνω στη στοίβα (Push). Ο υπάλληλος παίρνει πάντα το πιο πάνω χαρτί (που είναι και το πιο πρόσφατα τοποθετημένο σε αυτή) (Pop) και το ελέγχει.

Σε μία στοίβα, η διάταξη των δεδομένων δεν βασίζεται στο περιεχόμενό τους, αλλά στο χρόνο εισόδου τους στη στοίβα. Το γεγονός ότι η εισαγωγή και η αφαίρεση πραγματοποιούνται μόνο στην κορυφή της στοίβας έχει σαν αποτέλεσμα να αφαιρείται πάντα το πιο πρόσφατο στοιχείο της στοίβας. Η ταξινόμηση με κάποιο άλλο κριτήριο δεν έχει νόημα στη στοίβα.

3.1.1. Χαρακτηριστικά Στοίβας

- Η στοίβα (*Stack*) είναι μια γραμμική διάταξη στοιχείων στην οποία κάθε προσθήκη και αφαίρεση στοιχείων γίνεται μόνο στην κορυφή της.
- Στοιχεία εισάγονται με την ώθηση (*Push*).
- Στοιχεία εξάγονται με την απώθηση (*Pop*).
- Η μέθοδος φύλαξης που υλοποιεί η στοίβα λέγεται LIFO (Last In First Out).

3.2. Υλοποίηση και είδη στοίβας

Η στοίβα υλοποιεί τη λογική **LIFO** (Last In First Out). Τα δεδομένα εισάγονται στην κορυφή της στοίβας και η αφαίρεση ενός στοιχείου γίνεται πάντα από την κορυφή της στοίβας.

Η στατική στοίβα υλοποιείται με μονοδιάστατο πίνακα και χρησιμοποιεί ένα ***δείκτη Head*** που δείχνει στην κορυφή της στοίβας.

Οι λειτουργίες σε στοίβες είναι δύο:

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

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

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

3.3. Αλγόριθμοι στοίβας

Στη συνέχεια δίνονται δύο αλγόριθμοι που υλοποιούν τις λειτουργίες ώθησης και απώθησης σε μια στοίβα. Χρησιμοποιείται ο μονοδιάστατος πίνακας Stack μεγέθους N, η μεταβλητή Item που είναι το στοιχείο που θα μπει ή θα βγει από τη στοίβα, η μεταβλητή Head που περιέχει τη θέση της κεφαλής και μια λογική μεταβλητή Flag η οποία αν είναι αληθής, τότε η ζητούμενη λειτουργία έγινε με επιτυχία. Τα Stack, N, Head και Flag είναι καθολικές (global) μεταβλητές, ώστε να έχουν εμβέλεια σε όλο το πρόγραμμα (τόσο στο κυρίως πρόγραμμα όσο και στις διαδικασίες Push και Pop).

```
Procedure Push (Item: Real);
Begin
    If (Head < N) Then
    Begin
        Head:= Head + 1;
        Stack[Head]:= Item;
        Flag:= True;
    End
    Else Flag:= False;
End;
```

```
Procedure Pop (Var Item: Real);
Begin
    If (Head > 0) Then
    Begin
        Item:= Stack[Head];
        Head:= Head - 1;
        Flag:= True
    End
```

```
        Else Flag:= False;
End;
```

Η μεταβλητή Flag, αν είναι αληθής, σημειώνει την καλή εκτέλεση της ζητούμενης ενέργειας. Αν όμως είναι ψευδής τότε στην περίπτωση του αλγόριθμου Push σημαίνει ότι ο πίνακας Stack είναι πλήρης, ενώ στην περίπτωση του αλγόριθμου Pop σημαίνει ότι η στοίβα είναι άδεια.

Η στοίβα συνήθως δεν χρησιμοποιείται τόσο για καταχώρηση και αποθήκευση στοιχείων αλλά κυρίως για προσωρινή εισαγωγή και εξαγωγή με τη μέθοδο LIFO.

3.4. Χρήση

Οι στοίβες βρίσκουν μεγάλη χρήση στην επιστήμη των υπολογιστών. Χρησιμοποιούνται όχι μόνο για τη δημιουργία άλλων δομών δεδομένων αλλά και στο ίδιο το βασικό λογισμικό. Κλασσικό παράδειγμα τέτοιας εφαρμογής είναι όταν γίνεται κλήση υποπρογραμμάτων. Σε κάθε κλήση υποπρογράμματος η διεύθυνση επιστροφής φυλάσσεται σε μια στοίβα, απ' όπου ανασύρεται με την εντολή επιστροφής. Αν έχουν προηγηθεί διαδοχικές κλήσεις υποπρογραμμάτων σε κάθε εντολή επιστροφής από υποπρόγραμμα ο έλεγχος θα επιστρέφει σε κείνο το σημείο.

Οι στοίβες βρίσκουν χρήση και στη δημιουργία αναδρομικών διεργασιών. Σε κάθε αναδρομική διαδικασία, πριν από κάθε κλήση του εαυτού της φυλάσσονται σε μια στοίβα οι τιμές όλων των χρησιμοποιημένων μεταβλητών. Όταν η διαδικασία εξέλθει από το φυσιολογικό τέλος τότε ανασύρονται όλες οι τιμές από τη στοίβα και συνεχίζεται η λειτουργία της από το σημείο που κλήθηκε ο εαυτός της και κάτω. Η διαδικασία τερματίζεται οριστικά όταν η στοίβα είναι άδεια

3.5. Ασκήσεις αξιολόγησης

Άσκηση 1

Έστω ότι ο μονοδιάστατος πίνακας P που δίνεται στο παρακάτω σχήμα λειτουργεί με τη λογική της στοίβας. Ποια είναι η τιμή του δείκτη κεφαλής Head; Δώστε σχηματικά τη νέα μορφή του πίνακα P (να φαίνονται οι θέσεις και τα περιεχόμενα του πίνακα, καθώς και ο δείκτης Head) και περιγράψτε τις αλλαγές που γίνονται:

- μετά την εισαγωγή των στοιχείων 32, 22, 46
- μετά την εξαγωγή πέντε στοιχείων (αφού πρώτα έχουν εισαχθεί τα τρία παραπάνω).

1	2	3	4	5	6	7	8	9	10
13	72	29	37	26	12				

Άσκηση 2

Ποια είναι η τιμή του Head σε μια κενή στοίβα S, 10 θέσεων; Ποια θα είναι η κατάσταση της στοίβας και ποια η τιμή θα έχει ο δείκτης Head μετά την εισαγωγή των στοιχείων 23, 45, 14, 78, 51; Ποια θα είναι η κατάσταση της στοίβας και ποια θα είναι η τιμή του Head μετά την εξαγωγή των τριών στοιχείων (αφού πρώτα έχουν εισαχθεί τα παραπάνω);

4. ΟΥΡΕΣ

Σκοπός κεφαλαίου

Σε αυτό το κεφάλαιο θα αναλύσουμε τον ορισμό της ουράς και τη χρήση της. Επίσης, θα δούμε τις διαδικασίες εισαγωγής και εξαγωγής στοιχείων σε ουρές και τους αντίστοιχους αλγόριθμους. Οι ουρές (όπως και οι στοίβες) δεν αποτελούν ξεχωριστή δομή δεδομένων αλλά έναν διαφορετικό τρόπο διαχείρισης άλλων δομών δεδομένων.

Λέξεις - κλειδιά

Ουρά

First-In-First-Out (FIFO)

Εισαγωγή - Enqueue

Εξαγωγή - Dequeue

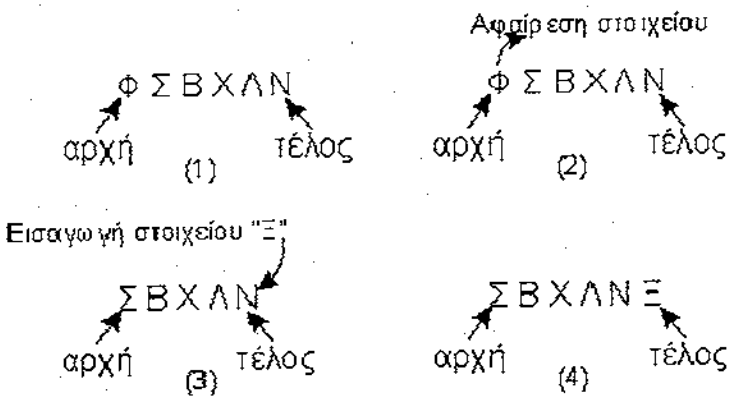
Δείκτες αρχής και τέλους

4.1. Ορισμός

Η έννοια της ουράς συναντάται πολύ συχνά στην καθημερινή ζωή. Δημιουργούνται ουρές όταν πρέπει να υπάρχει αναμονή για κάποια εξυπηρέτηση ανθρώπων, εργασιών ή προγραμμάτων. Η πιο συνηθισμένη μέθοδος εξυπηρέτησης την οποία υλοποιεί η ουρά είναι η μέθοδος FIFO (First In First Out), δηλαδή το στοιχείο της ουράς που εμφανίστηκε πρώτο, αυτό θα εξυπηρετηθεί πρώτο. Η δομή αυτή έχει μια προφανή αντιστοιχία με τις ουρές στις τράπεζες κλπ., όπου ο πρώτος που φεύγει από την ουρά είναι αυτός που βρίσκεται στην αρχή της. Ουρά μπορεί να θεωρηθεί μία σειρά από στοιχεία με δύο άκρα: τα στοιχεία μπαίνουν από το ένα άκρο (το πίσω) και βγαίνουν από το άλλο (το εμπρός).

Η ουρά είναι η δομή που εκφράζει μία συλλογή δεδομένων με γραμμική διάταξη, στην οποία η εισαγωγή γίνεται στη μία άκρη (τέλος της ουράς) και η διαγραφή στην άλλη (αρχή της ουράς).

Στο σχήμα που ακολουθεί δίνεται ένα παράδειγμα ουράς δεδομένων που είναι γράμματα.



Αρχικά η ουρά περιλαμβάνει τα γράμματα "Φ", "Σ", "Β", "Χ", "Λ" και "Ν". Το "Φ" είναι η αρχή της ουράς και το "Ν" το τέλος της. Στη συνέχεια, αφαιρείται ένα γράμμα, αυτό που βρίσκεται στην αρχή, δηλαδή το "Φ". Τώρα στην κορυφή βρίσκεται το "Σ". Στη συνέχεια προστίθεται το γράμμα "Ξ". Η προσθήκη γίνεται στο τέλος, δηλαδή μετά το "Ν".

Όπως στην καθημερινή ζωή στην ουρά που σχηματίζεται για ένα ταμείο, κάθε πελάτης που έρχεται κάθεται στο τέλος της ουράς. Όταν ελευθερωθεί το ταμείο, ο πελάτης που βρίσκεται στην αρχή της ουράς θα εξυπηρετηθεί.

Σε μια ουρά, η διάταξη των δεδομένων δεν βασίζεται στο περιεχόμενό τους, αλλά στο χρόνο εισόδου τους στην ουρά. Το γεγονός ότι η εισαγωγή γίνεται στο τέλος και η αφαίρεση στην αρχή της ουράς, έχει σαν αποτέλεσμα να γίνεται αφαίρεση πάντα του πιο παλιού στοιχείου της ουράς. Η ταξινόμηση με κάποιο άλλο κριτήριο δεν έχει νόημα στην ουρά.

4.1.1. Βασικά Χαρακτηριστικά Ουράς

- Η ουρά (queue) είναι μια γραμμική διάταξη στοιχείων στην οποία η προσθήκη στοιχείων γίνεται στο τέλος της και η αφαίρεση στοιχείων γίνεται από την αρχή της.
- Η μέθοδος φύλαξης που υλοποιεί η ουρά λέγεται FIFO (First In First Out).

