<>Εισαγωγή στις Αρχές της Επιστήμης των Η/Υ</>

Σύμφωνα με το ΠΣ της Β΄ Λυκείου

Γ. Γώγουλος, Γ. Κοτσιφάκης, Γ. Κυριακάκη
Α. Παπαγιάννης, Ε. Φραγκονικολάκης, Π. Χίνου

ΕΚΔΟΣΕΙΣ ΠΥΞΙΔΑ
www.pyxida.gr

Εισαγωγή στις Αρχές της Επιστήμης των Η/Υ

Γ. Γώγουλος,
Γ. Κοτσιφάκης,
Γ. Κυριακάκη,
Α. Παπαγιάννης,
Μ. Φραγκονικολάκης,
Π. Χίνου

Εισαγωγή στις Αρχές της Επιστήμης των Η/Υ

"Σύμφωνα με το ΠΣ της Β΄ Λυκείου"

Συγγραφική Ομάδα :
Γιώργος Γώγουλος, Γιώργος Κοτσιφάκης,
Γεωργία Κυριακάκη, Απόστολος Παπαγιάννης,
Μανόλης Φραγκονικολάκης, Παναγιώτα Χίνου

Συντονιστής ομάδας συγγραφής : Γιώργος Γώγουλος

Σελιδοποίηση – Σχεδιασμός Βιβλίου : Γεωργία Κυριακάκη
Φιλολογική επιμέλεια: Σίτσα Κοτσιφάκη
Σχεδιασμός Εξωφύλλου: Νίκος Ιγγλεζάκης

Εκδόσεις «ΠΕΚ - Πυξίδα Της Πόλης»
Υπεύθυνος έκδοσης : Ματθαίος Φραντζεσκάκης

Πρώτη Έκδοση : Δεκέμβριος 2014

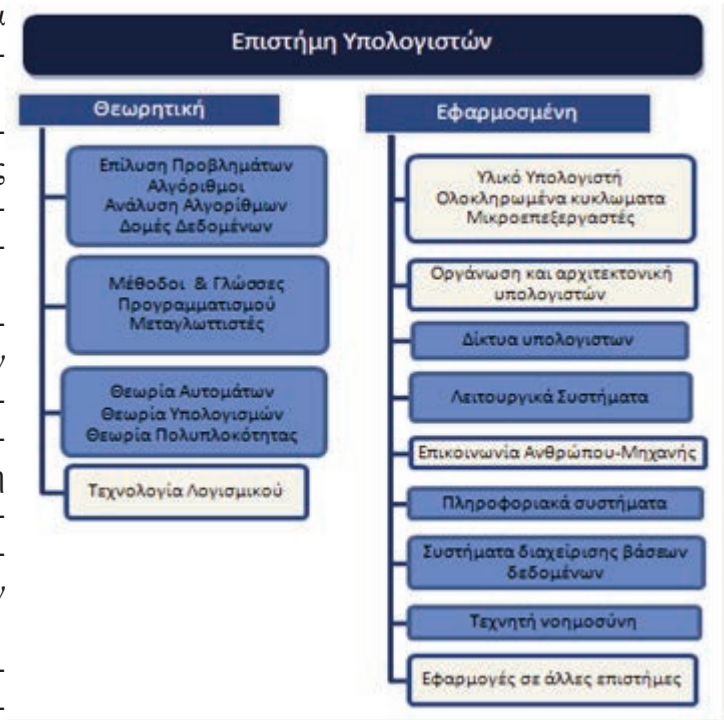
ISBN 978-618-81242-4-0

Κρύα Βρύση Δήμου Πλατανιά
73006 Κολυμβάρι Χανίων
Τηλ.: 2821074104 Φαξ: 2821036364
Web: http://ekdoseis-pek.blogspot.gr/
Web: www.pyxida.gr
Mail: info@pyxida.gr

Πρόλογος

Το βιβλίο αυτό απευθύνεται στους μαθητές της Β΄ τάξης Γενικού Λυκείου που παρακολουθούν το μάθημα γενικής παιδείας «Εισαγωγή στις Αρχές της Επιστήμης των Η/Υ», το οποίο έχει ως γενικό σκοπό να γνωρίσουν οι μαθητές βασικούς τομείς και θεμελιώδεις έννοιες της Επιστήμης των Υπολογιστών και παράλληλα να αναπτύξουν αναλυτική και συνθετική σκέψη.

Το βιβλίο καλύπτει το πρόγραμμα σπουδών του μαθήματος και διαρθρώνεται σε τρία κεφάλαια:



Στο 1ο Κεφάλαιο γίνεται μία εισαγωγή στις βασικές έννοιες και τομείς της Επιστήμης των Υπολογιστών, που ορισμένοι από αυτούς εξετάζονται αναλυτικότερα στα επόμενα κεφάλαια.

Στο 2ο Κεφάλαιο καλύπτονται έννοιες της Θεωρητικής Επιστήμης των Υπολογιστών όπως πρόβλημα, αλγόριθμος, δομές δεδομένων και προγραμματισμός δίνοντας έμφαση στη διασύνδεση της θεωρητικής προσέγγισης των εννοιών με καθημερινές εφαρμογές λογισμικού και το σύγχρονο γίγνεσθαι στην ανάπτυξη προγραμμάτων.

Το 3ο Κεφάλαιο εστιάζει σε βασικούς τομείς της Εφαρμοσμένης Επιστήμης των Υπολογιστών, όπως Λειτουργικά Συστήματα, Πληροφοριακά Συστήματα, Δίκτυα Υπολογιστών και Τεχνητή Νοημοσύνη.

Το βιβλίο επιχειρεί μέσα από δομημένα και στοχευμένα παραδείγματα να υποστηρίξει την κατανόηση των εννοιών και να δώσει ερεθίσματα για περαιτέρω διερεύνηση και μελέτη τομέων της Επιστήμης των Υπολογιστών. Ακολουθούνται σύγχρονες προτάσεις για τη διδασκαλία και τη μάθηση που προωθούν και δίνουν ιδιαίτερη έμφαση στη διαδικασία και όχι μόνο στο περιεχόμενο. Στην αρχή κάθε κεφαλαίου γνωστοποιούνται οι στόχοι του κεφαλαίου σε επίπεδο πρόθεσης ενώ στο τέλος των επιμέρους ενοτήτων δίνεται η δυνατότητα αυτό-αξιολόγησης του μαθητή σε επίπεδο προσδοκώμενων μαθησιακών αποτελεσμάτων.

Ιούνιος 2014
Οι συγγραφείς

Για την καλύτερη ανάγνωση και αξιοποίηση του περιεχομένου του βιβλίου, εκτός από σχήματα, πίνακες, πλαίσια και εικόνες έχουν χρησιμοποιηθεί διαφορετικά χρώματα σε κάθε κεφάλαιο του βιβλίου και αρκετά εικονίδια τα οποία χαρακτηρίζουν την πληροφορία που συνοδεύουν.

Χρώματα κεφαλαίων:

1 Κεφάλαιο 1

2 Κεφάλαιο 2

3 Κεφάλαιο 3

Τα εικονίδια και η σημασία τους αναφέρονται παρακάτω:

 Μαθησιακοί Στόχοι

 Λέξεις - κλειδιά

 Εισαγωγικές ερωτήσεις

 Ορισμός

 Ερωτήσεις - Θέματα για συζήτηση

 Ασκήσεις - Προβλήματα

 Αυτοαξιολόγηση

Περιεχόμενα

κεφάλαιο 1

Βασικές Έννοιες

Επιστήμη των Υπολογιστών

Στόχοι

Στόχος του κεφαλαίου είναι οι μαθητές:

- να γνωρίσουν βασικές έννοιες και τομείς της Επιστήμης των Υπολογιστών.

Λέξεις κλειδιά

Επιστήμη των Υπολογιστών	Πληροφορική, Θεωρητική Επιστήμη των Υπολογιστών, Εφαρμοσμένη Επιστήμη των Υπολογιστών

Εισαγωγικές ερωτήσεις

1. Γνωρίζετε τι είναι Επιστήμη και τι είναι Τεχνολογία; Πιστεύετε ότι η Πληροφορική είναι Επιστήμη ή Τεχνολογία;
2. Γνωρίζετε κλάδους της Επιστήμης των Υπολογιστών;

Εικόνα 1-1. Ταξινόμηση της Επιστήμης Υπολογιστών από την ACM (Association for Computing Machinery)

Επιστήμη Υπολογιστών (Computer Science)	
Λογισμικό και μηχανική λογισμικού (Software and its engineering)	Γλώσσες προγραμματισμού, προγραμματιστικά υποδείγματα, Λειτουργικά Συστήματα
Θεωρία υπολογισμού (Theory of computation)	Ανάλυση και σχεδίαση αλγορίθμων, Πολυπλοκότητα και υπολογισιμότητα αλγορίθμων
Μαθηματικά της Πληροφορικής (Mathematics of computing)	Διακριτά μαθηματικά, Θεωρία γράφων, Πιθανότητες και στατιστική, Μαθηματική ανάλυση και μαθηματική βελτιστοποίηση
Υλικό (Hardware)	Ολοκληρωμένα κυκλώματα, μνήμες και μέσα αποθήκευσης
Οργάνωση και Αρχιτεκτονική Η/Υ (Computer organization)	Έλεγχος και δοκιμή υλικού, συστήματα πραγματικού χρόνου
Δίκτυα υπολογιστών (Networks)	Αρχιτεκτονικές, πρωτόκολλα και υπηρεσίες δικτύων
Πληροφοριακά συστήματα (Information systems)	Συστήματα διαχείρισης δεδομένων, βάσεις δεδομένων, ανάκτηση πληροφοριών, συστήματα υποστήριξης αποφάσεων
Ασφάλεια και Ιδιωτικότητα (Security and privacy)	Κρυπτογράφηση, συστήματα ασφάλειας δεδομένων
Ανθρωποκεντρικός υπολογισμός (Human-centered computing)	Διεπαφή χρήστη-υπολογιστή, προσβασιμότητα
Μεθοδολογίες υπολογισμού (Computing methodologies)	Παράλληλοι υπολογιστές, Τεχνητή νοημοσύνη, Ρομποτική, Μηχανική όραση, Γραφική Υπολογιστών, Κατανεμημένα συστήματα
Εφαρμογές της Πληροφορικής σε άλλους τομείς (Applied computing)	Φυσικές επιστήμες, Μηχανική, Ηλεκτρονικό εμπόριο, Επιχειρήσεις, Αστρονομία, Αεροναυπηγική, Χημεία, Φυσική, Βιολογία, Εκπαίδευση, Νομικές και κοινωνικές επιστήμες, Καλές τέχνες

1. Βασικές έννοιες

1.1 Πληροφορική ή Επιστήμη Υπολογιστών;

Ο όρος «πληροφορική» προέρχεται από τη λέξη πληροφορία. Εμφανίστηκε τη δεκαετία του 1960 στη Γαλλία (Informatique) και τη Γερμανία (Informatik). Από την ετυμολογία της λέξης προκύπτει ο ορισμός:

Πληροφορική είναι η επιστήμη που ασχολείται με την αναπαράσταση, αποθήκευση και επεξεργασία της πληροφορίας.

Η Πληροφορική είναι άρρηκτα συνδεδεμένη με τη μηχανή που λέγεται ηλεκτρονικός υπολογιστής. Γι' αυτό και στον Αγγλόφωνο χώρο, έχει επικρατήσει η ονομασία Επιστήμη Υπολογιστών (Computer Science). Στη νέα επιστήμη όσο σημαντικός είναι ο υπολογιστής (computer), άλλο τόσο σημαντικές είναι οι διαδικασίες υπολογισμού (computing) που μπορεί να κάνει.

Επιστήμη Υπολογιστών είναι η επιστήμη που ασχολείται με τους υπολογιστές και τους υπολογισμούς.

Για μας, στο εξής, Πληροφορική και Επιστήμη Υπολογιστών αναφέρονται στην ίδια επιστήμη. Θεμελιώδεις έννοιες στην Επιστήμη των Υπολογιστών είναι:

- η ίδια η μηχανή, ο υπολογιστής και
- η υπολογιστική διαδικασία που η μηχανή μπορεί να επιτελέσει, το πρόγραμμα

Ιστορικοί σταθμοί της Επιστήμης Υπολογιστών	
300 π.Χ.	Αλγόριθμος του Ευκλείδη για τον υπολογισμό του μέγιστου κοινού διαιρέτη δύο φυσικών αριθμών, ένας από τους αρχαιότερους αλγόριθμους
100 π.Χ.	Μηχανισμός των Αντικυθήρων, μηχανικός υπολογιστής και όργανο αστρονομικών παρατηρήσεων
820 μ.Χ.	Ο al-Khwarizmi, Πέρσης μαθηματικός και αστρονόμος γράφει βιβλίο για τους αλγόριθμους
1822	Ο Charles Babbage σχεδιάζει την Αναλυτική Μηχανή του, την πρώτη μηχανή με αποθηκευμένα προγράμματα.
1940	Ο John von Neumann θέτει τις θεμελιώδεις αρχές σχεδίασης των σύγχρονων ΗΥ, γνωστές ως "αρχιτεκτονική von Neumann", στο Πανεπιστήμιο του Πρίνστον.
1944	Ο Aiken κατασκευάζει τον Mark I. Χρόνος για πρόσθεση: 1/3 s., για πολλαπλασιασμό: 6 s
1946	Οι Eckert και Maughly κατασκευάζουν τον ENIAC, τον πρώτο ηλεκτρονικό υπολογιστή με 18000 λυχνίες. Χρόνος για πολλαπλασιασμό: 6 ms
1949	Ο Wilkes κατασκευάζει τον EDSAC, τον πρώτο ψηφιακό υπολογιστή γενικής χρήσης με αποθηκευμένο πρόγραμμα (υπολογιστής τύπου von Neumann)
1969	Γέννηση του διαδικτύου με τη δημιουργία του δικτύου ARPANET
1991	Δημιουργία του Παγκόσμιου Ιστού (WWW) στο CERN

Η παραπάνω ιστορική αναδρομή εμφανίζει κυρίως τεχνολογικά επιτεύγματα και όχι επιστημονικά. Μήπως, λοιπόν, η Πληροφορική είναι απλώς η τεχνολογία των υπολογιστών και δεν είναι επιστήμη; Η απάν-

τηση είναι ότι η Πληροφορική είναι δισδιάστατη, είναι και τεχνολογία και επιστήμη. Απλά, ο ηλεκτρονικός υπολογιστής καταλαμβάνει τόσο κυρίαρχο ρόλο σε αυτήν που δεν είναι δυνατό να διαχωρίσουμε τα τεχνολογικά από τα επιστημονικά επιτεύγματα.

Θεωρητική Επιστήμη των Υπολογιστών
Επίλυση Προβλημάτων
Αλγόριθμοι
Ανάλυση Αλγορίθμων
Δομές Δεδομένων
Μέθοδοι Προγραμματισμού
Γλώσσες Προγραμματισμού
Μεταγλωττιστές
Θεωρία Αυτομάτων
Θεωρία Υπολογισμών
Θεωρία Πολυπλοκότητας
Τεχνολογία Λογισμικού

1.2 Θεωρητική Επιστήμη των Υπολογιστών

Θεμελιώδεις έννοιες της Θεωρητικής Επιστήμης των Υπολογιστών είναι ο **αλγόριθμος** και το **πρόγραμμα**. Μια μηχανικά εκτελέσιμη υπολογιστική διαδικασία αποτελεί αλγόριθμο. Το θεωρητικό μέρος της Επιστήμης Υπολογιστών ασχολείται με τη σχεδίαση, τη δημιουργία και την αναπαράσταση των αλγορίθμων.

Η μεταφορά του αλγορίθμου σε μια μορφή κατανοητή από τον ηλεκτρονικό υπολογιστή, δηλαδή σε μια **γλώσσα προγραμματισμού**, δημιουργεί ένα πρόγραμμα. Στη Θεωρητική Επιστήμη των Υπολογιστών εξετάζονται τεχνικές σχεδίασης προγραμμάτων και μελετώνται οι γλώσσες προγραμματισμού. Ένας άλλος τομέας της θεωρητικής Πληροφορικής ασχολείται με τα δεδομένα και τους τρόπους οργάνωσης αυτών, τις **δομές δεδομένων**.

Στο θεωρητικό μέρος της Επιστήμης Υπολογιστών ανήκει επίσης ο τομέας της **θεωρίας υπολογισμού**. Ο τομέας αυτός αποτελεί τη μαθηματική βάση της Επιστήμης των Υπολογιστών που έχει ως αντικείμενο να εκφράσει και να μελετήσει, με τη χρήση μαθηματικών μοντέλων, τη λειτουργία των υπολογιστικών μηχανών, την υπολογισιμότητα των προβλημάτων (αν μπορούν ή όχι να επιλυθούν από υπολογιστή), καθώς και τους υπολογιστικούς πόρους που απαιτούνται για την επίλυση προβλημάτων.

Εφαρμοσμένη Επιστήμη των Υπολογιστών
Υλικό Υπολογιστή
Ολοκληρωμένα κυκλώματα
Μικροεπεξεργαστές
Οργάνωση και αρχιτεκτονική υπολογιστών
Δίκτυα υπολογιστών
Λειτουργικά Συστήματα
Επικοινωνία Ανθρώπου-Μηχανής
Πληροφοριακά συστήματα
Συστήματα διαχείρισης βάσεων δεδομένων
Τεχνητή νοημοσύνη
Εφαρμογές σε άλλες επιστήμες

1.3 Εφαρμοσμένη Επιστήμη των Υπολογιστών

Η άρρηκτη σχέση της θεωρητικής επιστήμης με την υπολογιστική μηχανή δημιουργεί πεδία εφαρμογής τόσο στην ίδια την Πληροφορική όσο και σε άλλες Επιστήμες.

Η Εφαρμοσμένη Επιστήμη των Υπολογιστών ασχολείται με το **υλικό** (hardware) και την **οργάνωση** και **αρχιτεκτονική υπολογιστών**. Ασχολείται επίσης με τα **λειτουργικά συστήματα**, προγράμματα μεγάλης κλίμακας που τοποθετούνται ανάμεσα στο υλικό και το λογισμικό του υπολογιστή. Τα **πληροφοριακά συστήματα** συνδυάζουν υλικό, λογισμικό, δεδομένα, ανθρώπους και διαδικασίες. Η διασύνδεση υπολογιστών σε **δίκτυα**, δημιουργεί έναν άλλο τομέα. Ο τομέας της **τεχνητής νοημοσύνης** ασχολείται με τη δημιουργία προγραμμάτων που κάνουν τον υπολογιστή να προσομοιάζει σε νοήμον ον.

Τέλος, ένα μέρος της Εφαρμοσμένης Επιστήμης των Υπολογιστών διερευνά τις δυνατότητες εφαρμογής των ηλεκτρονικών υπολογιστών σε άλλες επιστήμες, όπως στη Μηχανική, στην Οικονομία, στην Ιατρική, στις Φυσικές επιστήμες, στις Ανθρωπιστικές επιστήμες και στις Καλές τέχνες. Οι εφαρμογές αυτές, συχνά δημιουργούν νέους επιστημονικούς τομείς όπως η Βιοϊατρική.

Καλώς ήρθατε στον κόσμο της Πληροφορικής!

κεφάλαιο 2

Θέματα Θεωρητικής Επιστήμης των Υπολογιστών

Πρόβλημα
Αλγόριθμοι
Προγραμματισμός

Στόχοι

Στόχοι του κεφαλαίου είναι οι μαθητές:

- να έρθουν σε επαφή με την έννοια του προβλήματος, τις κατηγορίες των προβλημάτων και τις διαδικασίες επίλυσης προβλημάτων.
- να επιλύουν απλά προβλήματα διατυπώνοντας τη λύση σε μορφή αλγορίθμου και προγράμματος.
- να γνωρίσουν διαφορετικούς αλγορίθμους για την επίλυση συγκεκριμένων προβλημάτων και να προβληματιστούν για τα χαρακτηριστικά τους.
- να αντιληφθούν τα στάδια ανάπτυξης εφαρμογών λογισμικού.
- να προβληματιστούν για τις δομές αναπαράστασης δεδομένων και τις λειτουργίες διαχείρισής τους μέσα από προγράμματα.

Λέξεις κλειδιά

Πρόβλημα	Επιλύσιμα - Μη επιλύσιμα - Ανοικτά προβλήματα, Υπολογιστικά προβλήματα, Κατανόηση - Επίλυση προβλήματος, Μεθοδολογίες επίλυσης προβλημάτων
Αλγόριθμος	Χαρακτηριστικά αλγορίθμου (Καθοριστικότητα, Περατότητα, Αποτελεσματικότητα, Είσοδος, Έξοδος), Ανάλυση Αλγορίθμων, Θεωρία Υπολογισμού, Υπολογισιμότητα, Πολυπλοκότητα αλγορίθμων, Σειριακή επεξεργασία, Παράλληλη επεξεργασία, Επαναληπτικοί και Αναδρομικοί αλγόριθμοι, Φυσική Γλώσσα, Διάγραμμα Ροής, Γλώσσες Περιγραφής Αλγορίθμων, Ψευδογλώσσα, Δομές δεδομένων, Εντολές αλγορίθμου, Δομές αλγορίθμων, Εκσφαλμάτωση, Λογικά λάθη, Συντακτικά λάθη, Τεκμηρίωση
Προγραμματισμός	Γλώσσες Προγραμματισμού, Προγραμματιστικό υπόδειγμα, Δομημένος προγραμματισμός, Αντικειμενοστραφής Προγραματισμός, Προγραμματιστικό περιβάλλον, Μεταφραστής, Διερμηνευτής Μεταγλωττιστής, Υποπρόγραμμα, Παράμετρος, Κύκλος ζωής εφαρμογής λογισμικού.

Εισαγωγικές ερωτήσεις

1. Γνωρίζετε ότι ο “τετραγωνισμός του κύκλου” είναι μη επιλύσιμο πρόβλημα;
2. Ποιος είναι ο αλγόριθμος που χρησιμοποιείτε καθημερινά στην προετοιμασία σας για το σχολείο;
3. Έχετε αναρωτηθεί από τι εξαρτάται η ταχύτητα εντοπισμού των στοιχείων ενός μαθητή σε ένα πρόγραμμα υπολογιστή;
4. Έχετε σκεφτεί τι σημαίνει “τερματίζω” μια επανάληψη;

2.1. Πρόβλημα

2.1.1 Η έννοια του προβλήματος

Τη λέξη πρόβλημα την έχουμε συναντήσει πολλές φορές. Έχουμε λύσει πολλά προβλήματα από την αρχή της σχολικής μας ζωής σε διάφορα μαθήματα όπως τα Μαθηματικά τη Φυσική, τη Χημεία κ.ά.. Όμως προβλήματα αντιμετωπίζουμε και καθημερινά στη ζωή μας όπως, για παράδειγμα, πώς θα πάω στο σχολείο, τί θα φάω σήμερα, πού θα πάω βόλτα με τους φίλους μου, πώς θα τακτοποιήσω το δωμάτιο μου. Τα προβλήματα αυτά θεωρούνται απλά, υπάρχουν όμως και πιο σύνθετα όπως η ρύπανση του περιβάλλοντος, η αντιμετώπιση φυσικών φαινομένων, η αποκωδικοποίηση του DNA, η επαγγελματική σταδιοδρομία, η ανεργία, η θεραπεία ασθενειών κ. ά.

Όλα τα προβλήματα είναι καταστάσεις που πρέπει να αντιμετωπιστούν αλλά δε γνωρίζουμε το πώς. Προβλήματα προκύπτουν όταν ένα εμπόδιο δυσκολεύει την επίτευξη ενός στόχου. Για παράδειγμα, μια βλάβη (το εμπόδιο) σε ένα εργοστάσιο παραγωγής εμποδίζει την εκπλήρωση των παραγγελιών (ο στόχος). Μόλις το εμπόδιο παρακαμφθεί, έχουμε φτάσει στη λύση του προβλήματος.

Στόχος είναι αυτό που έχουμε αποφασίσει ότι πρέπει να επιτύχουμε.

Εμπόδιο είναι αυτό που μας δυσκολεύει στην επίτευξη ενός στόχου.

Ο Jackson (1985) συνοψίζει:
Στόχος + Εμπόδιο = ΠΡΟΒΛΗΜΑ

Με άλλα λόγια, **πρόβλημα** είναι μια κατάσταση που απαιτεί λύση αλλά η λύση της δεν είναι γνωστή ούτε προφανής.

2.1.2 Κατηγορίες προβλημάτων

Αν παρατηρήσουμε τα προβλήματα που αναφέρθηκαν παραπάνω, διαπιστώνουμε τη διαφορετική τους φύση. Η κατηγοριοποίησή τους μπορεί να γίνει με βάση διάφορα κριτήρια. Αν ως κριτήριο θέσουμε τη δυνατότητα επίλυσής τους, τα προβλήματα διακρίνονται σε:

- **Επιλύσιμα**. Χαρακτηρίζονται τα προβλήματα των οποίων η λύση έχει διατυπωθεί ή η συνάφειά τους με άλλα, ήδη λυμένα, μας επιτρέπει να θεωρούμε βέβαιη την δυνατότητα επίλυσής τους. Παραδείγματα τέτοιων προβλημάτων είναι η επίλυση δευτεροβάθμιων εξισώσεων, ο υπολογισμός του ρεύματος σε ένα κύκλωμα, η κατασκευή μιας γέφυρας, η εξοικονόμηση ενέργειας, η οργάνωση μιας βιβλιοθήκης.
- **Ανοικτά**. Χαρακτηρίζονται τα προβλήματα των οποίων η λύση δεν έχει ακόμα βρεθεί, ούτε έχει αποδειχθεί ότι δεν επιδέχονται λύση. Ως ανοικτά προβλήματα μπορούμε να αναφέρουμε την ανακάλυψη ζωής σε άλλους πλανήτες, τη θεραπεία του καρκίνου, την πρόβλεψη των σεισμών.
- **Μη επιλύσιμα**. Χαρακτηρίζονται τα προβλήματα για τα οποία έχουμε καταλήξει στην παραδοχή ότι δεν μπορούν να λυθούν. Τέτοια προβλήματα είναι ο τετραγωνισμός του κύκλου, το ταξίδι στο παρελθόν, η γήρανση του ανθρώπου.

Εικόνα 2-1. Κατηγορίες προβλημάτων με κριτήριο την επιλυσιμότητα

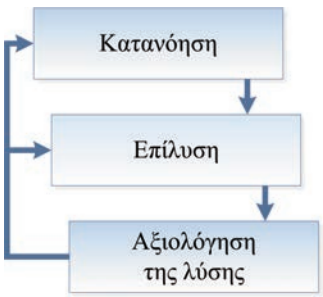
Η εικασία του Goldbach ότι όλοι οι θετικοί άρτιοι αριθμοί μπορούν να γραφούν ως άθροισμα δύο πρώτων αριθμών, παραμένει ανοικτό πρόβλημα.

Τα προβλήματα που δίνονται προς επίλυση στον υπολογιστή έχουν να κάνουν με τη διενέργεια υπολογισμών οι οποίοι απαιτούν μια σειρά

από λογικές και αριθμητικές πράξεις. Τα προβλήματα αυτά λέγονται υπολογιστικά και είναι αυτά με τα οποία θα ασχοληθούμε κυρίως σε αυτό το βιβλίο.

2.1.3 Υπολογιστικά προβλήματα

Στα υπολογιστικά προβλήματα ζητούμε να βρούμε την απάντηση που ικανοποιεί τα δεδομένα του προβλήματος. Η πλειοψηφία των προβλημάτων απαιτούν τη διενέργεια πράξεων υπάρχουν όμως, και υπολογιστικά προβλήματα απόφασης ή βελτιστοποιήσης. **Απόφασης** είναι ένα πρόβλημα στο οποίο η απάντηση είναι ένα ναι ή ένα όχι. Για παράδειγμα, δεδομένου ότι ο καιρός είναι άστατος, να πάω στο σχολείο με το ποδήλατο; **Βελτιστοποίησης** είναι ένα πρόβλημα στο οποίο αναζητούμε την απάντηση που ικανοποιεί κατά τον καλύτερο τρόπο τα δεδομένα του. Για παράδειγμα, δεδομένου ότι διαθέτω 100 ευρώ, ποιο είναι το καλύτερο κινητό που μπορώ να αγοράσω;



Εικόνα 2-2. Στάδια επίλυσης προβλήματος

2.1.4 Διαδικασία επίλυσης υπολογιστικών προβλημάτων

Η αντιμετώπιση υπολογιστικών προβλημάτων γίνεται με συγκεκριμένη διαδικασία που περιέχει σαφή βήματα.

1. Κατανόηση και Παρουσίαση προβλήματος

Σημαντικός παράγοντας για την **κατανόηση** του προβλήματος αλλά και τον καθορισμό της λύσης του είναι ο προσδιορισμός του χώρου του προβλήματος. Ο **χώρος του προβλήματος** καθορίζεται με τη βοήθεια ερωτήσεων («πώς», «πού», «τί», «γιατί», «πότε»). Απαντώντας στα ερωτήματα προσδιορίζουμε τα δεδομένα και τα ζητούμενα του προβλήματος. (Είσοδος – Έξοδος)

Δεδομένο προβλήματος ονομάζεται ένα γνωστό ή αποδεκτό στοιχείο το οποίο χρησιμοποιείται ως βάση ή προϋπόθεση για την επίλυση του προβλήματος.

Ζητούμενο προβλήματος είναι αυτό που ψάχνουμε για να βγούμε από τη δύσκολη κατάσταση στην οποία βρισκόμαστε.

Ανάλογα με τη φύση του προβλήματος τα δεδομένα και τα ζητούμενα μπορεί να είναι αριθμητικά, οικονομικά, λογικά κ.ά. Ανεξάρτητα από το είδος τους, τα δεδομένα και τα ζητούμενα πρέπει να έχουν τρεις (3) σημαντικές ιδιότητες: ορθότητα, πληρότητα και σαφήνεια.

Η **ορθότητα** είναι αναγκαίο να ελέγχεται κάθε φορά που επιδιώκεται η επίλυση ενός προβλήματος. Για παράδειγμα, έστω ότι σας ζητείται να ταξινομήσετε σε αλφαβητική σειρά τα επίθετα των συμμαθητών σας. Αν σας τα έχουν δώσει με ορθογραφικά λάθη, η ταξινόμηση που θα προκύψει θα είναι λανθασμένη. Πριν ξεκινήσουμε την προσπάθεια επίλυσης ενός προβλήματος πρέπει να ελέγξουμε την ορθότητα των δεδομένων.

Η **πληρότητα** πρέπει κι αυτή να ελέγχεται κάθε φορά που επιδιώκεται η επίλυση ενός προβλήματος. Έστω ότι στο παραπάνω παράδειγμα μας ζητάνε να ταξινομηθούν οι συμμαθητές μας με βάση την ημερομηνία γέννησης και μας δίνουν τα στοιχεία στην Εικόνα 2-3. Παρατηρούμε ότι τα δεδομένα είναι ελλιπή (εφόσον στην Παπαδάκη Μιχαέλα δεν υπάρχει ημερομηνία γέννησης) και δεν μπορούμε να επιλύσουμε το πρόβλημα.

Επίθετο	Όνομα	Ημερ.Γέννησης
Παπαδοπούλου	Κυριακή	3/22/98
Παπαδάκης	Κωνσταντίνος	2/25/98
Παπαδάκη	Μιχαέλα	…
Παπαδόπουλος	Μιχάλης	5/20/98

Εικόνα 2-3. Δεδομένα χωρίς πληρότητα

Η **σαφήνεια** είναι ενας σημαντικός παράγοντας για την ορθή επίλυση ενός προβλήματος. Τα δεδομένα αλλά και τα ζητούμενα, πρέπει να είναι **σαφή**, δηλαδή δεν πρέπει να υπάρχουν περιθώρια για παρερμηνείες. Οι διαθέσιμες επιλογές πρέπει να είναι συγκεκριμένες, έτσι ώστε να μην απαιτούνται διευκρινιστικές ερωτήσεις από το πρόσωπο που καλείται να λύσει το πρόβλημα.

Σαφήνεια δεδομένων

Για παράδειγμα, έστω ότι μας ζητάει η μητέρα μας να τη βοηθήσουμε στην παρασκευή ενός κέικ. Το επόμενο πρωί, έτοιμοι να ξεκινήσουμε τη διαδικασία παίρνουμε τη συνταγή και διαπιστώνουμε ότι δεν υπάρχουν τα υλικά που χρειάζονται. Το πρόβλημα που αρχικά είχαμε αντιληφθεί είναι διαφορετικό από αυτό που προέκυψε.

Σαφήνεια ζητουμένων

Έστω ότι μας ζητείται από μια φίλη μας να αγοράσουμε μια σοκολάτα. Η προθυμία να την εξυπηρετήσουμε μας οδηγεί στο Super Market. Εκεί έχουμε να επιλέξουμε ανάμεσα σε πάρα πολλές μάρκες, ποιότητες αλλά και μεγέθη. Στη συγκεκριμένη περίπτωση αυτό που κάνουμε είναι ή να επιλέξουμε κάποια στην τύχη, ή να ζητήσουμε νέες, πιο σαφείς, οδηγίες, τηλεφωνικά.

Η παρουσίαση του προβλήματος καθορίζει σε γενικές γραμμές την επιθυμητή σχέση εισόδου-εξόδου.

Ένας ακόμα σημαντικός παράγοντας για την ορθή επίλυση ενός προβλήματος είναι ο τρόπος παρουσίασής του. Η παρουσίαση του προβλήματος μπορεί να γίνει φραστικά (με λέξεις) ή αλγεβρικά (με μαθηματικά σύμβολα).

Παράδειγμα 2-1. Βρείτε δύο αριθμούς που το άθροισμα τους είναι 78 και το γινόμενο τους 1296.

Φυσική Γλώσσα	**Γλώσσα της Άλγεβρας**
Βρείτε δύο αριθμούς	x,y
που το άθροισμα τους είναι 78	x+y= 78
και το γινόμενο τους 1296	x•y= 1296

Εικόνα 2-4. Παρουσίαση του προβλήματος σε φυσική και αλγεβρική γλώσσα

2. Επίλυση του προβλήματος

Στην προηγούμενη παράγραφο περιγράψαμε τη διαδικασία εύρεσης των δεδομένων και των ζητουμένων μέσα από την παρουσίαση του προβλήματος. Προχωρώντας προς το στάδιο της επίλυσης (problem solving) καταλαβαίνουμε ότι σε πολλά προβλήματα η λύση τους δεν είναι άμεσα

γνωστή. Η επίλυση περιλαμβάνει την ανάλυση, μοντελοποίηση, σχεδίαση, και υλοποίηση μίας κατάλληλης λύσης του προβλήματος με τον υπολογιστή.

Βασικές έννοιες στην επίλυση προβλημάτων είναι η δομή, η ανάλυση και η σύνθεση. Με τον όρο **δομή,** εννοούμε τον τρόπο με τον οποίο επιμέρους στοιχεία σχετίζονται και συνδέονται μεταξύ τους ώστε να σχηματίζουν ενιαίο σύνολο. **Ανάλυση** είναι ο διαχωρισμός ενός συνόλου στα συστατικά του στοιχεία. **Σύνθεση** είναι η τοποθέτηση στοιχείων σε συσχετισμό μεταξύ τους έτσι ώστε να δημιουργείται ένα σύνολο.

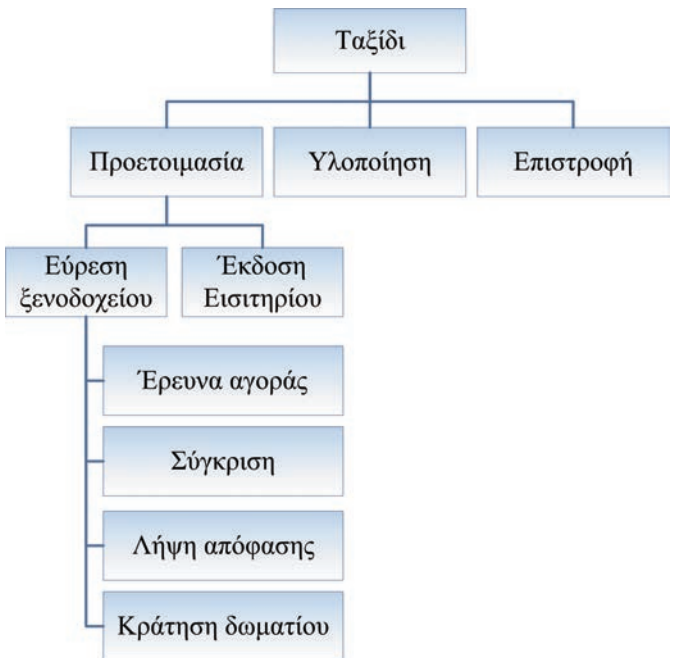
Όταν το ενιαίο σύνολο είναι το πρόβλημα, τα επιμέρους στοιχεία είναι τα μικρότερα προβλήματα στα οποία χωρίζεται, τα υποπροβλήματα. Εφαρμόζοντας την ανάλυση και τη σύνθεση στην επίλυση προβλημάτων δημιουργούνται τρεις βασικές μεθοδολογίες η Αναλυτική (Top Down), η Συνθετική (Bottom Up), και η Μικτή (Mixed).

Αναλυτική (Top Down problem solving) είναι η μεθοδολογία επίλυσης προβλημάτων που βασίζεται στη σχεδίαση από το γενικό στο ειδικό. Η γενική αρχή της είναι ότι για να λύσουμε κάποιο σύνθετο πρόβλημα πρέπει:

1. Να ορίσουμε τα υποπροβλήματα (sub-problems).
2. Να επαναλάβουμε την διαδικασία αυτή για κάθε ένα από τα υποπροβλήματα, όσο αυτό είναι αναγκαίο.
3. Όταν φτάσουμε σε υποπροβλήματα που δεν απαιτούν επιπλέον διάσπαση, προχωράμε στην άμεση επίλυσή τους, τότε έχει επιλυθεί και

Παράδειγμα 2-2. Ας δούμε ένα παράδειγμα εφαρμογής της αναλυτικής μεθοδολογίας: Θέλουμε να υλοποιήσουμε ένα ταξίδι στο εξωτερικό.

Εικόνα 2-5. Προετοιμασία ταξιδιού



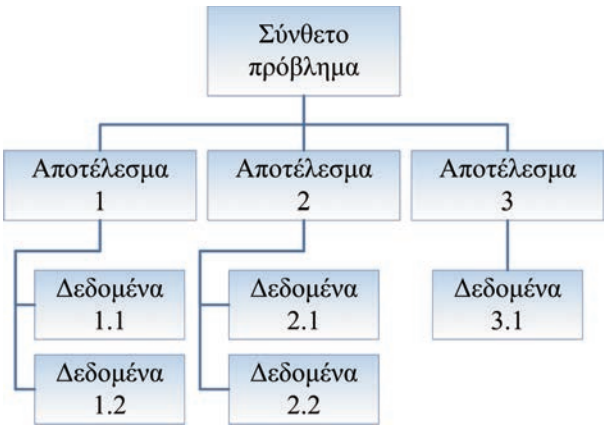
Υπάρχουν πολλοί τρόποι με τους οποίους μπορούμε να αναλύσουμε το πρόβλημα σε υποπροβλήματα. Ένας από όλους είναι να το αναλύσουμε αρχικά σε τρία υποπροβλήματα: προετοιμασία ταξιδιού, υλοποίηση και επιστροφή. Η προετοιμασία του ταξιδιού μπορεί να

περιλαμβάνει έκδοση εισιτηρίων και εύρεση ξενοδοχείου. Το πρόβλημα της εύρεσης ξενοδοχείου με τη σειρά του μπορεί να αναλυθεί στα υποπροβλήματα της έρευνας αγοράς, σύγκρισης, λήψης απόφασης και κράτησης δωματίου.

Με τον ίδιο τρόπο θα μπορούσαμε να αναλύσουμε και τα υπόλοιπα υποπροβλήματα προχωρώντας την ανάλυση σε ακόμα χαμηλότερα επίπεδα. Η διαγραμματική αναπαράσταση της ανάλυσης του προβλήματος δίνεται στην Εικόνα 2.5.

Συνθετική (bottom up problem solving) είναι η μεθοδολογία επίλυσης προβλημάτων που βασίζεται στη σχεδίαση από το ειδικό στο γενικό εμε τη σύνδεση των δεδομένων. Για το σκοπό αυτό:

1. Αρχίζουμε από δεδομένα που είναι ορθά.
2. Στη συνέχεια, με μια λογικά ορθή διαδικασία, παράγουμε νέα δεδομένα και αποτελέσματα
3. Η διαδικασία τερματίζει, όταν παραχθεί το ζητούμενο του προβλήματος.



Εικόνα 2-6. Συνθετική μεθοδολογία

Μεικτή (mixed), είναι η μεθοδολογία επίλυσης προβλημάτων που συνδυάζει την αναλυτική και τη συνθετική μέθοδο. Σύμφωνα με τη μεικτή μεθοδολογία, κάποια από τα υποπροβλήματα επιλύονται με την αναλυτική και κάποια με τη συνθετική μέθοδο.

Μπορείτε να σκεφτείτε ένα πρόβλημα που να επιλύεται με τη συνθετική ή τη μεικτή μεθοδολογία;

Η ανάλυση του προβλήματος μπορεί να αναπαρασταθεί είτε φραστικά είτε διαγραμματικά. Στη διαγραμματική αναπαράσταση, η περιγραφή έχει τη μορφή γενεολογικού δέντρου, στο οποίο κάθε πρόβλημα έχει «παιδιά» τα υποπροβλήματα στα οποία αναλύεται.