4.2. Υλοποίηση

Υπάρχουν δύο τρόποι υλοποίησης για τις ουρές, με πίνακες και με λίστες. Οι λειτουργίες είναι ίδιες, απλά διαφέρει η εσωτερική αποθήκευση των στοιχείων τους στη μνήμη.

Αν η ουρά αποθηκεύεται σε πίνακα, τα στοιχεία της έχουν συνεχόμενες θέσεις στη μνήμη του υπολογιστή. Υπάρχουν ακόμα οι δείκτες, εμπρός (Front) και πίσω (Rear), που δείχνουν το πρώτο και το τελευταίο στοιχείο της ουράς αντίστοιχα.

Η είσοδος και η έξοδος των στοιχείων σε μια ουρά υλοποιούνται με τους αλγόριθμους Enqueue και Dequeue. Σε αυτούς τους αλγορίθμους, η ουρά υλοποιείται με έναν πίνακα Q, N θέσεων. Item είναι το στοιχείο που εισέρχεται στην ουρά ή εξέρχεται από την ουρά και flag είναι η λογική μεταβλητή που καθορίζει αν η ζητούμενη λειτουργία έγινε καλώς ή όχι.

4.2.1. Εισαγωγή

Για την εισαγωγή στοιχείου στην ουρά ακολουθούνται τα εξής βήματα:

- Έλεγχος αν υπάρχει ελεύθερος χώρος.
- Αύξηση του δείκτη τέλους κατά 1.
- Εισαγωγή του στοιχείου στη θέση που δείχνει ο δείκτης τέλους.

```
Procedure Enqueue (Item: Real);
Begin
    If (Rear < N) Then
    Begin
        Rear:= Rear + 1;
        Q[Rear]:= Item;
        If (Front = 0) Then Front:= 1;
        Flag:= True
    End
    Else Flag:= False
End;
```

4.2.2. Εξαγωγή

Στην εξαγωγή, αντίστοιχα, αρχικά πραγματοποιείται έλεγχος για το αν υπάρχουν στοιχεία στην ουρά. Στη συνέχεια γίνεται εξαγωγή του στοιχείου που δείχνει ο δείκτης αρχής. Τέλος αυξάνεται ο δείκτης αρχής κατά 1.

```
Procedure Dequeue (Var Item: Real);
Begin
    If (Front <> 0) Then
    Begin
        Item:= Q[Front];
        Front:= Front + 1;
        Flag:= True;
        If (Front > Rear) Then
        Begin
            Rear:= 0;
            Front:= 0
        End
    End
    Else Flag:= False
End;
```

Αξίζει να προσεχθούν στον αλγόριθμο Dequeue δύο σημεία:

1. Προοδευτικά ο δείκτης τέλους θα φθάσει τη μέγιστη τιμή της διάστασης του πίνακα και δεν θα μπορούν να εισαχθούν νέα στοιχεία, ενώ η ουρά μπορεί να διαθέτει ελεύθερο χώρο στην αρχή της. Για το λόγο αυτό απαιτείται η μετακίνηση όλων των στοιχείων στην αρχή του πίνακα. Αυτή η μετακίνηση μπορεί να γίνεται με μια ρουτίνα, η οποία να καλείται αυτόματα όταν διαπιστωθεί ότι δεν υπάρχει χώρος για την εισαγωγή νέου στοιχείου. Έτσι αδυναμία εισαγωγής στοιχείου θα υπάρξει μόνο όταν ο πίνακας Q είναι πλήρης.

2. Όταν εξέλθει και το τελευταίο στοιχείο της ουράς, ο δείκτης Front γίνεται μεγαλύτερος από τον Rear κατά 1. Το γεγονός αυτό εκμεταλλεύεται ο αλγόριθμος για να ελέγξει, αν η ουρά είναι άδεια σε επόμενη αίτηση εξαγωγής. Με την ευκαιρία αυτή αποδίδονται οι αρχικές τιμές μηδέν και στους δύο δείκτες, καθώς η ουρά πλέον είναι άδεια. Με τον τρόπο αυτό αποφεύγεται η συχνή μετατόπιση των στοιχείων για την εύρεση νέων θέσεων.

Η μετακίνηση αυτή μπορεί να αποφεύγεται αν χρησιμοποιηθεί η δομή της κυκλικής ουράς (circular queue). Η ουρά αυτή είναι στην πραγματικότητα ένας **δακτύλιος** (ring). Ένα μειονέκτημα είναι ότι πρέπει να γίνεται έλεγχος ώστε να μην επικαλυφθούν τα πρώτα από τα τελευταία στοιχεία της ουράς.

5. ΛΙΣΤΕΣ

Σκοπός κεφαλαίου

Σε αυτό το κεφάλαιο αναλύεται η δομή της λίστας και οι διαδικασίες εισαγωγής και διαγραφής κόμβων σε μια λίστα με τη χρήση δεικτών.

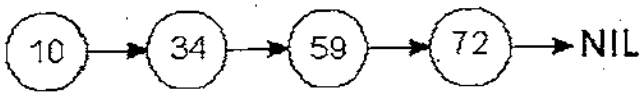
Λέξεις - κλειδιά

Δυναμική δομή

Συνδεδεμένη λίστα

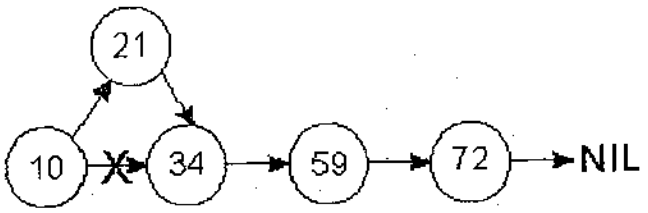
Δείκτης επόμενου κόμβου

5.1. Ορισμός λίστας

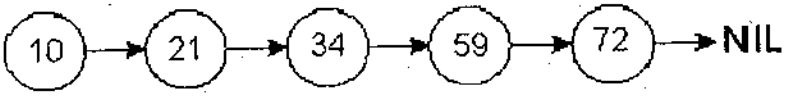


Συνδεδεμένη λίστα είναι ένα σύνολο από συνδεδεμένους κόμβους, όπως φαίνεται στο παραπάνω σχήμα. Κάθε κόμβος περιέχει δεδομένα (π. χ. μία απλή μεταβλητή ή μία εγγραφή) και μία σύνδεση προς τον επόμενο κόμβο. Οι συνδέσεις είναι δείκτες, δηλαδή οι διευθύνσεις του επόμενου κόμβου στη μνήμη. Ένας δείκτης με ειδική τιμή, καθορισμένη με σύμβαση (για παράδειγμα NIL) δείχνει το τέλος της λίστας. Ο τελευταίος κόμβος, επομένως, δεν έχει σύνδεση προς κάποιον άλλο κόμβο (ο σύνδεσμος προς τον επόμενο κόμβο είναι NIL). Η θέση του πρώτου κόμβου (κεφαλή) της λίστας είναι πάντα γνωστή, για να είναι δυνατή η χρησιμοποίηση της λίστας.

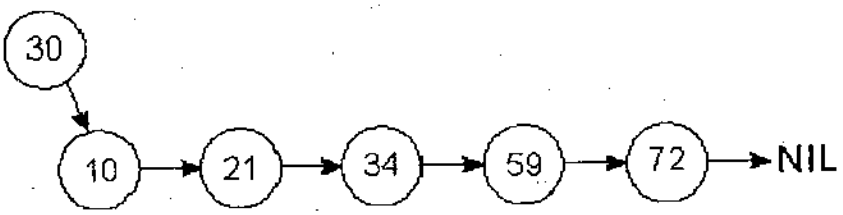
Οι συνδέσεις μεταξύ των κόμβων είναι δυναμικές, δηλαδή μπορούν να αλλάζουν. Αν λόγου χάρη στο παραπάνω παράδειγμα είναι επιθυμητό να προστεθεί ένας κόμβος με τον αριθμό 21, τότε για να παραμείνει ταξινομημένη η λίστα, θα πρέπει να παρεμβληθεί μεταξύ του πρώτου και του δεύτερου κόμβου, όπως φαίνεται στο σχήμα που ακολουθεί.



Ο πρώτος κόμβος συνδέεται με το νέο και ο νέος με τον δεύτερο. Φυσικά, η παλιά σύνδεση (πρώτου προς δεύτερο) παύει να ισχύει. Έτσι, η μορφή της λίστας θα είναι πλέον, όπως φαίνεται στο σχήμα που ακολουθεί.

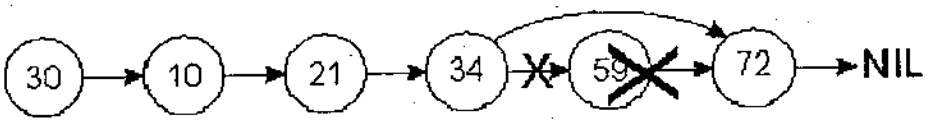


Σε μη ταξινομημένες λίστες η εισαγωγή ενός κόμβου μπορεί να γίνει πιο απλά. Να τεθεί π.χ. κεφαλή ο νέος κόμβος και να συνδεθεί προς την παλιά κεφαλή. Στο παράδειγμα που ακολουθεί προσθέτουμε κατ' αυτό τον τρόπο ένα κόμβο με το 30.



Στην περίπτωση αυτή μεταβάλλεται η αρχή της λίστας.

Η διαγραφή ενός κόμβου επιτυγχάνεται με την παράλειψή του από την σύνδεση που είχε ο προηγούμενος προς αυτόν και τη διαγραφή του, όπως φαίνεται στο παρακάτω σχήμα.



5.1.1. Κύρια Χαρακτηριστικά Λίστας

- Κάθε κόμβος περιέχει εκτός από τα δεδομένα του και έναν δείκτη.
- Οι κόμβοι φυλάσσονται σε τυχαίες θέσεις στη μνήμη, σε αντίθεση με τους πίνακες στους οποίους τα στοιχεία φυλάσσονται σε συνεχόμενες θέσεις στη μνήμη.
- Ο δείκτης κάθε κόμβου είναι η διεύθυνση του επόμενου κόμβου.
- Η διαχείριση της μνήμης γίνεται συνήθως από ειδικό υποσύστημα, τον κατανεμητή μνήμης (memory allocator).

- Σε περιπτώσεις όπου η μνήμη δεν ελευθερώνεται μέσω του κατανεμητή, ένα άλλο υποσύστημα φροντίζει για την αποκομιδή αχρήστων (garbage collection).

5.2. Ο τύπος δεδομένων δείκτη

Για να κατανοήσουμε καλύτερα τη λειτουργία της συνδεδεμένης λίστας, είναι απαραίτητο να εξηγήσουμε τον τύπο του δείκτη.

Ο δείκτης (*pointer*) είναι ένας τύπος δεδομένων που χρησιμοποιείται για να απεικονίσει απευθείας τη μνήμη του υπολογιστή. Στην PASCAL ο δείκτης σε κάποιο τύπο (για παράδειγμα σε έναν ακέραιο - Integer) ορίζεται ως εξής:

```
var q: ^integer;
```

5.2.1. Αρχικοποίηση

Πριν χρησιμοποιηθεί ένας δείκτης, θα πρέπει να δείχνει σε καθορισμένο χώρο στη μνήμη. Αν ο χώρος αυτός δεν υπάρχει, τότε θα πρέπει να δημιουργηθεί. Αυτό γίνεται με τη χρήση της διαδικασίας new, με την οποία αποδίδεται χώρος στο πρόγραμμα από το Λειτουργικό Σύστημα. Μπορεί επίσης να γίνει εκχώρηση στο δείκτη της τιμής ενός άλλου δείκτη.

Παράδειγμα:

```
new(p);
q:= p;
```

5.2.2. Χρήση

Μπορούμε να διαβάσουμε ή να αλλάξουμε τα στοιχεία στη θέση που δείχνει ο δείκτης χρησιμοποιώντας το όνομά του μαζί με το σύμβολο ^

Παράδειγμα:

```
p^:= 3;
writeln(p^);
```

5.3. Λειτουργίες

Στη συνέχεια περιγράφονται οι κυριότερες λειτουργίες σε λίστες.

Στους αλγορίθμους που ακολουθούν, σε κάθε κόμβο η πληροφορία και ο δείκτης στον επόμενο κόμβο ομαδοποιούνται σε μια εγγραφή. Χρησιμοποιούνται οι εξής συμβολισμοί: αν P είναι ο δείκτης ενός κόμβου, τότε P^.Info είναι το περιεχόμενο του κόμβου και P^.Next είναι ο δείκτης στον επόμενο. Αρχικά δίνεται η δομή των κόμβων της λίστας:

```
Type node_ptr = ^node;
node = record
        Info: integer;
        Next: node_ptr;
End;
```

5.3.1. Εισαγωγή

Η εισαγωγή (insertion) νέου κόμβου γίνεται με την αλλαγή του δείκτη του προηγούμενου κόμβου, ώστε να οδηγεί στον νέο, ενώ συγχρόνως ο δείκτης του νέου κόμβου οδηγεί στον επόμενο. Στον αλγόριθμο Insert_List προκειμένου να εισάγουμε ένα νέο κόμβο πρέπει να γνωρίζουμε το περιεχόμενο του κόμβου (έστω num) καθώς και τη θέση στην οποία θα τοποθετηθεί στη λίστα. (έστω p ο δείκτης του κόμβου μετά τον οποίο θα γίνει η εισαγωγή).

```
Procedure Insert_ List (p: node_ptr; num: integer);
        Var q: node_ptr;
Begin
        new (q);                    {* Δημιουργεί έναν νέο κόμβο με διεύθυνση q.*}
        q^. Info:= num;
        q^. Next:= p^. Next;
        p^. Next:= q;
End;
```

5.3.2. Διαγραφή

Η διαγραφή (deletion) ενός κόμβου μιας λίστας γίνεται βάζοντας το δείκτη του προηγούμενου κόμβου να δείχνει στον επόμενο. Και εδώ πρέπει να είναι γνωστή η θέση του κόμβου που θα διαγραφεί.

Στον αλγόριθμο Delete_Node, με p συμβολίζεται ο κόμβος που είναι πριν από αυτόν που θέλουμε να διαγραφεί, με q συμβολίζεται ο δείκτης του κόμβου που θέλουμε να διαγραφεί και το μόνο που γίνεται είναι ο δείκτης του p να δείχνει πλέον στον επόμενο του q. Έτσι ο κόμβος q αποσυνδέεται από τη λίστα, αλλά δεν διευκρινίζεται ποια είναι η τύχη του κόμβου αυτού, όπως και κάθε άλλου που θα ακυρωθεί στο μέλλον. Προκύπτει λοιπόν πρόβλημα διαχείρισης των ακυρωμένων κόμβων. Σε κάποιες εκδόσεις της PASCAL διατίθεται η διαδικασία dispose(s) με την οποία διαγράφεται οριστικά ο κόμβος στη διεύθυνση s, δηλαδή επιστρέφεται ο χώρος αυτός στο Λειτουργικό Σύστημα για να χρησιμοποιηθεί αλλού.

```
Procedure Delete_Node (p: node_ptr);
    Var q: node_ptr;
Begin
    q:= p^. Next;
    p^. Next:=q^. Next;
    dispose(q);
End;
```

5.3.3. Αναζήτηση

Η αναζήτηση (searching) βασίζεται στη διάσχιση των δεδομένων της λίστας μέσω των δεικτών.

5.3.4. Συνένωση

Στη συνένωση (concatenation), δύο λίστες συνδέονται σε μία όταν ο τελευταίος κενός δείκτης της πρώτης τροποποιηθεί, ώστε να δείχνει στον πρώτο κόμβο της δεύτερης.

5.3.5. Αντιστροφή

Η αντιστροφή (invertion) αφορά στην αντιστροφή της σειράς των κόμβων, έτσι ώστε ο πρώτος να γίνει ο τελευταίος. Ο αλγόριθμος που υλοποιεί τη λειτουργία της αντιστροφής σε μια λίστα χρησιμοποιεί τρεις δείκτες, p, q και r, που έχουν τις θέσεις τριών διαδοχικών κόμβων της λίστας.

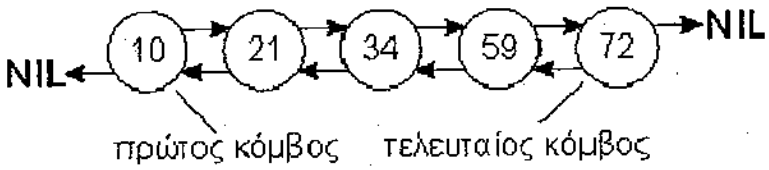
```
Procedure Inversion (var start: node_ptr);
    Var p, q, r: node_ptr;
Begin
    p:= start;
    q:= NIL;
    while p <> NIL do
    Begin
        r:= q;
        q:= p;
        p:= p ^.Next;
        q ^.Next:= r;
    End;
    start:= q;
End;
```

5.4. Παραλλαγές συνδεδεμένης λίστας

Υπάρχουν μορφές λιστών με ιδιαίτερα χαρακτηριστικά, για την ικανοποίηση συγκεκριμένων απαιτήσεων. Διακρίνουμε τις παρακάτω ειδικές μορφές συνδεδεμένων λιστών:

5.4.1. Διπλά συνδεδεμένη λίστα (doubly linked list)

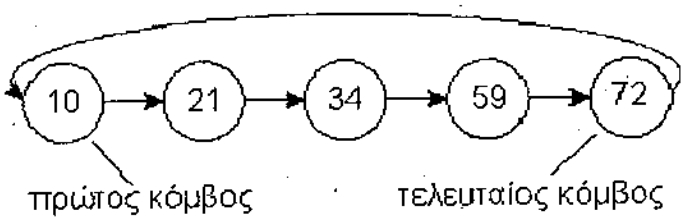
Κάθε κόμβος της λίστας έχει δείκτες στον προηγούμενο και στον επόμενο κόμβο. Επομένως, στις διπλά συνδεδεμένες λίστες κάθε κόμβος συνδέεται με τον επόμενο κόμβο, αλλά και με τον προηγούμενό του. Ένα παράδειγμα διπλά συνδεδεμένης λίστας δίνεται στο παρακάτω σχήμα.



5.4.2. Κυκλικά συνδεδεμένη λίστα

Ο τελευταίος κόμβος της λίστας δείχνει ξανά στον πρώτο. Η δομή αυτή ονομάζεται και δακτύλιος (*ring*).

Οι κυκλικά συνδεδεμένες λίστες έχουν το χαρακτηριστικό ότι ο τελευταίος κόμβος της λίστας, αντί για NIL έχει ως επόμενο τον πρώτο κόμβο. Μία τέτοια λίστα παρουσιάζεται στο σχήμα που ακολουθεί.



5.4.3. Κυκλικά διπλά συνδεδεμένη λίστα

Γίνεται σύνθεση των δύο παραπάνω παραλλαγών. Κάθε κόμβος της λίστας έχει δείκτες στον προηγούμενο και στον επόμενο κόμβο, αλλά στον τελευταίο κόμβο ο δείκτης προς τον επόμενο δείχνει στον πρώτο, ενώ στον πρώτο κόμβο ο δείκτης προς τον προηγούμενο δείχνει στον τελευταίο.

5.5. Χρήση συνδεδεμένης λίστας

Οι λίστες χρησιμοποιούνται για τη δυναμική αποθήκευση δεδομένων. Γενικά υπάρχουν δύο τρόποι να αποθηκευτούν συλλογές από συσχετιζόμενα δεδομένα, ως **πίνακες (arrays)** και ως **λίστες (lists)**. Το πρόβλημα με τους πίνακες είναι ότι το μέγεθός τους είναι αμετάβλητο. Έτσι, προκειμένου να αποθηκεύσουμε ένα σύνολο δεδομένων με χρήση πινάκων θα πρέπει

να γνωρίζουμε προκαταβολικά το πλήθος τους, έτσι ώστε να δεσμεύσουμε την απαραίτητη μνήμη για τον πίνακα. Στην πράξη, όμως, το μέγεθος αυτό δεν είναι πάντα γνωστό. Θα μπορούσαμε βέβαια εξ αρχής να δεσμεύσουμε μια μεγάλη ποσότητα μνήμης τέτοια ώστε να καλύψει οποιαδήποτε ποσότητα δεδομένων μπορεί να προκύψει στο πρόβλημά μας. Η λύση όμως αυτή μειονεκτεί. Κατ' αρχήν δεσμεύουμε περισσότερη μνήμη από όση χρειαζόμαστε, ενώ μελλοντικά ακόμα και αυτή μπορεί να αποδειχθεί ανεπαρκής.

Με τις **λίστες**, αντίθετα, έχουμε **δυναμική** δέσμευση μνήμης. Οι λίστες λειτουργούν με τέτοιο τρόπο που ο χώρος που απαιτείται για κάθε στοιχείο δεδομένων αποκτάται δυναμικά τη στιγμή που το χρειάζεται. Έτσι δεσμεύουμε ακριβώς τη μνήμη που χρειαζόμαστε και όχι παραπάνω, ενώ το πλήθος των δεδομένων που μπορούν να φορτωθούν στη μνήμη περιορίζεται από το φυσικό μέγεθος της μνήμης και από τίποτε άλλο.

5.6. Ασκήσεις αξιολόγησης

Άσκηση 1

Έχουμε μια λίστα με κόμβους που αντιπροσωπεύουν τους πελάτες μιας επιχείρησης. Κάθε κόμβος της λίστας είναι μια εγγραφή (record) με στοιχεία: Κωδικός πελάτη (συμβολοσειρά), Επωνυμία πελάτη (συμβολοσειρά), Υπόλοιπο για πληρωμή (πραγματικός αριθμός) και Δείκτης στον επόμενο κόμβο (δείκτης). Γράψτε σε γλώσσα PASCAL τον αλγόριθμο που υπολογίζει το συνολικό υπόλοιπο για όλους τους πελάτες.

Άσκηση 2

Δίνεται συνδεδεμένη λίστα με στοιχεία ακέραιους αριθμούς, που είναι ταξινομημένοι σε φθίνουσα διάταξη (ο μεγαλύτερος αριθμός βρίσκεται στην αρχή, δηλαδή στον πρώτο κόμβο της λίστας). Γράψτε σε γλώσσα PASCAL τον αλγόριθμο αναζήτησης στη λίστα του στοιχείου με τιμή Κ. Η δομή των κόμβων της λίστας είναι:

```
TYPE Pointer = ^ Node;

Node = RECORD

Number: Integer;

Next: Pointer;

END;
```

6. ΔΕΝΤΡΑ

Σκοπός κεφαλαίου

Σε αυτό το κεφάλαιο παρουσιάζονται τα δέντρα, με έμφαση στα δυαδικά δέντρα αναζήτησης. Παρουσιάζονται οι τρόποι παράστασης στη μνήμη, η αναζήτηση κόμβου, οι μέθοδοι διάσχισης κ.ά.