3. Αξιολόγηση της λύσης

Το στάδιο αξιολόγησης της λύσης είναι πολύ σημαντικό. Για να ελέγξουμε εάν έχουμε επιλύσει σωστά ένα πρόβλημα, αρχικά καταγράφουμε υποθετικά δεδομένα σύμφωνα με τις απαιτήσεις του προβλήματος. Στη συνέχεια, εφαρμόζουμε τα βήματα επίλυσης για να εξάγουμε τα αποτελέσματα. Τέλος, συγκρίνουμε τα αποτελέσματα που πήραμε με τα ζητούμενα του προβλήματος. Εάν διαπιστώσουμε λάθος, εντοπίζουμε το τμήμα της λύσης που εκτελεί τη λανθασμένη λειτουργία, το διορθώνουμε και επαναλαμβάνουμε τη διαδικασία ελέγχου, έως ότου τα αποτελέσματα να μη διαφέρουν από τα ζητούμενα. Πολλές φορές το λάθος μπο-

ρεί να οφείλεται και στην κατανόηση του προβλήματος. Στην περίπτωση αυτή η διαδικασία επίλυσης επαναλαμβάνεται από την αρχή.

Ασκήσεις - Προβλήματα

1. Να διατυπώσετε με αλγεβρικό τρόπο τα παρακάτω υπολογιστικά προβλήματα:
   α. Να υπολογιστεί το εμβαδό ενός τριγώνου, αν σας δίνεται ότι το εμβαδό του είναι το μισό του γινομένου της βάσης του επί το ύψος του.
   β. Η Κασσιανή είναι μεγαλύτερη της Άννα-Μαρίας κατά 10 χρόνια. Η Άννα-Μαρία είναι μικρότερη του Γιώργου κατά 3 χρόνια. Ο Κωνσταντίνος είναι 20 ετών και κατά 1 χρόνο μεγαλύτερος του Γιώργου. Πόσο χρονών είναι η Κασσιανή;
2. «Η Μάρθα θέλει να αγοράσει έναν ηλεκτρονικό υπολογιστή». Ποια είναι τα πιθανά ερωτήματα που θα πρέπει να απαντηθούν προκειμένου να προσδιοριστεί καλύτερα το ζητούμενο στο πρόβλημα της Μάρθας;»
3. Να λυθεί η εξίσωση: $ax^2+bx+c=0$. Αναλύστε σε υποπροβλήματα και παρουσιάστε σχηματικά το πρόβλημα.
4. «Ο Δημήτρης βρίσκεται στο λιμάνι του Πειραιά και θέλει να πάει στη Θεσσαλονίκη». Αναλύστε σε υποπροβλήματα και παρουσιάστε σχηματικά το πρόβλημα. Ποια μέθοδο χρησιμοποιήσατε και γιατί; Με βάση το είδος επίλυσης σε ποια ή ποιες κατηγορίες προβλημάτων θα το εντάσσατε;

Όταν ολοκληρώσετε την ενότητα ελέγξτε αν είστε σε θέση:

- να περιγράφετε την έννοια του προβλήματος.
- να αναγνωρίζετε και να απαριθμείτε τις κατηγορίες προβλημάτων.
- να διακρίνετε την ύπαρξη υπολογιστικών προβλημάτων και να αναφέρετε τις φάσεις επίλυσής τους.
- να προσδιορίζετε τα δεδομένα και τα ζητούμενα ενός προβλήματος
- να ελέγχετε την πληρότητα, την ορθότητα και την σαφήνεια των δεδομένων και των ζητούμενων.
- να αναλύετε ένα πρόβλημα σε απλούστερα και να διατυπώνετε τα αποτελέσματα της ανάλυσης με τρόπο σαφή.

2.2 Αλγόριθμοι

2.2.1 Η έννοια του αλγορίθμου

Στο προηγούμενο κεφάλαιο είδαμε ότι για την επίλυση ενός προβλήματος μπορούν να ακολουθηθούν διάφορες τεχνικές οι οποίες έχουν κατηγοριοποιηθεί και συστηματοποιηθεί. Παρόλα αυτά, η διεργασία αντιμετώπισης προβλημάτων παραμένει μια υψηλά διαισθητική λειτουργία του ανθρώπινου εγκεφάλου. Αναρωτηθήκατε το πώς εκτελούνται αριθμητικές πράξεις στο μυαλό μας με τόση ευκολία; Το πώς είμαστε σίγουροι ότι το αποτέλεσμά τους είναι σωστό; Στο μυαλό μας, με θαυμαστό τρόπο, μέσα από τη γνώση, δημιουργούνται διαδικασίες επίλυσης προβλημάτων που καταγράφονται σε αυτό ως σωστές. Με την εμφάνιση των ηλεκτρονικών υπολογιστών, το στοίχημα του ανθρώπου ήταν να μεταδώσει αυτή τη θαυμαστή ιδιότητα του εγκεφάλου του στον υπολογιστή. Η διαδικασία επίλυσης ενός προβλήματος οδηγεί στη δημιουργία ενός αλγορίθμου.

Οι αλγόριθμοι στην καθημερινή ζωή της Ματίνας

- Βάλε ελαφριά ρούχα. Πήγαινε στο μπαλκόνι που έχει ησυχία. Κατέβασε την τέντα. Πάρε ένα χυμό να δροσιστείς και φυσικά το βιβλίο σου, είπε η μητέρα μου.

Όχι πάλι, σκέφτηκα…. Εντολές, εντολές, εντολές…. Δε μου φτάνει το πρόβλημα μου, έχω και τους αλγόριθμους της μητέρας μου! Ξεκίνησα το διάβασμα. Ορισμός: $\sqrt{x}$ είναι ο αριθμός y για τον οποίο ισχύει $y^2 = x$. Ωραία σκέφτηκα. Η ρίζα του 9 είναι το 3. Η ρίζα του 49 είναι το 7. Ναι, αλλά πως θα υπολογίσω τη ρίζα του 2 ή τη ρίζα του 2371; Για τη λύση θα πρέπει να σκέφτομαι και να συνδυάζω ιδιότητες. Θα έχω όμως και τότε αποτέλεσμα; Αποτελεσματικότερο το κομπιουτεράκι της μητέρας μου. Εύκολο και σίγουρο. Πληκτρολόγησα το 2, πάτησα το σύμβολο $\sqrt{x}$ και να το αποτέλεσμα 1.41421356237309504880168872420 97 !!!!

Πώς το βρήκε τόσο γρήγορα; Πώς το σκέφτηκε; Γιατί εγώ πρέπει να διαβάζω, να συμπληρώνω φυλλάδια, να κάνω τόσο πολύπλοκους συνδυασμούς στο μυαλό μου και να μην μπορώ να υπολογίσω ΟΛΕΣ τις ρίζες, όπως το κομπιουτεράκι; Θα πρέπει, σκέφτηκα, να υπάρχει μια συγκεκριμένη γενική διαδικασία που αν την ακολουθήσω, θα μπορώ να υπολογίσω οποιαδήποτε ρίζα. Πρέπει να υπάρχει ένας αλγόριθμος!

Για επιβεβαίωση των σκέψεών μου έκανα αναζήτηση στον παγκόσμιο ιστό, στη μηχανή που λειτουργεί με τον εξυπνότερο και γρηγορότερο αλγόριθμο αναζήτησης. Φράση αναζήτησης: «Αλγόριθμος για τον υπολογισμό της τετραγωνικής ρίζας ενός αριθμού» και πρώτο αποτέλεσμα «Η προσεγγιστική μέθοδος του Newton για τον υπολογισμό της τετραγωνικής ρίζας αριθμού».

Αν y μια τιμή κατά προσέγγιση της $\sqrt{x}$, τότε το αποτέλεσμα της πράξης $\frac{x + \frac{x}{y}}{2}$ δίνει μια καλύτερη προσέγγιση, πιο κοντά στην πραγματική τιμή της ρίζας.

Άρχισα να το εφαρμόζω, ελπίζοντας να βρω την τιμή που μου έβγαλε το κομπιουτεράκι.

Πρώτη φορά $\frac{2+\frac{2}{2}}{2} = 1.5$, δεύτερη φορά $\frac{1.5 + \frac{2}{1.5}}{2} = 1.4166$, τρίτη φορά… $\frac{1.4166 + \frac{2}{1.4166}}{2} = 1.41421568$

Το εφάρμοσα μόλις τρεις φορές και παρατήρησα ότι το αποτέλεσμα σύγκλινε με την τιμή που είχε βγάλει το κομπιουτεράκι. Είχα ανακαλύψει μέσα από την παρατήρηση και την έρευνα την έννοια του αλγορίθμου, μια μηχανικά εκτελέσιμη υπολογιστική διαδικασία, μέσω της οποίας, αν ξέρεις πράξεις και την ακολουθήσεις πιστά, υπολογίζεις μια ποσότητα. Με αυτό τον τρόπο θα μπορούσα να υπολογίσω οποιαδήποτε τετραγωνική ρίζα.

Η έννοια του αλγορίθμου είναι θεμελιώδης για την επιστήμη της Πληροφορικής. Στα Μαθηματικά δίνουμε ορισμούς που περιγράφουν το «τι είναι». Διατυπώνουμε γενικές ιδιότητες και θεωρήματα. Συνδυάζοντας τα, λύνουμε ειδικά προβλήματα (π.χ. τον υπολογισμό της ρίζας του 2). Στην Πληροφορική κατασκευάζουμε αλγορίθμους που περιγράφουν το «πώς υπολογίζεται», δηλαδή δημιουργούμε συγκεκριμένες υπολογιστικές διαδικασίες οι οποίες εκτελούνται μηχανικά. Στη συνέχεια, τις εφαρμόζουμε για να λύνουμε γενικά προβλήματα (π.χ. υπολογισμός της ρίζας οποιουδήποτε αριθμού).

Αλγόριθμος είναι η ακριβής περιγραφή μιας σειράς βημάτων που απαιτούνται για την επίλυση ενός προβλήματος.

Ο αλγόριθμος αποτελεί τη βάση για τη δημιουργία ενός προγράμματος που θα εκτελεστεί από έναν ηλεκτρονικό υπολογιστή. Η έννοιά του είναι γενικότερη από εκείνη του προγράμματος και δε συνδέεται αποκλειστικά με την Πληροφορική. Μπορεί να εκφραστεί ακόμα και σε φυσική γλώσσα, Ελληνικά ή Αγγλικά.

Μια συνταγή μαγειρικής είναι ένας αλγόριθμος. Οι οδηγίες του καθηγητή για την επίλυση της δευτεροβάθμιας εξίσωσης είναι, επίσης, αλγόριθμος. Στην επίλυση προβλημάτων μέσω αλγορίθμων, υπάρχουν τρεις (3) διαφορετικοί διακριτοί ρόλοι:

- Ο λύτης, αυτός που καλείται να αντιμετωπίσει το πρόβλημα σχεδιάζοντας τον αλγόριθμο.
- Ο εκτελεστής, αυτός που εφαρμόζει πιστά τις εντολές του αλγορίθμου που έφτιαξε ο λύτης.
- Ο χρήστης, αυτός που ενεργοποιεί τον αλγόριθμο, καλώντας τον εκτελεστή να λύσει, όποτε θέλει, το πρόβλημα.

Επίλυση προβλήματος με Η/Υ

Λύτης = Προγραμματιστής

Εκτελεστής = Υπολογιστής

Χρήστης = Χρήστης Η/Υ

Ασκήσεις - Προβλήματα

1. Αναζητήσετε τον αλγόριθμο του Ευκλείδη για τον υπολογισμό του Μέγιστου Κοινού Διαιρέτη. Μελετήστε τη λειτουργία του και εφαρμόστε τον για τον υπολογισμό του ΜΚΔ των αριθμών 32 και 40.

Όταν ολοκληρώσετε την ενότητα ελέγξτε αν είστε σε θέση:

- να ορίζετε και να περιγράφετε την έννοια του αλγόριθμου
- να προσδιορίζετε τη θέση του αλγόριθμου στην καθημερινή ζωή
- να αναγνωρίζετε τη σημασία της αλγοριθμικής σκέψης στην επίλυση προβλημάτων και τη λήψη αποφάσεων
- να δίνετε παραδείγματα αλγορίθμων είτε για απλά καθημερινά προβλήματα είτε για απλά υπολογιστικά προβλήματα.
- να αναγνωρίζετε τους ρόλους του λύτη, του εκτελεστή και του χρήστη στη δημιουργία ενός αλγορίθμου.

2.2.2 Χαρακτηριστικά αλγορίθμου

Ο αλγόριθμος αποτελείται από μια σειρά εντολών που, αν εφαρμοσθούν πιστά, οδηγούν στην επίλυση ενός προβλήματος. Είναι, όμως, οποιαδήποτε σειρά από εντολές ένας αλγόριθμος; Η απάντηση είναι όχι. Οι εντολές ενός αλγορίθμου, αλλά και ο αλγόριθμος ως ολότητα, θα πρέπει να ικανοποιούν κάποια κριτήρια:

- **Είσοδος**: είναι τα στοιχεία που χρειάζεται ο αλγόριθμος για να εκτελεσθεί.
- **Έξοδος**: είναι τα στοιχεία που παράγει ο αλγόριθμος, τα αποτελέσματά του.
- **Καθοριστικότητα**: κάθε εντολή θα πρέπει να είναι μονοσήμαντη, δηλαδή να καθορίζει με απόλυτη σαφήνεια και ακρίβεια τον τρόπο εκτέλεσής της σε κάθε δυνατή περίπτωση.
- **Περατότητα**: εκτελώντας τα βήματα του αλγορίθμου, θα πρέπει να φθάνουμε σε πέρας (τέλος) σε κάθε δυνατή περίπτωση.
- **Αποτελεσματικότητα**: κάθε εντολή θα πρέπει να είναι διατυπωμένη με απλό τρόπο, ώστε να μπορεί να εκτελεσθεί.

Ας εξετάσουμε τα αλγοριθμικά κριτήρια μέσα από ένα πρόβλημα.

Πρόβλημα: πώς φτιάχνουμε ένα κέικ σοκολάτας;

Αλγόριθμος: Αρκεί να ανοίξουμε ένα βιβλίο ζαχαροπλαστικής και να ακολουθήσουμε πιστά οποιαδήποτε συνταγή.

Είσοδος: τα υλικά

Έξοδος: το ζεστό ζουμερό κέικ

Καθοριστικότητα: η εντολή «Ψήσε σε φούρνο ήδη προθερμασμένο σε θερμοκρασία 175 βαθμούς Κελσίου για ακριβώς 45 λεπτά» είναι απόλυτα σαφής, ενώ η εντολή «Ψήσε σε φούρνο μέχρι να φουσκώσει» σίγουρα δεν είναι!

Περατότητα: Μια εντολή επανάληψης «Ψήσε στο φούρνο μέχρι να σταματήσει η γη να κινείται» θα ήταν καταστροφική για τον αλγόριθμο και το κέικ. Ο αλγόριθμος δε θα τελειώσει ποτέ και το κέικ θα καιγόταν.

Αποτελεσματικότητα: Η εντολή «Άνοιξε το φούρνο και πιάσε το καυτό κέικ με τα δύο σου γυμνά χέρια», αν και απόλυτα σαφής, δε θα ήταν καταστροφική για το κέικ αλλά θα ήταν για τα χέρια μας και κατά συνέπεια για τον αλγόριθμο!

Είναι φανερό από το παραπάνω παράδειγμα ότι τα κριτήρια της καθοριστικότητας και αποτελεσματικότητας είναι σχετικά με αυτόν που θα κληθεί να εφαρμόσει τον αλγόριθμο. Κάθε εκτελεστής έχει διαφορετικές δυνατότητες, διαφορετικά δυνατά και αδύνατα σημεία. Ο άνθρωπος, για παράδειγμα, μπορεί να ακούσει, να διαβάσει, μπορεί να μιλήσει αλλά ακόμα και να εφαρμόσει τύπους, να σκεφθεί, να αυτοσχεδιάσει. Ο υπολογιστής, από την άλλη, μπορεί να εισάγει από το πληκτρολόγιο, να εμφανίσει, να κάνει αριθμητικές και λογικές πράξεις. Κάθε εκτελεστής έχει διαφορετικό σύνολο κατανοητών και εφικτών εντολών. Ο άνθρωπος έχει ένα ευρύ σύνολο εκτελέσιμων εντολών. Ο υπολογιστής μπορεί να εκτελεί μόνο βασικές πράξεις. Σε κάθε περίπτωση, μια σειρά εντολών θα πρέπει να λαμβάνει υπόψη το μελλοντικό εκτελεστή της, προκειμένου να ικανοποιεί τα αλγοριθμικά κριτήρια.

Ερωτήσεις - Θέματα για συζήτηση

1. Δίνεται η εντολή «Πολλαπλασίασε την ακτίνα επί 2 και επί 3.14» σε φυσική γλώσσα. Έχει καθοριστικότητα; Έχει αποτελεσματικότητα;
2. Ποιος έχει ευρύτερο σύνολο εκτελέσιμων εντολών, ο άνθρωπος ή ο υπολογιστής;
3. Συζητήσετε στην τάξη το πώς επηρεάζει ο εκτελεστής (άνθρωπος ή υπολογιστής) την αποτελεσματικότητα ενός αλγορίθμου.

Ασκήσεις - Προβλήματα

1. Περιγράψτε με φυσική γλώσσα αλγόριθμο υπολογισμού της περιφέρειας και του εμβαδού ενός κύκλου. Δώστε τον σε συμμαθητή σας να τον εκτελέσει και συζητήστε στην τάξη αν ικανοποιεί τα αλγοριθμικά κριτήρια.

Όταν ολοκληρώσετε την ενότητα ελέγξτε αν είστε σε θέση:

- να αναγνωρίζετε πότε μια σειρά από εντολές αποτελεί αλγόριθμο
- να αναγνωρίζετε τη σημασία του να τερματίζει και να δίνει αποτελέσματα ένας αλγόριθμος.
- να αναγνωρίζετε το ρόλο του λύτη και του εκτελεστή στην αποτελεσματικότητα ενός αλγορίθμου.

2.2.3 Ανάλυση αλγορίθμων, Θεωρία Υπολογισμού, Πολυπλοκότητα αλγορίθμων, Υπολογισιμότητα αλγορίθμων

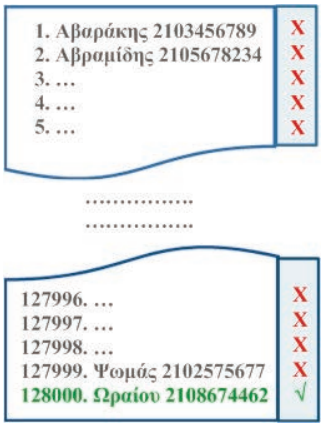
Οι αλγόριθμοι προκύπτουν από την ανάγκη εύρεσης λύσεων σε προβλήματα που μας απασχολούν και φυσικά μπορούμε να έχουμε πολλές λύσεις για το ίδιο πρόβλημα άρα και πολλούς αλγόριθμους. Πώς όμως μπορούμε να αξιολογήσουμε τους διαφορετικούς αλγόριθμους και να εντοπίσουμε ποιος είναι καλύτερος έναντι κάποιου άλλου; Υπάρχουν στοιχεία σε έναν αλγόριθμο που να μπορούν να μετρηθούν και να υπολογιστεί πόσο αποδοτικός είναι; Στα ερωτήματα αυτά προσπαθεί να απαντήσει ο τομέας της επιστήμης των υπολογιστών που ασχολείται με την **ανάλυση αλγορίθμων**.

Τα δύο στοιχεία ενός αλγορίθμου που μπορούν να μετρηθούν και να μας δώσουν μια εκτίμηση της αποδοτικότητάς του είναι ο χρόνος που χρειάζεται για να εκτελεστεί και ο χώρος, δηλαδή η μνήμη, που απαιτείται για να λειτουργήσει σωστά. Μεγαλύτερη έμφαση δίνουμε στη σωστή χρήση του χρόνου και όχι του χώρου, καθώς το κόστος των κυκλωμάτων μνήμης των υπολογιστών μειώνεται συνεχώς με την εξέλιξη της τεχνολογίας.

Η εκτίμηση του χρόνου εκτέλεσης ενός αλγορίθμου μπορεί να γίνει με δύο τρόπους: α) πειραματικά, χρονομετρώντας το διάστημα από τη στιγμή που ξεκινά να εκτελείται ένας αλγόριθμος μέχρι τη στιγμή που θα τερματίσει, β) θεωρητικά, προσπαθώντας να βρούμε μια μαθηματική σχέση που να συνδέει το χρόνο εκτέλεσης με το πλήθος των δεδομένων εισόδου. Επειδή οι αλγόριθμοι μπορούν να εφαρμόζονται σε διαφορετικό πλήθος δεδομένων κάθε φορά και φυσικά σε μηχανήματα διαφο-

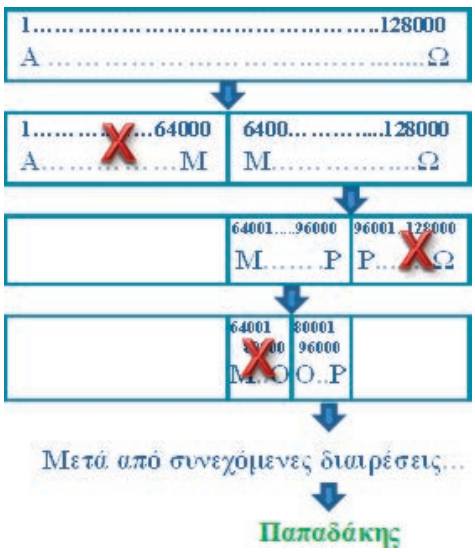
ρετικών δυνατοτήτων, η πειραματική προσέγγιση δε μας δίνει πολλά στοιχεία για την αποδοτικότητα του αλγορίθμου. Από την άλλη πλευρά, η θεωρητική προσέγγιση της εύρεσης μιας μαθηματικής σχέσης μας καλύπτει καλύτερα διότι έτσι μπορούμε να εκτιμήσουμε την αποδοτικότητα ενός αλγορίθμου ανεξάρτητα από το πλήθος των δεδομένων εισόδου ή της υπολογιστικής μηχανής που χρησιμοποιούμε.

Ας δούμε όμως καλύτερα ένα παράδειγμα. Έστω ότι έχουμε έναν τηλεφωνικό κατάλογο που περιλαμβάνει 128000 ονόματα και τηλέφωνα. Θέλουμε να φτιάξουμε έναν αλγόριθμο που, εάν του δίνουμε ένα όνομα ως είσοδο, αυτός να μας επιστρέφει το τηλέφωνο που αντιστοιχεί σε αυτό. Μια πρώτη λύση στο πρόβλημά μας είναι να αρχίσουμε να διαβάζουμε από την αρχή όλα τα ονόματα του τηλεφωνικού καταλόγου και μόλις βρούμε το όνομα που μας ενδιαφέρει να επιστραφεί το τηλέφωνό του. Η λύση αυτή είναι ιδιαίτερα χρονοβόρα, χρησιμοποιείται όμως στους υπολογιστές και ονομάζεται **σειριακή αναζήτηση**.



Εικόνα 2-7. Σειριακή αναζήτηση του ονόματος «Ωραίου»

Προσπαθώντας να αναλύσουμε τον αλγόριθμό μας παρατηρούμε ότι εάν το όνομα που δίνουμε προς αναζήτηση βρίσκεται στις πρώτες σελίδες του τηλεφωνικού καταλόγου, ο αλγόριθμός μας βρίσκει το αντίστοιχο τηλέφωνο γρήγορα, ενώ, εάν το όνομα που δίνουμε βρίσκεται στις τελευταίες σελίδες καθυστερεί σημαντικά. Εάν συμφωνήσουμε ότι ο χρόνος ανάγνωσης και σύγκρισης ενός ονόματος με αυτό που ψάχνουμε είναι ίσος με ένα δευτερόλεπτο, τότε ο χρόνος εκτέλεσης του αλορίθμου θα ισούται με τη θέση του ονόματος στον τηλεφωνικό κατάλογο. Έτσι, εάν αναζητούμε το 10ο στη σειρά όνομα, ο χρόνος εκτέλεσης θα είναι 10 δευτερόλεπτα. Αυτό που μας ενδιαφέρει συνήθως σε έναν αλγόριθμο είναι ο χρόνος της χειρότερης περίπτωσης που στο συγκεκριμένο παράδειγμα είναι 128000 δευτερόλεπτα και αντιστοιχεί στην αναζήτηση του τελευταίου ονόματος του τηλεφωνικού καταλόγου. Γενικεύοντας, αν ο τηλεφωνικός μας κατάλογος είχε n ονόματα, τότε ο χρόνος χειρότερης περίπτωσης θα συμβολίζονταν με O(n) θέλοντας να δείξουμε τη γραμμική σχέση ανάμεσα στο πλήθος των δεδομένων εισόδου και του χρόνου εκτέλεσης του αλγορίθμου. Ο συμβολισμός O ( ) λοιπόν μας δίνει ένα πάνω όριο για τον χρόνο εκτέλεσης ενός αλγορίθμου και αποτελεί ένα πολύ καλό μέτρο της αποδοτικότητάς του.



Εικόνα 2-8. Δυαδική Αναζήτηση του ονόματος «Παπαδάκης»

Μια δεύτερη λύση στο ίδιο πρόβλημα εκμεταλλεύεται την ιδιότητα της αλφαβητικής ταξινόμησης των ονομάτων του τηλεφωνικού καταλόγου. Όταν μας δίνεται ένα όνομα προς αναζήτηση, τότε το συγκρίνουμε με το όνομα που βρίσκεται στη μέση του τηλεφωνικού καταλόγου. Εάν το όνομα που ψάχνουμε προηγείται αλφαβητικά του μεσαίου ονόματος, τότε απορρίπτουμε αμέσως όλα τα ονόματα που βρίσκονται μετά τη μέση του τηλεφωνικού καταλόγου, ενώ, εάν το όνομα που ψάχνουμε έπεται αλφαβητικά του μεσαίου ονόματος, τότε απορρίπτουμε όλα τα ονόματα που βρίσκονται πριν τη μέση του τηλεφωνικού καταλόγου. Με μία και

μόνη σύγκριση έχουμε «ελέγξει» τα μισά ονόματα! Στη συνέχεια εφαρμόζουμε την τακτική μας στα 64000 πρώτα ή στα 64000 τελευταία ονόματα που έμειναν, συγκρίνοντας πάλι το μεσαίο όνομα με αυτό που αναζητούμε. Μετά τη νέα σύγκριση θα μας μείνουν προς έλεγχο 32000 ονόματα και συνεχίζουμε με την ίδια λογική μέχρι να βρούμε τελικά το όνομα που ψάχνουμε. Τα βήματα που θα εκτελέσει ο αλγόριθμός μας, στη χειρότερη περίπτωση, είναι μόλις 17! Ο αριθμός αυτός μπορεί να προκύψει μαθηματικά υπολογίζοντας τον δυαδικό λογάριθμο του 128000 ($\log_2 128000$) καθώς ο αλγόριθμός μας βασίστηκε σε συνεχόμενες διαιρέσεις του πλήθους των δεδομένων του προβλήματός μας με το 2. Στη γενική περίπτωση λοιπόν που ο αριθμός των ονομάτων του τηλεφωνικού καταλόγου ισούται με n, ο χρόνος χειρότερης περίπτωσης θα συμβολιζόταν με O(log n) (με log n συμβολίζουμε χάριν συντομίας τον δυαδικό λογάριθμο του n) . Ο αλγόριθμος που μόλις περιγράψαμε ονομάζεται **δυαδική αναζήτηση** και αποτελεί τον γρηγορότερο αλγόριθμο αναζήτησης, όταν τα δεδομένα μας είναι ταξινομημένα.

Οι περισσότεροι αλγόριθμοι έχουν χρονική πολυπλοκότητα που ανήκει σε μια από τις κατηγορίες: O(1), O(logn), O(n), O(nlogn), $O(n^2)$, $O(2^n)$

Η **πολυπλοκότητα αλγορίθμων** είναι ο ένας από τους δύο κλάδους του τομέα της επιστήμης των υπολογιστών που ονομάζεται **θεωρία υπολογισμού**. Ο τομέας αυτός εξετάζει αρχικά εάν μπορεί να λυθεί ένα πρόβλημα χρησιμοποιώντας κάποιο αλγόριθμο και, σε δεύτερη φάση, πόσο αποδοτικά μπορεί να λυθεί το συγκεκριμένο πρόβλημα. Ο κλάδος που αναζητά τι μπορεί να υπολογισθεί και τι όχι ονομάζεται **υπολογισιμότητα**. Για την απόδειξη της υπολογισιμότητας ή όχι κάποιου προβλήματος οι επιστήμονες της πληροφορικής χρησιμοποιούν διάφορα υπολογιστικά μοντέλα με το ισχυρότερο από αυτά να είναι η λεγόμενη **Μηχανή Τιούρινγκ** από το όνομα του μεγάλου βρετανού μαθηματικού Άλαν Τιούρινγκ που θεωρείται ο πατέρας της επιστήμης των υπολογιστών.

Εικόνα 2-9. Χρόνοι εκτέλεσης αλγορίθμων με χρόνο εκτέλεσης εντολής 1 ns

	Πλήθος δεδομένων προβλήματος				
Πολυπλοκότητα αλγορίθμου	**10**	**100**	**1000**	**10000**	**100000**
O(logn)	3,3 ns	6,6 ns	10 ns	13,3 ns	16,6 ns
O(n)	10 ns	100 ns	1000 ns	10000 ns	100000 ns
O(n^2)	100 ns	10000 ns	0,001 sec	0,1 sec	10 sec
O(n^3)	1000 ns	0,001 sec	1 sec	100 sec	278 ώρες
O(2^n)	1024 ns	401969368413 αιώνες			

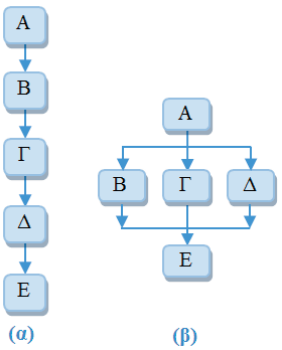
Όταν ολοκληρώσετε την ενότητα ελέγξτε αν είστε σε θέση:

- να περιγράφετε τι είναι η αποδοτικότητα ενός αλγορίθμου
- να αναγνωρίζετε και να συσχετίζετε τις έννοιες «πολυπλοκότητα» και «υπολογισιμότητα» αλγορίθμου στην επίλυση ενός προβλήματος

2.2.4 Βασικοί τύποι αλγορίθμων

Αλγόριθμοι σειριακής και παράλληλης επεξεργασίας

Οι εντολές ενός αλγορίθμου γράφονται η μια κάτω από την άλλη και εκτελούνται ακολουθιακά. Σε κάποιους αλγόριθμους, τμήματα εντολών μπορούν να εκτελεσθούν ταυτόχρονα. Ας θεωρήσουμε έναν αλγόριθμο που αποτελείται από 5 τμήματα, Α έως Ε. Στην Εικόνα 2-10 (α), βλέπουμε την εκτέλεση των τμημάτων του αλγορίθμου με σειριακή επεξεργασία. Αν τα τμήματα Β, Γ, Δ δεν εξαρτώνται το ένα από το άλλο, τότε μπορούν να εκτελεσθούν ταυτόχρονα, μειώνοντας δραστικά το χρόνο εκτέλεσης. Στην Εικόνα 2-10 (β), βλέπουμε την εκτέλεση των τμημάτων του αλγορίθμου με παράλληλη επεξεργασία.

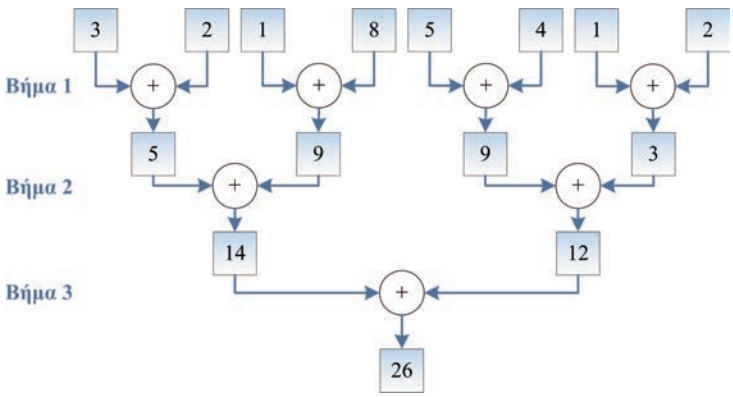


Εικόνα 2-10. Σειριακή και παράλληλη εκτέλεση αλγορίθμου

Για παράδειγμα, μπορούμε να εφαρμόσουμε τη σειριακή και παράλληλη επεξεργασία στον υπολογισμό του αθροίσματος 8 τυχαίων αριθμών. Ο αλγόριθμος σειριακής επεξεργασίας απαιτεί 8 βήματα εκτέλεσης:

Βήμα 1:	Πρόσθεσε τον πρώτο με το δεύτερο αριθμό.
Βήμα 2:	Πρόσθεσε το αποτέλεσμα με τον τρίτο αριθμό.
	…
Βήμα 8:	Πρόσθεσε το αποτέλεσμα με τον όγδοο αριθμό.

Στην Εικόνα 2-11 βλέπουμε την εφαρμογή ενός αλγορίθμου παράλληλης επεξεργασίας για την επίλυση του ίδιου προβλήματος. Η πρόσθεση του πρώτου με τον δεύτερο, του τρίτου με τον τέταρτο, του πέμπτου με τον έκτο και του έβδομου με τον όγδοο γίνονται ταυτόχρονα σε ένα βήμα.



Εικόνα 2-11. Παράλληλη εκτέλεση αλγορίθμου

Ο αλγόριθμος σειριακής επεξεργασίας απαιτεί οκτώ (8) βήματα, ενώ ο αλγόριθμος παράλληλης επεξεργασίας, μόλις τρία (3).

Αν θέλουμε να προσθέσουμε, χίλιους (1000) αριθμούς πόσα βήματα απαιτούνται με παράλληλη επεξεργασία; Πώς επηρεάζεται η απάντηση από το πλήθος των διαθέσιμων επεξεργαστών;

Αλγόριθμος σειριακής επεξεργασίας είναι ένας αλγόριθμος του οποίου τα βήματα εκτελούνται ακολουθιακά το ένα μετά το άλλο, από έναν επεξεργαστή.

Αλγόριθμος παράλληλης επεξεργασίας είναι ένας αλγόριθμος του οποίου τα βήματα μπορούν να εκτελούνται ταυτόχρονα από διαφορετικούς επεξεργαστές, μειώνοντας το χρόνο εκτέλεσης.

Μια επαναληπτική διαδικασία λέγεται **βρόχος**.

Ένας βρόχος που δεν σταματά ποτέ λέγεται **ατέρμων.**

Επαναληπτικοί αλγόριθμοι

Ένα από τα μεγάλα πλεονεκτήματα της χρήσης των υπολογιστών για την επίλυση προβλημάτων είναι η δυνατότητα επανάληψης μιας διαδικασίας πολλές φορές. Ο άνθρωπος, έστω και αν μπορεί να εκτελέσει μια ενέργεια, κουράζεται με την επανάληψή της, κάνει λάθη. Δεν ισχύει το ίδιο για τον υπολογιστή. Γι' αυτό και οι επαναληπτικοί αλγόριθμοι είναι πολύ διαδεδομένοι στους υπολογιστές.

Ο υπολογισμός του αθροίσματος δύο αριθμών είναι μια απλή πρόσθεση. Όταν, όμως, έχουμε χίλιους αριθμούς, τότε η πράξη της πρόσθεσης πρέπει να εκτελεστεί χίλιες φορές. Αυτή είναι μια επαναληπτική διαδικασία, η οποία μπορεί να εκφρασθεί ως εξής:

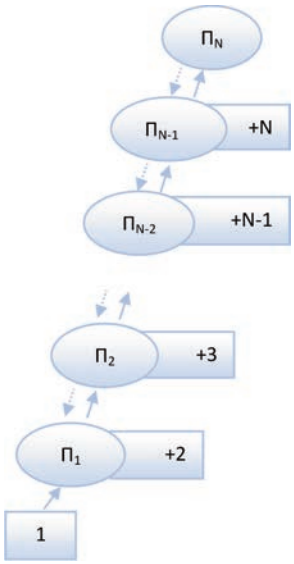
Βήμα 1:	Αρχικά το άθροισμα ισούται με μηδέν.
Βήμα 2:	Πρόσθεσε τον επόμενο αριθμό στο άθροισμα.
Βήμα 3:	Επανάλαβε 1000 φορές το βήμα 2.

Σε τρεις γραμμές ο αλγόριθμος έχει ολοκληρωθεί. Ο βρόχος μείωσε δραστικά το πλήθος των απαιτούμενων εντολών χωρίς να επηρεάσει την αποτελεσματικότητα του αλγορίθμου.

Αναδρομικοί αλγόριθμοι

Η αναδρομή είναι μια ιδιαίτερα διαισθητική αλγοριθμική μέθοδος. Η κατανόηση των αναδρομικών αλγορίθμων δεν είναι εύκολη υπόθεση. Απαιτεί ιδιαίτερο τρόπο σκέψης που κατακτάται μόνο με έντονη εξάσκηση. Ένας αναδρομικός αλγόριθμος χρησιμοποιεί τον ίδιο του τον εαυτό.

Αναδρομικός είναι ο αλγόριθμος που καλεί άμεσα ή έμμεσα τον εαυτό του μία ή περισσότερες φορές, επιλύοντας κάθε φορά ένα πρόβλημα της ίδιας φύσης με το αρχικό, αλλά μικρότερου μεγέθους.



Εικόνα 2-12. Αναδρομικός αλγόριθμος

Για παράδειγμα, ο υπολογισμός του γινομένου Π = 1*2*3*…*N μπορεί να γίνει αναδρομικά. Ας προσπαθήσουμε να λύσουμε ένα στιγμιότυπο του προβλήματος, τον υπολογισμό του 1*2*…*6 και μετά να γενικεύσουμε τη λύση, αντικαθιστώντας το 6 με N. Αναπτύσσουμε το εξής σκεπτικό: Για να υπολογίσω το γινόμενο 1*2*3*…*6, αρκεί να πολλαπλασιάσω τον αριθμό 6 με το γινόμενο 1*2*…*5. Για να υπολογίσω το γινόμενο 1*2*3*…*5, αρκεί να πολλαπλασιάσω τον αριθμό 5 με το γινόμενο 1*2*…*4. Συνεχίζοντας την απλούστευση του προβλήματος, θα φθάσω στο πρόβλημα υπολογισμού του γινομένου 1*2 που είναι τετριμμένο. Ακολουθώντας, τώρα, την αντίθετη φορά, έχοντας υπολογίσει το 1*2, υπολογίζω το (1*2)*3, μετά το ((1*2)*3)*4, κ.ο.κ., μέχρι να φθάσω στο ((1*2)*…)*6. Μπορώ να γενικεύσω το παραπάνω σκεπτικό για οποιοδήποτε ακέραιο N.

Αν συμβολίσω το γινόμενο 1*2*…*N με Π_N , τότε προκύπτει ο εξής μαθηματικός αναδρομικός ορισμός:

$\Pi_N = 1$, αν N=1

$\Pi_N = \Pi_{N-1} * N$, αν N>1

με τη χρήση του οποίου μπορώ να υπολογίσω αναδρομικά οποιοδήποτε γινόμενο Π_N.

Η δημιουργία και ο έλεγχος ορθότητας αναδρομικών αλγορίθμων

είναι ακόμα πιο δύσκολες διεργασίες που απαιτούν προγραμματιστική εμπειρία. Παρόλ' αυτά, η αναδρομή είναι πολύ ισχυρή μέθοδος, που δίνει τη δυνατότητα να περιγράψουμε με σύντομο τρόπο τη λύση δύσκολων προβλημάτων.

Ασκήσεις - Προβλήματα

1. Αναλύσετε τον υπολογισμό της αριθμητικής παράστασης ((x-y)*(x+y))/((2+x)*y) σε απλές εντολές μιας πράξης. Ποιες εντολές μπορούν να εκτελεστούν ταυτόχρονα; Πόσα βήματα απαιτούνται για τη σειριακή εκτέλεση από έναν επεξεργαστή; Πόσα βήματα απαιτούνται κατ' ελάχιστο για την παράλληλη επεξεργασία από τρεις επεξεργαστές;
2. Αναζητήσετε το «Πρόβλημα των πύργων του Ανόι». Επιλύσετε το πρόβλημα για 3 δίσκους και εκφράστε τη λύση του με φυσική γλώσσα. Χρησιμοποιήστε τη λύση που κατασκευάσατε, για να λύσετε το πρόβλημα για 4 δίσκους. Γενικεύσετε τη λύση για οποιοδήποτε πλήθος δίσκων. Εκφράσετε τη λύση σας αναδρομικά.

Όταν ολοκληρώσετε την ενότητα ελέγξτε αν είστε σε θέση:

- να αναγνωρίζετε την ύπαρξη συγκεκριμένων χαρακτηριστικών και τύπων αλγορίθμων
- να διακρίνετε έναν αλγόριθμο σειριακής επεξεργασίας από έναν αλγόριθμο παράλληλης επεξεργασίας
- να αναγνωρίζετε τις αλληλοεξαρτήσεις μεταξύ των εντολών ενός σειριακού αλγορίθμου και να εντοπίζετε ποια βήματα ενός αλγόριθμου που μπορούν να εκτελεσθούν παράλληλα.
- να προσδιορίζετε την έννοια της επαναληπτικής διαδικασίας και του βρόχου
- να αναγνωρίζετε πότε ένα πρόβλημα απαιτεί επαναληπτικό αλγόριθμο για την επίλυσή του.
- να διατυπώνετε σε φυσική γλώσσα έναν επαναληπτικό αλγόριθμο.
- να αναγνωρίζετε πότε ένας αλγόριθμος είναι αναδρομικός.

2.2.5 Αναπαράσταση αλγορίθμου

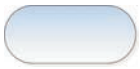
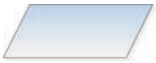
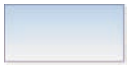
Η αναπαράσταση ενός αλγορίθμου μπορεί να γίνει με διάφορους τρόπους. Ο τρόπος περιγραφής σε συνδυασμό με αυτόν που θα κληθεί να τον εκτελέσει, επηρεάζουν την ικανοποίηση ή μη των αλγοριθμικών κριτηρίων. Επιπλέον, ο ίδιος αλγόριθμος, αν εκφρασθεί με διαφορετικούς τρόπους, μπορεί να ικανοποιεί ή όχι κάποια αλγοριθμικά κριτήρια. Για την αναπαράσταση των αλγορίθμων χρησιμοποιούνται διάφοροι τρόποι όπως η φυσική γλώσσα, το διάγραμμα ροής, οι γλώσσες περιγραφής αλγορίθμων και οι γλώσσες προγραμματισμού.