Λέξεις - κλειδιά

Δέντρο

Κόμβος

Δυαδικά δέντρα

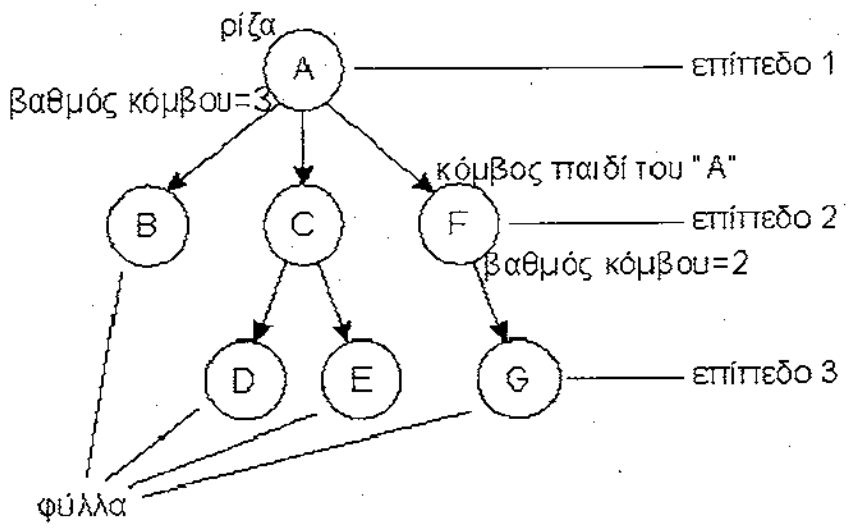
Υποδέντρο

6.1. Ορισμός

Δέντρο είναι μια δομή που περιλαμβάνει κόμβους δεδομένων με ιεραρχική διάταξη. Στο πρώτο επίπεδο υπάρχει μόνο ένας κόμβος, η ρίζα. Κάθε κόμβος μπορεί να συνδέεται με έναν ή περισσότερους κόμβους, τους κόμβους - παιδιά του. Κάθε κόμβος έχει μόνο έναν κόμβο - γονέα, εκτός από τη ρίζα, η οποία είναι ο μόνος κόμβος χωρίς κόμβο - γονέα. Οι κόμβοι που δεν έχουν κόμβους - παιδιά ονομάζονται φύλλα του δέντρου.

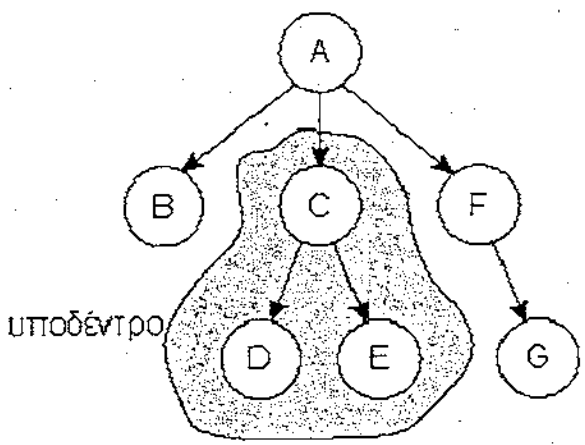
Το μεγαλύτερο επίπεδο στο οποίο μπορεί να βρίσκεται ένας κόμβος του δέντρου είναι το βάθος (ή ύψος) του δέντρου. Βαθμός ενός κόμβου του δέντρου είναι το πλήθος των κόμβων - παιδιών του. Βαθμός του δέντρου είναι ο μέγιστος από τους βαθμούς όλων των κόμβων του δέντρου.

Στο σχήμα που ακολουθεί δίνεται ένα παράδειγμα δέντρου.



Στο παραπάνω παράδειγμα το "A" είναι η ρίζα του δέντρου. Έχει τρεις κόμβους - παιδιά, το "B", το "C" και το "F", οπότε ο βαθμός της ρίζας είναι 3. Ο κόμβος "C" έχει δύο κόμβους - παιδιά (βαθμός κόμβου 2), το "D" και το "E". Ο κόμβος "F" έχει ένα κόμβο - παιδί, το "G". Οι κόμβοι "B", "D", "E" και "G" είναι κόμβοι - φύλλα (δεν έχουν κόμβους - παιδιά). Ο βαθμός του δέντρου είναι 3 και το βάθος επίσης 3.

Κάθε τμήμα του δέντρου από έναν κόμβο και κάτω ονομάζεται υποδέντρο του δέντρου. Στο ακόλουθο σχήμα ορίζεται ένα υποδέντρο του παραπάνω δέντρου.



6.1.1. Ορολογία

Ένα *δέντρο (tree)* είναι ένα σύνολο από *κόμβους (nodes)* που ενώνονται με *ακμές (edges)* και εκπληρεί την εξής ιδιότητα: αν ορίσουμε ως *μονοπάτι (path)* ένα σύνολο κόμβων ενωμένων με ακμές και ένα συγκεκριμένο κόμβο ως *ρίζα (root)* του δέντρου τότε κάθε άλλος κόμβος ενώνεται με τη ρίζα με ένα και μόνο ένα μονοπάτι.

Αν παραστήσουμε το δέντρο με τη ρίζα προς τα πάνω τότε κάθε κόμβος - εκτός από τη ρίζα - θα έχει ακριβώς ένα κόμβο από πάνω του ο οποίος και ονομάζεται *γονέας* του *(parent)*. Αντίστοιχα οι κόμβοι που βρίσκονται κάτω από τον γονέα ονομάζονται *παιδιά* του *(children)*.

Βαθμός (degree) ενός κόμβου ορίζεται ως ο αριθμός των παιδιών του. Βαθμός ενός δέντρου είναι ο μέγιστος βαθμός όλων των κόμβων του.

Κόμβοι χωρίς παιδιά ονομάζονται *τερματικοί κόμβοι (terminal nodes)* ή *εξωτερικοί κόμβοι (external nodes)* ή *φύλλα (leaves)*. Κόμβοι με παιδιά ονομάζονται μη *τερματικοί κόμβοι (non-terminal nodes)* ή *εσωτερικοί κόμβοι (internal nodes)* ή *κλαδιά (branches)*.

Επίπεδο (level) ενός κόμβου ορίζεται ως ο αριθμός των κόμβων στο μονοπάτι από τον κόμβο μέχρι τη ρίζα συμπεριλαμβανομένης και της ρίζας. Το μέγιστο επίπεδο των κόμβων του δέντρου ονομάζεται *βάθος (depth)* ή *ύψος (height)* του δέντρου.

Υποδέντρο (subtree) είναι ένα δέντρο το οποίο σχηματίζεται θέτοντας ως ρίζα κάποιον κόμβο του δέντρου και περιλαμβάνοντας όλους τους κόμβους από εκεί και κάτω.

Ένα δέντρο του οποίου κάθε κόμβος έχει δύο το πολύ ακμές ονομάζεται *δυαδικό δέντρο (binary tree)*.

6.1.2. Ποσοτικά στοιχεία

Ένα δέντρο βαθμού d και ύψους h μπορεί να έχει το πολύ $(d^h -1)/(d-1)$ κόμβους. Αυτό γίνεται επειδή στο επίπεδο 0 υπάρχει μόνο ένας κόμβος (η ρίζα), στο επίπεδο 1 μπορεί να υπάρχουν το πολύ d κόμβοι (ο μέγιστος αριθμός παιδιών ενός κόμβου), στο επίπεδο 2 μπορεί να υπάρχουν το πολύ d^2 κόμβοι κ.ο.κ. Άρα ο μέγιστος αριθμός κόμβων είναι:

$$1 + d + d^2 + d^3 + \ldots + d^{h-1} = (d^h - 1) / (d - 1)$$

Ένα δέντρο που περιέχει τον μέγιστο αριθμό κόμβων λέγεται *πλήρες*[1]. Με εφαρμογή του προηγούμενου τύπου προκύπτει ότι ο αριθμός των κόμβων ενός πλήρους δυαδικού δέντρου ύψους h είναι $2^h - 1$.

[1] Συμβατικά ορίζουμε ως πλήρες και ένα δέντρο που έχει σε όλα τα επίπεδα τον μέγιστο αριθμό κόμβων πλην του τελευταίου επιπέδου, με την προϋπόθεση ότι στο τελευταίο επίπεδο οι υπάρχοντες κόμβοι καταλαμβάνουν την αριστερή πλευρά του δέντρου.

Άμεση συνέπεια του προηγούμενου είναι ότι ένα πλήρες δυαδικό δέντρο με n κόμβους έχει ύψος $\lceil \log_2(n+1) \rceil$, όπου με τον συμβολισμό $\lceil x \rceil$ εννοούμε τον πλησιέστερο προς τα πάνω στον x ακέραιο. Για παράδειγμα, $\lceil 2.8 \rceil = 3$ και $\lceil 3 \rceil = 3$.

Μήκος μονοπατιού ενός κόμβου λέγεται ο αριθμός των ακμών που πρέπει να περάσουμε για να φθάσουμε από τη ρίζα στον κόμβο αυτό. Το μήκος μονοπατιού ισούται με το επίπεδο του κόμβου μείον 1.

Εσωτερικό μήκος μονοπατιού ενός δέντρου είναι το άθροισμα του μήκους μονοπατιών όλων των κόμβων του δέντρου.

Μέσο μήκος μονοπατιού είναι το πηλίκο του εσωτερικού μήκους μονοπατιού δια το πλήθος των κόμβων.

6.2. Αναπαράσταση των δέντρων στη μνήμη του υπολογιστή

Τα δέντρα χρησιμοποιούνται για την αναπαράσταση δεδομένων (κόμβοι) και των συσχετίσεών τους (ακμές). Ένας τρόπος αναπαράστασης δέντρου στη μνήμη είναι με τη χρήση συνδεδεμένης λίστας. Κάθε κόμβος της λίστας αποτελείται από τα δεδομένα που περιέχει και από ένα σύνολο δεικτών ίσων σε πλήθος με το βαθμό του δέντρου. Οι δείκτες δείχνουν στα παιδιά του κόμβου. Αν δεν υπάρχει κάποιο παιδί, τότε ο αντίστοιχος δείκτης είναι κενός (NIL).

Ένας άλλος τρόπος εφαρμόζεται στα πλήρη δυαδικά δέντρα και κάνει χρήση ενός μονοδιάστατου πίνακα. Στηρίζεται στην παρατήρηση ότι είναι πολύ αποδοτικό να κρατηθούν $2^h - 1$ θέσεις μνήμης, όπου h το ύψος του δέντρου. Σε κάθε θέση μπαίνει ένας κόμβος και η σχέση με τον γονέα και τα παιδιά του καθορίζεται από τις θέσεις.

Για την καταχώρηση των κόμβων πρώτα αριθμούνται κατά αύξουσα τάξη αρχίζοντας από το επίπεδο 1. Με αυτή τη σειρά οι κόμβοι αποθηκεύονται σε ένα μονοδιάστατο πίνακα info, δηλαδή ο k-στός κόμβος είναι ο info[k].

Για ένα πλήρες δυαδικό δέντρο ύψους h, με πλήθος κόμβων $N = 2^h - 1$, που έχει καταχωρηθεί στον πίνακα info με τον παραπάνω τρόπο, ισχύει ότι για κάθε κόμβο k (που βρίσκεται στη θέση k):

α) Ο γονέας του βρίσκεται στη θέση $\lfloor k/2 \rfloor$, εφόσον $k \neq 1$. Με τον συμβολισμό $\lfloor x \rfloor$ εννοούμε τον πλησιέστερο προς τα κάτω στον x ακέραιο, για παράδειγμα, $\lfloor 2.8 \rfloor = 2$ και $\lfloor 3 \rfloor = 3$. Εννοείται πως αν $k = 1$, τότε πρόκειται για τη ρίζα που δεν έχει γονέα.

β) Το αριστερό παιδί του κόμβου βρίσκεται στη θέση 2k, αν $2k \leq N$. Αν $2k > N$, τότε ο κόμβος δεν έχει αριστερό παιδί.

γ) Το δεξί παιδί του κόμβου βρίσκεται στη θέση $2k + 1$, αν $2k + 1 \leq N$. Αν $2k + 1 > N$, τότε ο κόμβος δεν έχει δεξί παιδί.

Παρόμοια λογική εφαρμόζεται και για μη πλήρη δυαδικά δέντρα. Η κατασκευή του πίνακα ακολουθεί την εξής λογική:

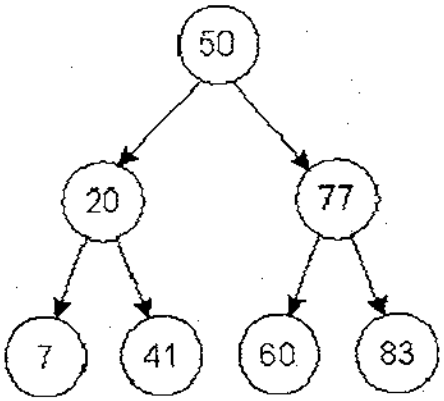
- Η ρίζα του δέντρου τοποθετείται στην πρώτη θέση του πίνακα.
- Για κάθε κόμβο που είναι στη θέση k του πίνακα, το αριστερό παιδί του (αν υπάρχει) τοποθετείται στη θέση 2k. Αν ο κόμβος δεν έχει αριστερό παιδί, στην αντίστοιχη θέση του πίνακα μπαίνει μια ειδική τιμή (π.χ. -1, αν υποθέσουμε ότι τα στοιχεία του δέντρου έχουν θετικές τιμές).
- Για κάθε κόμβο που είναι στη θέση k του πίνακα, το δεξί παιδί του (αν υπάρχει) τοποθετείται στη θέση $2k + 1$. Αν ο κόμβος δεν έχει δεξί παιδί, στην αντίστοιχη θέση του πίνακα μπαίνει μια ειδική τιμή.

6.3. Δυαδικά δέντρα αναζήτησης

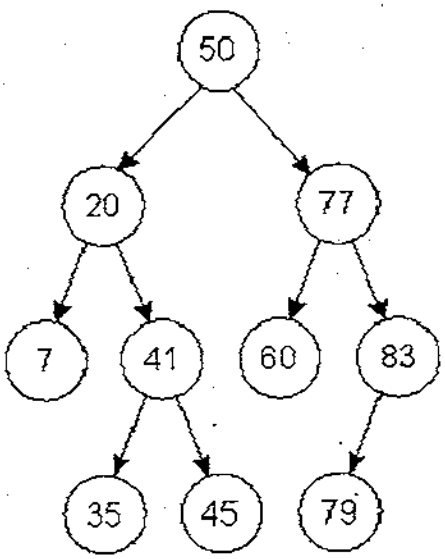
Τα δυαδικά δέντρα έχουν βαθμό 2, δηλαδή κάθε κόμβος μπορεί να έχει μέχρι δύο κόμβους - παιδιά. Επομένως, κάθε κόμβος μπορεί να έχει δύο υποδέντρα, το αριστερό υποδέντρο και το δεξί υποδέντρο. Τα δυαδικά δέντρα είναι αυτά που χρησιμοποιούνται περισσότερο στην πράξη.

Ειδικότερα, τα δυαδικά δέντρα αναζήτησης έχουν την ιδιότητα ότι είναι ταξινομημένα ως εξής: για κάθε κόμβο, το αριστερό υποδέντρο του περιέχει κόμβους με τιμές μικρότερες από την τιμή του κόμβου - γονέα και το δεξί υποδέντρο του περιέχει κόμβους με τιμές μεγαλύτερες από την τιμή του κόμβου - γονέα.

Ένα παράδειγμα δυαδικού δέντρου αναζήτησης φαίνεται στο σχήμα που ακολουθεί.

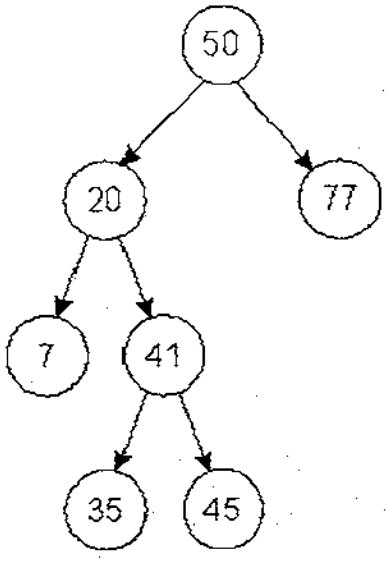


Το παραπάνω δυαδικό δέντρο αναζήτησης έχει επιπλέον την ιδιότητα ότι είναι ισοζυγισμένο, γιατί όλοι οι κόμβοι φύλλα βρίσκονται στο ίδιο επίπεδο (εδώ επίπεδο 3). Ισοζυγισμένα είναι, γενικότερα, τα δέντρα των οποίων οι κόμβοι φύλλα δεν απέχουν περισσότερο από ένα επίπεδο, όπως αυτό που φαίνεται στο σχήμα που ακολουθεί.



Στο παραπάνω δυαδικό δέντρο αναζήτησης όλοι οι κόμβοι φύλλα ανήκουν στο επίπεδο 3 (κόμβοι "7" και "60") ή στο επίπεδο 4 (κόμβοι "35", "45" και "79").

Το παρακάτω δυαδικό δέντρο αναζήτησης δεν είναι ισοζυγισμένο γιατί ο κόμβος φύλλο "77" ανήκει στο επίπεδο 2, ενώ οι κόμβοι "35" και "45" στο επίπεδο 4.



Λόγω της ιδιότητάς τους και του βαθμού (2) που έχουν τα δυαδικά δέντρα αναζήτησης, έχουν ορισμένα χαρακτηριστικά. Αναφέρουμε μερικά από αυτά:

- Ο μέγιστος αριθμός κόμβων σε δυαδικό δέντρο αναζήτησης βάθους k είναι $2^k - 1$.
- Για την εύρεση ενός στοιχείου, ο μέγιστος αριθμός προσπελάσεων σε δυαδικό δέντρο αναζήτησης βάθους k είναι k.
- Σε ένα ισοζυγισμένο δυαδικό δέντρο αναζήτησης με N συνολικά κόμβους, ο μέγιστος αριθμός προσπελάσεων είναι $\lceil \log_2 N \rceil$.
- Η αναζήτηση σε δυαδικό δέντρο αναζήτησης είναι εύκολη και γρήγορη. Ξεκινάμε συγκρίνοντας το στοιχείο που ψάχνουμε με τη ρίζα. Αν είναι μικρότερο από τη ρίζα συνεχίζουμε με το αριστερό παιδί της ρίζας. Αν είναι μεγαλύτερο από τη ρίζα συνεχίζουμε με το δεξί παιδί της ρίζας. Επαναλαμβάνουμε τη διαδικασία μέχρι να βρούμε το στοιχείο που ψάχνουμε ή μέχρι να μην υπάρχει το αντίστοιχο αριστερό ή δεξί παιδί του κόμβου που συγκρίνουμε με το στοιχείο που ψάχνουμε.

6.4. Διάσχιση δέντρων

Διάσχιση ενός δέντρου λέγεται η λειτουργία κατά την οποία επισκεπτόμαστε τους κόμβους του δέντρου με τρόπο ώστε να περάσουμε από όλους τους κόμβους και ακριβώς μία φόρα από κάθε κόμβο. Η διάσχιση ενός δέντρου μπορεί να γίνει με τους παρακάτω τρόπους, ανάλογα με τη σειρά επίσκεψης των κόμβων.

6.4.1. Προ-διαταγμένη διάσχιση (pre-order traversal)

Στην προ-διαταγμένη διάσχιση, η σειρά επίσκεψης των κόμβων είναι η εξής:

1. επίσκεψη της ρίζας
2. επίσκεψη του αριστερού υποδέντρου
3. επίσκεψη του δεξιού υποδέντρου

Στην υλοποίηση του αλγορίθμου προ-διαταγμένης διάσχισης που ακολουθεί σε γλώσσα PASCAL, χρησιμοποιείται η αναδρομική διαδικασία Preorder_Traversal. Οι ακμές των κόμβων υλοποιούνται με τη χρήση δεικτών της PASCAL, ενώ τα στοιχεία ομαδοποιούνται με τους δείκτες σε μία εγγραφή. Όταν ένας κόμβος δεν έχει αριστερό ή δεξί παιδί, στη θέση του δείκτη τοποθετείται η τιμή NIL.

Αρχικά δίνεται η δομή των κόμβων του δέντρου:

```
Type node_ptr = ^node;
node = record
    Info: integer;
    Left: node_ptr;
    Right: node_ptr;
End;
```

Ακολουθεί η αναδρομική διαδικασία Preorder_Traversal:

```
Procedure Preorder_Traversal (tree_node: node_ptr);
Begin
    If (tree_node <> NIL) Then
    Begin
        Writeln(tree_node^. Info);
        Preorder_Traversal(tree_node^. Left);
        Preorder_Traversal(tree_node^. Right);
    End
End;
```

6.4.2. Μετα-διαταγμένη διάσχιση (post-order traversal)

Στη μετα-διαταγμένη διάσχιση, η σειρά επίσκεψης των κόμβων είναι η εξής:

1. επίσκεψη του αριστερού υποδέντρου
2. επίσκεψη του δεξιού υποδέντρου
3. επίσκεψη της ρίζας

6.4.3. Ενδο-διαταγμένη διάσχιση (in-order traversal)

Στην ενδο-διαταγμένη διάσχιση, η σειρά επίσκεψης των κόμβων είναι η εξής:

1. επίσκεψη του αριστερού υποδέντρου
2. επίσκεψη της ρίζας
3. επίσκεψη του δεξιού υποδέντρου

6.5. Ασκήσεις αξιολόγησης

Άσκηση 1

Στον παρακάτω πίνακα είναι αποθηκευμένοι οι κόμβοι ενός δυαδικού δέντρου αναζήτησης με τιμές A_1, A_2, ..., A_8. Οι θέσεις με τιμές 0 υποδεικνύουν ότι δεν υπάρχουν κόμβοι στα αντίστοιχα σημεία του δέντρου.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A_1	A_2	A_3	A_4	0	A_5	A_6	0	A_7	0	0	0	0	A_8	0

α) Κατασκευάστε σχηματικά το αντίστοιχο δυαδικό δέντρο αναζήτησης.

β) Ποιο είναι το μικρότερο και το ποιο το μεγαλύτερο στοιχείο του δέντρου; Αιτιολογήστε την απάντησή σας.

γ) Ποιες σχέσεις (μεγαλύτερο - μικρότερο) υπάρχουν μεταξύ των στοιχείων A_5, A_7 και A_8;

Άσκηση 2

Δίνονται οι αριθμοί: 28, 34, 10, 45, 72, 31, 13, 66, 55, 21, 33, 61, 50.

α) Σχεδιάστε δυαδικό δέντρο αναζήτησης με στοιχεία τους παραπάνω αριθμούς, όταν αυτοί καταφθάνουν με τη σειρά που δίνεται (από αριστερά προς δεξιά).

β) Σχεδιάστε το πλήρες δυαδικό δέντρο αναζήτησης με στοιχεία τους ίδιους αριθμούς (η σειρά με την οποία τοποθετούνται τα στοιχεία δεν είναι αυτή που δίνεται παραπάνω).

γ) Αναφέρετε και περιγράψτε τους τρεις τρόπους διάσχισης δυαδικών δέντρων. Στη συνέχεια δώστε τη σειρά επίσκεψης των κόμβων του δέντρου του ερωτήματος (α) με καθέναν από τους τρεις τρόπους διάσχισης.

7. ΑΝΑΖΗΤΗΣΗ

Σκοπός κεφαλαίου

Σε αυτό το κεφάλαιο παρουσιάζονται οι βασικότεροι αλγόριθμοι αναζήτησης στοιχείων σε πίνακες.

Λέξεις - κλειδιά

Αναζήτηση

Σειριακή αναζήτηση

Δυαδική αναζήτηση

Αναζήτηση κατά ομάδες

Αναζήτηση παρεμβολής

7.1. Εισαγωγή

Η αναζήτηση εμφανίζεται παντού, από την καθημερινή ζωή μέχρι τις πιο εξειδικευμένες εφαρμογές. Στις εφαρμογές, η αναζήτηση δεδομένων γίνεται στην κύρια ή στη βοηθητική μνήμη. Αναζήτηση είναι η εύρεση του στοιχείου που έχει κάποια ιδιότητα (κριτήριο αναζήτησης) ή η απόφαση ότι στα δεδομένα που ψάχνουμε δεν υπάρχει στοιχείο με την ιδιότητα αυτή.