Η **φυσική γλώσσα** αποτελεί τον πιο απλό και ανεπεξέργαστο τρόπο παρουσίασης ενός αλγορίθμου, που με απλά λόγια και ελεύθερες εκφρά-

σεις περιγράφουμε τα βήματα. Ωστόσο, ο συγκεκριμένος τρόπος έκφρασης ενέχει αυξημένη πιθανότητα λάθους ή ασάφειας. Η φυσική γλώσσα ενδείκνυται μόνο σε καθημερινά προβλήματα στα οποία η ακρίβεια και η σαφήνεια δεν παίζουν σημαντικό ρόλο στην καθοριστικότητα και αποτελεσματικότητα του τρόπου επίλυσής τους.

Το **διάγραμμα ροής** είναι η αναπαράσταση του αλγορίθμου με τη χρήση γεωμετρικών σχημάτων. Είναι ο πλέον εποπτικός τρόπος παρουσίασης. Σε ένα διάγραμμα ροής μπορούμε, με μια ματιά, να αναγνωρίσουμε τις λογικές δομές που περιλαμβάνει ο αλγόριθμος. Από την άλλη, η χρήση σχημάτων περιορίζει τη δυνατότητα έκφρασης σε ένα σχετικά μικρό χώρο, όπως είναι μια εκτυπώσιμη σελίδα. Αυτό καθιστά σχεδόν αδύνατη την αναπαράσταση ενός αλγορίθμου εκατοντάδων ή χιλιάδων εντολών με ένα διάγραμμα ροής. Ο περιορισμός αυτός δεν υπάρχει σε μια γλώσσα. Τα σχήματα που χρησιμοποιούνται στα διαγράμματα του παρόντος βιβλίου και η έννοιά τους, φαίνονται στον παρακάτω πίνακα:

Εικόνα 2-13. Σύμβολα διαγραμμάτων ροής

Σχήμα		Έννοια
	Βέλος	Ροή (οδηγεί στην επόμενη εντολή)
	Έλλειψη	Αρχή - Τέλος αλγορίθμου
	Πλάγιο παραλληλόγραμμο	Είσοδος – Έξοδος
	Ορθογώνιο παραλληλόγραμμο	Επεξεργασία - Εκτέλεση πράξεων
	Ρόμβος	Συνθήκη

Οι **γλώσσες περιγραφής αλγορίθμων** αποτελούν ενδιάμεσο στάδιο αναπαράστασης. Προσεγγίζουν τις γλώσσες προγραμματισμού, είναι όμως απλούστερες. Στόχος τους είναι η κωδικοποιημένη αποτύπωση αλγορίθμων σε κείμενο, χωρίς, όμως, να ενδιαφέρουν οι λεπτομέρειες υλοποίησής τους από έναν υπολογιστή. Γι' αυτό και αναφέρονται ως ψευδογλώσσες ή ψευδοκώδικες.

Οι **γλώσσες προγραμματισμού** αποτελούν τις γλώσσες επικοινωνίας του ανθρώπου με τον υπολογιστή. Η προσπάθεια του ανθρώπου να αναθέσει την επίλυση προβλημάτων σε υπολογιστές οδήγησε στη δημιουργία γλωσσών που έχουν περιορισμένο σύνολο κωδικοποιημένων εντολών με αυστηρούς κανόνες σύνταξης. Με τη χρήση τους, ο άνθρωπος δίνει εντολές στον υπολογιστή και αυτός τις εκτελεί.

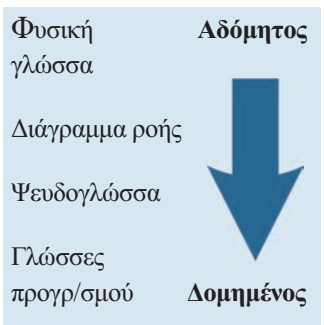
Ας δούμε τον αλγόριθμο υπολογισμού περιφέρειας και εμβαδού ενός κύκλου. ΣτηνΕικόνα 2-14, ο ίδιος αλγόριθμος έχει εκφρασθεί με τρεις διαφορετικούς τρόπους. Ανάμεσα στους τρεις τρόπους υπάρχει αντιστοιχία των εντολών. Γενικά, είναι εύκολο να μετατρέψουμε έναν αλγόριθμο από μια μορφή αναπαράστασης σε μια άλλη. Αν γνωρίζουμε καλά μια γλώσσα προγραμματισμού, τότε είναι εύκολο να μάθουμε και μια δεύ-

τερη, αρκεί να έχουμε κατανοήσει την αντιστοιχία των εντολών τους.

Φυσική Γλώσσα	Ψευδογλώσσα	Διάγραμμα ροής
Αρχή αλγορίθμου	**Αλγόριθμος** Κύκλος	
Ρώτα να μάθεις πόσο είναι η ακτίνα.	**Γράψε** 'Δώσε ακτίνα'	
Πάρε την ακτίνα.	**Διάβασε** ακ	
Πολλαπλασίασε την ακτίνα με το 2 και ότι βρεις με το 3.14. Το αποτέλεσμα είναι η περιφέρεια του κύκλου.	περ ← ακ*2*3.14	
Ύψωσε την ακτίνα στο 2 και πολλαπλασίασε ότι βρεις με το 3.14. Το αποτέλεσμα είναι το εμβαδό του κύκλου.	εμβ ← ακ^2*3.14	
Πες την περιφέρεια και το εμβαδό που υπολόγισες.	**Γράψε** περ, εμβ	
Τέλος αλγορίθμου	**Τέλος** Κύκλος	

Εικόνα 2-14. Τρεις αναπαραστάσεις του ίδιου αλγορίθμου

Ποιος από τους τρόπους αναπαράστασης είναι ο καλύτερος; Πόσο εύκολη είναι η δημιουργία διαγραμμάτων στον υπολογιστή; Ένα πρόγραμμα σε γλώσσα προγραμματισμού μπορεί να κατανοηθεί και εκτελεστεί από οποιονδήποτε άνθρωπο; Ένα κείμενο σε φυσική γλώσσα μπορεί να εκτελεστεί άμεσα από τον υπολογιστή; Οι απαντήσεις δεν είναι άμεσες ούτε εύκολες. Η φύση και το μέγεθος του προβλήματος, καθορίζουν τον καταλληλότερο τρόπο αναπαράστασης.



Συνοψίζοντας τους τρόπους αναπαράστασης αλγορίθμων, παρατηρούμε ότι ο βαθμός δόμησης του τρόπου αναπαράστασης αυξάνεται καθώς πηγαίνουμε από τον πλούτο και την ελευθερία της φυσικής γλώσσας προς τους αυστηρούς λεξικολογικούς και συντακτικούς κανόνες των γλωσσών προγραμματισμού. Η περιγραφή αλγορίθμου σε φυσική γλώσσα είναι ο λιγότερο δομημένος, ενώ οι γλώσσες προγραμματισμού ο πιο δομημένος τρόπος αναπαράστασης αλγορίθμων.

Ασκήσεις - Προβλήματα

1. Γράψτε αλγόριθμο που δέχεται τον προφορικό και το γραπτό βαθμό ενός μαθητή υπολογίζει το μέσο όρο του και εξάγει αποτέλεσμα προαγωγής. Ο μαθητής προάγεται αν έχει μέσο όρο πάνω από 9.5, διαφορετικά παραπέμπεται για επανεξέταση. Εκφράστε τον αλγόριθμο και με τους τρεις τρόπους αναπαράστασης, φυσική γλώσσα, διάγραμμα ροής και Ψευδογλώσσα

Όταν ολοκληρώσετε την ενότητα ελέγξτε αν είστε σε θέση:

- να αναγνωρίζετε τις διάφορες μορφές αναπαράστασης του αλγορίθμου.
- να συγκρίνετε και να επιλέγετε τον κατάλληλο τρόπο αναπαράστασης ανάλογα με το πρόβλημα και τον αλγόριθμο

2.2.6 Τα δεδομένα και η αναπαράστασή τους

Η έννοια των δεδομένων είναι πολύ βασική για την Πληροφορική. Τα δεδομένα προέρχονται από τον χώρο των προβλημάτων. Μπορούμε να πούμε ότι τα δεδομένα αποτελούν μια «αφαίρεση» του πραγματικού κόσμου. Για παράδειγμα, ένας μαθητής είναι μια οντότητα με μεγάλο πλήθος χαρακτηριστικών και ιδιοτήτων, όπως: ονοματεπώνυμο, όνομα πατρός, ημερομηνία γέννησης, αγαπημένη ομάδα, αγαπημένο φαγητό, ενδιαφέροντα, φίλοι κ.ά. Για το μαθητολόγιο του σχολείου είναι απαραίτητη η αναπαράσταση μόνο ενός μικρού μέρους από τα παραπάνω χαρακτηριστικά, όπως το ονοματεπώνυμο, το όνομα πατρός και η ημερομηνία γέννησης του μαθητή. Η Εικόνα 2-15 δείχνει την αντιστοίχηση ανάμεσα σε οντότητητες και έννοιες του πραγματικού κόσμου με έννοιες της Πληροφορικής, με ένα παράδειγμα.

Εικόνα 2-15. Αντιστοίχηση του πραγματικού κόσμου με έννοιες της Πληροφορικής

Πραγματικός κόσμος	Πληροφορική	Παράδειγμα
Οντότητες (αντικείμενα)	Δεδομένα	Μαθητής, Βαθμός
Σχέσεις οντοτήτων	Συσχετίσεις δεδομένων	Ο μαθητής έχει βαθμούς
Ενέργειες	Αλγόριθμοι	Ο μέσος όρος υπολογίζεται αθροίζοντας τους βαθμούς και διαιρώντας με το πλήθος τους

Ο μαθητής και ο βαθμός του αποτελούν οντότητες του πραγματικού κόσμου. Η Πληροφορική αναπαριστά αυτές τις οντότητες ως δεδομένα. Στον πραγματικό κόσμο ο μαθητής έχει βαθμούς. Η Πληροφορική συσχετίζει τα δεδομένα του μαθητή με τα δεδομένα “βαθμοί”. Ο μέσος όρος βαθμολογίας του μαθητή υπολογίζεται από τα δεδομένα των βαθμών του με συγκεκριμένο τρόπο. Ο υπολογισμός αυτός είναι μια ενέργεια που θα υλοποιηθεί από έναν αλγόριθμο.
Στόχος της Πληροφορικής είναι να μελετήσει, να αναπαραστήσει και να επεξεργαστεί τα δεδομένα για να δημιουργήσει *πληροφορία* και *γνώση* που θα αξιοποιηθούν στην επίλυση προβλημάτων.

Δεδομένα (data) είναι τα στοιχεία που αναπαρίστανται σε αλγορίθμους και προγράμματα και υφίστανται επεξεργασία.

Όλα τα στοιχεία που θα αναπαρασταθούν σε έναν αλγόριθμο ή ένα πρόγραμμα ονομάζονται δεδομένα, είτε είναι είσοδοι είτε έξοδοι (αποτελέσματα).

Θα πρέπει εδώ να σημειωθεί ότι η έννοια των δεδομένων στην Πληροφορική δεν είναι ταυτόσημη με την έννοια των δεδομένων μιας εκφώνησης σε ένα πρόβλημα π.χ. Μαθηματικών ή Φυσικής, αλλά είναι πιο γενική. Έστω για παράδειγμα το απλό πρόβλημα υπολογισμού του εμβαδού τριγώνου που έχει βάση β και ύψος υ. Τα στοιχεία που σχετίζονται με το πρόβλημα είναι η βάση, το ύψος, αλλά και το εμβαδόν του τριγώνου. Τα στοιχεία αυτά είναι τα δεδομένα που θα αναπαρασταθούν στον αλγόριθμο υπολογισμού του εμβαδού.

Όλα τα δεδομένα αποθηκεύονται στον υπολογιστή. Τα κυκλώματα των ψηφιακών υπολογιστών λειτουργούν σε δύο καταστάσεις, 0 και 1.

Το 0 και το 1 ονομάζονται **δυαδικά ψηφία** ή **bits** (**bi**nary digi**ts**). Άρα, τα ψηφιακά δεδομένα είναι σειρές από 0 και 1. Τα δυαδικά ψηφία ομαδοποιούνται σε **οκτάδες** που ονομάζονται **bytes**. Ένα byte μπορεί να παριστάνει ένα γράμμα, έναν αριθμό, ένα σύμβολο ή ένα τμήμα αυτών. Οι ακέραιοι αριθμοί συνήθως αναπαρίστανται με 4 ή 8 bytes. Το πλήθος των bits που χρησιμοποιείται για να αναπαρασταθούν τα δεδομένα καθορίζει το εύρος τους.

Ο αριθμός 27 παριστάνεται ως 00011011, ενώ το λατινικό γράμμα A παριστάνεται ως 01000001. Το 27 με 2 bytes θα παριστάνονταν ως 00000000 00011011.

Οι γλώσσες προγραμματισμού υψηλού επιπέδου δίνουν την δυνατότητα χρήσης *τύπων δεδομένων* που αποκρύπτουν από τον προγραμματιστή την αναπαράσταση σε bits και bytes. Ο **τύπος δεδομένων** (data type) σε μια γλώσσα προγραμματισμού αναφέρεται σε ένα είδος δεδομένων που μπορεί να παραστήσει η γλώσσα. Ένας τύπος δεδομένων καθορίζεται από το *όνομά* του, το *είδος* και το *εύρος* των τιμών του καθώς και από τις *πράξεις* που μπορούν να εφαρμοστούν σε αυτόν. Οι τύποι δεδομένων διακρίνονται σε **απλούς** και σε **σύνθετους**. Παραδείγματα απλών τύπων δεδομένων που υπάρχουν στις περισσότερες γλώσσες προγραμματισμού είναι ο **ακέραιος**, ο **πραγματικός**, ο **λογικός** και ο **χαρακτήρας**. Η Εικόνα 2-16 συνοψίζει τα χαρακτηριστικά των τεσσάρων αυτών απλών τύπων δεδομένων.

Τύπος δεδομένων	Είδος τιμών	Παραδείγματα τιμών	Βασικές πράξεις
Ακέραιος	Ακέραιοι αριθμοί με πρόσημο	0, 37, -156	Σύγκριση, πρόσθεση, αφαίρεση, πολλαπλασιασμός, διαίρεση
Πραγματικός	Πραγματικοί αριθμοί με υποδιαστολή	1.67, -0.0345, 3.14159	Σύγκριση, πρόσθεση, αφαίρεση, πολλαπλασιασμός, διαίρεση
Λογικός	Οι δύο λογικές τιμές Ψευδής και Αληθής	Ψευδής, Αληθής	Σύγκριση
Χαρακτήρας	Σύμβολα	'A', 'a', 'γ', '@', '8', '€'	Σύγκριση

Εικόνα 2-16. Απλοί τύποι δεδομένων

Οι απλοί τύποι δεδομένων χρησιμοποιούνται σε απλά προβλήματα, όπως το πρόβλημα υπολογισμού του εμβαδού τριγώνου. Η βάση και το ύψος του τριγώνου θα αναπαρασταθούν με τον τύπο δεδομένων "πραγματικός", εφόσον οι διαστάσεις είναι πραγματικοί αριθμοί. Το εμβαδόν του τριγώνου θα αναπαρασταθεί επίσης με τον τύπο δεδομένων "πραγματικός", εφόσον προκύπτει από υπολογισμό που περιέχει τη βάση και το ύψος. Στο μαθητολόγιο παριστάνονται δεδομένα όπως το φύλο, η τάξη και το τμήμα του μαθητή, οι προφορικοί βαθμοί και ο τελικός μέσος όρος και ο χαρακτηρισμός της φοίτησής του. Στον ακόλουθο πίνακα περιγράφονται οι τύποι δεδομένων για τα στοιχεία του μαθητολογίου. Ο προφορικός βαθμός είναι ακέραιος αριθμός με τιμές από 0 ως 20. το φύλο του μαθητή έχει την τιμή Α (Άρρεν) ή Θ (Θήλυ), άρα μπορεί να παρασταθεί ως χαρακτήρας. Η τάξη Α, Β ή Γ Λυκείου επίσης μπορεί να παρασταθεί με έναν χαρακτήρα, ενώ το τμήμα είναι ακέραιος αριθμός με τιμές 1, 2, 3, κλπ. Ο μέσος όρος βαθμολογίας είναι πραγματικός αριθ-

μός, π.χ. 18,7. Ο χαρακτηρισμός της φοίτησης ως επαρκούς είναι Αληθής αν η φοίτηση είναι επαρκής και Ψευδής στην αντίθετη περίπτωση, άρα είναι δεδομένο λογικού τύπου.

Εικόνα 2-17. Παραδείγματα δεδομένων του μαθητολογίου και των τύπων τους

Δεδομένα προβλήματος	Είδος και εύρος αποδεκτών τιμών	Τύπος δεδομένων
Βαθμός Α' τετραμήνου	Ακέραιος από 0 ως 20	Ακέραιος
Φύλο μαθητή	'Α' ή 'Θ'	Χαρακτήρας
Τάξη μαθητή	'Α', 'Β' ή 'Γ'	Χαρακτήρας
Τμήμα μαθητή	1, 2, 3 ή 4 …	Ακέραιος
Μέσος όρος τετραμήνων	Πραγματικός από 0 ως 20	Πραγματικός
Επαρκής φοίτηση	Αληθής ή Ψευδής	Λογικός

Σε πολλές γλώσσες προγραμματισμού υλοποιούνται σύνθετοι τύποι δεδομένων. Ένα τέτοιο παράδειγμα είναι οι συμβολοσειρές (strings) που αποτελούνται από μια σειρά χαρακτήρων για να αναπαραστήσουν δεδομένα κειμένου. Ένα παράδειγμα συμβολοσειράς είναι η λέξη «Πληροφορική».

Δομές δεδομένων

Σε περίπλοκα υπολογιστικά προβλήματα υπάρχει η ανάγκη να οργανωθούν τα δεδομένα με σύνθετους τρόπους. Ας θεωρήσουμε το παράδειγμα του μαθητολόγιου. Οι βαθμοί ενός μαθητή σε όλα τα μαθήματα είναι βολικό να καταχωρούνται σε έναν πίνακα που αποτελείται από γραμμές και στήλες.

Εικόνα 2-18. Βαθμολογίες ενός μαθητή σε τρία μαθήματα

	Α' τετράμηνο	Β' τετράμηνο	Μ.Ο. προφορικών	Γραπτός βαθμός	Μ.Ο. μαθήματος
Ιστορία	18	19	18.5	17	17.75
Μαθηματικά	17	16	16.5	18.5	17.5
Αρχαία	18	18	18	20	19

Ο πίνακας είναι μία δομή δεδομένων. Άλλες δομές δεδομένων είναι οι λίστες, οι ουρές, οι στοίβες, τα δέντρα και οι γράφοι.

Η δομή του πίνακα καθορίζει ότι τα δεδομένα οργανώνονται σε συνεχόμενες θέσεις.

Κάθε γραμμή παριστάνει ένα μάθημα, ενώ οι στήλες παριστάνουν τις περιόδους βαθμολόγησης. Αυτή η διάταξη μας διευκολύνει να συγκεντρώσουμε τους βαθμούς του μαθητή και να τους επεξεργαστούμε. Όταν τα δεδομένα είναι σύνθετα και πρέπει να συσχετιστούν μεταξύ τους, είναι σημαντικό να οργανωθούν ώστε, αφενός μεν να λύνεται σωστά το πρόβλημα, αφετέρου δε να γίνεται αποδοτική διαχείριση της μνήμης και των υπολογιστικών πόρων. Για το λόγο αυτό δημιουργούνται οι δομές δεδομένων.

Δομή δεδομένων είναι ένας τρόπος οργάνωσης, συσχέτισης και αποθήκευσης των δεδομένων.

Στις δομές δεδομένων εφαρμόζονται λειτουργίες όπως:

- **Εισαγωγή**: ένα νέο στοιχείο δεδομένων προστίθεται στη δομή και συσχετίζεται με τα στοιχεία που ήδη υπάρχουν σε αυτή.
- **Διαγραφή**: ένα στοιχείο αφαιρείται από τη δομή.

- **Προσπέλαση της δομής**: γίνεται πρόσβαση στα στοιχεία της με συγκεκριμένη μέθοδο.
- **Αναζήτηση στοιχείου** μέσα στη δομή.
- **Ταξινόμηση**: τα στοιχεία της δομής αναδιοργανώνονται έτσι ώστε να ακολουθούν συγκεκριμένη διάταξη.

Η επιλογή των κατάλληλων δομών δεδομένων είναι μέρος της λύσης του προβλήματος. Υπάρχουν δομές που είναι κατάλληλες για συγκεκριμένους τύπους προβλημάτων. Ένα από τα κριτήρια επιλογής δομών δεδομένων είναι ο τρόπος διαχείρισης της μνήμης του υπολογιστή. Με βάση αυτό το κριτήριο, οι δομές δεδομένων διακρίνονται σε δύο (2) κατηγορίες.

Οι **στατικές δομές δεδομένων** έχουν σταθερό μέγεθος το οποίο καθορίζεται κατά την δημιουργία (συγγραφή) του προγράμματος. Το μέγεθος αυτό δεν αλλάζει, επομένως οι δομές αυτές αποθηκεύουν συγκεκριμένο πλήθος δεδομένων.

Οι **δυναμικές δομές δεδομένων** δεν έχουν σταθερό μέγεθος και αυξομειώνονται με τη μέθοδο της δυναμικής παραχώρησης μνήμης. Εφαρμόζονται σε προβλήματα όπου το μέγεθος των δεδομένων δεν είναι γνωστό από την αρχή ή δε μπορεί να εκτιμηθεί, καθώς και σε περιπτώσεις όπου είναι κρίσιμο να μην δεσμεύεται μνήμη που δε θα χρησιμοποιηθεί.

Ερωτήσεις - θέματα για συζήτηση

1. Περιγράψτε ένα πρόβλημα από: α) τη Φυσική, β) την καθημερινότητά σας, γ) το σχολικό περιβάλλον, και εντοπίστε τα δεδομένα που πρέπει να αναπαρασταθούν από τον αλγόριθμο που θα λύσει το πρόβλημα.
2. Μπορείτε να περιγράψετε περιπτώσεις από τον πραγματικό κόσμο όπου δημιουργούνται: α) ουρές, β) στοίβες, γ) ιεραρχίες (δέντρα) και δ) γράφοι;

Ασκήσεις - Προβλήματα

1. Ένας μετεωρολογικός σταθμός θέλει να εξάγει στατιστικά για τη θερμοκρασία μίας πόλης. Ποια είναι τα δεδομένα που πρέπει να συγκεντρωθούν για: α) να υπολογιστεί η ώρα κατά την οποία καταγράφεται η μέγιστη θερμοκρασία της πόλης μέσα σε μία μέρα; β) να υπολογιστεί η μέση θερμοκρασία της πόλης στις 12:00 το μεσημέρι κατά τη διάρκεια των καλοκαιρινών μηνών;
2. Ποιον από τους βασικούς τύπους δεδομένων θα χρησιμοποιούσατε για τα παρακάτω στοιχεία: α) ηλικία, β) τιμή προϊόντος, γ) θερμοκρασία ατμόσφαιρας, δ) αν βρέχει ή όχι;
3. Περιγράψτε τη δομή δεδομένων: α) για τα μηνιαία έξοδα της οικογένειάς σας, β) για το εβδομαδιαίο πρόγραμμα των μαθημάτων σας στο σχολείο, γ) για τις κινήσεις των παικτών σε ένα παιχνίδι σκακιού, δ) για την ταξινόμηση των θηλαστικών (αναζητήστε πληροφορίες σε κάποιο βιβλίο βιολογίας).

Όταν ολοκληρώσετε την ενότητα ελέγξτε αν είστε σε θέση:

- να περιγράφετε τι είναι τα δεδομένα και ποιά είναι η σχέση τους με τα προγράμματα
- να διακρίνετε πώς αναπαριστά τα δεδομένα ο υπολογιστής και πώς οι γλώσσες προγραμματισμού υψηλού επιπέδου
- να αναγνωρίζετε και να ονομάζετε τα δεδομένα συγκεκριμένων προβλημάτων
- να περιγράφετε τι είναι οι τύποι δεδομένων και να επιλέγετε κατάλληλους τύπους δεδομένων για συγκεκριμένα προβλήματα
- να περιγράφετε τι είναι οι δομές δεδομένων
- να επιλέγετε στατικές ή δυναμικές δομές δεδομένων για συγκεκριμένα προβλήματα

2.2.7 Εντολές και δομές αλγορίθμου

Για την αναπαράσταση των αλγορίθμων του βιβλίου, θα χρησιμοποιήσουμε μια Ψευδογλώσσα. Πριν δούμε αναλυτικά τις εντολές της, θα αποσαφηνίσουμε κάποιες βασικές έννοιες ενός αλγορίθμου.

Σταθερά είναι μια ποσότητα που η τιμή της δεν αλλάζει κατά τη διάρκεια εκτέλεσης του αλγορίθμου.

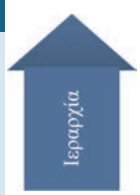
Μεταβλητή είναι μια ποσότητα που αναπαριστά ένα στοιχείο που έχει νόημα στον πραγματικό κόσμο. Η τιμή της μεταβλητής μπορεί να αλλάζει κατά τη διάρκεια εκτέλεσης του αλγορίθμου. Μια μεταβλητή αντιστοιχεί σε μια θέση μνήμης στον υπολογιστή. Το περιεχόμενό της θέσης μνήμης είναι η τιμή της μεταβλητής.

Κάθε μεταβλητή έχει όνομα που εκφράζει το περιεχόμενό της. Το όνομα πρέπει:
- να μην έχει κενά
- να μην έχει σύμβολα (+,-,*,$,%)
- να μην ξεκινά από αριθμό, αν και μπορεί να περιέχει.
- να μην είναι δεσμευμένη λέξη (λέξη που χρησιμοποιείται στη γλώσσα για ειδικό σκοπό)

Οι σταθερές και οι μεταβλητές αποτελούν τα δεδομένα που υφίστανται επεξεργασία από τις εντολές του αλγορίθμου. Τόσο οι σταθερές, όσο και οι μεταβλητές είναι κάποιου συγκεκριμένου είδους, που λέγεται **τύπος** του δεδομένου. Στην Ψευδογλώσσα υποστηρίζονται 4 βασικοί τύποι δεδομένων: Ακέραιος, Πραγματικός, Λογικός, Χαρακτήρες.

Τελεστής είναι το σύμβολο που χρησιμοποιείται για την εκτέλεση μιας πράξης. Υποστηρίζονται τρία είδη πράξεων, οι αριθμητικές, οι συγκριτικές και οι λογικές. Ο τελεστής ^ είναι η ύψωση σε δύναμη. Οι τελεστές div και mod υλοποιούν το πηλίκο και το υπόλοιπο της διαίρεσης

Αριθμητικοί	Συγκριτικοί	Λογικοί
^		ΟΧΙ (άρνηση)
* / div mod	>, < , >= , <= , = , <>	ΚΑΙ (σύζευξη)
+ -		Ή (διάζευξη)



Υψηλή

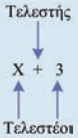
Χαμηλή

Εικόνα 2-19. Οι τελεστές της Ψευδογλώσσας και η προτεραιότητά τους

ακεραίων, αντίστοιχα. Ο συγκριτικός τελεστής <> είναι το διάφορο.

Έκφραση είναι ο συνδυασμός σταθερών και μεταβλητών (τελεστέοι) με πράξεις (τελεστές). Κάθε έκφραση μπορεί να αποτιμηθεί. Η τελική τιμή μιας έκφρασης εξαρτάται από την ιεραρχία των πράξεων και τη χρήση των παρενθέσεων. Η τιμή μιας έκφρασης μπορεί να είναι είτε αριθμός, οπότε μιλάμε για **αριθμητική έκφραση**, είτε ΑΛΗΘΗΣ ή ΨΕΥΔΗΣ οπότε μιλάμε για **λογική έκφραση**. Μια λογική έκφραση που περιέχει λογικούς τελεστές λέγεται **σύνθετη**, διαφορετικά **απλή**. Σε μια έκφραση εκτελούνται πρώτα οι αριθμητικοί, μετά οι συγκριτικοί και τέλος οι λογικοί τελεστές.

Μια έκφραση σχηματίζεται όταν εφαρμόζουμε πράξεις πάνω σε δεδομένα. π.χ.



Τελεστής: αυτός που τελεί, κάνει, πράττει = πράξη
Τελεστέος: αυτός που τελείται, παθαίνει την πράξη

Αν ο αριθμητικός τελεστής + αντικατασταθεί από το συγκριτικό > τότε η έκφραση από αριθμητική γίνεται λογική.

Γενική δομή αλγορίθμου

Γενικά, η έκφραση ενός αλγορίθμου σε Ψευδογλώσσα πρέπει να ακολουθεί την παρακάτω σύνταξη:

```
Αλγόριθμος Όνομα(ΔεδομέναΕισόδου↓, ΔεδομέναΕξόδου↑)

    Εντολές

Τέλος Όνομα
```

Ο αλγόριθμος αποτελείται από: την επικεφαλίδα και το κύριο σώμα.
Στην επικεφαλίδα πρώτα γράφεται η λέξη Αλγόριθμος. Ακολουθεί ένα όνομα που προσδιορίζει τη λειτουργία του αλγορίθμου και μέσα σε παρενθέσεις οι μεταβλητές εισόδου (↓) και εξόδου (↑).
Το κύριο σώμα περιέχει τις εντολές του αλγορίθμου.

Βασικές εντολές αλγορίθμου

Τρεις είναι οι βασικές εντολές της Ψευδογλώσσας και αντιστοιχούν στις τρεις βασικές λειτουργίες που επιτελούνται σε οποιοδήποτε κωδικοποιημένο αλγόριθμο: η εντολή εισόδου με την οποία εισάγονται δεδομένα στον αλγόριθμο, η εντολή εξόδου με την οποία εξάγονται τα αποτελέσματά του και η εντολή επεξεργασίας που εκτελεί πράξεις ανάμεσα στα δεδομένα.

α. Εντολή Εισόδου

Σύνταξη	**Διάβασε** λίστα_μεταβλητών
Λειτουργία	Με την εντολή αυτή ο εισάγονται τιμές από το χρήστη. Κατά την εκτέλεση της, σταματά προσωρινά η ροή εκτέλεσης του αλγόριθμου. Κάθε τιμή που εισάγεται εκχωρείται στην αντίστοιχη μεταβλητή. Όταν δοθούν τόσες τιμές, όσες και οι μεταβλητές στη λίστα, ολοκληρώνεται η είσοδος και συνεχίζει η εκτέλεση του αλγόριθμου.
Π.χ.	**Διάβασε** ΦΠΑ,Τ

Διάβασε ΦΠΑ,Τ

β. Εντολή εκχώρησης τιμής

Σύνταξη	μεταβλητή ← έκφραση
Λειτουργία	Με την εντολή αυτή εκτελούνται οι πράξεις που υπάρχουν στην έκφραση, υπολογίζεται η τιμή της και αποδίδεται στη μεταβλητή.
Π.χ.	Φ ← Τ*ΦΠΑ

Φ← Τ * ΦΠΑ

γ. Εντολή Εξόδου

Γράψε Φ

Σύνταξη	**Γράψε** λίστα_αποτελεσμάτων
Λειτουργία	Με την εντολή αυτή παρουσιάζονται τα αποτελέσματα του αλγόριθμου. Η μορφή της εξόδου εξαρτάται από τον εκτελεστή και δεν είναι σημαντική. Θα μπορούσε να είναι η αναφώνηση από τον εκτελεστή, η εμφάνιση σε μια οθόνη, η εκτύπωση σε έναν εκτυπωτή. Κάθε αποτέλεσμα μπορεί ναι είναι είτε ένα μήνυμα μέσα σε εισαγωγικά οπότε παρουσιάζεται αυτούσιο, ή μια μεταβλητή οπότε παρουσιάζεται η τιμή της ή μια παράσταση οπότε υπολογίζεται και παρουσιάζεται η τιμή της.
Π.χ.	**Γράψε** Φ

Δομές Αλγορίθμων

Οι βασικές αλγοριθμικές δομές είναι η δομή **ακολουθίας**, η δομή **επιλογής** και η δομή **επανάληψης**.

Α. Δομή Ακολουθίας

Η δομή ακολουθίας στην καθημερινή ζωή της Ελένης

Έχω πάρτι το βράδυ! Έχω όμως και διάβασμα. Πρέπει να τελειώσω στην ώρα μου, για να έχω χρόνο μετά να ετοιμαστώ και να είμαι στις 9 στο πάρτι. Ας οργανωθώ: 7:00-7:30 να διαβάσω. 7:30-7:45 να κάνω μπάνιο. 7:45-8:00 να στεγνώσω τα μαλλιά μου. 8:00-8:15 να μακιγιαριστώ. 8:15-8:30 να ντυθώ. 8:45-9:00 να πάω στο πάρτι. Όλα πρέπει να εκτελεσθούν στη σειρά και με απόλυτη ακρίβεια. Αν αλλάξω τη σειρά των βημάτων ή παραλείψω κάποιο βήμα, δε θα έχω το επιθυμητό αποτέλεσμα.

Η δομή ακολουθίας χρησιμοποιείται για την αντιμετώπιση απλών προβλημάτων όπου η εκτέλεση των εντολών είναι αποκλειστικά σειριακή. Πρακτικά, η δομή ακολουθίας περιέχεται σε κάθε αλγόριθμο.

Παράδειγμα 2-3. Στο ανταλλακτήριο συναλλάγματος, ισχύουν σήμερα οι ισοτιμίες της Εικόνας 2-10. Να μετατρέψετε ένα χρηματικό ποσό δοσμένο σε ευρώ σε αντίστοιχα ποσά σε δολάρια, λίρες και γεν.

EUR	USD	GBP	JPY
1	1.362	0.8013	138.75

Εικόνα 2-20. Ισοτιμίες συναλλάγματος

```
Αλγόριθμος Συνάλλαγμα(EUR↓,USD↑,GBP↑,JPY↑)
USD ← EUR * 1.362
GBP ← EUR * 0.8013
JPY ← EUR * 138.75
Τέλος Συνάλλαγμα
```

Β. Δομή Επιλογής

Η δομή επιλογής στην καθημερινή ζωή της Μαρίας

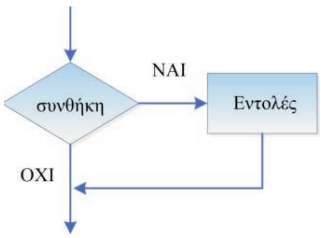
Ποιο δρόμο να ακολουθήσω για να γυρίσω από το σχολείο στο σπίτι; Φυσικά τον πιο σύντομο. Μήπως, όμως, ο πιο σύντομος δρόμος εξαρτάται από κάποιες συνθήκες; Πρώτα από όλα, με ποιο μεταφορικό μέσο μετακινούμαι, με τα πόδια, το ποδήλατο ή το αυτοκίνητο της μαμάς; Άλλος είναι ο συντομότερος δρόμος αν είμαι πεζή και άλλος με το αυτοκίνητο. Και αν είμαι με το αυτοκίνητο τι ώρα είναι και τι κίνηση έχει; Μήπως κάποιος δρόμος είναι κλειστός λόγω έργων; Αν δεν είναι, έχει καλώς. Αλλά αν είναι; Καλύτερα να πάω με τα πόδια και ας βρέχει !!!!

Στην επίλυση προβλημάτων, συχνά συναντάμε την ανάγκη να πάρουμε αποφάσεις και να επιλέξουμε διαφορετικούς τρόπους αντιμετώπισης, με βάση κάποια κριτήρια. Εξετάζουμε εναλλακτικές περιπτώσεις και σε κάθε μία εκτελούμε διαφορετικές εντολές. Οι περιπτώσεις διαμορφώνονται από τον έλεγχο των συνθηκών που επικρατούν. Στην καθημερινή ζωή, η απόφαση που παίρνουμε έχει τη λέξη Αν. «Αν ισχύει αυτό, θα κάνω εκείνο…». Παρόμοια, στην Ψευδογλώσσα, η δομή επιλογής εκφράζεται με την εντολή Αν σε τρεις μορφές: την απλή, τη σύνθετη και την πολλαπλή επιλογή.

Β.1. Απλή επιλογή

Η δομή απλής επιλογής χρησιμοποιείται όταν θέλουμε να εξετάσουμε μια συνθήκη και να εκτελέσουμε ή να παρακάμψουμε κάποιες εντολές.

Σύνταξη	**Αν** συνθήκη **τότε** **Αν** συνθήκη **τότε** εντολή Εντολές **Τέλος_αν**
Λειτουργία	Αν ισχύει η συνθήκη (δηλαδή αν είναι αληθής), τότε μόνο εκτελούνται οι εντολές. Σε κάθε περίπτωση, στη συνέχεια εκτελείται η εντολή που ακολουθεί τη δεσμευμένη λέξη τέλος_αν. Όταν θέλουμε να εκτελέσουμε μόνο μια βασική εντολή, η επιλογή μπορεί να γραφεί σε μια σειρά χωρίς τη λέξη τέλος_αν.

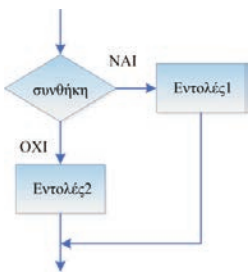


Εικόνα 2-21. Διάγραμμα ροής της απλής επιλογής

Β.2. Σύνθετη επιλογή

Η σύνθετη επιλογή είναι η πιο συχνά εμφανιζόμενη εντολή επιλογής. Με τον έλεγχο μιας συνθήκης, μπορούμε να διακρίνουμε δύο περιπτώσεις και σε κάθε μία να εκτελέσουμε διαφορετικές εντολές.

Σύνταξη	**Αν** συνθήκη **τότε** Εντολές1 **αλλιώς** Εντολές2 **Τέλος_αν**
Λειτουργία	Αν ισχύει η συνθήκη (δηλαδή αν είναι αληθής), τότε εκτελούνται μόνο οι Εντολές1. Αν η συνθήκη δεν ισχύει (αν είναι ψευδής) τότε εκτελούνται μόνο οι Εντολές2. Σε κάθε περίπτωση, η εκτέλεση του αλγόριθμου συνεχίζεται με την εντολή που ακολουθεί τη λέξη τέλος_αν.



Εικόνα 2-22. Διάγραμμα ροής της σύνθετης επιλογής

Ας δούμε τη χρήση των βασικών αυτών δομών σε δύο απλά υπολογιστικά προβλήματα.

Παράδειγμα 2-4. Να υπολογιστεί η απόλυτη τιμή ενός αριθμού.
Από τον μαθηματικό ορισμό της απόλυτης τιμής γνωρίζουμε ότι η απόλυτη τιμή ισούται με τον ίδιο τον αριθμό αν είναι θετικός ή με τον αντίθετό του αν είναι αρνητικός. Άρα, υπάρχουν δύο περιπτώσεις, που μπορούν να εξετασθούν με χρήση σύνθετης ή απλής επιλογής.

Αλγόριθμος 2-1(α). Απόλυτη τιμή αριθμού με σύνθετη επιλογή

```
Αλγόριθμος ΑπΤιμή1(x↓, ατ↑)
Αν x < 0 τότε
     ατ ← x * -1
Αλλιώς
     ατ ← x
Τέλος_αν
Τέλος ΑπΤιμή1
```

Αλγόριθμος 2-1(β). Απόλυτη τιμή αριθμού με απλή επιλογή

```
Αλγόριθμος ΑπΤιμή2(x↓, ατ↑)
ατ ← x
Αν x < 0 τότε
     ατ ← x * -1
Τέλος_αν
Τέλος ΑπΤιμή2
```

Η λογική της λύσης (β) με την απλή επιλογή έχει μια σημαντική ιδιότητα. Μπορεί να επεκταθεί για 3, ή περισσότερους αριθμούς και να εφαρμοσθεί επαναληπτικά.

Παράδειγμα 2-5. Να υπολογιστεί ο μεγαλύτερος από δύο αριθμούς.

Αλγόριθμος 2-2. Μέγιστος μεταξύ δύο αριθμών (α) με σύνθετη επιλογή, (β) με αοπλή επιλογή

```
Αλγόριθμος MAX1(x↑,y↓,μεγ↑)
Αν x < y τότε
     μεγ ← y
Αλλιώς
     μεγ ← x
Τέλος_αν
Τέλος MAX1
```

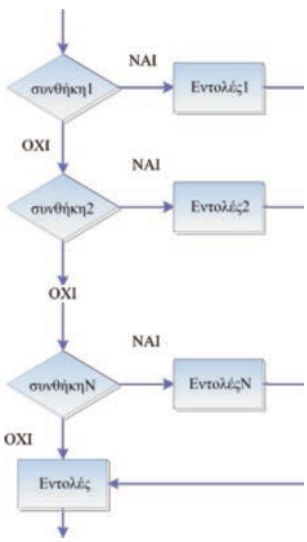
(α)

```
Αλγόριθμος MAX2(x↑,y↓,μεγ↑)
μεγ ← x
Αν μεγ < y τότε
     μεγ ← y
Τέλος_αν
Τέλος MAX2
```

(β)

Β.3. Πολλαπλή επιλογή

Όταν σε κάποιο σημείο του αλγορίθμου, θέλουμε να εξετάσουμε παραπάνω από δύο περιπτώσεις και να εκτελέσουμε διαφορετική ενέργεια σε κάθε μία, τότε χρησιμοποιούμε τη δομή της πολλαπλής επιλογής. Η εξέταση των περιπτώσεων γίνεται με τον έλεγχο διαφορετικών συνθηκών και υλοποιείται στην Ψευδογλώσσα με την εντολή πολλαπλής επιλογής:



Εικόνα 2-23. Διάγραμμα ροής της πολλαπλής επιλογής

Σύνταξη	**Αν** συνθήκη1 **τότε** εντολές1 **αλλιώς_αν** συνθήκη2 **τότε** εντολές2 … **αλλιώς_αν** συνθήκηΝ **τότε** εντολέςΝ **αλλιώς** εντολές **τέλος_αν**
Λειτουργία	Πρώτα ελέγχεται η σύνθηκη1. Αν ισχύει, τότε εκτελούνται οι εντολές1. Αν δεν ισχύει ελέγχεται η συνθήκη2. Η εκτέλεση συνεχίζεται ομοίως. Όταν μια συνθήκη είναι ψευδής, προχωράμε στον έλεγχο της επόμενης. Όταν μια συνθήκη είναι αληθής, εκτελούμε τις αντίστοιχες εντολές και μεταβαίνουμε στο τέλος της δομής. Η εκτέλεση του αλγορίθμου συνεχίζεται με την εντολή που βρίσκεται μετά το τελος_αν..

Παράδειγμα 2-6. Ο δείκτης μάζας σώματος (ΔΜΣ) είναι μία ένδειξη για το βαθμό παχυσαρκίας ενός ατόμου και υπολογίζεται από τον τύπο: βάρος/ύψος^2. Δεδομένων του βάρους και του ύψους ενός ατόμου, να εμφανίσετε χαρακτηρισμό που αφορά στο βαθμό παχυσαρκίας του, με βάση το ΔΜΣ.
Υπάρχουν τέσσερεις περιπτώσεις για τον χαρακτηρισμό του ατόμου, που μπορούν να εξετασθούν με μια δομή πολλαπλής επιλογής.

Οι πολλαπλές επιλογές μπορούν να γίνουν και με εμφωλευμένες δομές επιλογής

```
Αλγόριθμος ΧαρακτηρισμόςΑτόμου( βάρος↓ , ύψος↓ )
ΔΜΣ ← βάρος / ύψος ^ 2
Αν ΔΜΣ < 18.5 τότε
     Γράψε 'ελλιποβαρές άτομο'
Αλλιώς_αν ΔΜΣ < 25 τότε
     Γράψε 'άτομο με φυσιολογικό βάρος'
Αλλιώς_αν ΔΜΣ < 30 τότε
     Γράψε 'υπέρβαρο άτομο'
αλλιώς
     Γράψε 'άτομο που πάσχει από παχυσαρκία'
Τέλος_αν
Τέλος ΧαρακτηρισμόςΑτόμου
```

ΔΜΣ	Χαρακτηρισμός ατόμου
< 18,5	ελλιποβαρές
18,5 -24,9	φυσιολογικό βάρος
25 - 29,9	υπέρβαρο
> 30	πάσχει από παχυσαρκία

Αλγόριθμος 2-3. Χαρακτηρισμός ατόμου με βάση το ΔΜΣ με πολλαπλή επιλογή

Β.4. Εμφωλευμένες Δομές Επιλογής

Σε οποιαδήποτε από τις τρεις μορφές της δομής επιλογής, απλή, σύνθετη ή πολλαπλή, μπορούμε να βάλουμε στη θέση των εντολών μια άλλη δομή επιλογής. Έτσι δημιουργείται μια εμφωλευμένη δομή, δηλαδή μια δομή επιλογής μέσα σε μια άλλη. Τοποθετώντας μια εντολή Αν μέσα σε μια άλλη, παίρνουμε υποπεριπτώσεις. Η λογική αυτή μπορεί να επεκταθεί χωρίς περιορισμό, δηλαδή να έχουμε νέα εμφωλευμένη δομή μέσα σε μία ήδη εμφωλευμένη κ.ο.κ.

Παράδειγμα 2-7. Στο ταχυδρομείο, το κόστος αποστολής υπολογίζεται συναρτήσει του προορισμού και του βάρους της επιστολής, με βάση τον πίνακα. Δεδομένων του προορισμού και του βάρους μιας επιστολής, να εμφανίσετε το ποσό που στοιχίζει η αποστολή της.

		Βάρος επιστολής	
		Μέχρι και 100 γραμμάρια	Πάνω από 100 γραμμάρια
Προορισμός επιστολής	Εσωτερικό	1 €	2 €
	Εξωτερικό	2,50 €	4 €

Η θεώρηση των περιπτώσεων μπορεί να γίνει εναλλακτικά εξετάζοντας πρώτα το βάρος και μετά τον προορισμό εμφωλευμένα.

Δεν υπάρχει ένας μόνο συνδυασμός δομών επιλογής για την επίλυση του συγκεκριμένου προβλήματος. Αν εξετάσουμε πρώτα τις δύο περιπτώσεις για τον προορισμό με μια σύνθετη επιλογή και εμφωλευμένα πάρουμε υποπεριπτώσεις με σύνθετες επιλογές που εξετάζουν το βάρος, τότε προκύπτει εμφωλευμένη επιλογή που εξετάζει συνολικά 2 * 2 = 4 περιπτώσεις.

Αλγόριθμος 2-4. Υπολογισμός κόστους αποστολήςμε εμφωλευμένη επιλογή

```
Αλγόριθμος Επιστολή(προορισμός↓,βάρος↓,κόστος↑)
Αν προορισμός = 'Εσωτερικό' τότε
     Αν βάρος <= 100  τότε
          κόστος ← 1
     αλλιώς
          κόστος ← 2
     τέλος_αν
αλλιώς
     Αν βάρος <= 100  τότε
          κόστος ← 2.5
     αλλιώς
          κόστος ← 4
     τέλος_αν
τέλος_αν
Τέλος Επιστολή
```

Μια άλλη προσέγγιση είναι να εξετάσουμε τον προορισμό και το βάρος επιστολής μαζί, με μια σύνθετη λογική συνθήκη. Πρέπει προηγούμενα να έχουμε σκεφθεί όλες τις δυνατές περιπτώσεις. Ο αλγόριθμος που προκύπτει περιέχει πολλαπλή επιλογή με τέσσερις διακριτές περιπτώσεις.

Αλγόριθμος 2-5. Υπολογισμός κόστους αποστολήςμε πολλαπλή επιλογή

```
Αλγόριθμος Επιστολή2( προορισμός↓, βάρος↓, κόστος↑ )
Αν προορισμός = 'Εσωτερικό' και βάρος <= 100  τότε
     κόστος ← 1
αλλιώς_αν προορισμός = 'Εσωτερικό' και βάρος <= 100  τότε
     κόστος ← 2
αλλιώς_αν προορισμός = 'Εξωτερικό' και βάρος <= 100 τότε
     κόστος ← 2.5
αλλιώς
     κόστος ← 4
τέλος_αν
Τέλος Επιστολή2
```

Γενικά, στη δομή επιλογής, δεν υπάρχει ένας τρόπος λύσης. Μπορούμε να εξετάσουμε τις πιθανές περιπτώσεις με διαφορετικούς συνδυασμούς των τριών μορφών της εντολής Αν. Αυτό που είναι σημαντικό, είναι να επαληθεύσουμε ότι έχουμε λάβει υπόψη όλες τις πιθανές περιπτώσεις.

Γ. Δομή Επανάληψης

Η δομή επανάληψης στην καθημερινή ζωή της Κυριακής

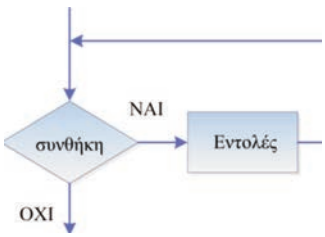
Σήμερα πήρα τον έλεγχο του Α' τετραμήνου. Γράφει τους βαθμούς μου σε κάθε ένα από τα 10 μαθήματα, αλλά δε γράφει το μέσο όρο μου. Πως θα τον υπολογίσω; Παίρνω το κομπιουτεράκι και αρχίζω. Πληκτρολογώ 15 και μετά +. Πληκτρολογώ 19 και μετά +. Πληκτρολογώ τον επόμενο βαθμό και μετά +. Στο τέλος πληκτρολογώ / και μετά 10. Ο μέσος όρος μου εμφανίζεται στην οθόνη. Επανέλαβα την ίδια δουλειά 10 φορές!

Πολλά προβλήματα απαιτούν την επανάληψη της ίδιας διαδικασίας. Η λογική των επαναληπτικών διαδικασιών εφαρμόζεται όταν μια σειρά από εντολές πρέπει να εφαρμοσθεί σε ένα σύνολο περιπτώσεων οι οποίες έχουν κάτι κοινό. Οι εντολές επανάληψης που υποστηρίζονται από την Ψευδογλώσσα είναι τρεις, η εντολή Όσο, η Μέχρις_ότου και η Για.

Γ.1. Εντολή Επανάληψης Όσο

Η εντολή Όσο χρησιμοποιείται για την επανάληψη της ίδιας διαδικασίας όσο ικανοποιείται μια συνθήκη.

Σύνταξη	**Όσο** συνθήκη **επανάλαβε** εντολές **τέλος_επανάληψης**
Λειτουργία	Αρχικά ελέγχεται αν ισχύει η συνθήκη. Αν ισχύει τότε εκτελούνται οι εντολές και ελέγχεται ξανά η συνθήκη. Η διαδικασία επαναλαμβάνεται Όσο η συνθήκη είναι αληθής. Όταν κάποτε ελεγχθεί η συνθήκη και είναι ψευδής, η επανάληψη τελειώνει και εκτελείται η εντολή που ακολουθεί τη λέξη τέλος_επανάληψης.



Εικόνα 2-24. Διάγραμμα ροής της επανάληψης Όσο

Παράδειγμα 2-8. Ο καθηγητής έβαλε τους βαθμούς του Α' τετραμήνου στην Πληροφορική και θέλει να βρει το μέσο όρο του τμήματος και πόσοι μαθητές είναι άριστοι (βαθμός πάνω από 18). Οι βαθμοί κυμαίνονται από 0 έως 20. Δεδομένου ότι το πλήθος των μαθητών είναι άγνωστο, ο καθηγητής θέλει να τελειώσει την εισαγωγή των βαθμών όταν δώσει την ειδική τιμή -1.

Αλγόριθμος 2-6. Μέσος όρος βαθμολογίας

```
Αλγόριθμος Πληροφορική
πλήθος_άριστων ← 0
πλήθος_μαθητών ←0
άθροισμα ← 0
Διάβασε βαθμός
Όσο βαθμός <> -1 επανάλαβε
     Αν βαθμός > 18 τότε
          πλήθος_άριστων ←πλήθος_άριστων + 1
     τέλος_αν
     άθροισμα ← άθροισμα + βαθμός
     πλήθος_μαθητών ← πλήθος_μαθητών + 1
     Διάβασε βαθμός
τέλος_επανάληψης
Αν πλήθος_μαθητών <> 0 τότε
     μέσος_όρος ← άθροισμα / πλήθος_μαθητών
     Γράψε μέσος_όρος, πλήθος_άριστων
αλλιώς
     Γράψε 'Δε δόθηκε ούτε ένας βαθμός'
τέλος_αν
Τέλος Πληροφορική
```

Χρησιμοποιούνται τρεις μεταβλητές που ενημερώνονται μέσα στην επανάληψη:
- πλήθος_άριστων: μετράει πόσοι είναι άριστοι (μετρητής),
- άθροισμα: αθροίζει τους βαθμούς (αθροιστής),
- πλήθος_μαθητών: μετράει πόσοι είναι όλοι οι μαθητές (μετρητής)

Η επανάληψη γίνεται για κάθε μαθητή. Το πλήθος των μαθητών δεν είναι γνωστό εκ των προτέρων. Μπορεί να είναι και μηδέν (0). Ενδείκνυται η χρήση της εντολής Όσο.

Εικόνα 2-25. Διάγραμμα ροής της επανάληψης μέχρις_ότου

Γ.2. Εντολή Επανάληψης Μέχρις_ότου

Η εντολή Μέχρις_ότου χρησιμοποιείται για την επανάληψη της ίδιας διαδικασίας μέχρι να ικανοποιηθεί μια συνθήκη.

Σύνταξη	**Αρχή_επανάληψης** εντολές **μέχρις_ότου** συνθήκη
Λειτουργία	Αρχικά, εκτελούνται οι εντολές. Μετά, ελέγχεται αν ισχύει η συνθήκη. Αν δεν ισχύει εκτελούνται ξανά οι εντολές. Η διαδικασία επαναλαμβάνεται μέχρι η συνθήκη να γίνει αληθής. Όταν ολοκληρωθεί η επανάληψη, εκτελείται η εντολή που ακολουθεί τη μέχρις_ότου.

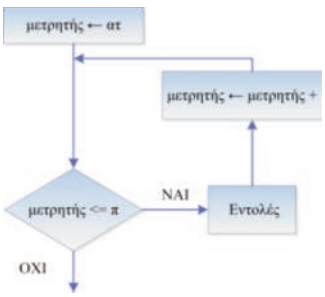
Παράδειγμα 2-9. Με το χαρτζιλίκι που μαζεύει στον κουμπαρά της, η Μάρθα θέλει να αγοράσει υπολογιστή. Η μητέρα της υποσχέθηκε την πρώτη εβδομάδα να της δώσει 25€ και κάθε εβδομάδα να της δίνει 5€ παραπάνω. Δεδομένης της τιμής του υπολογιστή, βρείτε σε πόσες εβδομάδες θα μαζέψει τα χρήματα από το χαρτζιλίκι της. Επίσης, εμφανίστε το ποσό που πιθανώς θα περισσέψει μετά την αγορά.

Στο παράδειγμα αυτό, η επανάληψη γίνεται για κάθε εβδομάδα. Το πλήθος των εβδομάδων δεν είναι γνωστό εκ των προτέρων αλλά είναι τουλάχιστον ένα (1). Ενδείκνυται η χρήση της εντολής Μέχρις_ότου

Αλγόριθμος 2-7. Συγκέντρωση ποσού για αγορά υπολογιστή

```
Αλγόριθμος Χαρτζιλίκι ( τιμήΗΥ↓ )
κουμπαράς ← 0
εβδομάδα ← 0
χαρτζιλίκι ← 25
Αρχή_επανάληψης
     κουμπαράς ← κουμπαράς + χαρτζιλίκι
     εβδομάδα ← εβδομάδα + 1
     χαρτζιλίκι ← χαρτζιλίκι + 5
μέχρις_ότου κουμπαράς >= τιμήΗΥ
Αν κουμπαράς > τιμήΗΥ τότε
     Γράψε 'Περίσσεψαν ', κουμπαράς - τιμήΗΥ
αλλιώς
     Γράψε 'Πάλι άδειος ο κουμπαράς!'
τέλος_αν
Τέλος Χαρτζιλίκι
```

Χρησιμοποιούνται τρεις μεταβλητές που ενημερώνονται μέσα στην επανάληψη:
- εβδομάδα: μετράει τις εβδομάδες (μετρητής)
- κουμπαράς: αθροίζει το χαρτζιλίκι κάθε βδομάδας (αθροιστής)
- χαρτζιλίκι: αυξάνεται κάθε βδομάδα κατά 5 €.



Εικόνα 2-26. Διάγραμμα ροής της επανάληψης Για

Γ.3. Εντολή Επανάληψης Για

Σύνταξη	**Για** μετρητής **από** ατ **μέχρι** ττ **[με_βήμα** βμ**]** εντολές **Τέλος_επανάληψης**
Λειτουργία	Με την εντολή Για επιτελείται η επαναληπτική εκτέλεση της ομάδας των εντολών για κάθε τιμή του μετρητή ξεκινώντας από την αρχική τιμή ατ μέχρι και την τελική τιμή ττ, μεταβαλλόμενος κάθε φορά κατά το βήμα βμ. Το βήμα μπορεί να παραληφθεί όταν ισούται με ένα. Ο αριθμός των επαναλήψεων της εντολής Για είναι προκαθορισμένος και γνωστός κατά την εκτέλεση του αλγόριθμου.

Χρησιμοποιείται όταν θέλουμε να εκτελεσθεί μια διαδικασία για προκαθορισμένο πλήθος επαναλήψεων.

Παράδειγμα 2-10. Το τμήμα Β1 έχει 25 μαθητές. Θέλουμε να καταχωρήσουμε τα ονόματα και τους βαθμούς των μαθητών στο μάθημα «Άλγεβρα», να υπολογίσουμε το μέσο όρο τους και να εμφανίσουμε το όνομα του μαθητή που πήρε τον καλύτερο βαθμό. Μόνο ένας από τους μαθητές έχει πάρει τον καλύτερο βαθμό του τμήματος και δεν είναι σίγουρο ότι ο βαθμός είναι το 20.

Αλγόριθμος 2-8. Μέσος όρος και μέγιστος βαθμός

```
Αλγόριθμος Άλγεβρα (μο↑, καλύτερος↑)
Σ ← 0
Για μαθητής από 1 μέχρι 25
    Διάβασε βαθμός, όνομα
    Αν μαθητής = 1 τότε μέγιστος ← βαθμός
    Σ ← Σ + βαθμός
    Αν βαθμός > μέγιστος τότε
          μέγιστος ←  βαθμός
          καλύτερος ←  όνομα
    τέλος_αν
τέλος_επανάληψης
μο ←  Σ / 25
Γράψε μο, καλύτερος
Τέλος Άλγεβρα
```

Η εύρεση του μέγιστου γίνεται με σύγκριση του βαθμού κάθε επόμενου μαθητή με τον μεγαλύτερο βαθμό από όλους τους προηγούμενους (μέγιστος). Η μεταβλητή καλύτερος ενημερώνεται με το αντίστοιχο όνομα.

Η επανάληψη γίνεται για κάθε μαθητή. Οι μαθητές είναι 25, άρα το πλήθος των επαναλήψεων είναι γνωστό και ενδείκνυται η χρήση της εντολής Για.

Στην Εικόνα 2-27 συνοψίζονται οι τρεις εντολές της δομής επανάληψης.

	Όσο	Μέχρις_ότου	Για
Εκτέλεση εντολών	όσο συνθήκη αληθής (μέχρι να γίνει ψευδής)	όσο συνθήκη ψευδής (μέχρι να γίνει αληθής)	όσο μετρητής <= τελική_τιμή, με βήμα > 0 όσο μετρητής >= τελική_τιμή, με βήμα < 0
Αριθμός επαναλήψεων	άγνωστος εκ των προτέρων	άγνωστος εκ των προτέρων	προκαθορισμένος
Ελάχιστος αριθμός επαναλήψεων	0	1	0

Εικόνα 2-27. Σύγκριση των δομών επανάληψης

Εμφωλευμένες δομές επανάληψης

Οποιαδήποτε δομή επανάληψης μπορεί να εμφωλευτεί σε μια άλλη. Η ιδέα είναι απλή. Θέλουμε να επαναλάβουμε μια διαδικασία. Αν για παράδειγμα στον αλγόριθμο "Άλγεβρα, θέλουμε να κάνουμε την ίδια εργασία όχι μόνο για το τμήμα Β1 αλλά και για τα πέντε τμήματα της τάξης Β τότε η επανάληψη Για μαθητής από 1 μέχρι 25 θα εμφωλευτεί σε μια νέα επανάληψη Για τμήμα από 1 μέχρι 5.

Παράδειγμα 2-11. Να εκτυπωθεί η προπαίδεια του 1 και να επεκταθεί για όλους τους αριθμούς από 1 μέχρι 10.

Αλγόριθμος 2-9. Εκτύπωση της προπαίδειας (α) του αριθμού 1, (β) των αριθμών 1 ως 10

```
Αλγόριθμος ΠροπαίδειαΤου1
Για α από 1 μέχρι 10
      Γράψε α,'*',1,'=',α*1
Τέλος_επανάληψης
Τέλος ΠροπαίδειαΤου1
```

(α)

```
Αλγόριθμος Προπαίδεια
Για β από 1 μέχρι 10
      Για α από 1 μέχρι 10
          Γράψε α,'*',β,'=',α*β
      Τέλος_επανάληψης
Τέλος_επανάληψης
Τέλος Προπαίδεια
```

(β)

Ας κλείσουμε την δομή επανάληψης, με τον αλγόριθμο που υλοποιεί την προσεγγιστική μέθοδο του Newton για τον υπολογισμό της τετραγωνικής ρίζας που αναφέρθηκε στην αρχή του κεφαλαίου.

Παράδειγμα 2-12. Να υπολογιστεί η τετραγωνική ρίζα πραγματικού αριθμού με δεδομένη ακρίβεια.

Όταν οι τιμές της ρίζας του x που υπολογίζονται σε δύο διαδοχικές επαναλήψεις διαφέρουν λιγότερο από την επιθυμητή ακρίβεια τερματίζει ο βρόχος

ΑπΤιμή1 είναι ο αλγόριθμος του Παραδείγματος 2.4

Αλγόριθμος 2-10. Προσεγγιστικός υπολογισμός της τετραγωνικής ρίζας αριθμού

```
Αλγόριθμος ΤετραγωνικήΡίζα (x↓, ακρίβεια↓, τρ↑)
νέα_προσέγγιση ← x /10
Αρχή_επανάληψης
     τρ ← νέα_προσέγγιση
     νέα_προσέγγιση ← (τρ + x / τρ) / 2
     ΑπΤιμή1(νέα_προσέγγιση - τρ , διαφορά)
μέχρις_ότου διαφορά < ακρίβεια
τρ ← νέα_προσέγγιση
Τέλος ΤετραγωνικήΡίζα
```

Αναδρομή

Η αναδρομή δίνει τη δυνατότητα να διατυπώσουμε αλγόριθμους που περιέχουν βρόχους, εναλλακτικά χωρίς βρόχους. Αποδεικνύεται ότι κάθε μη αναδρομικός αλγόριθμος μπορεί να μετασχηματισθεί σε ισοδύναμο αναδρομικό και κάθε αναδρομικός σε ισοδύναμο μη αναδρομικό. Στο παρακάτω παράδειγμα, βλέπουμε στην πράξη, την επίλυση ενός προβλήματος με δομή επανάληψης και με αναδρομή.

Παράδειγμα 2-13. Να υπολογιστεί το παραγοντικό θετικού ακέραιου αριθμού ν, το οποίο ορίζεται ως ν! = 1*2*…*ν.

Αλγόριθμος 2-11. Υπολογισμός του παραγοντικού, (α) με αναδρομή, (β) χωρίς αναδρομή

```
Αλγόριθμος ΠαραγΑ(ν↓, Π↑)
Αν ν=1 τότε
     Π ← 1
Αλλιώς
     ΠαραγΑ(ν-1, Π)
     Π ← Π * ν
Τέλος_αν
Τέλος ΠαραγΑ
```

```
Αλγόριθμος ΠαραγΕ(ν↓, Π↑)
Π ← 1
Για i από 1 μέχρι ν
     Π ← Π * i
Τέλος_επανάληψης
Τέλος ΠαραγΕ
```

Κατά μια έννοια, ο αναδρομικός αλγόριθμος είναι επαναληπτική διαδικασία αφού ξανακαλεί τον εαυτό του μέχρι να γίνει ν = 1. Ωστόσο, δεν περιέχει δομή επανάληψης!

Ερωτήσεις - θέματα για συζήτηση

1. Επεκτείνετε τον αλγόριθμο Μέγιστος_β (Παράδειγμα 2-3) ώστε να βρίσκει το μεγαλύτερο από τρεις αριθμούς. Γενικεύστε για οποιοδήποτε πλήθος αριθμών.
2. Μπορεί μια δομή πολλαπλής επιλογής να μετατραπεί σε ισοδύναμη με τη χρήση ανεξάρτητων δομών απλής επιλογής;

Ασκήσεις - Προβλήματα

1. Ποια από τις τρεις δομές αλγορίθμου ή ποιο συνδυασμό αυτών θα χρησιμοποιούσατε για τα παρακάτω προβλήματα: α) υπολογισμός γενικού μέσου όρου μαθητή σε δέκα μαθήματα β) προσαρμογή προφορικού βαθμού ώστε να μη διαφέρει από τον γραπτό πάνω από τρεις μονάδες γ) υπολογισμός ποσοστού άριστων μαθητών για 10 τμήματα με 20 μαθητές το καθένα. δ) υπολογισμός πηλίκου δύο αριθμών ε) εύρεση του ψηλότερου από 10 παίκτες μιας ομάδας μπάσκετ.
2. Γράψτε αλγόριθμο ο οποίος δέχεται τα ύψη τριών μαθητών σε εκατοστά και τυπώνει το μέσο ύψος τους σε μέτρα.
3. Κατασκευάστε το διάγραμμα ροής του αλγορίθμου ΧαρακτηρισμόςΑτόμου (Παράδειγμα 2-4). Ξαναγράψτε τον αλγόριθμο σε Ψευδογλώσσα χρησιμοποιώντας εμφωλευμένες δομές σύνθετης επιλογής. Κατασκευάστε το διάγραμμα ροής του νέου αλγορίθμου.
4. Δημιουργήστε αλγόριθμους για την επίλυση της πρωτοβάθμιας και δευτεροβάθμιας εξίσωσης. Εκφράστε τους με διάγραμμα ροής και Ψευδογλώσσα.
5. Δεδομένων των καρτεσιανών συντεταγμένων χ, ψ ενός σημείου που δεν βρίσκεται πάνω στους άξονες, να εμφανίσετε το τεταρτημόριο στο οποίο ανήκει. Γράψτε τον αλγόριθμο τόσο με εμφωλευμένες σύνθετες επιλογές όσο και με πολλαπλή επιλογή.
6. Ηλεκτρική εταιρία χρεώνει την κατανάλωση ρεύματος σύμφωνα με την παρακάτω κλίμακα:

- Τις πρώτες 200 μονάδες (0-200) προς 0,25 €./ μονάδα
- Τις επόμενες 1000 μονάδες (201-1200) προς 0,40 €./ μονάδα
- Τις πέρα των 1200 μονάδων προς 0,50 €./ μονάδα

Να γίνει αλγόριθμος που δέχεται τον αριθμό των μονάδων που καταναλώθηκαν από έναν πελάτη και εμφανίζει το ποσό των χρημάτων που χρωστάει ο πελάτης στην ηλεκτρική εταιρία.

7. Γράψτε αλγόριθμο σε Ψευδογλώσσα που δέχεται 100 αριθμούς από τον χρήστη και τους εμφανίζει α) με τη σειρά που δόθηκαν (επαναληπτικά) β) με σειρά αντίθετη από αυτή που δόθηκαν (αναδρομικά).
8. Γράψτε αλγόριθμο ο οποίος διαβάζει τον βαθμό ενός μαθητή (από 0 έως 20). Αν ο βαθμός είναι εκτός των επιτρεπτών ορίων τότε ζητείται νέος αριθμός. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να δοθεί σωστός αριθμός. Υλοποιήστε τον αλγόριθμο και με εντολή Όσο και με Μέχρις_ότου.

9. Γράψτε αλγόριθμο ο οποίος δέχεται αριθμούς και τυπώνει το μέσο όρο των αριθμών που δόθηκαν. Η εισαγωγή αριθμών σταματά όταν δοθεί το 0 ή αρνητικός αριθμός.
10. Γράψτε αλγόριθμο ο οποίος δέχεται αριθμούς και τυπώνει το μέσο όρο των αριθμών που δόθηκαν. Η εισαγωγή αριθμών σταματά όταν το άθροισμά τους ξεπεράσει το 1000.
11. Γράψτε αλγόριθμο ο οποίος δέχεται 100 αριθμούς και τυπώνει το μέσο όρο των αριθμών που δόθηκαν.
12. Να αναπτύξετε αλγόριθμο που δέχεται το πολύ 100 αριθμούς από τους οποίους το πολύ 20 θα είναι αρνητικοί, και τυπώνει το ποσοστό των θετικών και των αρνητικών αριθμών που εισήχθησαν. (το μηδέν δεν είναι ούτε θετικός ούτε αρνητικός)
13. Ασανσέρ έχει όριο βάρους 350 κιλά. Γράψτε αλγόριθμο ο οποίος δέχεται επαναληπτικά την εισαγωγή του βάρους κάθε ατόμου που θέλει να μπει στο ασανσέρ και σταματά όταν παραβιάζεται το όριο βάρους. Στο τέλος, τυπώνει το πλήθος και το συνολικό βάρος των ατόμων που μπήκαν στο ασανσέρ.
14. Υλοποιήσετε τον αλγόριθμο του Ευκλείδη για τον υπολογισμό του μέγιστου κοινού διαιρέτη α) με δομή επανάληψης β) με αναδρομή.

Όταν ολοκληρώσετε την ενότητα ελέγξτε αν είστε σε θέση:

- να περιγράφετε τα βασικά συνθετικά μέρη ενός αλγόριθμου (σταθερές, μεταβλητές, τελεστές, εκφράσεις).
- να διατυπώνετε με ακρίβεια τη σύνταξη και τη λειτουργία του συνόλου των εντολών (βασικών ή δομών) της ψευδογλώσσας.
- να αναγνωρίζετε από την εκφώνηση του προβλήματος, ποιες αλγοριθμικές δομές και με ποιον συνδυασμό, απαιτούνται για την επίλυση ενός δοσμένου προβλήματος.
- να συνδυάζετε, σε σειρά ή εμφωλευμένα, διάφορες αλγοριθμικές δομές για την επίλυση υπολογιστικών προβλημάτων.

2.2.8 Βασικές αλγοριθμικές λειτουργίες σε δομές δεδομένων

Στην προηγούμενη ενότητα είδαμε τις βασικές εντολές και αλγοριθμικές δομές, ενώ στην ενότητα 2.2.6 μάθαμε τι είναι οι δομές δεδομένων και ποιες λειτουργίες υποστηρίζονται σε αυτές. Σε αυτήν την ενότητα εφαρμόζουμε τις αλγοριθμικές δομές για να υλοποιήσουμε τη δομή δεδομένων του πίνακα.

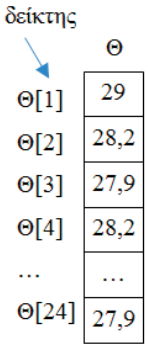
Πίνακες

Ο πίνακας, όπως ήδη αναφέρθηκε στην ενότητα 2.2.6, αποτελεί μια δομή δεδομένων, δηλαδή έναν τρόπο οργάνωσης των δεδομένων.

Ο **πίνακας** είναι μία διάταξη στοιχείων του ίδιου τύπου με συγκεκριμένο μέγεθος, στην οποία τα στοιχεία αναφέρονται με το ίδιο όνομα και με έναν ή περισσότερους δείκτες.

Οι πίνακες διατάσσουν τα στοιχεία τους σε γραμμές και στήλες. Ένας πίνακας Ν στοιχείων **μίας διάστασης** ή **μονοδιάστατος** αποτελείται από μία γραμμή (ή ισοδύναμα από μία στήλη) Ν στοιχείων. Τα στοιχεία αυτά βρίσκονται στις θέσεις 1 ως Ν, όπου Ν είναι το πλήθος των στοιχείων του πίνακα. Η θέση κάθε στοιχείου ονομάζεται **δείκτης** και είναι ένας ακέραιος με τιμές από 1 ως Ν. Όλα τα στοιχεία του πίνακα έχουν το ίδιο όνομα, αλλά καθένα έχει διαφορετικό δείκτη.

Έστω ότι θέλουμε να αποθηκεύσουμε τη θερμοκρασία μιας πόλης που καταγράφεται κάθε μία ώρα μέσα σε ένα 24ωρο, δηλαδή θέλουμε να αποθηκεύσουμε συνολικά 24 θερμοκρασίες. Μετά την καταγραφή και αποθήκευση των δεδομένων αυτών θέλουμε να τα επεξεργαστούμε ως εξής: να υπολογίσουμε τη μέση θερμοκρασία του 24ωρου, τη μέγιστη και την ελάχιστη. Εφόσον τα στοιχεία είναι πολλά, είναι ομοειδή και θέλουμε να τα αποθηκεύσουμε και να τα επεξεργαστούμε μαζί, είναι βολικό να χρησιμοποιήσουμε έναν πίνακα εικοσιτεσσάρων (24) πραγματικών αριθμών στον οποίον τα στοιχεία θα είναι αποθηκευμένα σε συνεχόμενες θέσεις. Όπως φαίνεται στην Εικόνα 2-28, ο πίνακας ονομάζεται Θ. Το πρώτο στοιχείο του πίνακα ονομάζεται Θ[1] ("Θ του 1", ή "Θ με δείκτη 1") και σε αυτό αποθηκεύουμε την τιμή για την πρώτη θερμοκρασία που καταγράφουμε, δηλαδή την τιμή 29. Οι δείκτες των στοιχείων του πίνακα είναι ακέραιοι αριθμοί με τιμές από 1 ως 24.Το τελευταίο στοιχείο του πίνακα ονομάζεται Θ[24] και συμφωνούμε ότι αντιστοιχεί στην τελευταία θερμοκρασία που θα καταγραφεί. Τα στοιχεία στις θέσεις 2 ως 23 συμφωνούμε ότι αντιστοιχούν στις ενδιάμεσες διαδοχικές καταγραφές θερμοκρασίας. Ο υπολογισμός της μέσης θερμοκρασία του 24ωρου γίνεται αθροίζοντας τα 24 στοιχεία του πίνακα και διαιρώντας το άθροισμα με το πλήθος τους. Η αποθήκευσή τους σε διαδοχικές θέσεις πίνακα, θα δούμε παρακάτω ότι διευκολύνει σημαντικά την υλοποίηση του αλγορίθμου.



Εικόνα 2-28. Μονοδιάστατος πίνακας Θ, 24 θερμοκρασιών

Η **διάσταση** του πίνακα καθορίζει το πλήθος των δεικτών που χρησιμοποιούνται για κάθε στοιχείο του. Ένας πίνακας **δύο διαστάσεων** ή **δισδιάστατος**, αποτελείται γενικά από Μ γραμμές και Ν στήλες, άρα από Μ * Ν στοιχεία. Κάθε στοιχείο αναφέρεται με το όνομα του πίνακα και δύο δείκτες: τον αριθμό της γραμμής και τον αριθμό της στήλης. Σε έναν πίνακα δύο διαστάσεων με όνομα Π, τα στοιχεία του αναφέρονται ως Π[i, j] όπου i ο δείκτης της γραμμής και j ο δείκτης της στήλης.

- Π[i, j]: το στοιχείο της γραμμής i και στήλης j, $1 \le i \le M$, $1 \le j \le N$.
- Π[1, j]: το στοιχείο της γραμμής 1 και στήλης j, $1 \le j \le N$.
- Π[i, 1]: το στοιχείο της γραμμής i και στήλης 1, $1 \le j \le N$.

Ας επεκτείνουμε το παράδειγμα του πίνακα θερμοκρασιών καταγράφοντας τις θερμοκρασίες της πόλης ανά ώρα του 24ωρου, για μία ολόκληρη εβδομάδα. Χρειαζόμαστε 7 * 24 = 168 στοιχεία για τα επτά 24ωρα. Μπορούμε επομένως να δημιουργήσουμε έναν πίνακα 168 στοιχείων. Μας βολεύει να διατηρήσουμε τη διάταξή τους ανά ημέρες και ώρες και γι' αυτό θα δημιουργήσουμε ένα πίνακα 168 στοιχείων δύο διαστάσεων, με όνομα Θ7, ο οποίος θα αποτελείται από 7 γραμμές και 24 στήλες. Επομένως για τον πίνακα αυτόν ισχύουν τα εξής:

- Κάθε μία από τις 7 γραμμές του πίνακα αποτελείται από εικοσιτέσσερα (24) στοιχεία.
- Κάθε μία από τις 24 στήλες του πίνακα αποτελείται από 7 στοιχεία.

• Τα στοιχεία του πίνακα ονομάζονται Θ7[κ, λ] όπου κ είναι ο δείκτης της γραμμής και λ ο δείκτης της στήλης στην οποία ανήκει το στοιχείο.Το κ μπορεί να πάρει τις ακέραιες τιμές 1 ως 7, ενώ το λ μπορεί να πάρει τις ακέραιες τιμές από 1 ως 24.

Επιπλέον, για να αποθηκεύσουμε τα δεδομένα του συγκεκριμένου προβλήματος θερμοκρασιών στον πίνακα Θ7, κάνουμε τις εξής συμβάσεις:

• Οι επτά (7) γραμμές αντιστοιχούν στα επτά (7) εικοσιτετράωρα, δηλαδή η 1η γραμμή αντιστοιχεί στο πρώτο 24ωρο της εβδομάδας, κ.ο.κ.

• Οι στήλες αντιστοιχούν στις ώρες των 24ωρων. Η 1η στήλη αντιστοιχεί στις θερμοκρασίες της 1ης ώρας όλων των 24ωρων.

• Το στοιχείο Θ7[1,1] που βρίσκεται στην 1η γραμμή και 1η στήλη του πίνακα αντιστοιχεί στην πρώτη θερμοκρασία που καταγράφουμε για το πρώτο 24ωρο.

• Το στοιχείο Θ7[κ, λ] για οποιοδήποτε κ με τιμές από 1 ως 7 και οποιοδήποτε λ με τιμές από 1 ως 24, αντιστοιχεί στη θερμοκρασία που καταγράφηκε για το κ 24ωρο την λ ώρα.

Ο πίνακας Θ7 μαζί με τα δεδομένα που περιέχει φαίνεται στην Εικόνα 2-29.

Θ7	Θ7[1,1]					Θ7[1, 24]
1η ημέρα	29	28,2	27,9	28,2	…	27,9
	28	27,8	28,4	28,6	…	26,1
	…	…	…	…	…	…
	30	30,3	31	28,6	…	25
7η ημέρα	30	30,3	31	28,6	…	25

Εικόνα 2-29. Πίνακας θερμοκρασιών 24τετραώρου για 7 ημέρες

Βασικές λειτουργίες σε μονοδιάστατους πίνακες

Οι συνηθέστερες λειτουργίες σε πίνακες μίας διάστασης είναι το διάβασμα και η εμφάνιση των στοιχείων τους, ο υπολογισμός του αθροίσματος και του μέσου όρου, η εύρεση του μέγιστου και του ελάχιστου, η αναζήτηση και η ταξινόμηση.

Διάβασμα και εμφάνιση

Γενικά, για να διαβάσουμε όλα τα στοιχεία ενός πίνακα μίας διάστασης με σειρά από το πρώτο μέχρι το τελευταίο, χρησιμοποιούμε επανάληψη η οποία μετράει από 1 μέχρι το πλήθος των στοιχείων του πίνακα. Ο μετρητής χρησιμοποιείται ως δείκτης των στοιχείων του πίνακα και μεταβάλλεται ώστε να λάβει την τιμή όλων των δεικτών.

Αλγόριθμος 2-12. Διάβασμα στοιχείων πίνακα μίας διάστασης

```
Αλγόριθμος ΔιάβασμαΠίνακα(Α↑)
Για i από 1 μέχρι N
      Διάβασε A[i]
Τέλος_επανάληψης
Τέλος  ΔιάβασμαΠίνακα
```

Άθροισμα και μέσος όρος

Για τον υπολογισμό του αθροίσματος των στοιχείων ενός πίνακα χρη-

σιμοποιούμε αθροιστή (π.χ. Σ), τον οποίο αρχικοποιούμε με την τιμή 0. Κατόπιν γράφουμε επανάληψη η οποία προσθέτει ένα ένα τα στοιχεία στον αθροιστή. Ο υπολογισμός του μέσου όρου (μέσης τιμής), μπορεί να γίνει μετά την άθροιση διαιρώντας το άθροισμα με το πλήθος των στοιχείων.

Αλγόριθμος 2-13. Άθροισμα και μέσος όρος στοιχείων πίνακα μίας διάστασης

```
Αλγόριθμος ΆθροισμαΜΟΠίνακα( Α↓ , Σ↑ , ΜΟ↑ )
Σ ← 0
Για i από 1 μέχρι N
      Σ ← Σ + A[i]
Τέλος_επανάληψης
ΜΟ ← Σ / N
Τέλος ΆθροισμαΜΟΠίνακα
```

Ελάχιστο και μέγιστο

Για τον υπολογισμό του ελάχιστου στοιχείου χρησιμοποιούμε μια μεταβλητή στην οποία θα αποθηκεύσουμε το αποτέλεσμα, π.χ. με όνομα min (minimum). Θεωρούμε ότι αρχικά το ελάχιστο στοιχείο του πίνακα είναι το πρώτο του. Κατόπιν γράφουμε μια δομή επανάληψης. Οποτεδήποτε σε κάποια θέση του πίνακα βρεθεί ένα στοιχείο που είναι μικρότερο από το μέχρι τώρα ελάχιστο, το ελάχιστο αντικαθίσταται με αυτό το μικρότερο στοιχείο. Η διαδικασία τερματίζει, όταν ελεγχθούν όλα τα στοιχεία του πίνακα. Αντίστοιχα γίνεται ο υπολογισμός για το μέγιστο.

Αλγόριθμος 2 14. Υπολογισμός ελάχιστου

```
Αλγόριθμος ΕλάχιστοΠίνακα( Α↓ , min↑ )
min ← A[1]
Για i από 2 μέχρι N
      Αν min < A[i] τότε min ← A[i]
Τέλος_επανάληψης
Τέλος ΕλάχιστοΠίνακα
```

Ταξινόμηση

Η ταξινόμηση συνίσταται στην αλλαγή της θέσης των στοιχείων του πίνακα ώστε να ακολουθούν μια συγκεκριμένη διάταξη, π.χ. αύξουσα ή φθίνουσα σειρά.

Υπάρχουν πολλοί αλγόριθμοι ταξινόμησης, όπως η **ταξινόμηση με εισαγωγή** (insertion sort), η **ταξινόμηση με επιλογή** (selection sort), η **ταξινόμηση ευθείας ανταλλαγής** ή **φυσαλίδα** (bubble sort), η **γρήγορη ταξινόμηση** (quicksort), κ.ά. Οι αλγόριθμοι αυτοί διαφέρουν στον τρόπο εξέτασης των στοιχείων και επομένως στην απόδοσή τους. Άλλοι είναι πιο γρήγοροι κι άλλοι πιο αργοί.