7.1.1. Χαρακτηριστικά αναζήτησης

- Οι αλγόριθμοι αναζήτησης (*searching algorithms*) επιτρέπουν την εύρεση ενός στοιχείου σε μια δομή δεδομένων με βάση ένα χαρακτηριστικό του.
- Αν τα στοιχεία είναι οργανωμένα σε εγγραφές (*records*) και κάθε μία από αυτές απαρτίζεται από πεδία (*fields*) τότε το πεδίο με βάση το οποίο γίνεται η αναζήτηση ονομάζεται κλειδί (*key*) της αναζήτησης.
- Το κόστος μιας αναζήτησης εκφράζεται ανάλογα με τον αριθμό συγκρίσεων που απαιτούνται κατά μέσο όρο για την εύρεση της αναζητούμενης εγγραφής.

7.2. Σειριακή αναζήτηση

Είναι η πιο απλή και άμεση μέθοδος αναζήτησης. Στη σειριακή αναζήτηση (*sequential search*) η εύρεση της εγγραφής στον πίνακα γίνεται ελέγχοντας μία - μία τις εγγραφές από την αρχή μέχρι το τέλος του πίνακα. Αν ο πίνακας έχει N στοιχεία, ο μέσος αριθμός αναζητήσεων είναι N / 2, εφόσον το στοιχείο υπάρχει στον πίνακα. Αν το στοιχείο δεν υπάρχει, τότε ο μέσος αριθμός αναζητήσεων είναι N / 2 ή N, ανάλογα αν ο πίνακας είναι ταξινομημένος ή όχι.

Η μέθοδος της σειριακής αναζήτησης για μη ταξινομημένο πίνακα δίνεται στον αλγόριθμο Sequential_Search που ακολουθεί υλοποιημένος σε PASCAL. Ο πίνακας P είναι καθολική (global) μεταβλητή, που έχει δηλωθεί ως: Var P: Array[1..N] Of Real;

```
Procedure Sequential_Search (N: Integer; K: Real; Var Found: Booloean; Index: Integer);
{* N: μέγεθος πίνακα, K: ζητούμενο στοιχείο, Index: θέση, Found: λογική μεταβλητή *}
Var I: Integer;
Begin
    Found:= False;
    I:= 1;
    While (I <= N) Do
        If (P[I] = K) Then
        Begin
            Index:= I;
            I:= N + 1;
            Found:= True;
        End
        Else I:= I + 1;
End;
```

Μετά την εκτέλεση της Sequential_Search, αν η μεταβλητή Found είναι αληθής, τότε το στοιχείο K υπάρχει στη θέση Index, αλλιώς το στοιχείο δεν υπάρχει.

7.3. Δυαδική αναζήτηση

Με τη δυαδική αναζήτηση κάθε έλεγχος στον πίνακα ή βρίσκει το ζητούμενο στοιχείο ή περιορίζει στο μισό το πλήθος των στοιχείων μέσα στα οποία βρίσκεται αυτό που ψάχνουμε. Η δυαδική αναζήτηση απαιτεί τα στοιχεία του πίνακα να είναι ταξινομημένα.

Ο αλγόριθμος αρχίζει συγκρίνοντας την τιμή του στοιχείου που αναζητείται με αυτήν του «μεσαίου» στοιχείου του πίνακα. Αν οι δύο τιμές συμπίπτουν, η αναζήτηση τελειώνει αφού το στοιχείο βρέθηκε. Αν η τιμή του ζητούμενου στοιχείου είναι μεγαλύτερη από την τιμή του στοιχείου με το οποίο το συγκρίνουμε, συνεχίζεται το ψάξιμο στα στοιχεία που βρίσκονται μετά από αυτό που μόλις εξετάστηκε, αλλιώς κρατούνται μόνο αυτά που βρίσκονται πριν από το τρέχον. Αυτή η διαδικασία συνεχίζεται μέχρι να βρεθεί το ζητούμενο στοιχείο ή να μην υπάρχουν πια άλλα στοιχεία προς εξέταση. Ο αριθμός αναζητήσεων είναι $\log_2 N+1$, όπου N είναι το μέγεθος του πίνακα.

Ο αλγόριθμος που περιγράφει τη δυαδική αναζήτηση υλοποιείται στην PASCAL με τη διαδικασία Binary_Search που ακολουθεί. Στη διαδικασία αυτή, τα L (Lower) και U (Upper) είναι τα όρια της περιοχής στην οποία ψάχνουμε κάθε φορά και M (Middle) είναι το μέσο αυτής της περιοχής. Συγκρίνουμε το στοιχείο K που ψάχνουμε με το στοιχείο του πίνακα που βρίσκεται στη θέση M. Αν το K είναι ίσο με το P[M], τότε το στοιχείο βρέθηκε στη θέση M. Αν το K είναι μικρότερο από το P[M], τότε απορρίπτονται από τη συνέχεια της αναζήτησης τα στοιχεία που βρίσκονται στις θέσεις M έως U, επομένως η νέα περιοχή στην οποία θα συνεχίσουμε την αναζήτηση είναι στις θέσεις L έως M - 1. Αν το K είναι μεγαλύτερο από το P[M], τότε απορρίπτονται από τη συνέχεια της αναζήτησης τα στοιχεία που βρίσκονται στις θέσεις L έως M, επομένως η νέα περιοχή στην οποία θα συνεχίσουμε την αναζήτηση είναι στις θέσεις M + 1 έως U.

```
Procedure Binary_Search (N: Integer; K: Real; Var Found: Booloean; Index: Integer);
{* N: μέγεθος πίνακα, K: ζητούμενο στοιχείο, Index: θέση, Found: λογική μεταβλητή *}
Var L, M, U: Integer;
Begin
    Found:= False;
    L:= 1;
    U:= N;
```

```
    Repeat
        M:= (L + U) Div 2;
        If (K < P[M]) Then U:= M - 1
            Else If (K > P[M]) Then L:= M + 1
                Else Found:= True;
    Until (U < L) OR (Found = True);
    If Found = True Then Index:= M;
End;
```

Μετά την εκτέλεση της Binary_Search, αν η μεταβλητή Found είναι αληθής, τότε το στοιχείο K υπάρχει στη θέση Index, αλλιώς το στοιχείο δεν υπάρχει.

Με τη μέθοδο αυτή, μετά από κάθε σύγκριση το τμήμα του πίνακα το οποίο ανιχνεύουμε μειώνεται στο μισό, δηλαδή στο 1/2, 1/4, 1/8 κ.ο.κ. του αρχικού. Έτσι, ο μέγιστος αριθμός συγκρίσεων της μεθόδου είναι $\lceil \log_2 N \rceil$, όπου N το πλήθος των στοιχείων του πίνακα. Ο αριθμός αυτός είναι και ο αριθμός των συγκρίσεων για ανεπιτυχή αναζήτηση.

7.4. Αναζήτηση κατά ομάδες

Αυτή η μέθοδος είναι μια τεχνική που μπορεί να χρησιμοποιηθεί για να βελτιώσει την απόδοση του αλγόριθμου σειριακής αναζήτησης στην περίπτωση που ο πίνακας είναι ταξινομημένος. Σύμφωνα με την τεχνική αυτή, ο πίνακας χωρίζεται σε ομάδες και εξετάζεται το τελευταίο στοιχείο κάθε ομάδας. Αν το στοιχείο αυτό είναι μεγαλύτερο από εκείνο που αναζητείται, γίνεται σειριακή αναζήτηση στα στοιχεία της ομάδας αυτής, αλλιώς συνεχίζεται η αναζήτηση με το τελευταίο στοιχείο της επόμενης ομάδας.

Ο μέσος αριθμός συγκρίσεων είναι N/(2M)+M/2 τόσο για επιτυχή αναζήτηση όσο και για ανεπιτυχή, όπου N το μέγεθος του πίνακα και M το πλήθος των στοιχείων της ομάδας. Αποδεικνύεται ότι ο αλγόριθμος κάνει τα ελάχιστα βήματα, όταν το πλήθος των στοιχείων σε κάθε ομάδα είναι ίσος με την τετραγωνική ρίζα του N.

Η μέθοδος αυτή περιγράφεται με τον αλγόριθμο Block_Search που παρουσιάζεται στη συνέχεια υλοποιημένος σε γλώσσα PASCAL:

```
Procedure Block_Search (N: Integer; K: Real; Var Found: Booloean; Index: Integer);
{* N: μέγεθος πίνακα, K: ζητούμενο στοιχείο, Index: θέση, Found: λογική μεταβλητή *}
Var I, J, Q: Integer;
Begin
    Q:= Round(Sqrt(N));
    I:= Q;
    J:= N;
    Found:= False;
    {* Εύρεση ομάδας *}
    Repeat
        If (K < P[I]) Then
        Begin
            J:= I;
            I:= N + 1;
        End
        Else I:= I + Q;
        If (I > N) Then I:= N;
    Until (I = N);
    {* Εύρεση στοιχείου μέσα στην ομάδα *}
    For I:= J - Q To J Do
        If (K = P[I]) Then
        Begin
            Found:= True;
            Index:= I;
            I:= J + 1;
        End;
        Else If (K > P[I]) Then I:= J + 1;
```

```
End;
```

Μετά την εκτέλεση της Block_Search, αν η μεταβλητή Found είναι αληθής, τότε το στοιχείο K υπάρχει στη θέση Index, αλλιώς το στοιχείο δεν υπάρχει.

7.5. Αναζήτηση παρεμβολής

Στη αναζήτηση παρεμβολής (*interpolation search*), η εύρεση της εγγραφής στον (ταξινομημένο) πίνακα γίνεται υπολογίζοντας την πιθανή θέση της εγγραφής σε σχέση με τις τιμές των στοιχείων στα άκρα της περιοχής αναζήτησης. Ανάλογα με τη σύγκριση της τιμής της εγγραφής με την εγγραφή στην πιθανή θέση αναπροσαρμόζεται το αντίστοιχο άκρο. Αν ο πίνακας έχει στοιχεία ομοιόμορφα κατανεμημένα θα απαιτηθούν log(log N) συγκρίσεις.

Ο αλγόριθμος παρεμβολής που ακολουθεί λαμβάνει υπ' όψη το περιεχόμενο των στοιχείων του πίνακα, προκειμένου να καθορίσει το σημείο αναζήτησης, ενώ η δυαδική αναζήτηση δουλεύει μόνο με τις θέσεις των στοιχείων.

```
Procedure Interpolation_Search (N: Integer; K: Real; Var Found: Booloean; Index: Integer);
{* N: μέγεθος πίνακα, K: ζητούμενο στοιχείο, Index: θέση, Found: λογική μεταβλητή *}
Var I, L, U: Integer;
Temp: Real;
Begin
    L:= 1;
    U:= N;
    Found:= False;
    While (P[L] <= K) AND (K <= P[U]) AND (Found = False) Do
    Begin
        Temp:= (U - L) / (P[U] - P[L]) * (K - P[L])
        I:= Round(Temp) + L ;
        If (K > P[I]) Then L:= I + 1
            Else If (K < P[I]) Then U:= I - 1;
```

```
                    Else
                    Begin
                        Found:= True;
                        Index:= I;
                    End
    End;
End;
```

Μετά την εκτέλεση της Interpolation_Search, αν η μεταβλητή Found είναι αληθής, τότε το στοιχείο K υπάρχει στη θέση Index, αλλιώς το στοιχείο δεν υπάρχει.