Ο αλγόριθμος της ταξινόμησης με εισαγωγή κατά αύξουσα σειρά λειτουργεί ως εξής: βρίσκει τη θέση που πρέπει να τοποθετηθεί (εισαχθεί) κάθε στοιχείο του πίνακα σε σχέση με τα προηγούμενά του και το εισάγει σ' αυτή τη θέση.

Η ταξινόμηση ξεκινάει από το δεύτερο στοιχείο του πίνακα (τρέχον στοι-

χείο) το οποίο πρέπει να τοποθετηθεί στη σωστή θέση ως προς το προηγούμενό του. Γίνεται η σύγκριση των δύο στοιχείων (2ο με 1ο) και αν το τρέχον (2ο) είναι μικρότερο από το πρώτο, ανταλλάσσουν θέσεις μεταξύ τους. Έτσι τα δύο πρώτα στοιχεία ταξινομούνται μεταξύ τους. Ο αλγόριθμος κατόπιν συνεχίζει επαναληπτικά με τα επόμενα στοιχεία του πίνακα. Κάθε στοιχείο που εξετάζεται (τρέχον στοιχείο), συγκρίνεται διαδοχικά με τα προηγούμενά του και αν το τρέχον στοιχείο είναι μικρότερο από κάποιο προηγούμενό του, ανταλλάσσεται με αυτό που συγκρίθηκε. Η ταξινόμηση του πίνακα με τον αλγόριθμο αυτόν επιτυγχάνεται σταδιακά.Ο αλγόριθμος της ταξινόμησης με εισαγωγή μπορεί να γραφτεί σε ψευδογλώσσα ως εξής:

Αλγόριθμος 2-15. Ταξινόμηση με εισαγωγή

```
Αλγόριθμος ΤαξινόμησηΜεΕισαγωγή (Α↓↑)
Για  i  από 2 μέχρι N
       j ← i
       όσο A[j]< A[j-1] και  j > 1 επανάλαβε
              t  ← A[j]
              A[j] ← A[j-1]
              A[j-1] ← t
              j← j-1
       τέλος_επανάληψης
τέλος_επανάληψης
Τέλος ΤαξινόμησηΜεΕισαγωγή
```

Αναζήτηση

Η αναζήτηση είναι μια λειτουργία που εκτελείται συχνά σε πίνακες. Το πρόβλημα της αναζήτησης είναι η εύρεση μιας δεδομένης τιμής σε έναν πίνακα. Υπάρχουν δύο βασικοί αλγόριθμοι αναζήτησης σε πίνακα, η σειριακή και η δυαδική.

Σειριακή αναζήτηση

Ο αλγόριθμος της σειριακής αναζήτησης σε μονοδιάστατο πίνακα, ελέγχει με τη σειρά ένα προς ένα τα στοιχεία του πίνακα μέχρι να βρεθεί το στοιχείο που αναζητούμε ή να εξαντληθεί ο πίνακας.

Αλγόριθμος 2-16. Σειριακή αναζήτηση

```
Αλγόριθμος ΣειριακήΑναζήτηση(Α↓, key↓, θέση↑)
θέση ← -1
i ← 1
Όσο i <= N και θέση = -1 επανάλαβε
       Αν A[i] = key  τότε θέση ← i
       i ← i  + 1
Τέλος_επανάληψης
Αν θέση <> -1 τότε
       Γράψε 'Βρέθηκε στη θέση', θέση
αλλιώς
       Γράψε 'Δεν υπάρχει στον πίνακα'
Τέλος_αν
Τέλος ΣειριακήΑναζήτηση
```

Η μεταβλητή «θέση» χρησιμοποιείται για να αποθηκευτεί η θέση στην οποία βρέθηκε το στοιχείο. Αν όμως η τιμή της παραμείνει -1 ως το τέλος, τότε το στοιχείο δεν υπάρχει στον πίνακα.

Δυαδική αναζήτηση

Η δυαδική αναζήτηση είναι ένας ταχύτερος αλγόριθμος αναζήτησης σε ταξινομημένο πίνακα. Διαιρεί τον πίνακα σε δύο μέρη, επιλέγοντας το μεσαίο στοιχείο κάθε φορά και συγκρίνοντάς το με το στοιχείο που ψάχνουμε.

Η λογική της δυαδικής αναζήτησης μας είναι γνωστή από την αναζήτηση στον έντυπο τηλεφωνικό κατάλογο. Ο τηλεφωνικός κατάλογος περιέχει τα τηλέφωνα ταξινομημένα αλφαβητικά ως προς το επώνυμο. Έστω ότι ψάχνουμε να βρούμε το τηλέφωνο της κυρίας Τακτικάκη. Αρχικά ανοίγουμε τον κατάλογο σε κάποια τυχαία σελίδα, περίπου στη μέση του. Βλέπουμε ποιο γράμμα υπάρχει σε κείνη τη σελίδα. Αν ανοίξαμε μια σελίδα με επίθετα από 'Κ' και εμείς ψάχνουμε επίθετο από 'Τ' , πρέπει να κοιτάξουμε παρακάτω. Ανοίγουμε ξανά περίπου στη μέση, από τη σελίδα του 'Κ' που βρισκόμαστε μέχρι το τέλος του καταλόγου. Ας υποθέσουμε ότι ανοίγουμε σε μια σελίδα που περιέχει επώνυμα που αρχίζουν με το γράμμα 'Ρ'. Πρέπει να ξανανοίξουμε λίγο παρακάτω. Ξανανοίγουμε στο μισό του υπολοίπου και βλέπουμε το γράμμα 'Φ'. Άρα πρέπει ν' ανοίξουμε προς τα πίσω αυτή τη φορά. Τελικά ανοίγουμε στο 'Τ', κάνουμε ένα μικρό ξεφύλλισμα προς το 'Τα' και βρίσκουμε το τηλέφωνο που αναζητούμε.

Αλγόριθμος 2-17. Δυαδική αναζήτηση

```
Αλγόριθμος ΔυαδικήΑναζήτηση (Α↓, key↓, θέση↑)
θέση ← 0
start  ← 1
end ← N
όσο end >= start  και θέση =0 επανάλαβε
     μέση  ←  start + ((end - start) div 2)
       Αν A[μέση] = key τότε
             θέση  ← μέση
       αλλιώς_αν A[μέση] < key τότε
         start ←  μέση + 1
       αλλιώς
         end ←  μέση - 1
       τέλος_αν
τέλος_επανάληψης
Τέλος ΔυαδικήΑναζήτηση
```

Βασικές λειτουργίες σε πίνακες δύο ή περισσότερων διαστάσεων

Η επεξεργασία των στοιχείων πινάκων δύο διαστάσεων γίνεται συνήθως με εμφωλευμένες επαναλήψεις οι οποίες διασχίζουν τα στοιχεία του πίνακα γραμμή προς γραμμή ή στήλη προς στήλη. Η λειτουργία αναζήτησης στοιχείων σε πίνακα δύο διαστάσεων μπορεί επίσης να εφαρμοστεί σε μία γραμμή, σε μία στήλη ή σε ολόκληρο τον πίνακα. Εκτός από μονοδιάστατους και δισδιάστατους μπορούμε να δημιουργήσουμε και πολυδιάστατους πίνακες, στους οποίους κάθε στοιχείο διακρίνεται από περισσότερους από δύο δείκτες, εφόσον αυτό είναι αναγκαίο για το πρόβλημα.

Στον αλγόριθμο 2-18 γίνεται εφαρμογή των λειτουργιών πινάκων μίας και δύο διαστάσεων σε ένα πραγματικό πρόβλημα με θερμοκρασίες.

Αλγόριθμος 2-18. Πρόβλημα «Θερμοκρασίες»

```
Αλγόριθμος Θερμοκρασίες
Για  i  από 1 μέχρι 7
       Για  j  από 1 μέχρι 24
              Διάβασε  Θ7[i, j]
       τέλος_επανάληψης
τέλος_επανάληψης
Για  i  από 1 μέχρι 7
       Η[i]← 0
τέλος_επανάληψης
Για  j  από 1 μέχρι 24
       Ω[j]← 0
τέλος_επανάληψης
Για  i  από 1 μέχρι 7
       Για  j  από 1 μέχρι 24
              Η[i]← Η[i] + Θ7[i, j]
              Ω[j]← Ω[j] + Θ7[i, j]
       τέλος_επανάληψης
τέλος_επανάληψης
Για  i  από 1 μέχρι 7
       Η[i]← Η[i] / 24
       Γράψε 'Ημέρα ', i, ', μέση θερμοκρασία: ',  Η[i]
τέλος_επανάληψης
Για  j  από 1 μέχρι 24
       Ω[j]← Ω[j] /7
       Γράψε 'Ώρα ', j-1, ':00,  μέση θερμοκρασία: ', Ω[j]
τέλος_επανάληψης
Τέλος Θερμοκρασίες
```

Ερωτήσεις - θέματα για συζήτηση

1.Ανατρέξτε στο πρόβλημα με τις θερμοκρασίες επτά ημερών - εικοσιτεσσάρων ωρών. Ο πίνακας Θ7 αποτελείται από 7 γραμμές και 24 στήλες. Θα μπορούσαν τα συγκεκριμένα δεδομένα να αποθηκευτούν σε πίνακα με 24 γραμμές και 7 στήλες; Περιγράψτε τις συμβάσεις που θα καθορίζατε για τον πίνακα αυτόν.

2. Στο πρόβλημα των θερμοκρασιών επτά ημερών, τι ταξινομήσεις θα είχε νόημα να κάνουμε στον πίνακα με τις θερμοκρασίες; Τι θα γινόταν αν ταξινομούσαμε τον δισδιάστατο πίνακα Θ7 γραμμή προς γραμμή;

3. Ποια είναι η απόδοση των αλγορίθμων α) ταξινόμησης με εισαγωγή, β) σειριακής αναζήτηση και γ) δυαδικής αναζήτησης; Μπορείτε να τις εκφράσετε με το συμβολισμό Ο( );

Ασκήσεις - Προβλήματα

1. Να διαβαστούν τα στοιχεία μονοδιάστατου πίνακα: α) από το τελευταίο προς το πρώτο, β) πρώτα στις μονές και κατόπιν στις ζυγές θέσεις, γ) από τα άκρα προς το μέσο του πίνακα (πρώτο, τελευταίο,

δεύτερο, προτελευταίο, …).

2. Πρώτος είναι ένας αριθμός που διαιρείται μόνο με τον εαυτό του και τη μονάδα. Να γραφτεί αλγόριθμος που υπολογίζει και τοποθετεί σε πίνακα τους 20 πρώτους θετικούς αριθμούς.
3. Οι δόσεις για την αγορά ενός υπολογιστή υπολογίζονται ως εξής: ο πελάτης επιλέγει το ποσό της προκαταβολής και το πλήθος των μηνιαίων δόσεων, ενώ το ποσό των δόσεων προσαυξάνεται με ένα ετήσιο επιτόκιο. Να γραφτεί αλγόριθμος που υπολογίζει και καταχωρεί σε πίνακα το ποσό κάθε δόσης για την αγορά του υπολογιστή.
4. Αλλάξτε τον αλγόριθμο της σειριακής αναζήτησης ώστε να είναι πιο αποδοτικός σε ταξινομημένους πίνακες.
5. Σε ένα κατάστημα πώλησης ηλεκτρονικών αγαθών, πωλούνται 10 διαφορετικά μοντέλα smartphones. Να γραφτεί αλγόριθμος που: α) να διαβάζει και να καταχωρεί σε πίνακα τα μοντέλα των τηλεφώνων και την τιμή τους, β) να τα εμφανίζει ταξινομημένα με αύξουσα τιμή, γ) να ζητάει μια τιμή από το χρήστη και να εμφανίζει όλα τα smartphones με τιμή μικρότερη ή ίση από αυτήν.

Όταν ολοκληρώσετε την ενότητα ελέγξτε αν είστε σε θέση:

- να εντοπίζετε και να περιγράφετε δομές δεδομένων για συγκεκριμένα προβλήματα.
- να αναγνωρίζετε σε ποια προβλήματα είναι κατάλληλη και σε ποια είναι απαραίτητη η χρήση πινάκων.
- να αποφασίζετε τη διάσταση των πινάκων που θα χρησιμοποιήσετε.
- να επιλέγετε τον κατάλληλο αλγόριθμο αναζήτησης για συγκεκριμένο πρόβλημα.
- να εφαρμόζετε τις λειτουργίες πινάκων μίας ή δύο διαστάσεων σε συγκεκριμένα προβλήματα.
- να σχεδιάζετε και να υλοποιείτε λειτουργίες σε πίνακες.

2.2.9 Εκσφαλμάτωση σε λογικά λάθη

Στη διάρκεια συγγραφής αλγορίθμων διαπιστώνονται λάθη τα οποία οφείλονταν στη λανθασμένη σύνταξη εντολών, στη λάθος αποτύπωση εκφράσεων λόγω λανθασμένης χρήσης τελεστών, στη χρήση δεσμευμένων λέξεων ως ονόματα μεταβλητών, ασυμβατότητες στους τύπους δεδομένων κ.ά. Όλα αυτά τα λάθη προκαλούσαν αδυναμία μετάφρασης του προγράμματος με αποτέλεσμα να εντοπίζονται με σχετική ευκολία. Τέτοιου είδους λάθη ονομάζονται **συντακτικά λάθη**.

Εκτός όμως, από τα συντακτικά υπάρχουν και τα **λογικά λάθη** τα οποία εμφανίζονται στη διάρκεια εκτέλεσης του αλγορίθμου και οφείλονται σε σφάλματα κατά την υλοποίηση. Αυτού του είδους τα λάθη παράγουν ανεπιθύμητη έξοδο (λανθασμένη λύση) ή ανεπιθύμητη συμπεριφορά και συχνά είναι δύσκολο να εντοπιστούν. Η εργασία ελέγχου αφορά στον έλεγχο του αλγορίθμου με δοκιμαστικά δεδομένα, με

τη μορφή πίνακα τιμών.

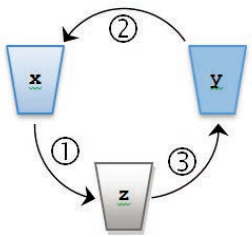
Εκτέλεση αλγορίθμου με πίνακα τιμών

Ο προγραμματιστής εκτελεί τον αλγόριθμο στο χαρτί με εικονικά δεδομένα, εξάγει τα αποτελέσματα και συγκρίνει τα δικά του αποτελέσματα με τα αναμενόμενα.

Παράδειγμα 2-14. Αλγόριθμος αντιμετάθεσης μεταβλητών. Μια διαδικασία πολύ χρήσιμη στον προγραμματισμό είναι η ανταλλαγή των τιμών δύο μεταβλητών. Σκοπός της αντιμετάθεσης των μεταβλητών x και y, είναι να βρεθεί η αρχική τιμή του x στη μεταβλητή y και το αντίθετο. Η λύση θα έλεγε κανείς ότι είναι απλή. Βάλε την τιμή του x στο y και μετά βάλε την τιμή του y στο x.

		x	y
	Αρχικά:	5	6
Βήμα 1:	Δώσε στο x την τιμή του y	6	
Βήμα 2:	Δώσε στο y την τιμή του x		6

Ας δούμε τον πίνακα τιμών του αλγόριθμου αυτού. Έστω ότι αρχικά x=5 και y=6. Δίνοντας στο x την τιμή του y, η αρχική τιμή του x χάνεται. Και οι δυο μεταβλητές έχουν την ίδια τιμή, την τιμή του y!



Εικόνα 2-30. Ανταλλαγή τιμών μεταβλητών

		x	y	z
	Αλγόριθμος Αντιμετάθεση(x↑↓, y↑↓)	5	6	
Βήμα 1:	z ← x			5
Βήμα 2:	x ← y	6		
Βήμα 3:	y ← z		5	
	Τέλος Αντιμετάθεση			

Εικόνα 2-31. Πίνακας Τιμών του αλγορίθμου αντιμετάθεσης

Η λύση βρίσκεται στη χρήση τρίτης βοηθητικής μεταβλητής, όπου αποθηκεύεται προσωρινά η τιμή του x πριν αλλάξει. Στο σχήμα που ακολουθεί, φαίνεται η χρονική σειρά των εντολών που οδηγούν στην αντιμετάθεση των x, y.

Ασκήσεις - Προβλήματα

1. Ελέγξτε αν ο παραπάκω αλγόριθμος κάνει αντιμετάθεση των μεταβλητών x, y.

```
Αλγόριθμος ΊσωςΑντιμετάθεση(x↑↓, y↑↓)
        x ← x + y
        y ← x - y
        x ← x - y
Τέλος ΊσωςΑντιμετάθεση
```

Όταν ολοκληρώσετε την ενότητα ελέγξτε αν είστε σε θέση:

- να περιγράφετε τα είδη λαθών ενός αλγορίθμου
- να εκτελείτε έναν αλγόριθμο κάνοντας πίνακα τιμών
- να εντοπίζετε και να διορθώνετε λογικά λάθη σε απλά παραδείγματα αλγορίθμων

2.2.10 Τεκμηρίωση

Στους αλγόριθμους που είδαμε διαπιστώσαμε ότι θα ήταν πιο κατανοητοί αν υπήρχαν κάποια σχόλια τα οποία να διευκρινίζουν τις λειτουργίες τους. Τα σχόλια σε έναν αλγόριθμο εισάγονται με το θαυμαστικό (!) και συνήθως τοποθετούνται στις επικεφαλίδες, σε ομάδες εντολών, σε δομές ελέγχου. Τοποθετούνται επίσης για διευκρινήσεις, αλλά και στα ονόματα των αναγνωριστικών. Η λογική της χρήσης των σχολίων είναι ο αλγόριθμος να μπορεί να γίνει κατανοητός από οποιονδήποτε προγραμματιστή.

```
Αλγόριθμος μέγιστος
!Υπολογισμός μεγίστου τριών αριθμών
Διάβασε x,y,z       ! Εισαγωγή τριών αριθμών
Αν x>=y τότε ! Σύγκριση των δύο αριθμών
      Max← x ! μέγιστος είναι ο πρώτος
Αλλιώς
      Max← y ! μέγιστος είναι ο δεύτερος
τέλος_αν      ! βρήκα το μέγιστο των δύο πρώτων
Αν z> Max τότε      ! Σύγκριση τρίτου με το μέγιστο των δύο
      Max←z  ! Μέγιστος όλων είναι ο τρίτος
τέλος_αν      ! Αλλιώς κράτα τον προηγούμενο μέγιστο
Γράψε Max
Τέλος μέγιστος
```

Αλγόριθμος 2-19. Τεκμηρίωση με σχόλια στον αλγόριθμο υπολογισμού του μέγιστου

Όταν ολοκληρώσετε την ενότητα ελέγξτε αν είστε σε θέση:

- να αναγνωρίζετε τη σημασία της τεκμηρίωσης σε έναν αλγόριθμο
- να εντοπίζετε και να περιγράφετε τη λειτουργία σχολίων σε απλά παραδείγματα αλγορίθμων

2.3 Προγραμματισμός

2.3.1 Γλώσσες προγραμματισμού και Προγραμματιστικά υποδείγματα

Με την πάροδο του χρόνου τα προβλήματα που καλούνταν να λύσουν οι υπολογιστές γινόταν όλο και πιο πολύπλοκα. Απαιτούσαν μεγαλύτερης έκτασης προγραμματισμό και συμμετοχή πολυπληθών ομάδων από προγραμματιστές. Προέκυψε λοιπόν η ανάγκη για τη δημιουργία αποτελεσματικών και τυποποιημένων διαδικασιών προγραμματισμού οι οποίες να μας δίνουν την δυνατότητα να δημιουργούμε πιο κατανοητό κώδικα, με λιγότερα σφάλματα, σε συνεργασία με άλλους προγραμματιστές. Οι τυποποιημένες αυτές διαδικασίες προγραμματισμού, ή αλλιώς προγραμματιστικά στυλ που καθιερώθηκαν, αποτέλεσαν τα προγραμματιστικά υποδείγματα για τα οποία συζητάμε στην παρούσα ενότητα.

Προγραμματιστικό υπόδειγμα ή **προγραμματιστικό παράδειγμα** (programming paradigm) ονομάζεται ένα θεμελιώδες στυλ προγραμματισμού υπολογιστών, με το οποίο ένα υπολογιστικό πρόβλημα και η αλγοριθμική λύση του προσεγγίζονται με συγκεκριμένες μεθόδους.

Είναι δηλαδή ένα σύνολο εννοιών οι οποίες εκφράζουν έναν συγκεκριμένο τρόπο σκέψης, και κατά συνέπεια έκφρασης της υλοποίησης, και διαμορφώνουν τον τρόπο σχεδιασμού ενός προγράμματος.

Τα πιο κοινά προγραμματιστικά υποδείγματα είναι: το **προστακτικό** (imperative), το **δηλωτικό** (declarative), το **λογικό** (logical), το **συναρτησιακό** (functional), το **δομημένο** (structured), το **αντικειμενοστραφές** ή **οντοκεντρικό** (object-oriented) και το υπόδειγμα **χειρισμού γεγονότων** ή **συμβάντων** (event-driven).

Σε πολλά από τα υποδείγματα υπάρχει επικάλυψη εννοιών, όπως και επιρροές μεταξύ τους. Έτσι κάποια από τα υποδείγματα απορρέουν από κάποια άλλα εξειδικεύοντάς τα ή εμπλουτίζοντάς τα. Δεν υπάρχει συμφωνία στην κοινότητα των προγραμματιστών, ποιο από τα υποδείγματα παρέχει την καλύτερη προσέγγιση στην ανάπτυξη λογισμικού. Αυτό είναι φυσικό, μια που κάθε ένα από αυτά αναπτύχθηκε και εξελίχθηκε στον χρόνο έχοντας στο μυαλό την ανάγκη αποτελεσματικότερης λύσης συγκεκριμένων προβλημάτων στην ανάπτυξη λογισμικού.

Ένα από τα χαρακτηριστικά μιας γλώσσας προγραμματισμού είναι η υποστήριξη που παρέχει σε συγκεκριμένα υποδείγματα. Έτσι για παράδειγμα, η γλώσσα Smalltalk έχει άμεση υποστήριξη για αντικειμενοστραφή προγραμματισμό. Κάποιες γλώσσες προγραμματισμού έχουν σχεδιαστεί για να ακολουθούν ένα μόνο υπόδειγμα, ενώ άλλες μπορούν να υποστηρίξουν πολλαπλά υποδείγματα. Έτσι, προγράμματα που γράφτηκαν στη γλώσσα C++ μπορεί να ακολουθούν αμιγώς το δομημένο υπόδειγμα, αμιγώς το αντικειμενοστραφές ή να περιέχουν στοιχεία και από τα δύο υποδείγματα. Οι δημιουργοί λογισμικού είναι αυτοί που αποφασίζουν πώς να χρησιμοποιήσουν καλύτερα τα στοιχεία του κάθε υποδείγματος.

Το διαδικαστικό (procedural) υπόδειγμα και το δομημένο (structured) κάποιες φορές εμφανίζονται στη βιβλιογραφία σαν διαφορετικά υποδείγματα. Σε αυτή την περίπτωση το ένα απορρέει από το άλλο και οι διαφορές τους είναι ελάχιστες. Σε άλλες περιπτώσεις όμως, όπως και στην δική μας, αντιμετωπίζονται σαν ταυτόσημες έννοιες.

Όταν αναφερόμαστε σε ένα υπόδειγμα πολύ συχνά αντικαθιστούμε τη λέξη «υπόδειγμα» με τη λέξη «προγραμματισμός», εννοώντας ακριβώς το ίδιο. Για παράδειγμα, οι φράσεις «προστακτικό υπόδειγμα» και «προστακτικός προγραμματισμός» είναι ταυτόσημες.

Προστακτικός προγραμματισμός

Στον προστακτικό προγραμματισμό, το ζητούμενο κατασκευάζεται / υπολογίζεται αλλάζοντας την κατάσταση του υπολογιστή μέσω εντολών. Η ροή εκτέλεσης του προγράμματος στον προστακτικό προγραμματισμό είναι σαφής. Εντολές δείχνουν πως υπολογίζεται το ζητούμενο, βήμα προς βήμα.

Το προστακτικό υπόδειγμα είναι πολύ κοντά σε αυτό που πράγματι συμβαίνει στον υπολογιστή μας όταν εκτελούμε ένα πρόγραμμα (γι' αυτό είναι συνήθως πιο οικείο). Διάφορες οδηγίες που επεξεργάζεται ο υπολογιστής έχουν ως αποτέλεσμα την αλλαγή των περιεχομένων της μνήμης του. Η γλώσσα μηχανής, η πιο χαμηλού επιπέδου γλώσσα και η μόνη που καταλαβαίνει άμεσα ο υπολογιστής, είναι μια καθαρά προστακτική γλώσσα.

Το υπόδειγμα του προστακτικού προγραμματισμού ακολουθούν οι δομημένες (ή διαδικαστικές) γλώσσες προγραμματισμού, όπως η Pascal, η C και η Fortran, αλλά και πολλές αντικειμενοστραφείς γλώσσες όπως η Java, η C++ και η C#.

Επειδή οι γλώσσες πολύ χαμηλού επιπέδου δεν είναι πρακτικές για τη συγγραφή μεγάλων και δύσκολων προγραμμάτων, οι σχεδιαστές γλωσσών προγραμματισμού αντιμετώπισαν αυτό το πρόβλημα με ποικίλους τρόπους. Σε μερικές περιπτώσεις παρέμειναν στο προστακτικό προγραμματισμό, αλλά εφηύραν αφαιρετικές έννοιες πιο "υψηλού επιπέδου", δηλαδή έννοιες που είναι πλησιέστερες στον άνθρωπο / προγραμματιστή και πιο μακριά από τον υπολογιστή. Η ιδέα της μεταβλητής, μιας θέσης μνήμης στην οποία ο προγραμματιστής μπορεί να δώσει όνομα, είναι το πιο χαρακτηριστικό και το πιο θεμελιώδες παράδειγμα μιας τέτοιας αφαίρεσης. Σε άλλες περιπτώσεις σχεδίασαν γλώσσες που να υποστηρίζουν δηλωτικό προγραμματισμό και φυσικά και το απαραίτητο λογισμικό για τη μετάφρασή τους σε γλώσσα μηχανής.

Παραδείγματα γλωσσών δηλωτικού προγραμματισμού είναι η Haskell, η LISP και η Prolog. Επίσης, αν και δεν μπορούμε να τις κατατάξουμε απόλυτα στις γλώσσες προγραμματισμού, το δηλωτικό υπόδειγμα ακολουθούν οι γλώσσες SQL, HTML και CSS.

Δηλωτικός προγραμματισμός

Στο δηλωτικό προγραμματισμό, σε αντίθεση με τον προστακτικό, το ζητούμενο αποτέλεσμα υπολογίζεται περιγράφοντας απλώς τις επιθυμητές ιδιότητες του. Η ροή εκτέλεσης ενός προγράμματος δηλωτικού προγραμματισμού δεν είναι φανερή. Ο προγραμματιστής καθορίζει μόνο τα χαρακτηριστικά του ζητούμενου αποτελέσματος και όχι πως θα υπολογιστεί το αποτέλεσμα.

Λογικός Προγραμματισμός

Στο λογικό προγραμματισμό, ο προγραμματιστής ορίζει μια σειρά από γεγονότα και κανόνες, και κατάλληλο λογισμικό συνάγει τις απαντήσεις στις ερωτήσεις που θέτει βάσει αυτών των κανόνων και των γεγονότων. Ο λογικός προγραμματισμός είναι τύπος δηλωτικού προγραμματισμού.

Χαρακτηριστικό παράδειγμα γλώσσας λογικού προγραμματισμού είναι η Prolog.

Συναρτησιακός Προγραμματισμός

Στην περίπτωση του συναρτησιακού προγραμματισμού, ο υπολογισμός περιγράφεται σα μία συνάρτηση που συσχετίζει τις παραμέτρους της (είσοδος) με το αποτέλεσμά της (έξοδος). Οι συναρτήσεις στο υπόδειγμα αυτό είναι μαθηματικές. Το αποτέλεσμα που επιστρέφουν εξαρτάται απο-

κλειστικά και μόνο από την είσοδο που τους δίνουμε. Δύο κλήσεις της ίδιας συνάρτησης με τις ίδιες παραμέτρους θα παράγουν αναγκαστικά το ίδιο αποτέλεσμα. Αυτό λέγεται σε αντιπαραβολή με τις "συναρτήσεις" προστακτικών γλωσσών προγραμματισμού όπου ο όρος "συνάρτηση" χρησιμοποιείται καταχρηστικά, καθώς μία τέτοια "συνάρτηση" μπορεί να επιστρέφει διαφορετικό αποτέλεσμα κάθε φορά που καλείται, ακόμα και αν οι παράμετροι είναι ίδιες μια που το αποτέλεσμα είναι δυνατό να δέχεται παρεμβολές από εξωτερικές πηγές (π.χ. τιμή μιας εξωτερικής στη «συνάρτηση» μεταβλητής).

Χαρακτηριστική γλώσσα συναρτησιακού προγραμματισμού είναι η Haskell.

Δομημένος προγραμματισμός

Ο δομημένος (ή διαδικαστικός ή διαδικασιακός) προγραμματισμός είναι ένα είδος δηλωτικού προγραμματισμού, ο οποίος βασίζεται στην έννοια της κλήσης διαδικασιών. Η διαδικασία, γνωστή επίσης και ως υποπρόγραμμα, ρουτίνα, υπορουτίνα ή συνάρτηση, είναι ένα αυτοτελές σύνολο εντολών προς εκτέλεση το οποίο είναι δυνατό να δέχεται παραμέτρους (είσοδος) και να επιστρέφει αποτέλεσμα (έξοδος).

Χαρακτηριστικές γλώσσες δομημένου προγραμματισμού είναι η Pascal, η C και η Fortran. Το υπόδειγμα του δομημένου προγραμματισμού όμως το υποστηρίζει και η πλειοψηφία των αντικειμενοστραφών γλωσσών, όπως η Java, η C++, η PHP, η Python και η C#.

Ο δομημένος προγραμματισμός βασίζεται στην αρχή του διαίρει και βασίλευε, καθώς διασπά το βασικό πρόβλημα σε μικρότερα έως ότου αυτά να είναι αρκετά μικρά, περιεκτικά και εύκολα προς κατανόηση και επίλυση.

Στο δομημένο προγραμματισμό έχουν προστεθεί ανακυκλώσεις και διαδικασίες για να αντικαταστήσουν τις εντολές ελέγχου ροής τύπου goto του (απλού) προστακτικού προγραμματισμού, όπως επίσης και τοπικές μεταβλητές για να βελτιώσουν την προγραμματιστική διαδικασία.

Αντικειμενοστραφής ή οντοκεντρικός προγραμματισμός

Ο αντικειμενοστραφής ή οντοκεντρικός προγραμματισμός είναι μία μεθοδολογία ανάπτυξης προγραμμάτων προστακτικού τύπου. Κεντρική ιδέα στον αντικειμενοστραφή προγραμματισμό είναι η κλάση (class), μία αυτοτελής και αφαιρετική αναπαράσταση κάποιας κατηγορίας αντικειμένων (objects), είτε φυσικών αντικειμένων του πραγματικού κόσμου είτε νοητών, εννοιολογικών αντικειμένων, σε ένα περιβάλλον προγραμματισμού. Πρακτικώς είναι ένας τύπος δεδομένων, ή αλλιώς το προσχέδιο μίας δομής δεδομένων με δικά της περιεχόμενα, τόσο μεταβλητές όσο και διαδικασίες. Οι διαδικασίες των κλάσεων συνήθως καλούνται μέθοδοι (methods) και οι μεταβλητές τους γνωρίσματα (attributes) ή πεδία (fields).

Ο αντικειμενοστραφής προγραμματισμός εμφανίστηκε στα τέλη της δεκαετίας του 1960 και καθιερώθηκε κατά τη δεκαετία του 1990, αντικαθιστώντας σε μεγάλο βαθμό το παραδοσιακό υπόδειγμα του δομημένου προγραμματισμού.

Προγραμματισμός χειρισμού γεγονότων ή συμβάντων

Στον προγραμματισμό χειρισμού γεγονότων ή συμβάντων, η ροή του προγράμματος εξαρτάται από εισερχόμενα γεγονότα. Η βασική ιδέα είναι ότι υπάρχει ένας κύριος ατέρμονος βρόχος (endless loop), ο οποίος

παρακολουθεί τα γεγονότα (events). Όταν ένα γεγονός λαμβάνει χώρα, τότε καλείται μια συνάρτηση που ονομάζεται συνάρτηση ανάκλησης (callback function) και μεταβάλλεται η ροή του προγράμματος.

Η μεθοδολογία αυτή χρησιμοποιείται κυρίως σε γραφικά περιβάλλοντα αλληλεπίδρασης χρήστη-υπολογιστή (graphical user interfaces). Για παράδειγμα, σε μια γραφική διεπαφή χρήστη, έχουμε ως γεγονότα τα πατήματα πλήκτρων ή επιλογών με το ποντίκι.

Χαρακτηριστικά παραδείγματα γλωσσών προγραμματισμού χειρισμού γεγονότων είναι η Actionscript, η Visual Basic και η Javascript.

Οι γλώσσες που υποστηρίζουν προγραμματισμό χειρισμού γεγονότων συνήθως υποστηρίζουν και άλλα προγραμματιστικά υποδείγματα, όπως το δομημένο ή το αντικειμενοστραφές.

Τα τελευταία χρόνια, λόγω της υποστήριξής της από όλους τους μοντέρνους περιηγητές ιστού (web browsers), μεγάλη εκτίμηση έχει κερδίσει η γλώσσα Javascript, η οποία χρησιμοποιείται πλέον στην πλειοψηφία υλοποιήσεων διεπαφών χρήστη (user interfaces) και υποσυστημάτων επικοινωνίας με το διακομιστή (server) σε εφαρμογές ιστού (web applications).

Παράδειγμα Χρήσης Γλωσσών Προγραμματισμού σε Εφαρμογή Ιστού

Με την τεράστια διάδοση του διαδικτύου, έχει επικρατήσει η τάση, η πλειοψηφία του λογισμικού που αναπτύσσεται να είναι κατάλληλο για λειτουργία μέσω του ιστού. Αυτός ο τύπος εφαρμογών (web applications) απαιτεί την χρήση αρκετών γλωσσών προγραμματισμού και τεχνολογιών για την υλοποίησή του.

Οι εφαρμογές αυτές είναι αποθηκευμένες σε έναν διακομιστή ιστού (web server) και ο χρήστης τους έχει πρόσβαση σε αυτές μέσω ενός περιηγητή ιστού (Internet Explorer, Chrome, Firefox, κ.ά.) που είναι εγκατεστημένος στον υπολογιστή του. Από τη στιγμή που ο χρήστης θα ζητήσει την εκτέλεση μιας τέτοιας εφαρμογής, ο περιηγητής ιστού επικοινωνεί με το διακομιστή ιστού και παραλαμβάνει την διεπαφή χρήστη (user interface) της εφαρμογής (μέσω πρωτοκόλλου http), την οποία εκτελεί. Η διεπαφή χρήστη συνήθως αποτελείται από αρχεία γραμμένα σε HTML, σε CSS και σε Javascript και από εικόνες που τη συμπληρώνουν αισθητικά.

Εικόνα 2-32. Εφαρμογή ιστού

Τα αρχεία HTML καθορίζουν την δομή της διεπαφής (επικεφαλίδες, παράγραφοι, πίνακες, φόρμες, κ.ά.), τα αρχεία CSS καθορίζουν τη μορφοποίηση των αντικειμένων της διεπαφής (χρώματα, θέσεις, γραμματοσειρές, μεγέθη, κ.α.) ενώ τα αρχεία Javascript αναλαμβάνουν το ενεργό κομμάτι της διεπαφής (διαχείριση γεγονότων πληκτρολόγησης και χρήσης ποντικιού, ανανέωση αποτελεσμάτων στην οθόνη, κ.λ.π.).

Το κεντρικό κομμάτι μιας εφαρμογής ιστού είναι αυτό που παραμένει στο διακομιστή ιστού και εκτελείται από αυτόν κάθε φορά που ο χρήστης ζητάει την εκτέλεση κάποιας λειτουργίας μέσω του περιηγητή ιστού. Το κεντρικό κομμάτι της εφαρμογής είναι γραμμένο σε κάποια από τις γλώσσες με τις οποίες μπορεί να συνεργαστεί ο διακομιστής ιστού, όπως οι Java, PHP και Python.

Σε περίπτωση που θα χρειαστεί πληροφορία που είναι αποθηκευμένη

σε κάποια βάση δεδομένων, αυτή, στις περισσότερες περιπτώσεις, θα αποκτηθεί με τη χρήση εντολών σε γλώσσα SQL.

Τα αποτελέσματα της εφαρμογής πριν σταλούν στον περιηγητή ιστού για προβολή μετατρέπονται σε HTML ή σε κάποια μορφή δεδομένων που μπορεί να διαχειριστεί η Javascript και αυτός αναλαμβάνει την παρουσίασή τους στον χρήστη. Η Εικόνα 2-32 παρουσιάζει σε απλοποιημένη μορφή τη λειτουργία μιας συνηθισμένης εφαρμογής ιστού.

2.3.2 Σχεδίαση και συγγραφή κώδικα

Στην ενότητα 2.2 γνωρίσαμε έννοιες των αλγορίθμων και δημιουργήσαμε αλγόριθμους για την επίλυση προβλημάτων. Σε αυτήν την ενότητα μετατρέπουμε τους αλγόριθμους σε πηγαίο κώδικα (source code) γραμμένο σε συγκεκριμένη γλώσσα προγραμματισμού. Ο πηγαίος κώδικας, αφού μεταφραστεί, θα εκτελεστεί από τον υπολογιστή για να παράγει αποτελέσματα.

Επιλογή γλώσσας προγραμματισμού

Η δημιουργία προγραμμάτων που μπορούν να εκτελεστούν από τον υπολογιστή απαιτεί τη χρήση μιας γλώσσας προγραμματισμού και ενός προγραμματιστικού περιβάλλοντος γι' αυτή τη γλώσσα. Κάθε γλώσσα προγραμματισμού είναι καταλληλότερη για συγκεκριμένου είδους προγράμματα, συγκεκριμένες ανάγκες χρηστών και συγκεκριμένο είδος υπολογιστών. Άλλες γλώσσες είναι κατάλληλες για προγραμματισμό στο Διαδίκτυο, άλλες για εκπαιδευτικούς σκοπούς, άλλες για προγραμματισμό συστημάτων, κ.ο.κ. Επιπλέον, διαφορετικές γλώσσες προγραμματισμού απαιτούν διαφορετικές γνώσεις από τον προγραμματιστή.

Σε αυτήν την ενότητα γράφουμε κώδικα σε γλώσσα προγραμματισμού Python για να υλοποιήσουμε αλγόριθμους που έχουμε αναπτύξει σε προηγούμενες ενότητες. Η παρουσίαση της Python στο παρόν βιβλίο δεν είναι πλήρης, ούτε εξαντλητική και έχει ως μοναδικό στόχο την εισαγωγή σε έννοιες που συναντώνται σε πολλές γλώσσες προγραμματισμού. Μπορεί να γίνει επιλογή οποιασδήποτε άλλης γλώσσας για τη συγγραφή και εκτέλεση των προγραμμάτων αυτής της ενότητας.

Χαρακτηριστικά της γλώσσας Python

- Απλή σύνταξη
- Γενικής χρήσης
- Δυναμική τυποποίηση των δεδομένων
- Δομές δεδομένων υψηλού επιπέδου
- Επεκτάσιμη
- Ενσωματώνεται εύκολα σε άλλες γλώσσες προγραμματισμού
- Οντοκεντρική
- Χρησιμοποιεί διερμηνευτή

=>

- ευκολία μάθησης,
- γρήγορη συγγραφή προγραμμάτων,
- μικρότερα προγράμματα, σύγχρονη προσέγγιση

Το όνομα της Python προέρχεται από την τηλεοπτική εκπομπή του BBC «Monty Python's Flying Circus».

Προγραμματιστικό περιβάλλον και εργαλεία προγραμματισμού

Τα προγραμματιστικά περιβάλλοντα περιλαμβάνουν ένα σύνολο από προγράμματα και «εργαλεία» που βοηθούν τον προγραμματιστή να δημιουργήσει, να μεταφράσει και να εκτελέσει τα προγράμματά του: το **συντάκτη** (editor), το **μεταφραστή** (translator) και τον **εκσφαλματωτή** (debugger).

Ο συντάκτης βοηθάει στη συγγραφή του προγράμματος, δηλαδή του **πηγαίου κώδικα** (source code). Μερικοί συντάκτες μπορούν να αναγνωρίζουν αυτόματα την ορθογραφία της γλώσσας προγραμματισμού, να κάνουν έξυπνη στοίχιση και να χρησιμοποιούν διάφορα χρώματα για να κάνουν τον κώδικα πιο ευανάγνωστο. Συνήθως ο συντάκτης είναι μέρος ενός **ολοκληρωμένου προγραμματιστικού περιβάλλοντος** (In-

tegrated Development Environment – IDE). Και ένα απλό πρόγραμμα επεξεργασίας κειμένου όμως μπορεί να παίξει το ρόλο του συντάκτη. Ο πηγαίος κώδικας, με όποιο πρόγραμμα και αν συνταχθεί, αποθηκεύεται σε αρχεία κειμένου, με κατάληξη που παραπέμπει στη γλώσσα προγραμματισμού, π.χ. .c (γλώσσα C) , .cpp (C++), .java (Java), .js (Javascript), .py (Python) κ.λ.π..

Μετά τη συγγραφή του πηγαίου κώδικα ο μεταφραστής αναλαμβάνει να τον μετατρέψει σε γλώσσα μηχανής. Η μετάφραση μπορεί να γίνει από δύο διαφορετικά είδη προγραμμάτων: από διερμηνευτή (interpreter) ή από μεταγλωττιστή (compiler). Ο **διερμηνευτής** αναλύει και εκτελεί το πρόγραμμα εντολή προς εντολή, δηλαδή μετατρέπει κάθε εντολή σε γλώσσα μηχανής, την εκτελεί και συνεχίζει την ίδια διαδικασία μέχρι το τέλος. Η γλώσσα Python χρησιμοποιεί διερμηνευτή.

Ο **μεταγλωττιστής** από την άλλη, διαβάζει όλο το πρόγραμμα, κάνει λεξικογραφική, συντακτική και σημασιολογική ανάλυση, παράγει ένα ισοδύναμο πρόγραμμα σε γλώσσα μηχανής και συχνά βελτιστοποιεί τον κώδικα. Το εκτελέσιμο πρόγραμμα που παράγεται τελικά από τη μεταγλώττιση είναι ανεξάρτητο από το πηγαίο και μπορεί να εκτελεστεί οποτεδήποτε το επιθυμεί ο χρήστης. Οι γλώσσες προγραμματισμού C και Pascal χρησιμοποιούν μεταγλωττιστή.

Λεξικογραφική ανάλυση: αναγνώριση λέξεων

Συντακτική ανάλυση: γραμματική ανάλυση (parsing)

Σημασιολογική ανάλυση: δήλωση συναρτήσεων, έλεγχος τύπων δεδομένων, κ.ά.

Προγράμματα στη γλώσσα Python

Ένα πρόγραμμα στη γλώσσα Python μπορεί να γραφτεί απευθείας στη γραμμή εντολών της γλώσσας, σε ένα γραφικό περιβάλλον ή σε έναν απλό συντάκτη επεξεργασίας κειμένου και να αποθηκευτεί σε αρχείο με κατάληξη .py. Το βασικό περιβάλλον της Python είναι μια απλή γραμμή εντολών (command line) και κάτω από κάθε εντολή εμφανίζεται το αποτέλεσμά της μόλις αυτή εκτελεστεί. Η εκτέλεση γίνεται με την αλλαγή γραμμής. Δηλαδή, ένα πρόγραμμα σε Python αποτελείται από μία σειρά γραμμών οι οποίες διαχωρίζονται με το χαρακτήρα NEWLINE. Οι εντολές τοποθετούνται μέσα στις γραμμές. Στη συνέχεια παρουσιάζονται τα βασικά στοιχεία της γλώσσας Python που επιτρέπουν τη μετατροπή των αλγορίθμων που έχουμε ήδη δει σε απλά προγράμματα.

Αναγνωριστικά: είναι ονόματα που χρησιμοποιούνται για να ταυτοποιήσουν μεταβλητές, συναρτήσεις κ.ά. Ξεκινούν με ένα γράμμα του λατινικού αλφαβήτου (A–Z, a-z) ή με το χαρακτήρα _. Ο πρώτος χαρακτήρας ακολουθείται από μηδέν ή περισσότερα γράμματα, _ και ψηφία (0 ως 9). Έγκυρα αναγνωριστικά είναι για παράδειγμα τα: a, my_name, emvado_kyklou, aktina1, κλπ.

Δεσμευμένες λέξεις: είναι λέξεις που δε μπορούν να χρησιμοποιηθούν ως αναγνωριστικά γιατί χρησιμοποιούνται για ειδικό σκοπό. Δεσμευμένες λέξεις της Python είναι μεταξύ άλλων οι λέξεις and, break, continue, def, elif, else, for, from, if, in, not, or, return, while. Η σημασιολογία και η χρήση τους παρουσιάζονται παρακάτω μέσα από τα παραδείγματα.

Αριθμητικές και μαθηματικές εκφράσεις. Η Python μπορεί να χρησιμοποιηθεί για τον υπολογισμό αριθμητικών εκφράσεων που αποτε-

λούνται από αριθμητικούς τελεστές και τελεστέους, δηλαδή σύμβολα πράξεων. Αν στη γραμμή εντολών της γλώσσας γράψουμε 2 + 3 και αλλάξουμε γραμμή, ο διερμηνευτής υπολογίζει την τιμή της παράστασης και εμφανίζει το αποτέλεσμα στην οθόνη. Εκτός από τους γνωστούς αριθμητικούς τελεστές +, -, *, /, μπορούμε να χρησιμοποιήσουμε και μαθηματικές συναρτήσεις που αποτελούν μέρος της βιβλιοθήκης της γλώσσας.

Μεταβλητές. Είναι θέσεις μνήμης που δεσμεύονται για την αποθήκευση τιμών. Ο διερμηνευτής της Python δεσμεύει χώρο στη μνήμη του υπολογιστή για τις μεταβλητές ανάλογα με τον τύπο της κάθε μεταβλητής (τύπος δεδομένων). Σε αντίθεση με γλώσσες όπως η Pascal, η C και η Java, ο προγραμματιστής της Python δεν είναι απαραίτητο να δηλώσει τον τύπο των μεταβλητών που χρησιμοποιεί. Η δήλωση γίνεται αυτόματα όταν η μεταβλητή πάρει τιμή για πρώτη φορά.

Εκχώρηση ή ανάθεση τιμής. Γίνεται με το σύμβολο =. Μια εκχώρηση τιμής έχει τη μορφή όνομα_μεταβλητής = τιμή. Αριστερά του συμβόλου = υπάρχει ένα έγκυρο αναγνωριστικό - όνομα μεταβλητής και δεξιά η τιμή που θα εκχωρηθεί στη μεταβλητή. Η τιμή αυτή μπορεί να προκύπτει από τον υπολογισμό μιας έκφρασης.

Πρόγραμμα 2-1. Εκχωρήσεις τιμών σε μεταβλητές

```
#δήλωση και εκχώρηση τιμής στην μεταβλητή count
count = 5
# δήλωση  και εκχώρηση τιμής στην μεταβλητή price
price = 13.45
name = "Άννα Μαρία"  #δήλωση, εκχώρηση στην μεταβλητή name
print(name)            #θα τυπώσει Άννα Μαρία
a = b = c = 1          #πολλαπλή δήλωση και εκχώρηση τιμής
#πολλαπλή δήλωση, εκχώρηση τιμών
age, name2 = 17, "Αλέξανδρος"
```

Απλοί τύποι δεδομένων. Η γλώσσα Python υποστηρίζει απλούς και σύνθετους τύπους δεδομένων. Απλοί και βασικοί τύποι δεδομένων είναι οι αριθμοί: ακέραιοι (int), δίτιμοι ακέραιοι (bool) και πραγματικοί (float). Οι ακέραιοι παριστάνουν ακέραιες ποσότητες με εύρος που εξαρτάται από τη διαθέσιμη κύρια μνήμη του υπολογιστή. Οι δίτιμοι ακέραιοι ή λογικοί αριθμοί παριστάνουν τις λογικές τιμές True (Αληθής) και False (Ψευδής) και αντιστοιχούν σε 1 και 0. Οι πραγματικοί (float) παριστάνουν αριθμούς κινητής υποδιαστολής (floating point). Οι σύνθετοι τύποι δεδομένων περιγράφονται παρακάτω.

Είσοδος και έξοδος. Η είσοδος δεδομένων από το χρήστη (πληκτρολόγιο) γίνεται με την συνάρτηση input() που είναι μέρος της βιβλιοθήκης της Python. Η έξοδος γίνεται με τη συνάρτηση print().

Ας δούμε τώρα τον απλό αλγόριθμο σειριακής εκτέλεσης που υπολογίζει την περιφέρεια και το εμβαδόν ενός κύκλου εισάγοντας την ακτίνα του. Η πρώτη εντολή θα εισάγει και θα αποθηκεύσει την ακτίνα

του κύκλου στην μεταβλητή aktina. Η είσοδος δεδομένων γίνεται με τη συνάρτηση input(), όπου μέσα στις παρενθέσεις τοποθετούμε σε εισαγωγικά (‘’) το μήνυμα που θα δει ο χρήστης και θα του ζητάει να εισάγει δεδομένα.

Πρόγραμμα 2-2. Υπολογισμός της περιφέρειας και του εμβαδού κύκλου

```
aktina  = float(input('Δώσε την ακτίνα'))
pi = 3.14
periferia = aktina*2*pi
emvado = aktina**2*pi
print (periferia )
print (emvado)
```

Η ποσότητα «ακτίνα κύκλου» είναι πραγματική. Η Python δεν απαιτεί να δηλώσουμε τον τύπο της μεταβλητής aktina πριν την χρησιμοποιήσουμε, αλλά για να διασφαλίσουμε ότι η τιμή που θα δοθεί είναι πραγματική, μετατρέπουμε την είσοδο σε πραγματικό αριθμό με τη συνάρτηση float(). Με παρόμοιο τρόπο, μια τιμή μπορεί να μετατραπεί σε ακέραια με τη συνάρτηση int().

Δομές ελέγχου. Οι δομές ελέγχου βοηθούν στη λήψη απόφασης με βάση ένα ή περισσότερα κριτήρια. Οι εντολές της Python που υποστηρίζουν έλεγχο είναι οι **if**, **if..else** και **if…elif…else**. Η σύνταξη της εντολής if σε απλή μορφή φαίνεται στο παράδειγμα υπολογισμού της απόλυτης τιμής. Μετά τη δεσμευμένη λέξη if ακολουθεί μια λογική έκφραση, π.χ. $x < 0$ και κατόπιν ο χαρακτήρας ‘:’. Οι εντολές που θα εκτελεστούν αν η συνθήκη είναι αληθής τοποθετούνται σε νέες γραμμές κάτω από τη γραμμή του if και μάλιστα ξεκινούν όλες με έναν τουλάχιστον κενό χαρακτήρα (space ή tab) για να οριοθετηθεί η ομάδα εντολών της επιλογής.

Ο αλγόριθμος εύρεσης του μέγιστου μεταξύ δύο αριθμών περιλαμβάνει σύνθετη επιλογή με την εντολή else. Η πολλαπλή επιλογή elif, μέσω της οποίας ελέγχονται πολλές συνθήκες, μπορεί να εφαρμοστεί στον αλγόριθμο ελέγχου του Δείκτη Μάζας Σώματος.

Πρόγραμμα 2-3. Απόλυτη τιμή αριθμού

```
x = float(input('Δώσε τιμή για το x '))
if x < 0 :
    x = -x
print("Η απόλυτη τιμή είναι:", x)
##########################################################
```

Πρόγραμμα 2-4. Μέγιστο δύο αριθμών

```
x=float(input('Δώσε τιμή για το x '))
y=float(input('Δώσε τιμή για το y '))
if x < y :
    megistos = y
else :
    megistos = x
print("Ο μεγαλύτερος από τους δύο είναι: ", megistos)
```

Οι εμφωλευμένες επιλογές ελέγχουν μία συνθήκη μέσα σε μία άλλη, όπως φαίνεται στο παράδειγμα υπολογισμού του κόστους αποστολής

εσωτερικού με βάση τον προορισμό και το βάρος της επιστολής. Μια εμφωλευμένη επιλογή μπορεί να υλοποιηθεί και με σύνθετες λογικές εκφράσεις που αποτελούνται and (και), or (ή) και not (όχι).

Πρόγραμμα 2-5. Χαρακτηρισμός ατόμου με βάση το ΔΜΣ

```
b=float(input('Δώσε βάρος '))
h=float(input('Δώσε ύψος '))
dms = b/(h**2)              #υπολογισμός του ΔΜΣ
if dms < 18.5 :
    print("ελλιποβαρές άτομο")
elif dms <25:      print("άτομο με φυσιολογικό βάρος")
elif dms <30:      print("υπέρβαρο άτομο ")
else:     print("άτομο που πάσχει από παχυσαρκία")
##########################################################
```

Πρόγραμμα 2-6. Εκτύπωση του κόστους αποστολής

```
d=input('Δώσε προορισμό (home ή abroad):  ')
w=float(input('Δώσε βάρος επιστολής σε γραμμάρια: ')
if d ==’home’:
      if w<=100 :  print(‘Το κόστος είναι 1 ευρώ’)
      else :       print(‘Το κόστος είναι 2 ευρώ’)
```

Σύνθετοι τύποι δεδομένων. Στους βασικούς τύπους δεδομένων ανήκουν και οι σειρές (sequences) ή λίστες οι οποίες παριστάνουν διατεταγμένα σύνολα πεπερασμένου μεγέθους τα στοιχεία των οποίων έχουν δείκτη. Όταν το μήκος μιας σειράς είναι N, οι δείκτες των στοιχείων είναι οι αριθμοί 0, 1, ..., N-1 και το στοιχείο i της σειράς με όνομα A, επιλέγεται γράφοντας A[i].

Οι σειρές της Python χρησιμοποιούνται και για την αναπαράσταση των πινάκων που συζητήθηκαν στην ενότητα 2.2.8, με τη διαφορά ότι οι σειρές της Python είναι δυναμικές και όχι στατικές και μπορούν σε αυτές να προσθαφαιρούνται στοιχεία ανά πάσα στιγμή και να αλλάζει το μέγεθός τους. Επομένως, η αναπαράσταση των πινάκων στην Python γίνεται με δυναμική παραχώρηση μνήμης.

Πρόγραμμα 2-7. Αναπαράσταση πίνακα με σειρές της Python

```
grades = [18, 19, 20]      #πίνακας με τρεις βαθμούς
print(grades[0])           #εκτυπώνει 18
grades[0] = 19             #αλλαγή του πρώτου στοιχείου
print(grades)              #εκτυπώνει [19, 19, 20]
grades[2] = grades[2]-1    #μειώνει το τρίτο στοιχείο κατά 1
print(grades)              #εκτυπώνει [19, 19, 19]
```

Δομές επανάληψης. Οι δομές επανάληψης (επαναληπτικές διαδικασίες ή βρόχοι (loops)) οι οποίες υποστηρίζονται από την Python είναι η while και η for.

Ο βρόχος **while** (όσο) επαναλαμβάνει ένα σύνολο εντολών όσο μια συνθήκη είναι αληθής, ελέγχοντας την τιμή της πριν εκτελεστούν οι εντολές. Ας θεωρήσουμε το πρόβλημα του υπολογισμού του μέσου όρου ηλικιών που δίνονται από το χρήστη. Το διάβασμα ηλικιών μπορεί να

συμφωνηθεί ότι θα γίνεται όσο οι ηλικίες που εισάγονται είναι θετικοί αριθμοί. Τιμή ηλικίας μικρότερη ή ίση του μηδενός σημαίνει ότι ο χρήστης επιθυμεί τον τερματισμό της εισόδου δεδομένων.

Ο βρόχος **for** (για) χρησιμοποιεί έναν μετρητή, π.χ. i και μία έκφραση in range( ) για να εκφράσει ότι ο μετρητής παίρνει τιμές σε ένα συγκεκριμένο διάστημα, όπως φαίνεται στο παράδειγμα υπολογισμού του αθροίσματος 100 αριθμών που εισάγονται από το πληκτρολόγιο.

```
s=0
count = 0
age = int(input('Δώσε ηλικία'))
while age>0 :
       count = count +1
       s=s+age
       age= int(input('Δώσε  ηλικία'))
if count > 0 :
      print ("Ο μέσος όρος ηλικιών είναι " , s/count)
#####################################################
s=0
for i in range (100) :
       x=float(input('Δώσε τιμή για το x '))
       s=s+x
print ("Το αθροισμα είναι" , s)
```

Πρόγραμμα 2-8. Υπολογισμός του μέσου όρου ηλικιών που εισάγονται από το πληκτρολόγιο

Πρόγραμμα 2-9. Υπολογισμός του αθροίσματος 100 αριθμών, που εισάγονται από το πληκτρολόγιο

Συναρτήσεις. Συχνά, ένα τμήμα κώδικα που έχουμε γράψει πρέπει να χρησιμοποιηθεί πολλές φορές και σε διαφορετικά σημεία ενός ή περισσότερων προγραμμάτων. Ας θυμηθούμε το παράδειγμα των θερμοκρασιών που αναπτύξαμε στην ενότητα 2.2.8, όπου το ζητούμενο τώρα είναι να υπολογίσουμε τη μέγιστη θερμοκρασία ανά ημέρα, αλλά και ανά ώρα των επτά ημερών. Ο αλγόριθμος υπολογισμού του μέγιστου είναι ο ίδιος, σε ό,τι δεδομένα και αν εφαρμοστεί (σε οποιοδήποτε πρόβλημα, με οποιονδήποτε τύπο και οποιοδήποτε πλήθος δεδομένων). Υπάρχει τρόπος να γράψουμε τον αλγόριθμο μία φορά και να τον χρησιμοποιήσουμε όποτε μας χρειάζεται. Όλες οι γλώσσες προγραμματισμού δίνουν τη δυνατότητα τμηματοποίησης και επαναχρησιμοποίησης του πηγαίου κώδικα. Ο μηχανισμός που το επιτρέπει αυτό είναι τα υποπρογράμματα.

Τα **υποπρογράμματα** είναι τμήματα πηγαίου κώδικα που εκτελούν συγκεκριμένη λειτουργία και μπορούν να διαχωριστούν από το υπόλοιπο πρόγραμμα.

Οι περισσότερες γλώσσες προγραμματισμού ονομάζουν τα υποπρογράμματα **συναρτήσεις** ή **μεθόδους**. Έχουμε ήδη χρησιμοποιήσει συναρτήσεις για την είσοδο και την εκτύπωση δεδομένων, αλλά και για τον υπολογισμό μαθηματικών ποσοτήτων. Η συνάρτηση input( ) είναι υποπρόγραμμα της γλώσσας Python. Το ίδιο και οι συναρτήσεις print( ), sqrt( ) κ.λ.π. Οι συναρτήσεις αυτές είναι υλοποιημένες από τους δημιουργούς της γλώσσας για να τις χρησιμοποιήσουν οι προγραμματιστές

Ένα υποπρόγραμμα:
- Είναι ένα τμήμα προγράμματος που επιτελεί συγκεκριμένη λειτουργία.
- Γράφεται, ελέγχεται και διορθώνεται ανεξάρτητα από άλλα τμήματα του προγράμματος.
- Έχει σαφή όρια (αρχή και τέλος).
- Έχει είσοδο και έξοδο.
- Μπορεί να «κληθεί» από διαφορετικά σημεία του ίδιου προγράμματος.
- Μπορεί να κληθεί από διαφορετικά προγράμματα.
- Επικοινωνεί με το υπόλοιπο πρόγραμμα ανταλλάσσοντας τιμές μέσω των παραμέτρων.

και να μη χρειαστεί να τις ξαναγράψουν. Τις συναρτήσεις αυτές τις «καλέσαμε» από τα δικά μας προγράμματα.

Μπορούμε όμως να δημιουργήσουμε μια δική μας συνάρτηση όποτε θέλουμε ένα τμήμα του προγράμματος να σχεδιαστεί και να υλοποιηθεί ανεξάρτητα από το υπόλοιπο πρόγραμμα. Στο παράδειγμα με τις θερμοκρασίες, διακρίναμε το ανεξάρτητο υποπρόβλημα υπολογισμού του μέγιστου, το οποίο θέλουμε να λυθεί σε δύο διαφορετικές περιπτώσεις, ανά ημέρα και ανά ώρα των επτά ημερών. Άρα, το αυτόνομο υποπρόγραμμα που θα γράψουμε, θα πρέπει να μπορεί να εφαρμοστεί, ανεξάρτητα από το αν πρόκειται για τις θερμοκρασίες επτά ημερών ή για τις θερμοκρασίες ενός 24ωρου. Στην πρώτη περίπτωση, το υποπρόγραμμα θα πρέπει να «δέχεται» επτά θερμοκρασίες για μια συγκεκριμένη ώρα, ενώ στη δεύτερη, εικοσιτέσσερις θερμοκρασίες μίας συγκεκριμένης ημέρας. Σε κάθε περίπτωση το υποπρόγραμμα θα πρέπει να υπολογίζει και να επιστρέφει το σωστό μέγιστο. Άρα, για το υποπρόγραμμα υπολογισμού του μέγιστου, τα δεδομένα εισόδου είναι μεταβλητά. Ο μηχανισμός με τον οποίο υποστηρίζεται η επικοινωνία ενός υποπρογράμματος με το υπόλοιπο πρόγραμμα και η ανταλλαγή τιμών μεταξύ των δύο, ονομάζεται “παράμετροι”.

Οι **παράμετροι** είναι μεταβλητές που επιτρέπουν το πέρασμα τιμών από ένα τμήμα προγράμματος σε ένα άλλο.

Κατά τη σύλληψη, σχεδίαση και συγγραφή ενός υποπρογράμματος καθορίζονται οι παράμετροί του, δηλαδή οι ποσότητες που θα αποτελέσουν είσοδο στο υποπρόγραμμα και οι ποσότητες που θα αποτελέσουν έξοδο από αυτό.

Πρόγραμμα 2-10. Ορισμός και κλήση της συνάρτησης εύρεσης του μέγιστου

```
#def είναι δεσμευμένη λέξη που δηλώνει τον
#  ορισμό συνάρτησης
#  max2 είναι το όνομα της συνάρτησης
#  a, b είναι οι παράμετροι της συνάρτησης
def max2(a, b) :
      if a > b : return a
      return b

#κλήση της συνάρτησης max2
max2(10, 14)              #θα τυπώσει 14
#Κλήση της συνάρτησης max2 με παραμέτρους X, Y ή Y, X
X = 20
Y = 17
max2(X, Y)    # το X αντιστοιχίζεται στην παράμετρο a
max2(Y, X)    # το Y αντιστοιχίζεται στην παράμετρο a
#Κλήση της max2 με 1η παράμετρο το αποτέλεσμα
# της κλήσης max2(X, Y)
Z = 24
max2(max2(X, Y), Z)       #τι θα εκτυπωθεί;
```

Οι παράμετροι κατά την κλήση του υποπρογράμματος αντιστοιχίζονται σε μεταβλητές ή ποσότητες του τμήματος προγράμματος που καλεί το υποπρόγραμμα. Αν αλλάξει η τιμή της παραμέτρου μέσα στο υπο-

πρόγραμμα, τότε αλλάζει και η τιμή της μεταβλητής που έχει αντιστοιχιστεί σε αυτή την παράμετρο κατά την κλήση του υποπρογράμματος.

Ασκήσεις - Προβλήματα

1. Να γράψετε σε κώδικα: α) μια συνάρτηση η οποία θα επιστρέφει την περίμετρο ενός κύκλου με παράμετρο την ακτίνα του και β) μια συνάρτηση η οποία θα επιστρέφει το εμβαδόν του κύκλου.
2. Να δημιουργήσετε σε κώδικα μια συνάρτηση που να δέχεται ως παραμέτρους το βάρος και το ύψος ενός ατόμου και να εμφανίζει στην οθόνη το χαρακτηρισμό του (ελλιποβαρές, φυσιολογικό, υπέρβαρο ή παχύσαρκο) με βάση το Δείκτη Μάζας Σώματος.
3. Να γράψετε σε κώδικα, συνάρτηση που υπολογίζει και επιστρέφει το μέγιστο μιας σειράς Ν ακεραίων, καθώς και τη θέση στην οποία βρέθηκε το μέγιστο στοιχείο.
4. Να μετατρέψετε τον αλγόριθμο της σειριακής αναζήτησης σε κώδικα συνάρτησης. Η συνάρτηση θα δέχεται ως παράμετρο μία σειρά αριθμών, το πλήθος τους και το κλειδί αναζήτησης και θα επιστρέφει τη θέση στην οποία βρέθηκε το κλειδί, ή -1 αν δε βρεθεί.
5. Να μετατρέψετε α) τον αλγόριθμο της δυαδικής αναζήτησης και β) τον αλγόριθμο της ταξινόμησης με εισαγωγή σε κώδικα συναρτήσεων.

Όταν ολοκληρώσετε την ενότητα ελέγξτε αν είστε σε θέση:

- να γράφετε εντολές σε γλώσσα Python.
- να μετατρέπετε γνωστούς αλγόριθμους σε γλώσσα Python και να συνθέτετε δικά σας προγράμματα.
- να δημιουργείτε συναρτήσεις.
- να χρησιμοποιείτε διερμηνευτή της γλώσσας Python για να εκτελείτε τα προγράμματά σας.
- να διορθώνετε τα προγράμματά σας όταν έχουν συντακτικά λάθη.

2.3.3 Κύκλος ζωής εφαρμογής λογισμικού

Από τα πρώτα χρόνια του προγραμματισμού μέχρι σήμερα τα προγράμματα έχουν γίνει εξαιρετικά πολύπλοκα ώστε ακόμα και οι ίδιοι οι δημιουργοί τους αδυνατούν να τα ελέγξουν. Συχνά περιέχουν σφάλματα τα οποία εμφανίζονται μετά την παράδοση στον πελάτη, ορισμένες φορές μάλιστα ύστερα από χρόνια λειτουργίας. Για την αντιμετώπιση αυτών των προβλημάτων, από το τέλος της δεκαετίας του ΄60, αναγνωρίστηκε ότι το λογισμικό απαιτεί μεθοδικό σχεδιασμό.

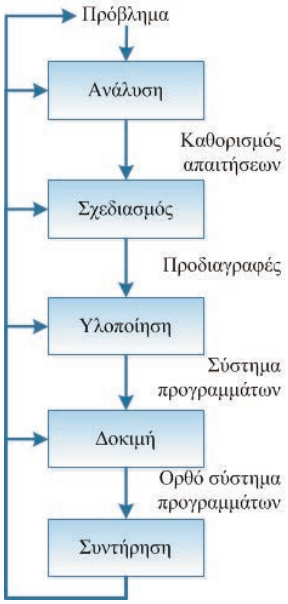
Τα προβλήματα των πολύ μεγάλων προγραμμάτων, συζητήθηκαν σε συνέδρια που διοργανώθηκαν από το ΝΑΤΟ τη δεκαετία του 1960.

Η εργασία ανάπτυξης ενός μεγάλου έργου λογισμικού είναι μια διαδικασία που διαιρείται σε πολλές φάσεις, καθεμιά από τις οποίες έχει το δικό της σημαντικό ρόλο. Οι φάσεις αυτές είναι:

1. **Ανάλυση του προβλήματος.** Το πρόβλημα που πρέπει να λυθεί ορίζεται και αναλύεται σε συνεργασία πελάτη και κατασκευαστή. Περιγράφονται οι απαιτήσεις του προς ανάπτυξη λογισμικού, επισημαίνονται ιδιαιτερότητες, υποδεικνύονται σημεία στα οποία απαιτείται ιδιαίτερη προσοχή και περιγράφονται οι συνθήκες κάτω

από τις οποίες είναι δυνατή η υλοποίηση. Ακόμα, αναλύονται στοιχεία που αφορούν το κόστος αλλά και τους τρόπους διανομής του τελικού προϊόντος στην αγορά κ.ά. Η φάση αυτή θεωρείται σημαντική γιατί θέτει τα θεμέλια για όλες τις επόμενες. Το αποτέλεσμα είναι μια κατανοητή, πλήρης και σαφής περιγραφή χαρακτηριστικών και συμπεριφοράς της εφαρμογής ή, αλλιώς, οι προδιαγραφές της εφαρμογής.

2. **Σχεδιασμός.** Το σύστημα λογισμικού σχεδιάζεται στο σύνολό του, αναλύεται σε μέρη τα οποία επίσης σχεδιάζονται λεπτομερώς και ορίζεται η σύνδεση μεταξύ τους. (βλ. 2.1.4. Αναλυτική Μεθοδολογία επίλυσης προβλημάτων) Το αποτέλεσμα που προκύπτει είναι η συνολική δομή του συστήματος και οι προδιαγραφές όλων των λειτουργικών μονάδων του.
3. **Υλοποίηση των προγραμμάτων.** Γράφεται ο κώδικας κάθε λειτουργικής μονάδας ξεκινώντας από τη σχεδίαση και δημιουργία των φορμών επικοινωνίας χρήστη – υπολογιστή (διεπαφή) και τη διάταξή τους στην οθόνη. Ακολουθεί η κωδικοποίηση των αλγορίθμων και των δομών δεδομένων. Στο στάδιο αυτό εργάζονται κυρίως οι προγραμματιστές οι οποίοι συνεργάζονται και με άλλες ομάδες, όπως για παράδειγμα τους γραφίστες.
4. **Δοκιμή.** Οι λειτουργικές μονάδες γίνονται ένα σύνολο και ελέγχονται στο να πληρούν τις προδιαγραφές της ανάλυσης. Στο στάδιο αυτό ανιχνεύονται και διορθώνονται, κατά το δυνατόν, τα σφάλματα. Αποτέλεσμα είναι η παράδοση του προγράμματος στον εντολέα.
5. **Συντήρηση.** Η τελευταία φάση του κύκλου ζωής, η οποία έχει και τη μεγαλύτερη διάρκεια, περιλαμβάνει την έναρξη της λειτουργίας του προγράμματος, τη συντήρηση, υποστήριξη και αναβάθμισή του.



Εικόνα 2-33. Κύκλος ζωής λογισμικού

Όλα τα στάδια σχηματίζουν κλειστό κύκλο από τον οποίο φαίνεται ότι σε περίπτωση τροποποιήσεων θα πρέπει να εκτελεστούν από την αρχή ορισμένες ή και όλες οι φάσεις. Η μια φάση θα πρέπει να έχει ολοκληρωθεί απόλυτα για να ξεκινήσει η επόμενη.

Ένα παράδειγμα που θα μας δώσει να καταλάβουμε καλύτερα τον Κύκλο Ζωής του λογισμικού είναι η δημιουργία μια εφαρμογής πολυμέσων. Έστω ότι ένας Δήμος της χώρας μας θέλει να προβεί στη δημιουργία μιας εφαρμογής για την τουριστική προβολή της πόλης.

Όπως οποιοδήποτε νέο προϊόν, έτσι και μια εφαρμογή πολυμέσων ξεκινά συνήθως από μια έρευνα που θα καταγράψει τις επιθυμίες του κοινού (ομάδα στόχος). Μπορεί να είναι μια εφαρμογή που να περιέχει μόνο τα κυριότερα αξιοθέατα, χάρτες, ενδιαφέρουσες διαδρομές, πολιτιστικά στοιχεία του τόπου, διασκέδαση κ.λ.π. Τα αποτελέσματα της έρευνας θα καθορίσουν τις προδιαγραφές της νέας εφαρμογής. Στο στάδιο αυτό θα γίνει και ο χρονοπρογραμματισμός εργασιών αλλά και η εξασφάλιση πόρων (φάση ανάλυσης).

Για τη δημιουργία μιας πολυμεσικής εφαρμογής συνεργάζονται πάρα πολλές ειδικότητες, όπως ο διευθυντής παραγωγής (γενικός συντονιστής), ο ειδικός συλλογής υλικού, ο σχεδιαστής του περιβάλλοντος διεπαφής, οι προγραμματιστές και οι σεναριογράφοι. Το τμήμα σχεδιασμού,

έχοντας υπόψη τις προδιαγραφές, θα δημιουργήσει τη γενική δομή της εφαρμογής (διαγράμματα ροής, χάρτες πλοήγησης κ.λ.π.). Επιπλέον, θα καθορίσει και τις προδιαγραφές λειτουργικότητας (φάση σχεδιασμού).

Το τμήμα παραγωγής συγκεντρώνει και επεξεργάζεται το πολυμεσικό υλικό (κείμενο, γραφικά, ήχος, εικόνα, βίντεο) καθώς και την ενσωμάτωσή του στην εφαρμογή με βάση το σενάριο, όπως αυτό καθορίστηκε στη φάση της σχεδίασης (φάση υλοποίησης).

Η έτοιμη πλέον εφαρμογή θα πρέπει να περάσει από ένα τελικό έλεγχο της σωστής λειτουργία της. Με βάση τις παρατηρήσεις που θα γίνουν πραγματοποιείται η τελική εφαρμογή (φάση δοκιμής).

Το δοκιμασμένο πλέον λογισμικό είναι έτοιμο να παραχθεί και να δοθεί προς χρήση στους τελικούς αποδέκτες. Εισέρχεται λοιπόν στη φάση λειτουργίας και συντήρησης.

κεφάλαιο 3

Θέματα Εφαρμοσμένης Επιστήμης των Υπολογιστών

Λειτουργικά Συστήματα
Δίκτυα
Πληροφοριακά Συστήματα
Τεχνητή Νοημοσύνη

Στόχοι

Στόχοι του κεφαλαίου είναι οι μαθητές:

- Να συσχετίσουν έννοιες και τομείς της επιστήμης των υπολογιστών με τρόπους / προγράμματα επικοινωνίας και αλληλεπίδρασης με τον υπολογιστή και διαχείρισης των βασικών του μονάδων
- Να έρθουν σε επαφή με τρόπους διαχείρισης και ανάκτησης δεδομένων που χρησιμοποιούν σε καθημερινές τους δραστηριότητες
- Να γνωρίσουν θέματα δικτύων υπολογιστών και σύγχρονων υπηρεσιών τους
- Να προβληματιστούν και να αναγνωρίσουν εφαρμογές της επιστήμης υπολογιστών στον τομέα της τεχνητής νοημοσύνης
- Να εκτιμήσουν τα επιτεύγματα στο χώρο της εφαρμοσμένης επιστήμης των υπολογιστών

Λέξεις κλειδιά

Λειτουργικά Συστήματα	Λειτουργικό Σύστημα, φλοιός, πυρήνας, Διαχείριση, Διεργασία, Κεντρική Μονάδα Επεξεργασίας, Μνήμη, Μονάδα εισόδου – εξόδου, Σύστημα Αρχείων
Δίκτυα	Δίκτυο υπολογιστών, κατηγορίες δικτύων, συστήματα και συσκευές δικτύου, Τοπολογίες δικτύων, σύγχρονες υπηρεσίες διαδικτύου
Πληροφοριακά Συστήματα	Πληροφοριακό Σύστημα, Βάση Δεδομένων, Σύστημα Διαχείρισης Βάσεων Δεδομένων, Σχεσιακό μοντέλο δεδομένων
Τεχνητή Νοημοσύνη	Τεχνητή Νοημοσύνη, νευρωνικά δίκτυα, έμπειρα συστήματα, γλώσσες προγραμματισμού

Εισαγωγικές ερωτήσεις

1. Έχεις σκεφτεί τι κρύβει η επικοινωνία σου με τον υπολογιστή μετά το διπλοπάτημα ενός εικονιδίου/ εφαρμογής;
2. Ποια δίκτυα χρησιμοποιείς στις καθημερινές σου δραστηριότητες;
3. Γνωρίζεις πώς και που αποθηκεύονται τα δεδομένα που χρησιμοποιείς σε μια εφαρμογή όπως το Facebook;
4. Μπορεί μια μηχανή να σκεφτεί;
5. Τι είναι η ρομποτική;

3.1 Λειτουργικά Συστήματα

Στην ενότητα αυτή, ερχόμαστε σε επαφή με την έννοια του Λειτουργικού Συστήματος και περιγράφουμε τα βασικά χαρακτηριστικά του.

3.1.1 Η έννοια του Λειτουργικού Συστήματος

Όπως γνωρίζουμε από το γυμνάσιο, ένας υπολογιστής προκύπτει από τη σύνθεση δύο αλληλένδετων μερών:

1. Του **υλικού** (hardware), δηλαδή όλων εκείνων των στοιχείων ενός υπολογιστή που έχουν υλική υπόσταση όπως ο επεξεργαστής, η μνήμη, οι κάρτες επέκτασης οι μονάδες εισόδου/εξόδου κ.τ.λ.
2. Του **λογισμικού** (software), δηλαδή των προγραμμάτων που περιέχουν τις εντολές που ορίζουν πώς θα συμπεριφερθούν οι συσκευές του υπολογιστή και τι αποτελέσματα θα παράγουν.

Το λογισμικό, με τη σειρά του, μπορεί να διαιρεθεί σε δύο μεγάλες κατηγορίες: στα **προγράμματα συστήματος** και στα **προγράμματα εφαρμογών**. Το βασικότερο από τα προγράμματα συστήματος ενός υπολογιστή είναι το λειτουργικό σύστημα. Ας προσεγγίσουμε την έννοια του λειτουργικού συστήματος με ένα παράδειγμα:

Η λειτουργία ενός υπολογιστή μπορεί να παρομοιασθεί με τη λειτουργία ενός εστιατορίου. Σε ένα εστιατόριο οι πελάτες (χρήστες) δίνουν παραγγελίες (εντολές) στους σερβιτόρους (προγράμματα - εφαρμογές). Οι σερβιτόροι «περνούν» τις παραγγελίες στον σεφ (λειτουργικό σύστημα) ο οποίος είναι υπεύθυνος για τη σωστή εκτέλεση των συνταγών (αλγορίθμων), τη σωστή διαχείριση του ανθρώπινου δυναμικού (μονάδες επεξεργασίας) και τη σωστή διαχείριση των υλικών και των συσκευών (υπολογιστικοί πόροι). Όπως λοιπόν σε ένα εστιατόριο ο σεφ είναι ο βασικός υπεύθυνος για τη σωστή λειτουργία του, έτσι και σε έναν υπολογιστή το λειτουργικό σύστημα αποτελεί το βασικό πρόγραμμα (ή ομάδα προγραμμάτων) που συντονίζει το υλικό και το λογισμικό του υπολογιστή και φροντίζει για τη σωστή διαχείριση των πόρων του. Προχωρώντας λοιπόν σε ένα τυπικό ορισμό του λειτουργικού συστήματος μπορούμε να πούμε ότι:

Εικόνα 3-1. Επίεδα επικοινωνίας Χρήστη-Μηχανής

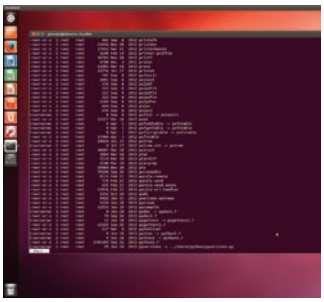
Το **Λειτουργικό Σύστημα (Λ.Σ.)** είναι ένα σύνολο προγραμμάτων που λειτουργεί ως σύνδεσμος ανάμεσα στο υλικό και το λογισμικό, ελέγχει όλους τους πόρους του υπολογιστή φροντίζοντας για την αποδοτική χρήση τους, και παρέχει τη βάση πάνω στην οποία θα αναπτυχθούν τα προγράμματα εφαρμογών.

Συνεχίζοντας με το παράδειγμα του εστιατορίου, παρατηρούμε τα εξής: Οι συνταγές ενός εστιατορίου δεν αναφέρουν σε ποια συγκεκριμένη κουζίνα θα εκτελεστούν, αλλά θεωρούν ότι υπάρχει ένα «βασικό περιβάλλον» όπου θα μπορούν να υλοποιηθούν. Υπεύθυνος για την εκτέλεσή τους είναι ο σεφ που γνωρίζει τις συσκευές και τη λειτουργία τους καθώς και τον αριθμό των υπαλλήλων που έχει. Κατ' αναλογία, τα προγράμματα που εκτελούνται σε έναν υπολογιστή δε γνωρίζουν τις ακρι-

βείς δυνατότητες του μηχανήματος, αλλά θεωρούν ότι υπάρχει ένα «βασικό περιβάλλον» το οποίο τους παρέχει όλα τα απαραίτητα στοιχεία για να παράγουν τα ζητούμενα αποτελέσματα. Υπεύθυνος για την εκτέλεσή τους είναι το Λ.Σ. που γνωρίζει τους πόρους που διαθέτει ο υπολογιστής και μπορεί να τους διαχειριστεί κατάλληλα. Σε ένα εστιατόριο με πολλούς υπαλλήλους κουζίνας μπορούν πολλές συνταγές να ετοιμάζονται ταυτόχρονα με την εποπτεία του σεφ που δίνει εντολές. Αντίστοιχα, σε ένα υπολογιστικό σύστημα με πολλούς επεξεργαστές ή πολλούς πυρήνες επεξεργαστών, πολλά προγράμματα μπορούν να εκτελούνται ταυτόχρονα κάτω από την εποπτεία του λειτουργικού συστήματος που ορίζει ποιο πρόγραμμα θα εκτελεστεί σε ποιον επεξεργαστή, πώς θα κατανεμηθούν οι υπόλοιποι πόροι και με ποια σειρά θα παραχθούν τα αποτελέσματα των προγραμμάτων.

Εικόνα 3-2. Το γραφικό περιβάλλον των Windows

3.1.2 Βασικές εργασίες του Λειτουργικού Συστήματος


Εικόνα 3-3. Διερμηνευτής εντολών του LINUX

Ένα Λ.Σ. είναι μια σύνθετη ομάδα προγραμμάτων που επιτελούν μια διπλή λειτουργία: από τη μια πλευρά έρχονται σε επαφή με τους χρήστες του υπολογιστή δίνοντάς τους τη δυνατότητα να εκτελέσουν εφαρμογές και να δώσουν εντολές προς το υλικό, ενώ, από την άλλη, διαχειρίζονται τους πόρους του υπολογιστή και φροντίζουν για την αποδοτικότερη χρήση τους. Η πρώτη λειτουργία υλοποιείται μέσω του **φλοιού** (shell) του λειτουργικού συστήματος, ενώ η δεύτερη μέσω του **πυρήνα** (kernel).

Τα κυριότερα στοιχεία του φλοιού είναι το γραφικό περιβάλλον χρήστη (Graphical User Interface) και ο διερμηνευτής εντολών (command shell). Όλα τα σύγχρονα Λ.Σ. έχουν εντυπωσιακά γραφικά περιβάλλοντα προσπαθώντας να προσελκύσουν περισσότερους χρήστες. Συνήθως μπερδεύουμε την έννοια του Λ.Σ. με το γραφικό του περιβάλλον γιατί με αυτό ερχόμαστε σε επαφή, παρόλο που, στην πραγματικότητα, επιτελεί δευτερεύουσες λειτουργίες. Το καθοριστικό στοιχείο που προσδίδει τα χαρακτηριστικά ενός Λ.Σ. είναι ο πυρήνας του.