7.6. Ασκήσεις αξιολόγησης

Άσκηση 1

Περιγράψτε τη λογική του αλγορίθμου δυαδικής αναζήτησης σε πίνακα. Επιδείξτε τη λειτουργία του αλγορίθμου, αναζητώντας το στοιχείο 3 στον πίνακα:

A = [1, 3, 4, 7, 10, 13, 22, 28, 31, 37, 44, 58, 59, 69, 71]

Σε κάθε βήμα της αναζήτησης, δώστε τις τιμές των μεταβλητών L (Lower), U (Upper) και M (Middle).

Άσκηση 2

Δίνεται πίνακας με 10 ακέραιους αριθμούς ταξινομημένους σε αύξουσα διάταξη:

1	2	3	4	5	6	7	8	9	10
12	25	38	59	61	66	70	77	82	90

α) Περιγράψτε αναλυτικά τα βήματα της διαδικασίας δυαδικής αναζήτησης στοιχείου με τιμή 70. Σε κάθε βήμα της αναζήτησης, δώστε τις τιμές των μεταβλητών L (Lower), U (Upper) και M (Middle).

β) Στη συνέχεια περιγράψτε τη διαδικασία σειριακής αναζήτησης του ίδιου στοιχείου και διατυπώστε τα συμπεράσματά σας από τη σύγκριση των δύο τρόπων αναζήτησης.

8. ΤΑΞΙΝΟΜΗΣΗ

Σκοπός κεφαλαίου

Σε αυτό το κεφάλαιο παρουσιάζονται οι βασικοί αλγόριθμοι ταξινόμησης.

Λέξεις - κλειδιά

Ταξινόμηση

Σύζευξη

8.1. Ορισμός

Ταξινόμηση είναι η τοποθέτηση ενός συνόλου στοιχείων σε μια ιδιαίτερη σειρά. Η σειρά αυτή είναι είτε αύξουσα είτε φθίνουσα. Σκοπός της ταξινόμησης είναι η διευκόλυνση της αναζήτησης δεδομένων μέσα από ένα σύνολο από αυτά.

8.1.1. Χαρακτηριστικά αλγορίθμων ταξινόμησης

- Οι αλγόριθμοι ταξινόμησης (*sorting algorithms*) θέτουν τα στοιχεία μιας δομής δεδομένων σε σειρά σύμφωνα με μια ορισμένη σχέση διάταξης.
- Αν τα στοιχεία είναι οργανωμένα σε εγγραφές (*records*) και κάθε μία από αυτές απαρτίζεται από πεδία (*fields*), τότε το πεδίο με βάση το οποίο γίνεται η ταξινόμηση ονομάζεται κλειδί (*key*) της ταξινόμησης.
- Το κόστος μιας ταξινόμησης εκφράζεται ανάλογα με τον αριθμό συγκρίσεων και μετακινήσεων που απαιτούνται κατ' ελάχιστο, κατά μέγιστο και κατά μέσο όρο.
- Συχνά για να αποφευχθεί το κόστος των μετακινήσεων, αντί να ταξινομηθούν τα στοιχεία ταξινομείται ένας πίνακας με *δείκτες* στα αντίστοιχα στοιχεία.

8.2. Ταξινόμηση με παρεμβολή

Η μέθοδος αυτή χωρίζει τα στοιχεία που ταξινομούνται σε τρία μέρη: το στοιχείο που εξετάζεται κάθε φορά, την ακολουθία προορισμού που είναι τα ήδη ταξινομημένα στοιχεία και την ακολουθία εισόδου, για παράδειγμα:

6 12 18 42	36	55 7 98
Ακολουθία προορισμού	Εξεταζόμενο στοιχείο	Ακολουθία εισόδου

Η μέθοδος αρχίζει εξετάζοντας το δεύτερο στοιχείο και έχοντας στην ακολουθία προορισμού μόνο το πρώτο στοιχείο. Για κάθε στοιχείο που εξετάζεται, βρίσκεται η σωστή του θέση στην ακολουθία προορισμού και τοποθετεί εκεί μετακινώντας τα μεγαλύτερα στοιχεία της ακολουθίας κατά μία θέση.

Το μέγιστο πλήθος συγκρίσεων είναι I - 1 και το ελάχιστο 1 για κάθε τιμή του δείκτη I (θέση στοιχείου που εξετάζεται).

8.2.1. Κύρια στοιχεία

- Στην ταξινόμηση με παρεμβολή (*insertion sort*) τα στοιχεία χωρίζονται σε:
  - αυτά τα οποία έχουν ταξινομηθεί,
  - αυτό το οποίο θα ταξινομηθεί, και
  - αυτά τα οποία δεν έχουν ταξινομηθεί.
- Σε κάθε βήμα το στοιχείο που πρέπει να ταξινομηθεί εισάγεται στη σωστή θέση αυτών που έχουν ταξινομηθεί.
- Για την εύρεση του σημείου εισαγωγής μπορεί να χρησιμοποιηθεί και ο αλγόριθμος της δυαδικής αναζήτησης. Για να ευρεθεί το σημείο εισαγωγής του εξεταζόμενου στοιχείου στην ακολουθία προορισμού, συγκρίνουμε το εξεταζόμενο στοιχείο με καθένα από τα στοιχεία της ακολουθίας προορισμού, ξεκινώντας από το τελευταίο στοιχείο της ακολουθίας προορισμού μέχρι να βρεθεί η κατάλληλη θέση (δηλαδή μέχρι να βρεθεί στοιχείο της ακολουθίας προορισμού μικρότερο από το εξεταζόμενο στοιχείο). Κάθε φορά μειώνουμε το στοιχείο της ακολουθίας προορισμού μία θέση προς τα δεξιά.

8.2.2. Αλγόριθμος

Ακολουθεί ο αλγόριθμος ταξινόμησης με παρεμβολή, υλοποιημένος στην PASCAL με τη διαδικασία Straight_Insertion_Sort. Ο πίνακας A είναι καθολική (global) μεταβλητή, που έχει δηλωθεί ως: Var A: Array[1..N] Of Real;

```
Procedure Straight_Insertion_Sort (N: Integer);
Var  I, J: Integer;
     Element: Real;
Begin
     For J:= 2 To N do
     Begin
          Element:= A[J];
          I:= J - 1;
          Repeat
               If (A[J] < A[I]) Then
               Begin
                    A[I + 1]:= A[I];
                    I:= I - 1;
               End
          Until (I <= 0) OR (A[J] >= A[I]);
          A[I + 1]:= Element;
     End
End;
```

Στον αλγόριθμο αυτό το μέγιστο πλήθος συγκρίσεων για κάθε τιμή του δείκτη J (δηλαδή για κάθε στοιχείο που εξετάζεται) είναι J - 1 και το ελάχιστο 1. Ο μέσος όρος είναι J / 2. Για το σύνολο των στοιχείων το πλήθος των συγκρίσεων είναι κατ' ελάχιστο N - 1, κατά μέσο όρο

$(N^2 + N - 2) / 4$ και κατά μέγιστο $(N^2 + N) / 2 - 1$, ενώ το πλήθος των μετακινήσεων είναι κατά 1 μεγαλύτερο από το αντίστοιχο πλήθος συγκρίσεων.

Ο προηγούμενος αλγόριθμος μπορεί να βελτιωθεί σημαντικά, αν χρησιμοποιηθεί η μέθοδος της δυαδικής αναζήτησης για την εύρεση της θέσης κάθε στοιχείου στην ακολουθία προορισμού. Ο αλγόριθμος τότε ονομάζεται δυαδική παρεμβολή. Με τον τρόπο αυτό βελτιώνονται οι τιμές των συγκρίσεων, ενώ οι τιμές των μετακινήσεων παραμένουν οι ίδιες.

8.3. Ταξινόμηση με επιλογή

Η μέθοδος βασίζεται στα ακόλουθα τρία βήματα:

a. επιλογή του στοιχείου με την ελάχιστη τιμή

b. ανταλλαγή του με το πρώτο στοιχείο

c. επανάληψη των (a), (b) με τα υπόλοιπα στοιχεία.

Ο αλγόριθμος ταξινόμησης με επιλογή κάνει πάντα $C = (N^2 - N) / 2$ συγκρίσεις.

8.3.1. Κύρια στοιχεία

- Στην ταξινόμηση με επιλογή (*selection sort*), τα στοιχεία χωρίζονται σε:
    - αυτά τα οποία έχουν ταξινομηθεί,
    - αυτά τα οποία δεν έχουν ταξινομηθεί.
- Σε κάθε βήμα επιλέγεται το ελάχιστο στοιχείο από αυτά τα οποία δεν έχουν ταξινομηθεί και εισάγεται στο τέλος των στοιχείων που έχουν ταξινομηθεί.
- Οι συγκρίσεις που απαιτούνται είναι $(N^2 - N) / 2$.

8.3.2. Αλγόριθμος

Ακολουθεί ο αλγόριθμος ταξινόμησης με επιλογή, υλοποιημένος στην PASCAL με τη διαδικασία Straight_Selection_Sort.

```
Procedure Straight_Selection_Sort (N: Integer);
```

```
Var  I, J: Integer;
     Max: Real;
Begin
     For I:= 1 To N - 1 do
     Begin
          Max:= A[I];
          K:= I;
          For J:= I + 1 To N Do
               If (A[J] > Max) Then
               Begin
                    Max:= A[J];
                    K:= J;
               End;
          A[K]:= A[I];
          A[I]:= Max;
     End
End;
```

Η μέθοδος ταξινόμησης με επιλογή κατά μία έννοια είναι αντίθετη με αυτή της παρεμβολής. Η τελευταία σε κάθε βήμα λαμβάνει υπ' όψη της μόνο ένα στοιχείο της ακολουθίας εισόδου και όλα τα στοιχεία της ακολουθίας προορισμού. Αντίθετα η ταξινόμηση με επιλογή λαμβάνει υπόψη της όλα τα στοιχεία της ακολουθίας εισόδου, προκειμένου να βρει ένα, το μικρότερο και να το βάλει σαν επόμενο στοιχείο της ακολουθίας προορισμού.

8.4. Ταξινόμηση με αντιμετάθεση

Με τη μέθοδο αυτή, αρχίζουμε από το τελευταίο στοιχείο και το συγκρίνουμε με το προηγούμενό του. Αν είναι μικρότερο αντιμεταθέτουμε τα δυο στοιχεία. Μετά συνεχίζουμε συγκρίνοντας το προτελευταίο με το προηγούμενό του κ.ο.κ. Η διαδικασία τερματίζεται όταν

φτάσουμε σε κάποιο στοιχείο που όλα τα προηγούμενά του είναι ταξινομημένα. Όλες αυτές οι ενέργειες επαναλαμβάνονται τόσες φορές όσες και τα στοιχεία. Η μέθοδος αυτή λέγεται και ταξινόμηση φυσαλίδας γιατί τα μικρότερα στοιχεία «ανεβαίνουν» σαν φυσαλίδες στην αρχή του πίνακα.

8.4.1. Κύρια στοιχεία

- Στην ταξινόμηση με αντιμετάθεση (*exchange sort*), τα στοιχεία ταξινομούνται με διαδοχική αντιμετάθεση ζευγών που δεν ακολουθούν τη διάταξη της ταξινόμησης.
- Ο αλγόριθμος ταξινόμησης με αντιμετάθεση μπορεί να βελτιωθεί εναλλάσσοντας σε κάθε πέρασμα τη φορά του ελέγχου.
- Στην πρώτη περίπτωση ονομάζεται Bubble Sort (ταξινόμηση φυσαλίδας), ενώ στη δεύτερη ονομάζεται Shake Sort.