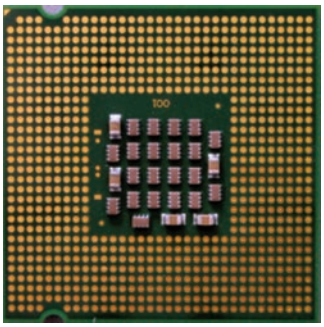
Εικόνα 3-4. Σχέση Φλοιού - Πυρήνα

Στον πυρήνα ενός λειτουργικού συστήματος συναντάμε τέσσερα τμήματα τα οποία διαχειρίζονται την Κεντρική Μονάδα Επεξεργασίας, τη Μνήμη, το Σύστημα Αρχείων και τις συσκευές εισόδου/εξόδου αντίστοιχα. Ας δούμε λοιπόν αυτά τα τέσσερα τμήματα του πυρήνα ενός λειτουργικού συστήματος πιο αναλυτικά:

Διαχείριση της Κεντρικής Μονάδας Επεξεργασίας

Η κεντρική μονάδα επεξεργασίας, ή αλλιώς επεξεργαστής, αποτελεί τον πιο κρίσιμο πόρο ενός υπολογιστή και θα θέλαμε να τον χρησιμοποιούμε όσο πιο αποδοτικά γίνεται. Με ποιον τρόπο μπορούμε να εκμεταλλευτούμε καλύτερα τις δυνατότητες της κεντρικής μονάδας επεξεργασίας; Ας επιστρέψουμε στο παράδειγμα του εστιατορίου: Σε ένα εστιατόριο έρχονται ταυτόχρονα στη κουζίνα πολλές παραγγελίες. Ας υποθέσουμε ότι ο σεφ έχει μόνον έναν υπάλληλο στη κουζίνα, στον οποίο δίνει εντολή να ασχολείται με μία συνταγή κάθε φορά και να μην προχωρά στην επόμενη μέχρι να ολοκληρώσει την παλιά. Προφανώς,

αυτός ο τρόπος διαχείρισης των παραγγελιών είναι εξαιρετικά αργός και σύντομα οι πελάτες θα αρχίσουν να διαμαρτύρονται. Για να αυξήσει ο σεφ την απόδοση της κουζίνας, διατάζει τον υπάλληλό του να ασχολείται για μικρά χρονικά διαστήματα με κάθε παραγγελία και να εκμεταλλεύεται τους «νεκρούς» χρόνους· όσο, για παράδειγμα, μία συνταγή είναι στο φούρνο, να προετοιμάζει την επόμενη. Με τον τρόπο αυτό αυξάνεται δραματικά η ταχύτητα διεκπεραίωσης των παραγγελιών και οι πελάτες μένουν ευχαριστημένοι. Με ανάλογο τρόπο πράττει και ένα Λ.Σ.: χωρίζει το χρόνο σε μικρά τμήματα και διαθέτει κυκλικά σε κάθε πρόγραμμα που πρέπει να εκτελεστεί ένα τμήμα χρόνου. Έτσι, ενώ ο επεξεργαστής εκτελεί κάθε χρονική στιγμή ένα μόνο πρόγραμμα, σε διάστημα ενός δευτερολέπτου έχει εκτελέσει πολλά διαφορετικά προγράμματα δίνοντάς μας την ψευδαίσθηση του παραλληλισμού. Οι σύγχρονοι επεξεργαστές επιτυγχάνουν και πραγματικό παραλληλισμό καθώς έχουν πολλούς επεξεργαστικούς πυρήνες στο ίδιο ολοκληρωμένο κύκλωμα. Η αντιστοιχία με το παράδειγμα του εστιατορίου θα ήταν ο σεφ να είχε πολλούς υπαλλήλους στη κουζίνα οι οποίοι ταυτόχρονα να μπορούν να ετοιμάζουν πολλές διαφορετικές παραγγελίες.



Εικόνα 3-5. Ολοκληρωμένο κύκλωμα επεξεργαστή

Στην ορολογία των λειτουργικών συστημάτων κάθε εκτελέσιμο πρόγραμμα ονομάζεται **διεργασία**. Μια διεργασία λοιπόν είναι ένα πρόγραμμα ή ένα αυτόνομο τμήμα προγράμματος που εκτελείται ή περιμένει να εκτελεστεί από την κεντρική μονάδα επεξεργασίας. Η συχνή εναλλαγή διεργασιών στην ΚΜΕ με σκοπό την αποδοτικότερη χρήση της ονομάζεται **πολυεπεξεργασία**. Το τμήμα του λειτουργικού συστήματος που ασχολείται με την επιλογή των διεργασιών και των χρόνων εκτέλεσης αυτών ονομάζεται **χρονοδρομολογητής**. Ας δούμε ένα παράδειγμα: έχουμε τρεις διεργασίες που χρειάζονται χρόνο στην ΚΜΕ, στο υποσύστημα της κάρτας γραφικών και στο δίσκο. Στη σειριακή εκτέλεση η πρώτη διεργασία τελειώνει τη χρονική στιγμή 6, η δεύτερη τη χρονική στιγμή 11 και η τρίτη τη χρονική στιγμή 15, ενώ η ΚΜΕ ελευθερώνεται για πιθανή επόμενη διεργασία μόνον όταν τελειώνει και η τρίτη διεργασία δηλαδή μετά τη χρονική στιγμή 15.

Σύστημα	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ΚΜΕ	Δ1	Δ1					Δ2	Δ2				Δ3			
Γραφικά			Δ1	Δ1					Δ2				Δ3		
Δίσκος					Δ1	Δ1				Δ2	Δ2			Δ3	Δ3

Εικόνα 3-6: Σειριακή εκτέλεση διεργασιών

Σύστημα	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ΚΜΕ	Δ1	Δ2	Δ3	Δ1	Δ2										
Γραφικά				Δ3	Δ1	Δ1	Δ2								
Δίσκος					Δ3	Δ3	Δ1	Δ1	Δ2	Δ2					

Εικόνα 3-7: Εκτέλεση με καταμερισμό χρόνου

Στην πολυεπεξεργασία οι διεργασίες εκτελούνται κυκλικά στην ΚΜΕ για μια μόνο χρονική στιγμή κάθε φορά. Με τον τρόπο αυτό επιτυγχάνουμε καλύτερη χρήση της καθώς από τη χρονική στιγμή 6 η ΚΜΕ είναι

ήδη ελεύθερη για πιθανή επόμενη διεργασία, ενώ συνολικά και οι τρείς διεργασίες έχουν τελειώσει ήδη από τη χρονική στιγμή 10.

«Φόρτωση» προγραμμάτων
Όταν κάνουμε διπλό κλικ στο εικονίδιο ενός προγράμματος, ο υπολογιστής καθυστερεί κάποια δευτερόλεπτα να εμφανίσει το περιβάλλον του προγράμματος στην οθόνη. Η καθυστέρηση αυτή οφείλεται στη μεταφορά των απαραίτητων δεδομένων από τη δευτερεύουσα στην κύρια μνήμη του συστήματος. Τη μεταφορά αυτή την αναλαμβάνει το Λ.Σ. Εάν κλείσετε το πρόγραμμα και το ξανανοίξετε αμέσως τότε η καθυστέρηση μειώνεται σημαντικά διότι κάποια δεδομένα υπάρχουν ήδη στην κύρια μνήμη του συστήματος από την προηγούμενη εκτέλεση.

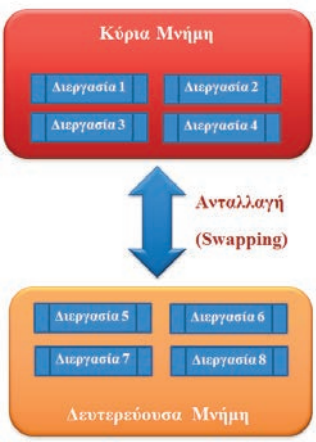
Διαχείριση της Μνήμης

Το σύστημα μνήμης των υπολογιστών βασίζεται σε μια αυστηρή ιεραρχική οργάνωση η οποία περιλαμβάνει:

1. Τους **καταχωρητές** (registers) που είναι κελιά μνήμης ενσωματωμένα στο κύκλωμα της ΚΜΕ.
2. Τη **λανθάνουσα μνήμη** (cache memory) η οποία βρίσκεται ενσωματωμένη στο ολοκληρωμένο του επεξεργαστή.
3. Την **κύρια μνήμη** (main memory) ή μνήμη RAM η οποία αποτελεί ξεχωριστό άρθρωμα της μητρικής πλακέτας.
4. Τη **δευτερεύουσα μνήμη** (secondary memory) η οποία αντιστοιχεί κυρίως στο σκληρό δίσκο.

Εικόνα 3-8. Άρθρωμα Μνήμης RAM

Όσο απομακρυνόμαστε από την ΚΜΕ τόσο χαμηλώνει η ταχύτητα της μνήμης αλλά επίσης τόσο μεγαλύτερα ποσά διαθέσιμης μνήμης έχουμε. Η μνήμη που βρίσκεται κοντά στην ΚΜΕ είναι πολύ γρήγορη αλλά συνάμα πολύ ακριβή. Γι' αυτό το λόγο είναι σχετικά μικρή σε μέγεθος. Η πρόκληση για τους επιστήμονες της πληροφορικής είναι να διατηρούν τα δεδομένα που χρησιμοποιεί συχνά μια διεργασία όσο το δυνατόν πιο κοντά στην ΚΜΕ ώστε να επιτυγχάνουν υψηλές ταχύτητες εκτέλεσης των προγραμμάτων. Το πρόβλημα γίνεται ακόμη πιο περίπλοκο διότι, όπως είδαμε στην προηγούμενη ενότητα, την ίδια χρονική στιγμή πολλές διεργασίες βρίσκονται υπό εκτέλεση και, κατά συνέπεια, μοιράζονται τη μνήμη ενός υπολογιστικού συστήματος. Αυτή τη δύσκολη εργασία προσπαθεί να διεκπεραιώσει το τμήμα του λειτουργικού συστήματος που ασχολείται με τη διαχείριση μνήμης.



Εικόνα 3-9. Ανταλλαγή Διεργασιών

Όταν δε χωρούν όλες οι υπό εκτέλεση διεργασίες στην κύρια μνήμη ενός υπολογιστή, τα σύγχρονα λειτουργικά συστήματα κρατούν συνήθως έναν αριθμό διεργασιών στην κύρια μνήμη και τις υπόλοιπες τις αποθηκεύουν στη δευτερεύουσα μνήμη. Όταν φτάνει η σειρά μιας διεργασίας που είναι αποθηκευμένη στη δευτερεύουσα μνήμη να εκτελεστεί, τότε συμβαίνει το φαινόμενο της **ανταλλαγής** (swapping). Το Λ.Σ. επιλέγει μια διεργασία που βρίσκεται στην κύρια μνήμη με κάποιον αλγόριθμο και τη μεταφέρει στη δευτερεύουσα, ώστε να ελευθερώσει χώρο για τη διεργασία που πρόκειται να εκτελεστεί. Αυτή η ανταλλαγή διεργασιών δημιουργεί σοβαρές χρονικές καθυστερήσεις και οι σχεδιαστές των λειτουργικών συστημάτων αναζητούν τρόπους ώστε να γίνεται όσο το δυνατόν πιο σπάνια.

Διαχείριση του Συστήματος Αρχείων

Λειτουγικό Σύστημα	Συστήματα αρχείων
Windows	NTFS, FAT32
Linux	Ext4, BTRFS
Mac OS X	HFS+

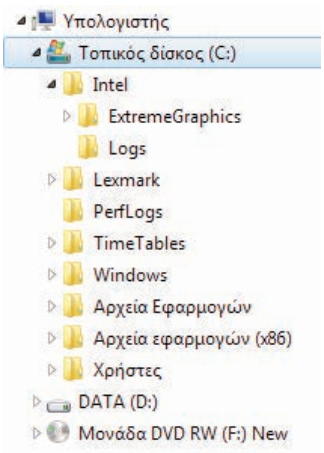
Εικόνα 3-10. Ονομασίες Συστημάτων Αρχείων

Κάθε υπολογιστική μηχανή έχει κάποιο είδος μνήμης μόνιμης αποθήκευσης όπου διατηρεί τα δεδομένα των προγραμμάτων που εκτελεί. Όπως σε μια τακτοποιημένη βιβλιοθήκη δεν υπάρχουν ανακατεμένα έγγραφα, φάκελοι και βιβλία σε τυχαίες θέσεις, έτσι και στη μνήμη μόνιμης αποθήκευσης του υπολογιστή τα δεδομένα ακολουθούν μια συγκεκριμένη οργάνωση, ώστε να μπορούν να ανακτώνται εύκολα όταν ζητηθούν από τους χρήστες ή από τον ίδιο τον υπολογιστή. Υπεύθυνο για την οργάνωση των δεδομένων είναι το Λ.Σ.

Η βασική μονάδα οργάνωσης δεδομένων σε όλα τα λειτουργικά συστήματα είναι το **αρχείο** το οποίο περιέχει συγκεντρωμένα ομοειδή δεδομένα (εντολές προγραμμάτων, κείμενο, δεδομένα πολυμέσων κτλ). Για το λόγο αυτό, αναφερόμαστε σε σύστημα αρχείων όταν θέλουμε να μιλήσουμε για τον τρόπο με τον οποίο διαχειρίζεται τα δεδομένα του υπολογιστή ένα Λ.Σ.. Κάθε αρχείο, καταλαμβάνει κάποιον αριθμό από ψηφιακές λέξεις (bytes) στη μνήμη μόνιμης αποθήκευσης του υπολογιστή. Το Λ.Σ. χωρίζει τα μέσα αποθήκευσης σε μικρά **μπλοκ** (clusters) που μπορεί να έχουν μέγεθος από 512 bytes έως μερικά kilobytes. Ανάλογα με το μέγεθος του αρχείου που διαχειρίζεται, το Λ.Σ. αποθηκεύει το αρχείο σε έναν αριθμό από συνεχόμενα μπλοκ μνήμης μέχρι να καλύψει το μέγεθός του. Κάθε αρχείο έχει ένα όνομα και συνήθως μια κατάληξη που δείχνει το είδος των δεδομένων που περιέχει. Το Λ.Σ. αποθηκεύει ένα σύνολο πληροφοριών για κάθε αρχείο, όπως την ημερομηνία δημιουργίας του, την ημερομηνία τροποποίησής του, το όνομα χρήστη που το δημιούργησε κτλ, και μας παρέχει όλες εκείνες τις εντολές για να μπορούμε να δημιουργούμε, να τροποποιούμε, να μετονομάζουμε, να αντιγράφουμε και να διαγράφουμε αρχεία.

Δημιουργήστε ένα αρχείο κειμένου στον υπολογιστή σας και παρατηρήστε ότι, όσο μικρό και αν είναι, το μέγεθός του στον δίσκο είναι 4KB. Αυτό συμβαίνει γιατί το μέγεθος μπλοκ στα Windows είναι 4KB!

Άλλο ένα ισχυρό εργαλείο που παρεχε το Λ.Σ. για την οργάνωση των αρχείων, είναι ο **φάκελος** ή **κατάλογος** (folder ή directory). Ο φάκελος εξομοιώνει τους φακέλους οργάνωσης εγγράφων ενός γραφείου. Περιέχει δηλαδή ένα πλήθος από καταχωρήσεις που αναφέρονται σε αρχεία ή άλλους φακέλους και μας βοηθά στο να έχουμε οργανωμένα τα αρχεία του συστήματός μας. Σχηματικά, τα συστήματα αρχείων ακολουθούν μια ανεστραμμένη δενδρική δομή: Στη ρίζα του δέντρου βρίσκεται ο βασικός φάκελος του μέσου αποθήκευσης που οργανώνουμε και μέσα σε αυτόν υπάρχουν αρχεία ή άλλοι φάκελοι που με τη σειρά τους περιέχουν νέους φακέλους ή αρχεία κ.ο.κ. Το Λ.Σ. μας παρέχει επίσης ένα σύνολο εντολών για να μπορούμε να διαχειριζόμαστε φακέλους ανάλογα με τις ανάγκες μας.



Εικόνα 3-11. Δενδρική δομή φακέλων

Διαχείριση των Εισόδων και Εξόδων

Ένας υπολογιστής χωρίς συσκευές εισόδου και εξόδου θα μας ήταν ουσιαστικά άχρηστος καθώς δε θα γινόταν εφικτή η αλληλεπίδραση μαζί του. Για το λόγο αυτό, όταν αναφερόμαστε στους υπολογιστές, εννοούμε και όλες εκείνες τις περιφερειακές συσκευές όπως οθόνη, πληκτρολόγιο, ποντίκι, εκτυπωτές κτλ. που είναι υπεύθυνες για την είσοδο ή την έξοδο

δεδομένων προς και από τον υπολογιστή. Ένα από τα πολλά καθήκοντα του λειτουργικού συστήματος, είναι να ελέγχει όλες αυτές τις συσκευές εισόδου/εξόδου.

Ο έλεγχος αυτός γίνεται μέσω κατάλληλων προγραμμάτων που ονομάζονται **οδηγοί** (drivers). Τα μικρά αυτά προγράμματα, είναι απαραίτητα για τη λειτουργία των συσκευών και είτε βρίσκονται ενσωματωμένα μέσα στο Λ.Σ., είτε μας παρέχονται με την αγορά των περιφερειακών συσκευών και πρέπει να τα εγκαταστήσουμε στον υπολογιστή μας. Το Λ.Σ. συνεργάζεται με τον οδηγό της κάθε συσκευής και επικοινωνεί μαζί του μέσω μιας συμφωνημένης από πριν διεύθυνσης μνήμης ή ενός σήματος διακοπής (interrupt) όπου το Λ.Σ. έχει δεσμεύσει αποκλειστικά για τη συγκεκριμένη συσκευή.

3.1.3 Γνωστά Λειτουργικά Συστήματα

Στην αγορά υπολογιστών υπάρχει ένας σημαντικός χωρισμός των λειτουργικών συστημάτων σε δύο βασικές κατηγορίες: Στα δωρεάν, ανοικτού κώδικα, λειτουργικά συστήματα και στα εμπορικά. Το πιο γνωστό από όλα τα λειτουργικά συστήματα ανοικτού κώδικα ονομάζεται **LINUX** από το συνδυασμό του ονόματος του Φινλανδού δημιουργού του Linus Torvalds και του λειτουργικού από το οποίο προήλθε, το UNIX. Το LINUX κυκλοφορεί σε πολλές διανομές, δηλαδή έτοιμα πακέτα που περιέχουν, εκτός από τον πυρήνα, το γραφικό περιβάλλον του λειτουργικού καθώς και πολλά άλλα βοηθητικά προγράμματα, (Ubuntu, Fedora, Debian κ.α). Υπάρχουν δεκάδες ακόμη λειτουργικά ανοικτού κώδικα, ενδεικτικά αναφέρουμε το **FreeBSD**.

Κατηγοριοποίηση Λειτουργικών Συστημάτων			
Παραδοσιακοί υπολογιστές	Εμπορικά		Windows, MAC OS X, Solaris
	Δωρεάν	Linux	Ubuntu, Fedora, openSUSE, Debian
		Υπόλοιπα	FreeBSD, Haiku, KolibriOS
Κινητές συσκευές	Εμπορικά		iOS, Windows Phone
	Ανοικτού Κώδικα		Android
Ενσωματωμένα (embedded) συστήματα			LynxOS, VxWorks, FreeRTOS

Από την άλλη πλευρά έχουμε τα εμπορικά λειτουργικά συστήματα με τον κυριότερο εκπρόσωπο αυτής της κατηγορίας να είναι η οικογένεια των λειτουργικών συστημάτων **Windows**. Τα Windows προέκυψαν από την εξέλιξη του λειτουργικού συστήματος DOS τη δεκαετία του ’80. Άλλα Λ.Σ. αυτής της κατηγορίας είναι το **MAC OS X** και, για μεγάλα υπολογιστικά συστήματα, το **Solaris**.

Ειδικού σκοπού Λ.Σ. συναντάμε σε ενσωματωμένα (embedded) υπολογιστικά συστήματα με ιδιαίτερα χαρακτηριστικά, όπως το μικρό μέγεθος και η υψηλή ταχύτητα απόκρισης.

Λειτουργικά συστήματα δε συναντάμε μόνο στους παραδοσιακούς υπολογιστές γραφείου, αλλά σε όλες τις συσκευές που έχουν επεξεργαστική ισχύ όπως σε κινητά τηλέφωνα, σε υπολογιστές παλάμης, σε παι-

χνιδομηχανές κτλ. Η αγορά αυτή, κυριαρχείται από τρία κυρίως λειτουργικά συστήματα, το **iOS** της Apple, το **Android** της Google και το **Windows Phone** της Microsoft.

Ασκήσεις – Προβλήματα

1. Αναζητήστε πληροφορίες για την εικονική μνήμη (virtual memory). Πότε και πώς τη χρησιμοποιεί ένα Λ.Σ.;
2. Φτιάξτε μια λίστα με τις βασικές εντολές διαχείρισης αρχείων του διερμηνευτή εντολών (command shell) του LINUX.
3. Ανοίξτε τη διαχείριση εργασιών των Windows και περιηγθείτε στις καρτέλες της εφαρμογής. Ποιά είναι η διαφορά των εφαρμογών από τις διεργασίες;
4. Συζητήστε στην τάξη με ποιο τρόπο εγκαθιστούμε νέα προγράμματα στα λειτουργικά της οικογένειας windows και με ποιο σε εκδόσεις του λειτουργικού linux. Πειραματιστείτε στο εργαστήριο του σχολείου.
5. Κατασκευάστε έναν πίνακα στον οποίο θα συνοψίζετε τα κύρια χαρακτηριστικά των τριών πιο διαδεδομένων λειτουργικών συστημάτων για κινητά τηλέφωνα.

Όταν ολοκληρώσετε την ενότητα ελέγξτε αν είστε σε θέση:

- Να περιγράφετε τις βασικές αρμοδιότητες ενός Λ.Σ.
- Να εξηγείτε τον τρόπο με τον οποίο τα λειτουργικά συστήματα διαχειρίζονται τους πόρους ενός υπολογιστικού συστήματος.
- Να ονομάζετε γνωστά λειτουργικά συστήματα και να διακρίνετε τις κύριες διαφορές τους.

3.2 Δίκτυα

Η ταχύτητα μετάδοσης των δεδομένων σε ένα δίκτυο μετριέται σε bit ανα δευτερόλεπτο (bit per second - bps) και στα πολλαπλάσιά του.
1 Kbps = 10^3 bps
1 Mbps = 10^6 bps
1 Gbps = 10^9 bps

Τα δίκτυα υπολογιστών είναι ένας σημαντικός τομέας της εφαρμοσμένης επιστήμης των υπολογιστών και σε αυτή την ενότητα γνωρίζουμε τα βασικά τους στοιχεία, τις κατηγορίες τους, και τις σύγχρονες υπηρεσίες που μας παρέχουν.

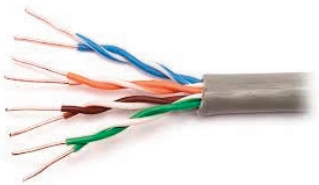
3.2.1 Ορισμός δικτύου Η/Υ

Τον όρο δίκτυο τον συναντάμε συχνά σε πολλές ανθρώπινες δραστηριότητες. Αναφερόμαστε, για παράδειγμα, σε δίκτυο ύδρευσης, εννοώντας τον τρόπο μεταφοράς νερού στα σπίτια μας, δίκτυο ηλεκτρισμού, αναφερόμενοι στον τρόπο μεταφοράς της ηλεκτρικής ενέργειας και οδικό δίκτυο, εννοώντας τους δρόμους μιας περιοχής. Με ανάλογο τρόπο, όταν θέλουμε να αναφερθούμε στη διασύνδεση υπολογιστών χρησιμοποιούμε τον όρο Δίκτυο Η/Υ.

Δίκτυο υπολογιστών ονομάζεται η σύνδεση δύο ή περισσότερων υπολογιστών με σκοπό την ανταλλαγή δεδομένων και τη διαμοίραση πόρων.

Στον παραπάνω ορισμό ο όρος υπολογιστής δεν αναφέρεται αποκλειστικά στους «παραδοσιακούς υπολογιστές», αλλά περιλαμβάνει οποιαδήποτε συσκευή έχει επεξεργαστική ισχύ. Ολοένα και περισσότερες συσκευές ενσωματώνουν σήμερα μικρούς «υπολογιστές» που τους δίνουν τη δυνατότητα να συνδέονται και να ανταλλάσουν δεδομένα. Οι συσκευές αυτές όταν συνδέονται σε ένα δίκτυο ονομάζονται **κόμβοι**.

3.2.2 Στοιχεία δικτύου



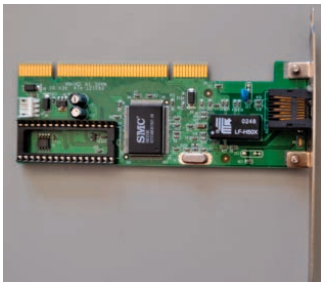
Εικόνα 3-12. Χαλκινα ζεύγη καλωδίων

Εικόνα 3-13. Οπτικές ίνες

Ένα δίκτυο υπολογιστών αποτελείται από πολλά δομικά στοιχεία που επιτρέπουν την διασύνδεση και την ανταλλαγή δεδομένων μεταξύ των κόμβων του. Τα κυριότερα από τα δομικά αυτά στοιχεία είναι:

- **Γραμμές μετάδοσης**. Για να κατασκευάσουμε ένα δίκτυο, θα χρειαστούμε οπωσδήποτε έναν τρόπο για να διασυνδέσουμε τα υπολογιστικά συστήματα που μετέχουν σ' αυτό. Οι γραμμές μετάδοσης ενός δικτύου συνήθως αποτελούνται από χάλκινα καλώδια ή οπτικές ίνες. Χρησιμοποιούμε τα χάλκινα συνεστραμμένα ζεύγη καλωδίων κυρίως λόγω του χαμηλού κόστους, αλλά και της ήδη μεγάλης εγκατεστημένης βάσης τέτοιων καλωδίων σε παγκόσμιο επίπεδο (όλα τα τηλεφωνικά καλώδια είναι αυτού του είδους). Οι οπτικές ίνες από την άλλη πλευρά έχουν σαφή πλεονεκτήματα όπως οι μεγάλες ταχύτητες, η υψηλή χωρητικότητα καναλιών επικοινωνίας, αλλά και η διάδοση των σημάτων σε μεγάλες αποστάσεις χωρίς αλλοιώσεις. Για το λόγο αυτό οι τηλεφωνικές εταιρίες έχουν αντικαταστήσει, σε μεγάλο βαθμό, το κεντρικό τους δίκτυο με οπτικές ίνες και έχουν αφήσει μόνο το τελευταίο κομμάτι της σύνδεσης των κτηρίων με τον κεντρικό κορμό του δικτύου σε χάλκινα καλώδια.
- **Κάρτες δικτύου** (network interface controllers – NICs). Οι κάρτες

δικτύου είναι το μέρος εκείνο του υλικού ενός υπολογιστή που του δίνει τη δυνατότητα να συνδεθεί σε ένα δίκτυο και να ανταλλάξει δεδομένα. Οι κάρτες δικτύου μπορεί να είναι τόσο ξεχωριστές ηλεκτρονικές συσκευές όσο και ενσωματωμένες στο υπάρχον υλικό της υπολογιστικής συσκευής. Κάθε κάρτα δικτύου έχει μια μοναδική παγκόσμια διεύθυνση που ονομάζεται διεύθυνση MAC. Ανάλογα με το μέσο μετάδοσης στο οποίο συνδέονται οι κάρτες δικτύου χωρίζονται σε ενσύρματες και ασύρματες.



Εικόνα 3-14. Κάρτα Δικτύου

- **Διανομείς** (Hubs/ Switches). Όταν το δίκτυό μας έχει περισσότερους από δύο υπολογιστές, χρησιμοποιούμε αυτές τις συσκευές ώστε να μπορεί κάθε υπολογιστής να λαμβάνει και να μεταδίδει δεδομένα προς όλους τους υπόλοιπους. Η διαφορά των hubs από τα switches είναι ότι τα hubs όταν λαμβάνουν δεδομένα από έναν υπολογιστή, τα μεταδίδουν προς όλους τους υπόλοιπους με τους οποίους είναι συνδεδεμένα, ενώ τα switches δρουν πιο «έξυπνα» στέλνοντας τα δεδομένα μόνον προς τον υπολογιστή που αυτά προορίζονται.
- **Ασύρματα σημεία πρόσβασης** (Wireless acess points). Οι συσκευές αυτές συνδέουν μεταξύ τους ασύρματες υπολογιστικές συσκευές βοηθώντας έτσι στη δημιουργία ασύρματων δικτύων. Κάθε τέτοια συσκευή συνδέεται συνήθως ενσύρματα σε ένα τοπικό δίκτυο το οποίο στη συνέχεια το μοιράζει ασύρματα στις υπόλοιπες ασύρματες συσκευές του δικτύου.
- **Επαναλήπτες** (Repeaters). Όταν το μήκος των καλωδίων ενός δικτύου είναι αρκετά μεγάλο (πάνω από 100 μέτρα), τα μεταφερόμενα σήματα εξασθενούν ώστε δεν είναι πλέον αξιόπιστη η λήψη τους. Για το λόγο αυτό, ανάμεσα σε κόμβους που βρίσκονται σε μεγάλες αποστάσεις, παρεμβάλλονται επαναλήπτες οι οποίοι λαμβάνουν το σήμα, το καθαρίζουν από τον ανεπιθύμητο θόρυβο και το αναμεταδίδουν ενισχυμένο προς τον επόμενο κόμβο ή επαναλήπτη.

Εικόνα 3-15. Διανομείς (switches)

Ένα οικιακό ADSL Modem/Router
Εάν έχουμε πρόσβαση στο διαδίκτυο από το σπίτι μας, τότε, κατά πάσα πιθανότητα, έχουμε μια σύνθετη δικτυακή συσκευή που ονομάζεται ADSL Modem/Router. Η συσκευή αυτή συνδυάζει 4 συσκευές από αυτές που περιγράψαμε πιο πάνω:
1. Είναι διαμορφωτής/αποδιαμορφωτής (Modem) εφόσον συνδέεται στην τηλεφωνική μας γραμμή και μετατρέπει τα ψηφιακά δεδομένα που λαμβάνουμε ή στέλνουμε, σε αναλογικά σήματα.
2. Είναι δρομολογητής διότι περιέχει στοιχεία που το βοηθούν να επιλέξει την κατάλληλη διαδρομή για τα δεδομένα που φεύγουν από το οικιακό μας δίκτυο (πχ. όταν πληκτρολογούμε τη διεύθυνση μιας ιστοσελίδας, ο δρομολογητής γνωρίζει πού θα στείλει την αίτηση για να λάβουμε τις πληροφορίες που ζητήσαμε).
3. Λειτουργεί ως διανομέας (switch) στο τοπικό μας δίκτυο καθώς μπορούμε συνήθως να συνδέσουμε ενσύρματα έως και τέσσερις υπολογιστές με αυτό.
4. Τέλος, λειτουργεί και ως ασύρματο σημείο πρόσβασης καθώς πάνω στη συσκευή αυτή μπορούν να συνδεθούν οι ασύρματες συσκευές του σπιτιού μας και να έχουν πρόσβαση στο διαδίκτυο

- **Δρομολογητές** (Routers). Όλα τα δεδομένα που περνούν πάνω από ένα δίκτυο είναι χωρισμένα σε πακέτα με το κάθε ένα από αυτά να περιέχει τόσο τη διεύθυνση του αποστολέα όσο και τη διεύθυνση του

παραλήπτη. Οι δρομολογητές αναλαμβάνουν το έργο της προώθησης των πακέτων αυτών προς τον τελικό παραλήπτη επιλέγοντας κάθε φορά τη βέλτιστη διαδρομή.

Εικόνα 3-16. ADSL Modem/Router

- **Διαμορφωτές/Αποδιαμορφωτές** (Modems). Για να μπορέσουμε να μεταδώσουμε ψηφιακά δεδομένα πάνω από τα παλιά χάλκινα καλώδια των τηλεφωνικών δικτύων, χρειαζόμαστε μια συσκευή που να μετατρέπει-διαμορφώνει τα ψηφιακά σήματα των υπολογιστών σε αναλογικά σήματα τα οποία μπορούν να ταξιδέψουν μέσα από τα τηλεφωνικά καλώδια. Επίσης όταν τα αναλογικά δεδομένα φτάσουν στον προορισμό τους, πρέπει πάλι να μετατραπούν-αποδιαμορφωθούν σε ψηφιακά. Οι συσκευές αυτές ονομάζονται διαμορφωτές/αποδιαμορφωτές ή στα αγγλικά, όπως είναι πιο γνωστές, MODEM.

3.2.3 Κατηγορίες Δικτύων

Με κριτήριο το μέσο μετάδοσης των πληροφοριών, τα δίκτυα Η/Υ χωρίζονται σε:

- **Ενσύρματα δίκτυα**. Το μέσο μετάδοσης είναι κάποιο είδος καλωδίου (χάλκινα σύρματα, οπτικές ίνες).
- **Ασύρματα δίκτυα**. Το μέσο μετάδοσης είναι η ηλεκτρομαγνητική ακτινοβολία (WiFi, δίκτυα κινητών τηλεφώνων, δορυφορικά δίκτυα).

"Σερφάρισμα" στο διαδίκτυο

Όλοι έχουμε πληκτρολογήσει μια ηλεκτρονική διεύθυνση ενός ιστότοπου σε έναν φυλλομετρητή. Μετά από λίγα δευτερόλεπτα η σελίδα που ζητήσαμε εμφανίζεται στην οθόνη του υπολογιστή μας. Πώς όμως ο υπολογιστής μας γνωρίζει με ποιον απομακρυσμένο υπολογιστή θα συνδεθεί για να μας δείξει τις πληροφορίες που του ζητήσαμε; Κάθε υπολογιστής που είναι συνδεδεμένος στο διαδίκτυο έχει και μία μοναδική διεύθυνση που ονομάζεται διεύθυνση IP (Internet Protocol). Η διεύθυνση αυτή αποτελείται από 4 αριθμούς από το 0 έως το 255 χωρισμένους με τελείες. Έτσι για παράδειγμα η διεύθυνση 152.148.4.35 είναι μια έγκυρη IP διεύθυνση ενώ ή 153.264.302.45 δεν είναι. Οι υπολογιστές καταλαβαίνουν μόνο αυτού του τύπου τις διευθύνσεις, γι αυτό όταν εμείς γράφουμε μια ηλεκτρονική διεύθυνση με γράμματα π.χ. www.sch.gr, ένας άλλος υπολογιστής που ονομάζεται DNS (Domain Name Server) αναλαμβάνει να μεταφράσει αυτή τη διεύθυνση στην αντίστοιχη διεύθυνση IP. Στη συνέχεια, η αίτησή μας περνά από ένα πλήθος διαφορετικών δρομολογητών που είναι συνδεδεμένοι στο διαδίκτυο για να καταλήξει στον απομακρυσμένο υπολογιστή που ζητήσαμε και να μας στείλει πίσω τις πληροφορίες της ιστοσελίδας. Στην διπλανή εικόνα βλέπουμε τη διαδρομή που κάνει μια αίτησή μας προς την ιστοσελίδα www.in.gr

```
Παρακολούθηση της διαδρομής προς: www.in.gr [88.198.190.148]
με μέγιστο πλήθος αναπηδήσεων 30:

  1     1 ms     1 ms     1 ms  oxygen.lan [192.168.1.254]
  2    28 ms    29 ms    29 ms  10.12.255.54
  3    27 ms    29 ms    29 ms  62.169.243.65
  4    38 ms    32 ms    36 ms  62.169.252.138
  5    34 ms    32 ms    36 ms  62.169.249.170
  6    35 ms    32 ms    36 ms  195.22.193.37
  7    74 ms    72 ms    76 ms  te1-7-1-0.milano50.mil.seabone.net [93.186.129.8
3]
  8     *       67 ms    69 ms  ae6.bar1.Milan.Level3.net [4.68.111.165]
  9    77 ms    79 ms    79 ms  ae-0-11.bar2.Milan1.Level3.net [4.69.142.190]
 10    80 ms    78 ms    79 ms  vl-3506-ve-120.ebr1.Frankfurt1.Level3.net [4.69.
159.117]
 11    81 ms    82 ms    86 ms  ae-1-19.bar1.Munich1.Level3.net [4.69.153.245]
 12    94 ms    99 ms    92 ms  ae-0-11.bar2.Munich1.Level3.net [4.69.153.254]
 13    93 ms    92 ms    96 ms  62.140.25.102
 14    81 ms    82 ms    86 ms  core11.hetzner.de [213.239.203.137]
 15    83 ms    87 ms    81 ms  juniper1.rz2.hetzner.de [213.239.245.54]
 16    92 ms    95 ms    92 ms  hos-tr2.ex3k8.rz2.hetzner.de [213.239.235.249]
 17    83 ms    85 ms    82 ms  static.88-198-190-148.clients.your-server.de [88
.198.190.148]

Η παρακολούθηση ολοκληρώθηκε.
```

Για να φτάσει η αίτησή μας στον υπολογιστή που ζητάμε, περνάει από 17 διαφορετικά σημεία. Παρατηρήστε ότι ενώ η σελίδα που ζητάμε είναι ελληνική, η αίτησή μας περνάει από το Μιλάνο, τη Φρανκφούρτη και το Μόναχο για να καταλήξει σε έναν υπολογιστή της Γερμανίας όπου και ολοκληρώνεται η διαδρομή καθώς η σελίδα μας βρίσκεται αποθηκευμένη εκεί και όχι στην Ελλάδα όπως πιθανώς νομίζαμε!

Με κριτήριο την περιοχή που καλύπτουν, κατατάσσουμε τα δίκτυα Η/Υ σε:

- **Τοπικά δίκτυα** (Local Area Networks – LAN). Τα στοιχεία που τα

αποτελούν βρίσκονται σε ακτίνα μερικών δεκάδων ή εκατοντάδων μέτρων (π.χ. το δίκτυο του εργαστηρίου υπολογιστών του σχολείου μας).

- **Μητροπολιτικά Δίκτυα** (Metropolitan Area Networks – MAN). Εκτείνονται σε αποστάσεις μερικών χιλιομέτρων (π.χ το δίκτυο υπολογιστών ενός μητροπολιτικού δήμου με πολλά διάσπαρτα γραφεία μέσα στον αστικό ιστό).
- **Δίκτυα Ευρείας Περιοχής** (Wide Area Networks – WAN). Τα στοιχεία των δικτύων είναι δυνατόν να βρίσκονται σε αποστάσεις πολλών εκατοντάδων χιλιομέτρων (π.χ. το δίκτυο υπολογιστών μιας τράπεζας με πολλά υποκαταστήματα ανά την Ελλάδα).

Το πιο διαδεδομένο πρωτόκολλο τοπικών δικτύων νομάζεται **ethernet** και οι ταχύτητές του μπορούν να φτάσουν τα 100Gbps.

Η ενοποίηση όλων αυτών των μικρών ή μεγάλων δικτύων σχηματίζει το γνωστό σε όλους μας **διαδίκτυο**. Το διαδίκτυο δεν έχει συγκεκριμένη δομή αλλά ούτε και εποπτεύεται κεντρικά από κάποιον οργανισμό. Την τελευταία 20ετία εξαπλώθηκε ραγδαία ενώ ήδη έχουν μπει οι βάσεις για να συμπεριλάβει ακόμη περισσότερες συσκευές και υπηρεσίες.

Με κριτήριο την τεχνική προώθησης της πληροφορίας που χρησιμοποιούμε, τα δίκτυα χωρίζονται σε:

- **Δίκτυα εκπομπής** (broadcast communication networks). Στα δίκτυα αυτού του τύπου το μήνυμα μεταδίδεται σε όλους τους κόμβους ενός δικτύου και ο κάθε κόμβος το δέχεται ή το απορρίπτει. Παραδείγματα τέτοιων δικτύων είναι τα ασύρματα δίκτυα καθώς το μέσο μετάδοσης είναι κοινό για όλους.
- **Δίκτυα μεταγωγής** (switched networks). Σε αυτού του τύπου τα δίκτυα οι συνδέσεις μεταξύ των κόμβων είναι από σημείο σε σημείο και η πληροφορία μεταδίδεται από τον ένα κόμβο στον άλλο ακολουθώντας μια συγκεκριμένη διαδρομή. Τα δίκτυα μεταγωγής χωρίζονται σε 2 βασικές υποκατηγορίες:

  i) Δίκτυα μεταγωγής κυκλώματος (circuit switched networks). Τα δίκτυα αυτά μοιάζουν με τα γνωστά μας τηλεφωνικά δίκτυα όπου υπάρχει μια φυσική γραμμή επικοινωνίας μεταξύ των δύο συνομιλητών.

Έχετε μήνυμα στον υπολογιστή σας!
Εάν θέλατε να ταχυδρομήσετε μια φωτογραφία σε έναν φίλο σας θα την κόβατε ποτέ σε πολλά μικρά κομμάτια και στη συνέχεια θα τα ταχυδρομούσατε ξεχωριστά σε διαφορετικούς φακέλους με διαφορετικά γραμματόσημα περιμένοντας από τον φίλο σας να τα ξαναενώσει; Σίγουρα όχι! Κι όμως, αυτή ακριβώς είναι η διαδικασία που ακολουθείται στο γνωστό σε όλους ηλεκτρονικό ταχυδρομείο (email). Εάν θέλουμε να στείλουμε ηλεκτρονικά μια ψηφιακή μας φωτογραφία, ο υπολογιστής αναλαμβάνει να την χωρίσει σε μικρά κομμάτια που λέγονται πακέτα. Στο κάθε πακέτο, ο υπολογιστής προσθέτει έναν αύξοντα αριθμό, την ηλεκτρονική διεύθυνση του αποστολέα και την ηλεκτρονική διεύθυνση του παραλήπτη. Στη συνέχεια στέλνει τα πακέτα προς το modem/router του σπιτιού μας το οποίο με τη σειρά του κωδικοποιεί κατάλληλα μέσω των τηλεφωνικών γραμμών. Τα πακέτα αφού ακολουθήσουν μια διαδρομή με πολλούς ενδιάμεσους σταθμούς-δρομολογητές, όχι απαραίτητα την ίδια πάντα, φτάνουν τελικά στον υπολογιστή του παραλήπτη όπου ξαναενώνονται σχηματίζοντας την αρχική φωτογραφία. Και όλα αυτά μέσα σε λίγα δέκατα του δευτερολέπτου!