8.4.2. Αλγόριθμος

Ακολουθεί ο αλγόριθμος ταξινόμησης με αντιμετάθεση, υλοποιημένος στην PASCAL με τη διαδικασία Bubble_Sort.

```
Procedure Bubble_Sort (N: Integer);
Var  I, J: Integer;
     Temp: Real;
Begin
     For I:= 2 To N do
          For J:= N Downto I Do
               If (A[J - 1] > A[J]) Then
                    Begin
                         Temp:= A[J];
                         A[J]:= A[J - 1];
                         A[J - 1]:= Temp;
```

```
            End;
End;
```

Η υπεροχή του αλγορίθμου Shake_Sort έναντι του απλού Bubble_Sort είναι μόνο στον αριθμό συγκρίσεων και όχι στον αριθμό μετακινήσεων στοιχείων. Οι τελευταίες, όμως, είναι πιο χρονοβόρες από τις συγκρίσεις και έτσι η βελτίωση δεν είναι ιδιαίτερα σημαντική. Συγκρίνοντάς τους επίσης με τους αλγορίθμους παρεμβολής και επιλογής, παρατηρούμε ότι υστερούν.

Και οι τρεις προηγούμενες μέθοδοι είναι απλές, αλλά μειονεκτούν σε πλήθος συγκρίσεων και αντιμεταθέσεων στοιχείων.

Οι δύο επόμενες μέθοδοι είναι πιο σύνθετες προγραμματιστικά αλλά πλεονεκτούν στο χρόνο που απαιτούν για την ταξινόμηση των στοιχείων ενός πίνακα.

8.5. Ταξινόμηση παρεμβολής με φθίνοντα διαστήματα

Η μέθοδος αυτή προτάθηκε από τον Shell και λέγεται Shellsort. Για την εφαρμογή αυτής της μεθόδου πρώτα ορίζεται μια φθίνουσα ακολουθία ακεραίων με τελευταίο τον 1, έστω h1, h2, ..., ht με ht = 1. Μετά, σε t βήματα ταξινομούνται πρώτα τα στοιχεία που απέχουν h1 μεταξύ τους, μετά αυτά που απέχουν h2 μεταξύ τους κ.ο.κ. μέχρι που φτάνουνε στα στοιχεία που απέχουν ht = 1 δηλαδή όλα τα στοιχεία. Για παράδειγμα, έστω ότι ο πίνακας προς ταξινόμηση είναι ο εξής:

1	2	3	4	5	6	7	8
44	55	12	42	94	18	6	67

Αν h = 2, θα ταξινομηθούν (στις θέσεις που βρίσκονται) ξεχωριστά τα 44, 12, 94, 6 και τα 55, 42, 18, 67. Μετά τη μερική ταξινόμηση θα έχουμε:

1	2	3	4	5	6	7	8
6	18	12	42	44	55	94	67

Με το επόμενο βήμα (h = 1) θα γίνει η ολική ταξινόμηση.

Είναι φανερό ότι ο αλγόριθμος αυτός σε κάθε πέρασμα εκμεταλλεύεται το γεγονός ότι στο προηγούμενο πέρασμα τα στοιχεία της ομάδας ταξινομήθηκαν μερικώς. Είναι επίσης φανερό

ότι οποιαδήποτε διαστήματα ταξινόμησης είναι κατάλληλα αρκεί το τελευταίο να είναι 1. Ωστόσο, η ακολουθία διαστημάτων που είναι δυνάμεις του 2, δεν είναι η καλύτερη.

8.6. Ταξινόμηση με διαμερισμό

Η μέθοδος αυτή είναι η καλύτερη γνωστή μέχρι σήμερα και ονομάζεται γρήγορη ταξινόμηση (quick sort). Η μέθοδος αυτή βασίζεται στην ανταλλαγή μεταξύ των πιο απομακρυσμένων στοιχείων, το χωρισμό του πίνακα σε δύο μέρη και την αναδρομική κλήση της μεθόδου για καθένα από τα δύο μέρη. Για να εξηγήσουμε την ταξινόμηση με διαμερισμό, θα χρησιμοποιήσουμε ως παράδειγμα τον πίνακα που ακολουθεί:

1	2	3	4	5	6	7	8
44	55	12	42	94	18	6	67

Η μέθοδος συνοπτικά εξελίσσεται ως εξής: λαμβάνεται το «μεσαίο» στοιχείο του πίνακα, έστω Χ. Έπειτα σαρώνονται οι δύο υπο-πίνακες, ψάχνοντας για ένα στοιχείο του αριστερού υπο-πίνακα που είναι μεγαλύτερο από το Χ και ένα στοιχείο του δεξιού υπο-πίνακα που είναι μικρότερο του Χ. Στο σχήμα έχουμε Χ = 42, 44 > 42 και 6 < 42, άρα γίνεται αντιμετάθεση των 44 και 6. Η διαδικασία συνεχίζεται για όλα τα στοιχεία των δύο υπο-πινάκων. Στο ίδιο σχήμα γίνεται μια αντιμετάθεση των στοιχείων 55 και 18. Έτσι στο τέλος όλα τα στοιχεία του αριστερού υπο-πίνακα έχουν τιμή μικρότερη του Χ και όλα τα στοιχεία του δεξιού υπο-πίνακα έχουν τιμή μεγαλύτερη του Χ. Έχει επιτευχθεί έτσι ένας πρώτος διαμερισμός του αρχικού πίνακα. Όμως ο σκοπός μας είναι η ταξινόμηση του πίνακα και όχι ο διαμερισμός, αλλά η απόσταση από το διαμερισμό στην ταξινόμηση είναι πολύ μικρή.

Στη συνέχεια ο κάθε υπο-πίνακας χωρίζεται σε δύο άλλους υπο-πίνακες και η προηγούμενη διαδικασία της σάρωσης και ανταλλαγής επαναλαμβάνεται για τα δύο ζεύγη υπο-πινάκων. Η συνέχεια είναι προφανής, οι προηγούμενοι πίνακες χωρίζονται στα δύο και επαναλαμβάνονται τα ίδια μέχρι να φτάσουμε σε υπο-πίνακες με ένα μόνο στοιχείο.

Το λεπτό σημείο της μεθόδου βρίσκεται στο διαμερισμό κάθε πίνακα σε δύο υπο-πίνακες. Αφού εκτελεστεί η σάρωση και η ανταλλαγή για τον έναν πρέπει να γίνει και για τον άλλο και για να επιτευχθεί αυτό χρειάζεται να «θυμάται» τα όριά του. Αυτό υποδεικνύει τη χρήση αναδρομικότητας. Ο αναδρομικός αλγόριθμος που ακολουθεί υλοποιεί τη μέθοδο αυτή.

8.6.1. Κύρια στοιχεία

- Στην ταξινόμηση με διαμερισμό (*partition exchange sort*) ή γρήγορη ταξινόμηση (*quick sort*), επιλέγεται ένα στοιχείο του πίνακα και τα υπόλοιπα στοιχεία μοιράζονται σε δύο υπο-πίνακες ανάλογα με τη σχέση τους με το στοιχείο αυτό.
- Στη συνέχεια ο αλγόριθμος καλείται αναδρομικά για τους δύο υπο-πίνακες.
- Οι συγκρίσεις που απαιτούνται είναι N.
- Οι μετακινήσεις που απαιτούνται είναι N * log(N) / 6.

8.6.2. Αλγόριθμος

Ακολουθεί ο αλγόριθμος ταξινόμησης με διαμερισμό, υλοποιημένος στην PASCAL με τη διαδικασία Quick_Sort. Οι παράμετροι Bottom και Top αναπαριστούν το πρώτο και το τελευταίο στοιχείο του συνόλου των στοιχείων του πίνακα πάνω στα οποία θα εφαρμοστεί ο αλγόριθμος.

```
Procedure Quick_Sort (Bottom, Top: Integer);
Var  I, J, Mid: Integer;
     Temp, X: Real;
Begin
     I:= Bottom;
     J:= Top;
     Mid:= (Bottom + Top) DIV 2
     X:= A[Mid];
     Repeat
          While (A[I] < X) Do I:= I + 1;
          While (A[J] > X) Do J:= J - 1;
          If (I < J) Then
          Begin
               Temp:= A[I];
```

```
                A[I]:= A[J];
                A[J]:= Temp;
                I:= I + 1;
                J:= J - 1;
            End;
        Until (I > J);
        If (Bottom < J) Quick_Sort(Bottom, J);
        If (I < Top) Quick_Sort(I, Top);
End;
```

Στην πρώτη κλήση του αλγορίθμου οι μεταβλητές Top και Bottom έχουν τιμές 1 και N αντίστοιχα. Ορίζουν δηλαδή όλο τον πίνακα. Σε κάθε κλήση της διαδικασίας Quick_Sort, οι μεταβλητές αυτές έχουν τα όρια του νέου υπο-πίνακα, αλλά ταυτόχρονα τα όρια του προηγούμενου υπο-πίνακα φυλάσσονται στη στοίβα, απ' όπου ανακαλούνται με τη διαδικασία της αναδρομής.

Η επιλογή του αλγορίθμου εξαρτάται από την επιλογή του στοιχείου X. Αν επιλέγεται το μεσαίο στοιχείο και οι τιμές των στοιχείων του πίνακα είναι τυχαίες, τότε η μέθοδος αυτή είναι πολύ γρήγορη. Όμως η επίδοση εκφυλίζεται στην περίπτωση που ο πίνακας είναι ήδη ταξινομημένος ή ταξινομημένος αντίστροφα.

Ο αλγόριθμος Quick_Sort σε κάθε πέρασμα επεξεργάζεται όλους τους υπο-πίνακες. Άρα απαιτούνται N συγκρίσεις. Ο αριθμός των περασμάτων είναι στην καλύτερη περίπτωση logN και στη χειρότερη N. Έτσι ο συνολικός αριθμός των συγκρίσεων είναι στην καλύτερη περίπτωση N logN και στη χειρότερη N^2.

8.7. Σύζευξη

Ο όρος σύζευξη αναφέρεται στη δημιουργία ενός ταξινομημένου πίνακα από το συνδυασμό δύο ή περισσότερων ταξινομημένων πινάκων. Η σύζευξη αποκαλείται και συγχώνευση.

Η λογική του αλγορίθμου είναι απλή. Εξετάζονται τα πρώτα στοιχεία κάθε πίνακα και το μικρότερο μεταφέρεται στον νέο πίνακα. Στη συνέχεια εξετάζεται το μεγαλύτερο από τα δύο

αυτά στοιχεία και το επόμενο στοιχείο του πίνακα από τον οποίο μεταφέρθηκε το πρώτο στοιχείο. Αυτό επαναλαμβάνεται μέχρι να εξαντληθούν και οι δύο πίνακες.

8.7.1. Αλγόριθμος

Ακολουθεί ο αλγόριθμος σύζευξης, υλοποιημένος στην PASCAL με τη διαδικασία Merge. Οι πίνακες A, B και C είναι καθολικές (global) μεταβλητές, που έχουν δηλωθεί ως:

```
Var     A: Array[1..M + 1] Of Integer;
        B: Array[1..N + 1] Of Integer;
        C: Array[1..M + N] Of Integer;
```

```
Procedure Merge;
Var  I, J, K: Integer;
Begin
     I:= 1;
     J:= 1;
     A[M+1]:= Maxint;
     B[N+1]:= Maxint;
     For K:= 1 To M + N Do
          If (A[I] < B[J]) Then
          Begin
               C[K]:= A[I];
               I:= I + 1;
          End
          Else
          Begin
               C[K]:= B[J];
               J:= J + 1;
```

```
        End
End;
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

- Χρήστος Κοίλιας (1993). *Δομές δεδομένων και οργανώσεις αρχείων*, Εκδόσεις νέων τεχνολογιών
- Κοκκος Α., Λιονταράκης Α., Ματράλης Χ., Παναγιωτακόπουλος Χ. (1998), *Ανοικτή και εξ αποστάσεως εκπαίδευση*, Εκδόσεις Ελληνικό Ανοικτό Πανεπιστήμιο
- Τσακαλίδης Α.(1990), *Δομές δεδομένων,* Εκδόσεις Πανεπιστημίου Πατρών
- Χατζηλυγερούδης Ι. (2000)*Δομές δεδομένων,* Εκδόσεις Ελληνικό Ανοικτό Πανεπιστήμιο
- Πηγές από το internet