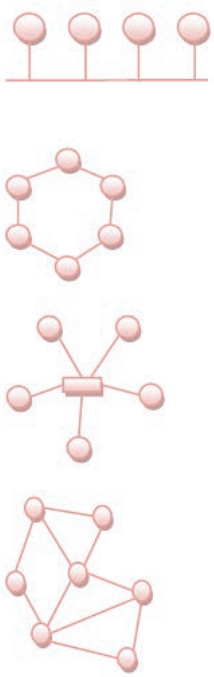
Εικόνα 3-17. Αναπαράσταση δικτύου μεταγωγής πακέτου

ii) Δίκτυα μεταγωγής πακέτου (packet switched networks). Στα δίκτυα αυτά η πληροφορία που πρόκειται να μεταδοθεί σπάει σε

μικρά πακέτα δεδομένων τα οποία και δρομολογούνται ανεξάρτητα πάνω στο δίκτυο. Δεν είναι αναγκαστικό να ακολουθήσουν όλα τα πακέτα την ίδια διαδρομή, ούτε να φτάσουν με συγκεκριμένη σειρά στο κόμβο λήψης. Όταν τα πακέτα φτάσουν στον προορισμό τους ενώνονται ώστε να ξαναδημιουργήσουν την αρχική πληροφορία. Η τεχνική μεταγωγής πακέτου είναι το κύριο χαρακτηριστικό του γνωστού μας διαδικτύου (internet).

3.2.4 Τοπολογίες δικτύων

Η τοπολογία ενός δικτύου είναι ο τρόπος με τον οποίο είναι συνδεδεμένοι οι κόμβοι ενός δικτύου. Υπάρχουν πολλές διαφορετικές τοπολογίες δικτύων προσπαθώντας να συμβιβάσουν την ταχύτητα μετάδοσης και την αξιοπιστία των δικτύων με το κόστος κατασκευής και συντήρησης. Οι πιο γνωστές από αυτές είναι:



Εικόνα 3-18. Τοπολογίες διαύλου, δακτυλίου, αστέρα και κατανεμημένη

- **Διαύλου** (bus). Όλοι οι υπολογιστές του δικτύου συνδέονται σε ένα και μόνο καλώδιο που λειτουργεί ως δρόμος επικοινωνίας και ανταλλαγής δεδομένων. Είναι η πιο απλή αλλά συνάμα και η πιο αργή τοπολογία δικτύων καθώς μόνο ένας υπολογιστής μπορεί να μεταδίδει δεδομένα κάθε φορά στο κοινό καλώδιο.
- **Δακτυλίου** (ring). Κάθε κόμβος του δικτύου συνδέεται τόσο με τον αριστερό όσο και με τον δεξιό του γείτονα, σχηματίζοντας έτσι ένα δακτύλιο. Χρησιμοποιείται κυρίως σε δίκτυα οπτικών ινών υψηλών ταχυτήτων.
- **Αστέρα** (star). Οι υπολογιστές συνδέονται ακτινωτά με μια κεντρική συσκευή που λειτουργεί ως τροχονόμος και δρομολογεί κατάλληλα τις αιτήσεις των υπολογιστών για μεταφορά δεδομένων. Τα σύγχρονα τοπικά δίκτυα ακολουθούν αυτή την τοπολογία καθώς έχει πολλά πλεονεκτήματα με κυριότερο τις υψηλές ταχύτητες μετάδοσης δεδομένων.
- **Κατανεμημένη** (mesh). Εάν επεκτείνουμε την τοπολογία αστέρα, συνδέοντας πολλά επιμέρους δίκτυα αστέρα και προσθέτοντας επιπλέον συνδέσεις μεταξύ των κόμβων, καταλήγουμε στην κατανεμημένη τοπολογία στην οποία η πληροφορία μπορεί να ακολουθήσει πολλές εναλλακτικές διαδρομές μεταξύ δύο κόμβων.

3.2.5 Σύγχρονες υπηρεσίες δικτύων

Το διαδίκτυο έχει γίνει αναπόσπαστο μέρος της καθημερινής μας ζωής και προσφέρει πλήθος υπηρεσιών που δε θα μπορούσαμε να ις φανταστούμε πριν από μερικά χρόνια. Ενδεικτικά αναφέρουμε τις εξής:

Μετάδοση φωνής μέσω διαδικτύου (Voice over IP – VOIP)

Παρόλο που το διαδίκτυο δεν κατασκευάστηκε αρχικά για μετάδοση φωνής, τα τελευταία χρόνια, ακόμη και μεγάλοι τηλεφωνικοί κολοσσοί, έχουν υιοθετήσει τεχνολογίες μετάδοσης φωνής μέσω διαδικτύου (VOIP) για την εξυπηρέτηση τηλεφωνικών κλήσεων.

Τηλεδιασκέψεις πραγματικού χρόνου

Εκτός από τη μετάδοση φωνής, πολύ διαδεδομένη είναι και η μετάδοση εικόνας σε πραγματικό χρόνο μέσω διαδικτύου. Πολλές εταιρίες παρέχουν ειδικά προγράμματα για τηλεδιασκέψεις είτε για προσωπική είτε για επαγγελματική χρήση, ενώ πολλές εφαρμογές τηλεδιάσκεψης συναντάμε για εκπαιδευτικούς και επιστημονικούς σκοπούς.

Διαδικτυακή Τηλεόραση (IPTV)

Η γιγάντωση του διαδικτύου και η αύξηση των ταχυτήτων μετάδοσης δεδομένων έκανε δυνατή την παροχή υπηρεσιών τηλεόρασης μέσω διαδικτύου. Αυτές περιλαμβάνουν τόσο ζωντανές μεταδόσεις τηλεοπτικών καναλιών όσο και μεταδόσεις κατ' απαίτηση (on demand).

Υπηρεσίες web 2.0 – Κοινωνικό Λογισμικό

Ο όρος αυτός περιλαμβάνει ένα σύνολο υπηρεσιών του παγκόσμιου ιστού στις οποίες οι χρήστες του διαδικτύου δεν είναι απλά θεατές ή αναγνώστες αλλά παραγωγοί πληροφοριών που τις δημοσιεύουν με εύκολο τρόπο στο διαδίκτυο. Παραδείγματα τέτοιων υπηρεσιών είναι τα ιστολόγια (blogs), τα wikis, οι σελίδες κοινωνικής δικτύωσης (facebook κ.α.), οι υπηρεσίες διαμοιρασμού πολυμέσων (youtube, flikr) κ.α.

Νεφοπληροφορική (cloud computing)

Ο όρος νεφοπληροφορική χρησιμοποιείται για να δηλώσει ένα είδος πληροφορικής όπου δεδομένα και εφαρμογές δεν βρίσκονται τοπικά σε έναν Η/Υ, αλλά σε κάποιον απομακρυσμένο εξυπηρετητή στον οποίο οι χρήστες συνδέονται για να τις χρησιμοποιήσουν. Ένα τέτοιο μοντέλο εργασίας έχει σημαντικά πλεονεκτήματα καθώς οι χρήστες έχουν ένα ενιαίο περιβάλλον εργασίας που τους ακολουθεί παντού και σε όλες τις υπολογιστικές συσκευές τους. Ωστόσο θέτει ζητήματα ασφάλειας των δεδομένων καθώς αυτά βρίσκονται αποθηκευμένα στο αχανές διαδίκτυο και όχι τοπικά σε κάποιο προσωπικό υπολογιστή.

Ασκήσεις – Προβλήματα

1. Κατασκευάστε ένα σχεδιάγραμμα του δικτύου του εργαστηρίου σας, σημειώνοντας τη διεύθυνση IP κάθε υπολογιστή.
2. Αναζητήστε πληροφορίες για τις δικτυακές εντολές ping, ipconfig και tracert. Εκτελέστε τις στη γραμμή εντολών των windows και σχολιάστε στην τάξη το αποτέλεσμα.
3. Κατεβάστε ένα μεγάλο αρχείο από το διαδίκτυο, χρονομετρώντας τη διάρκεια λήψης του. Στη συνέχεια αξιοποιήστε την πληροφορία αυτή για να υπολογίσετε την ταχύτητα σύνδεσης στο διαδίκτυο.

Όταν ολοκληρώσετε την ενότητα ελέγξτε αν είστε σε θέση:

- Να ονομάζετε τα βασικά στοιχεία ενός δικτύου.
- Να κατηγοριοποιείτε τα δίκτυα βάσει συγκεκριμένων κριτηρίων.
- Να περιγράφετε τις σύγχρονες υπηρεσίες δικτύου.

3.3 Πληροφοριακά συστήματα

Η ψηφιακή επανάσταση δημιούργησε την κοινωνία της πληροφορίας (information society), στην οποία η πληροφορία αποτελεί σημαντικό παράγοντα σε όλους τους τομείς της ανθρώπινης δραστηριότητας. Οι οργανισμοί, οι επιχειρήσεις και οι κοινωνίες στηρίζονται στην πληροφορία και στη γνώση που προέρχεται από αυτήν, για να λειτουργήσουν.

Ποιος είναι ο ρόλος της Επιστήμης Υπολογιστών στη δημιουργία γνώσης;

Η αυτοματοποιημένη δημιουργία και διαχείριση της γνώσης γίνεται εφικτή και αποδοτική μέσα από συστήματα που στηρίζονται στους υπολογιστές. Τα συστήματα αυτά ονομάζονται Πληροφοριακά Συστήματα.

Δεδομένα	23061912	Ανεπεξέργαστο στοιχείο δεδομένων	
Πληροφορία	6/23/12	Μορφοποιημένο στοιχείο δεδομένων	Ημερομηνία
Γνώση	23-06-1912 η ημ/νία γέννησης του Alan Turing	Συσχέτιση με άλλα δεδομένα	Ημερομηνία γέννησης του Alan Turing
Σοφία	Ο Alan Turing γεννήθηκε πριν το Β' Παγκ. Πόλεμο	Συσχέτιση με άλλα δεδομένα άλλων πεδίων	Ο Turing αποκρυπτογράφησε το «Αίνιγμα»

Εικόνα 3-19. Η μετάβαση από τα δεδομένα στην πληροφορία, τη γνώση και τη σοφία

Πληροφοριακό Σύστημα (Π.Σ.) είναι ένα σύνολο από ανθρώπους, διαδικασίες, δεδομένα, λογισμικό και υλικό υπολογιστών, που αλληλεπιδρούν μεταξύ τους με οργανωμένο τρόπο για να υποστηρίξουν την ανθρώπινη δραστηριότητα.

Στον πυρήνα των Πληροφοριακών Συστημάτων βρίσκονται τα συστήματα διαχείρισης δεδομένων, δηλαδή τα συστήματα που επιτρέπουν την αποθήκευση, ανάκτηση και επεξεργασία των δεδομένων.

Μέχρι τα μέσα της δεκαετίας του 1960 τα συστήματα αυτά αποτελούνταν από αρχεία (files) δεδομένων αποθηκευμένα σε μαγνητικές ταινίες, στα οποία γινόταν ενημέρωση με διεργασίες δέσμης (batch processes). Η κατασκευή ταχύτερων αποθηκευτικών μέσων με μεγάλη χωρητικότητα οδήγησε στην ανάγκη για καλύτερη οργάνωση των δεδο-

Τύπος συστήματος	Σκοπός	Παραδείγματα εφαρμογών
Σύστημα Αυτοματισμού Γραφείου	Υποστήριξη εργασιών σε ημερήσια βάση (ατομική παραγωγικότητα)	Επεξεργαστής κειμένου, επεξεργαστής λογιστικών φύλλων
Σύστημα Επεξεργασίας Συναλλαγών	Καθημερινή επεξεργασία των δεδομένων της επιχείρησης σε επίπεδο λειτουργίας	Κράτηση εισιτηρίων και ενημέρωση διαθεσιμότητας
Σύστημα Υποστήριξης Αποφάσεων	Εργαλεία ανάλυσης βάσεων δεδομένων με στόχο την υποστήριξη αποφάσεων	Ανάλυση αγοράς, πρόβλεψη της ζήτησης προϊόντων
Σύστημα διαχείρισης πελατών	Υποστήριξη της αλληλεπίδρασης μεταξύ επιχείρησης και πελατών	Αυτοματοποίηση των πωλήσεων

Εικόνα 3-20. Τύποι Πληροφοριακών Συστημάτων

μένων και, γι' αυτό το λόγο, δημιουργήθηκαν οι βάσεις δεδομένων. Αρχικά, οι βάσεις δεδομένων στηρίζονταν σε συστήματα τοπικής αποθήκευσης και επομένως άμεσης πρόσβασης. Η εξάπλωση όμως των δικτύων άλλαξε σημαντικά την αρχιτεκτονική των συστημάτων διαχείρισης δεδομένων, εφόσον επέτρεψε την κατανομή των πόρων τους πάνω σε δίκτυα (κατανεμημένα συστήματα). Σήμερα η αποθήκευση των δεδομένων έχει μετατοπιστεί στα «σύννεφα» (cloud storage), εφόσον ο χρήστης δεν αντιλαμβάνεται που και πώς είναι αποθηκευμένα τα δεδομένα του, στα οποία έχει πρόσβαση μέσω του Διαδικτύου.

Βάσεις δεδομένων

Οι περισσότεροι από εμάς ερχόμαστε καθημερινά σε επαφή με βάσεις δεδομένων. Η ανάληψη χρημάτων από μια Αυτόματη Ταμειακή Μηχανή (ATM) τράπεζας, η κράτηση εισιτηρίων, η αγορά αγαθών,από ένα πολυκατάστημα ή από το Διαδίκτυο είναι μόνο μερικά από τα παραδείγματα όπου χρησιμοποιούμε βάσεις δεδομένων.

Βάση Δεδομένων (Database) είναι μια οργανωμένη συλλογή από αποθηκευμένα δεδομένα που σχετίζονται μεταξύ τους, διαθέτουν συνέπεια και συνοχή, και έχουν δημιουργηθεί για συγκεκριμένο σκοπό, συγκεκριμένη ομάδα χρηστών και συγκεκριμένες εφαρμογές.

Ας δούμε το παράδειγμα ενός συστήματος ηλεκτρονικών κρατήσεων. Τα δεδομένα του συστήματος αφορούν ξενοδοχεία, οχήματα για ενοικίαση, πελάτες, κρατήσεις, πληρωμές, κ.λ.π. Τα δεδομένα αυτά σχετίζονται μεταξύ τους με συγκεκριμένους τρόπους. Οι πελάτες κάνουν κρατήσεις για δωμάτια ξενοδοχείων ή οχήματα. Μία κράτηση μπορεί να γίνει μόνο εφόσον είναι διαθέσιμο το προϊόν που ενδιαφέρει τον πελάτη και ακολουθείται από μια σειρά ενεργειών όπως η ενημέρωση της διαθεσιμότητας, η καταχώρηση της πληρωμής, η αποστολή της απόδειξης ή του τιμολογίου κλπ. Όλα αυτά τα στοιχεία πρέπει να αποθηκεύονται, να οργανώνονται έτσι ώστε να συσχετίζονται μεταξύ τους και να ενημερώνονται με συνεπή τρόπο από πολλούς χρήστες ταυτόχρονα. Γι' αυτό το λόγο, το ηλεκτρονικό σύστημα κρατήσεων πρέπει να υλοποιηθεί με βάσεις δεδομένων οι οποίες θα πλαισιωθούν από λογισμικό για τη σωστή λειτουργία τους. Το λογισμικό αυτό ονομάζεται Σύστημα Διαχείρισης Βάσεων Δεδομένων (Database Management System).

Οι ανθρώπινες κοινωνίες για να παραμείνουν «ανθρώπινες» και βιώσιμες πρέπει να παράγουν, να επεξεργάζονται, να μοιράζονται και να χρησιμοποιούν την πληροφορία για να κτίσουν γνώση, περνώντας πλέον από την κοινωνία της πληροφορίας στην κοινωνία της γνώσης με στόχο την βελτίωση της ανθρώπινης ζωής. (UNESCO, 2005)

Σύστημα Διαχείρισης Βάσεων Δεδομένων είναι ένα σύνολο προγραμμάτων που διευκολύνουν τη δημιουργία και τη λειτουργία μιας βάσης δεδομένων παρέχοντας τη δυνατότητα ορισμού, χειρισμού και διαμοίρασης δεδομένων ανάμεσα σε πολλούς χρήστες και εφαρμογές.

Ένα Σύστημα Διαχείρισης Βάσεων Δεδομένων:

- αποθηκεύει μεγάλο όγκο δεδομένων και παρέχει γρήγορη πρόσβαση σε αυτά με τη σωστή οργάνωσή τους,
- εξασφαλίζει την ορθότητα των δεδομένων έτσι ώστε να μην υπάρχει

πλεονασμός,
- εξασφαλίζει τη συνέπεια στα δεδομένα ανάμεσα σε διαφορετικούς χρήστες που τα ενημερώνουν ταυτόχρονα,
- παρέχει μηχανισμούς «ανάνηψης» σε περιπτώσεις αστοχίας,
- παρέχει εργαλεία στους χρήστες για να χειρίζονται τα δεδομένα,
- παρέχει έλεγχο πρόσβασης και ασφάλεια στα δεδομένα,
- διευκολύνει την ανάπτυξη εφαρμογών που χειρίζονται τα δεδομένα

Το σχεσιακό μοντέλο δεδομένων (relational database model) έχει τα θεμέλιά του στη θεωρία συνόλων και στη λογική.

Οι πρώτες Βάσεις Δεδομένων μέχρι τη δεκαετία του 1980, οργάνωναν τα δεδομένα κυρίως ως ιεραρχίες ή δίκτυα. Το 1970, ο Codd πρότεινε ένα είδος οργάνωσης που ονομάστηκε σχεσιακό μοντέλο δεδομένων. Οι Σχεσιακές Βάσεις Δεδομένων (Relational Databases) ακολουθούν το σχεσιακό μοντέλο δεδομένων και έχουν επικρατήσει ως το κυρίαρχο είδος Βάσεων Δεδομένων. Άλλα μοντέλα δεδομένων είναι το οντοσχεσιακό (object-relational) και το οντοκεντρικό (object-oriented).

Οι σχεσιακές βάσεις δεδομένων χρησιμοποιούν τη γλώσσα SQL (Structured Query Language) που έχει καθιερωθεί ως πρότυπο και σε αυτήν οφείλεται μέρος της επιτυχημένης πορείας των σχεσιακών βάσεων δεδομένων.

Ένα από τα σημαντικότερα χαρακτηριστικά των Βάσεων Δεδομένων είναι ότι υποστηρίζουν ερωτήσεις (queries) πάνω στα δεδομένα. Οι ερωτήσεις γίνονται από χρήστες ή προγράμματα σε μια γλώσσα ερωταπαντήσεων (query language). Οι γλώσσες αυτές είναι υψηλού επιπέδου και μπορούν να χρησιμοποιηθούν και από κάποιον που δεν έχει ιδιαίτερες προγραμματιστικές γνώσεις.

Ένα άλλο μοντέλο δεδομένων που έγινε δημοφιλές εξαιτίας του Παγκόσμιου Ιστού (WWW) είναι το μοντέλο της XML (eXtensible Markup Language), το οποίο περιγράφει ημιδομημένα δεδομένα (έγγραφα) με ιεραρχική οργάνωση. Το μοντέλο της XML χρησιμοποιείται για την ανταλλαγή δεδομένων ανάμεσα σε εφαρμογές του Παγκόσμιου Ιστού.Τα τελευταία χρόνια έχουν αναπτυχθεί Συστήματα Διαχείρισης Βάσεων Δεδομένων που το υποστηρίζουν.

Κατανεμημένες Βάσεις Δεδομένων

Αγορές μέσω πιστωτικής κάρτας
Όταν πραγματοποιούμε αγορές μέσω πιστωτικής κάρτας, επικοινωνούμε χωρίς να το γνωρίζουμε με το πληροφοριακό σύστημα μιας τράπεζας. Δίκτυα υπολογιστών, βάσεις δεδομένων και αλγόριθμοι κρυπτογραφίας συνεργάζονται αρμονικά ώστε η τράπεζα να ελέγξει το καταναλωτικό προφίλ μας και να εγκρίνει ή όχι την αγορά μέσα σε λίγα δευτερόλεπτα.

Τα δεδομένα μιας επιχείρησης ή ενός οργανισμού μπορεί να μη βρίσκονται συγκεντρωμένα σε ένα σημείο. Για παράδειγμα, σε μια επιχείρηση με πολλά υποκαταστήματα εκτελούνται αγορές σε διαφορετικά σημεία. Τα υποκαταστήματα δημιουργούν και αποθηκεύουν ένα μικρό μέρος των δεδομένων της επιχείρησης στα οποία χρειάζονται γρήγορη πρόσβαση, ανεξάρτητα από τα υπόλοιπα υποκαταστήματα. Το κεντρικό κατάστημα της επιχείρησης είναι απαραίτητο να έχει πρόσβαση στα δεδομένα κάθε υποκαταστήματος και να ενημερώνει τα δικά του συγκεντρωτικά στοιχεία για να παρακολουθεί τη διαθεσιμότητα των προϊόντων. Πώς εξασφαλίζεται η συνέπεια των δεδομένων που είναι απομακρυσμένα αλλά ανήκουν στον ίδιο οργανισμό; Με τη δημιουργία κατανεμημένων (distributed) υπολογιστικών συστημάτων.

Ένα **κατανεμημένο υπολογιστικό σύστημα** αποτελείται από επί μέρους τμήματα, όχι απαραίτητα ομοιογενή, τα οποία διασυνδέονται μέσω δικτύου υπολογιστών και συνεργάζονται μεταξύ τους για να επιλύσουν συγκεκριμένα προβλήματα. Ο στόχος των κατανεμημένων συστημάτων είναι να διευκολύνουν την επίλυση μεγάλων και πολύπλοκων προβλημάτων, διαιρώντας τα σε μικρότερα τα οποία θα λυθούν με συντονισμό

διαφορετικών απομακρυσμένων τμημάτων. Μια βάση δεδομένων μπορεί να είναι κατανεμημένη σε διαφορετικούς απομακρυσμένους υπολογιστές. Μερικές από τις προκλήσεις στις κατανεμημένες βάσεις δεδομένων είναι η ασφαλής και γρήγορη ανταλλαγή δεδομένων και η εγκυρότητα των δεδομένων που προσπελαύνονται από διαφορετικούς χρήστες.

Μαθήματα

Κωδικός	Τίτλος	Περιγραφή
10	Αρχές της Επιστήμης των Υπολογιστών	Μάθημα της Β' Λυκείου

Μαθητές

ΑΜ	Επώνυμο	Όνομα
78	Μυλωνάς	Γιώργος

Βαθμολογίες

ΑΜ_μαθητή	Κωδικός_μαθήματος	Βαθμός
78	10	18

```
SELECT Βαθμός
FROM Βαθμολογίες
WHERE ΑΜ_μαθητή = 78
```

Εινόνα 3-21. Σχεσιακό σχήμα Βάσεων Δεδομένων μαθητών και ερώτηση σε SQL

Αρχιτεκτονική αποθήκευσης σε νέφος

Όπως αναφέρθηκε στην ενότητα 2.3.5. νεφοπληροφορική, αναπτύχθηκε για να προσφέρει υπολογιστικούς πόρους (υλικό, λογισμικό και δεδομένα) ως υπηρεσίες του Διαδικτύου, χωρίς να είναι ορατό στο χρήστη πού βρίσκονται αυτοί οι πόροι. Το βασικό της πλεονέκτημα είναι ο περιορισμός του κόστους αγοράς και συντήρησης υλικού και λογισμικού. Το αποτέλεσμα της είναι ότι δίνει τη δυνατότητα αποθήκευσης όγκου δεδομένων μεγέθους της τάξης των terabytes (1 terabyte = 2^{40} bytes ≈ 10^{12} bytes). Οι παραδοσιακές βάσεις δεδομένων δεν μπορούν να χειριστούν τέτοιο όγκο δεδομένων. Έτσι δημιουργήθηκε η ανάγκη για αλλαγή της παραδοσιακής αρχιτεκτονικής των Συστημάτων Διαχείρισης Βάσεων Δεδομένων σε αρχιτεκτονικές νέφους (cloud storage).

Ασκήσεις – Προβλήματα

1. Εντοπίστε τη βάση δεδομένων που χρησιμοποιείται στο σχολείο σας και περιγράψτε τα είδη των στοιχείων που αποθηκεύονται σε αυτήν.
2. Περιγράψτε τα δεδομένα της βάσης που αφορά σε μια συλλογή τραγουδιών διαφόρων καλλιτεχνών και ειδών μουσικής. Τι ερωτήσεις μπορεί να γίνονται σε αυτά τα δεδομένα από χρήστες που αγοράζουν τραγούδια από το Διαδίκτυο;
3. Αναζητήστε υπηρεσίες του Διαδικτύου που χρησιμοποιούν αποθήκευση σε νέφος.

Όταν ολοκληρώσετε την ενότητα ελέγξτε αν είστε σε θέση:

- να αναγνωρίζετε τη σημασία της πληροφορίας στη δημιουργία της γνώσης και να εξηγείτε τι είναι ένα Πληροφοριακό Σύστημα.
- να αναγνωρίζετε τα Συστήματα Διαχείρισης Βάσεων Δεδομένων ως σημαντικό τμήμα των Πληροφοριακών Συστημάτων και να περιγράφετε το ρόλο τους.
- να αναγνωρίζετε το σχεσιακό μοντέλο δεδομένων ως κυρίαρχο είδος Βάσεων Δεδομένων και τη γλώσσα ερωταπαντήσεων SQL ως βασικό χαρακτηριστικό επικοινωνίας με τη Βάση Δεδομένων.

3.4 Τεχνητή Νοημοσύνη (Τ.Ν.)

Ο αρχαίος Έλληνας φιλόσοφος Αριστοτέλης ήταν ο πρώτος που διατύπωσε ερωτήματα για την ανθρώπινη νόηση και το λογισμό. "Από πού προέρχεται η γνώση; Πώς η γνώση οδηγεί στη δράση;". Τα ερωτήματα αυτά τον οδήγησαν στη θεμελίωση των νόμων της Λογικής. Πολύ αργότερα, το 17ο αιώνα, ο Γερμανός φιλόσοφος και επιστήμονας G.W. Leibniz οραματίστηκε μια καθολική γλώσσα συλλογιστικής η οποία θα κατέληγε σε υπολογισμούς. Η ιδέα ότι η σκέψη διέπεται από νόμους που μπορούν να μετατραπούν σε υπολογισμούς, οδήγησε στην προσπάθεια κατασκευής «σκεπτόμενων» μηχανών. Μπορεί όμως μια μηχανή να σκεφτεί;

Το 1950 ο Alan Turing, «πατέρας» της θεωρητικής επιστήμης των υπολογιστών, πρότεινε ένα τεστ για να διαπιστώσουμε, όχι αν μια μηχανή μπορεί να σκεφτεί, αλλά αν μπορεί να ενεργεί έξυπνα. Σύμφωνα λοιπόν με τον Turing, μια μηχανή είναι ευφυής αν συνομιλήσει για 5 λεπτά με έναν άνθρωπο και ο άνθρωπος δεν καταλάβει ότι μιλάει με μηχανή. Ο Turing επίσης διατύπωσε τι δεν μπορεί και δε θα μπορέσει ποτέ να κάνει μια μηχανή: "μια μηχανή δεν μπορεί να είναι ευγενική, φιλική, να έχει χιούμορ, να ερωτευτεί, να κάνει λάθη, να φάει παγωτό, να κάνει κάτι πραγματικά καινούργιο …".

Από τα μέσα του 20ου αιώνα και μετά, ένας ολόκληρος επιστημονικός κλάδος ασχολείται με τη δημιουργία ευφυών μηχανών, ο κλάδος της Τεχνητής Νοημοσύνης.

Τεχνητή Νοημοσύνη (Artificial Intelligence – AI) είναι η επιστήμη που επιχειρεί να δημιουργήσει οντότητες με ευφυΐα, συστηματοποιώντας και αυτοματοποιώντας τις νοητικές λειτουργίες.

Η γέννηση της σύγχρονης Τεχνητής Νοημοσύνης (Τ.Ν.) έγινε με τα νευρωνικά δίκτυα (neural networks). Το 1943 οι Warren McCulloch και Walter Pitts πρότειναν την ιδέα των τεχνητών νευρώνων που συνδέονται μεταξύ τους, ενεργοποιούν ο ένας τον άλλον σύμφωνα με λογικούς κανόνες και μπορούν να μάθουν, όπως ο ανθρώπινος εγκέφαλος. Το 1956 ο John McCarthy πρότεινε τον όρο Τεχνητή Νοημοσύνη και το 1958 ο ίδιος πρότεινε τη γλώσσα προγραμματισμού LISP, η οποία με λίγους τελεστές και με ειδικό συμβολισμό για τις συναρτήσεις αποτελεί μια υπολογιστικά καθολική (computationally universal ή Turing-complete) γλώσσα αλγορίθμων. Ένα από τα πρώτα προγράμματα Τ.Ν. ήταν ο Γενικός Λύτης Προβλημάτων (General Problem Solver) των Allen Newell, J.C. Shaw και Herbert Simon που προσομοίωνε τον ανθρώπινο τρόπο επίλυσης προβλημάτων. Από τα μέσα της δεκαετίας του '60 μέχρι τα μέσα του '70 οι περιορισμοί σε υπολογιστικούς πόρους οδήγησαν στην παρακμή του κλάδου. Στις αρχές του 1970 δημιουργήθηκε η γλώσσα λογικού προγραμματισμού Prolog από την ομάδα του Alain Colmerauer στη Γαλλία. Η Τ.Ν. ανέκαμψε την δεκαετία του '80 με τη δημιουργία των έμπειρων συστημάτων (expert systems) που έδιναν έμφαση στην

Ο υπολογιστής είναι μηχανή. Οι μηχανές δεν ερωτεύονται. Άρα ο υπολογιστής δεν ερωτεύεται.
Μηχανή (Η/Υ) ∧
¬ Ερωτεύεται (Μηχανή)
=> ¬ Ερωτεύεται (Η/Υ)

αναπαράσταση και την επεξεργασία της γνώσης. Το πρώτο επιτυχημένο εμπορικό έμπειρο σύστημα δημιουργήθηκε το 1982 από την εταιρεία DEC. Σήμερα, οι εξελίξεις στην T.N. την έχουν καταστήσει επιστημονικό κλάδο και χάρη σε αυτήν ένας υπολογιστής μπορεί να παίξει σκάκι αυτόνομα, να οδηγήσει αυτοκίνητα και ελικόπτερα, να διαγνώσει ασθένειες, να ελέγξει την ορθογραφία ενός κειμένου και να μαθαίνει. Αυτές οι ικανότητες βέβαια δε σημαίνουν ότι ο υπολογιστής διαθέτει διορατικότητα και αντίληψη, αλλά ότι μπορεί να επιδείξει ευφυή συμπεριφορά.

Τι χρειάζεται όμως να έχει μια μηχανή για να χαρακτηριστεί ευφυής; Χρειάζεται:

- **να επικοινωνεί με τον άνθρωπο στη γλώσσα του.** Ο κλάδος της T.N. που ασχολείται με την επικοινωνία υπολογιστή - ανθρώπου με ανθρώπινες γλώσσες ονομάζεται Επεξεργασία Φυσικής Γλώσσας (Natural Language Processing). Μερικά από τα θέματα που πραγματεύεται ο κλάδος είναι η Ανάλυση Διαλόγου (Discourse Analysis), η Μηχανική Μετάφραση (Machine Translation), η Οπτική Αναγνώριση Χαρακτήρων (Optical Character Recognition), η Αναγνώριση Φωνής (Speech Recognition), κ.ά.
- **να κατέχει γνώση.** Ο τομέας της Αναπαράστασης Γνώσης (Knowledge Representation) ασχολείται με τους τρόπους που αναπαρίσταται και αποθηκεύεται η συμβολική γνώση σε μια μηχανή. Η συμβολική γνώση εκφράζει λογικές προτάσεις και ο στόχος της αναπαράστασής της είναι να διευκολυνθεί η επεξεργασία της.
- **να συλλογίζεται.** Η Αυτοματοποιημένη Συλλογιστική (Automated Reasoning) χρησιμοποιεί την αναπαράσταση των λογικών προτάσεων για να δημιουργεί αυτόματα νέες λογικές προτάσεις.
- **να μαθαίνει**. Η Μηχανική Μάθηση (Machine Learning) ασχολείται με τον τρόπο που οι υπολογιστές μπορούν να βελτιώσουν τη συμπεριφορά τους με βάση αυτά που γνωρίζουν.
- **να βλέπει, να αισθάνεται**. Η Υπολογιστική Όραση (Computer Vision) ασχολείται με την επεξεργασία εικόνων με στόχο την εξαγωγή συμπερασμάτων για το περιεχόμενό τους και τη λήψη αποφάσεων.
- **να κινείται αυτόνομα και να χειρίζεται άλλα αντικείμενα**. Η Ρομποτική (Robotics) μελετά και κατασκευάζει προγραμματιζόμενες μηχανές που μπορούν να κινηθούν αυτόνομα έχοντας αίσθηση του πραγματικού κόσμου.

Οι επιστημονικές εξελίξεις στους παραπάνω κλάδους έχουν καταστήσει την Τεχνητή Νοημοσύνη μια πραγματικότητα. Μερικοί από τους πλέον σύγχρονους τομείς εφαρμογής της σήμερα, είναι:

1. Ο Αυτόνομος Σχεδιασμός (Automated Planning). Το σύστημα MAPGEN της NASA δέχεται στόχους υψηλού επιπέδου και δημιουργεί αυτόνομα το σχέδιο για τον έλεγχο του μη επανδρωμένου οχήματος Mars Exploration Rover που εξερευνά τον πλανήτη Άρη.
2. Τα Παιχνίδια (Game Playing). Το 1997 ο υπολογιστής Deep Blue της IBM νίκησε τον παγκόσμιο πρωταθλητή Kasparov στο σκάκι. Οι μηχανές που μπορούν να παίζουν παιχνίδια επιδεικνύουν ευφυή συμπεριφορά και γι' αυτό το λόγο αποτελούν αντικείμενο μελέτης.

«Φαίνεται πιθανό ότι, εφόσον ξεκινήσουν οι μηχανές να σκέπτονται, δε θα τους πάρει πολύ χρόνο για να υπερβούν τις ισχνές μας δυνάμεις... Θα μπορούν να συνδιαλέγονται και να ακονίζουν το μυαλό τους. Σε κάποιο στάδιο θα πρέπει να αναμένουμε ότι οι μηχανές θα πάρουν τον έλεγχο», Alan Turing.

3. Ο Αυτόνομος έλεγχος (Autonomous control). Αυτοκίνητα χωρίς τιμόνι, πετάλια και μοχλό ταχυτήτων που κινούνται αυτόνομα μπορούν να αποτελέσουν μεταφορικό μέσο για άτομα με προβλήματα όρασης ή κίνησης. Ήδη από το 2012 η εταιρεία Google ανακοίνωσε ότι τα οχήματά της συμπλήρωσαν 500.000 χιλιόμετρα αυτόνομης οδήγησης χωρίς ατύχημα.
4. Η Διάγνωση (Diagnosis). Λογισμικά ιατρικής διάγνωσης μπορούν με βάση τα συμπτώματα και με ανάλυση των πιθανοτήτων να διαγνώσουν ασθένειες με μεγάλη ακρίβεια.
5. Η Διακίνηση Οχημάτων και Αγαθών (Logistics) χρησιμοποιεί μεθόδους της Τ.Ν. για να προγραμματίσει σε μικρό χρονικό διάστημα τη σωστή και αποδοτική μεταφορά μεγάλων στόλων οχημάτων και τη διανομή αγαθών.

Ερωτήσεις - Θέματα για συζήτηση

1. Αναζητήστε σύγχρονες εφαρμογές της Τεχνητής Νοημοσύνης με ευρεία χρήση.
2. Σχολιάστε την άποψη του Turing που διατυπώνεται στο τέλος της ενότητας σχετικά με τις μηχανές.

Όταν ολοκληρώσετε την ενότητα ελέγξτε αν είστε σε θέση:

- Να διατυπώνετε ερωτήματα σχετικά με τη “νοημοσύνη” των μηχανων.
- Να απαριθμείτε τους σύγχρονους τομείς της Τεχνητής Νοημοσύνης και τις πλέον σύχγρονες εφαρμογές της.
- Να αναφέρετε γλώσσες προγραμματισμού της Τ.Ν.

Βιβλιογραφία

1. H. Abelson, G. J. Sussman, *Structure and Interpretation of Computer Programs*, 2nd Edition, 1996.
2. P. F. Drucker, *Landmarks of Tomorrow: A Report on the New "Post-Modern"* World, 1959.
3. R. Elmasri, S. B. Navathe, *Fundamentals of Database Systems,* Sixth Edition, Addison-Wesley, 2010.
4. A. Hodges, Άλαν Τιούρινγκ: *Το αίνιγμα*, εκδόσεις Τραυλός, 2004.
5. H. Lewis, Χ. Παπαδημητρίου. *Στοιχεία Θεωρίας Υπολογισμού*, εκδόσεις Κριτική, 2005
6. M. Sipser, *Εισαγωγή στη Θεωρία Υπολογισμού*, Πανεπιστημιακές Εκδόσεις Κρήτης, 2007
7. E. Horowitz, S. Sahni, *Fundamentals of Data Structures in Pascal*, W. H. Freeman; Fourth Edition, 1993.
8. B. Kernighan και D. Ritchie, *Η Γλώσσα Προγραμματισμού C*, Εκδόσεις Κλειδάριθμος, Αθήνα 1990.
9. D. Knuth, *The art of Computer Programming, Volume 1: Fundamental Algorithms*. Third Edition, Reading, Massachusetts: Addison-Wesley, 1997.
10. G. Polya, *Πώς να το λύσω*, Εκδόσεις Καρδαμίτσα, Αθήνα 1991.
11. P. Rechenberg, *Εισαγωγή στην Πληροφορική – Μία ολοκληρωμένη παρουσίαση*, Εκδόσεις Κλειδάριθμος, 1992.
12. S. Russel, P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall; 3rd Edition, 2010.
13. A. Silberschatz, P.B. Galvin, G. Gagne, *Operating System Concepts*, Wiley, 8th edition, 2009.
14. A. Tanenbaum, D.J. Wetherall, *Computer Networks*, Pearson, 5th Edition, 2002.
15. A. S. Tanenbaum, *Modern Operating Systems*, 3rd Edition, 2008.
16. J. Valacich and C. Schneider, *Information Systems Today: Managing in the Digital World*, Fourth Edition, Published by Prentice Hall, 2010.
17. Α. Βακάλη, κ.ά., *Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον*, Ο.Ε.Δ.Β, 2008.
18. Ν. Αντωνάκος κ.ά.,*Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον*, Ο.Ε.Δ.Β, 1999.
19. Ν. Ιωαννίδης, Κ. Μαρινάκης, Σ. Μπακογιάννης, *Δομημένη σχεδίαση Προγράμματος Αλγοριθμική*, Εκδόσεις Ν. Τεχνολογιών, 1991.
20. Γ. Παπακωνσταντίνου, κ.ά., *Τεχνολογία Υπολογιστικών Συστημάτων & Λειτουργικά Συστήματα,* Τάξη Γ΄ Ενιαίου Λυκείου, Ο.Ε.Δ.Β, 1999.
21. Ι. Μανωλόπουλος, *Δομές Δεδομένων – Μία προσέγγιση με Pascal*, Εκδόσεις Art of Text, 1997.

Πηγές on-line

1. M. Armbrust, e.a. (2009). *Above the Clouds: A Berkeley View of Cloud Computing*, TR No. UCB/EECS-2009-28, [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html.
2. D. Evans (August 19, 2011). *Introduction to Computing: Explorations in Language, Logic, and Machines*, [Online]. Available: http://www.computingbook.org/FullText.pdf.
3. R. Sedgewick, K. Wayne (2003). *An Introduction to Computer Science*, [Online]. Available: ftp://ftp.cs.princeton.edu/pub/people/ rs/book0/pdf/IntroCS.book.pdf.
4. R. Sedgewick, K. Wayne (2013, Oct. 13). *Introduction to Programming in Java*, Booksite [Online]. Available: http://introcs.cs.princeton.edu/java/home/.
5. R. Sedgewick, K. Wayne (2014, Jun 15). *Algorithms*, (4th .Edition), Booksite [Online]. Available: http://algs4.cs.princeton.edu/home/.
6. R. Sedgewick, P. Flajolet. (2013). *An Introduction to the Analysis of Algorithms*, Booksite [Online]. Available: http://aofa.cs.princeton.edu/ home/.
7. R. Toal (2004). *Programming Paradigms*, [Online]. Available: http://cs.lmu.edu/~ray/notes/paradigms.
8. P. Van Roy (2009). *Programming Paradigms for Dummies: What Every Programmer Should Know*, [Online]. Available: http://www.info.ucl.ac.be/~pvr/VanRoyChapter.pdf.
9. UNESCO. *UNESCO World Report: Towards Knowledge Societies, 2005*, http://unesdoc.unesco.org/images/0014/001418/141843e.pdf.
10. Wikipedia. *Programming paradigm* (2014, Jun 19), [Online]. Available: http://en.wikipedia.org/wiki/Programming_paradigm.
11. Wikipedia. *Comparison of programming languages* (2014, July 06), [Online]. Available: http://en.wikipedia.org/wiki/ Comparison_of_programming_languages.
12. Κασσιός *Γ., Εισαγωγή στο Συναρτησιακό Προγραμματισμό*, http://cgi.di.uoa.gr/~kassios/courses/fp/notes/0.pdf.

Λεξικό Βασικών Όρων

Αλγόριθμος. Η ακριβής περιγραφή μιας σειράς βημάτων που απαιτούνται για την επίλυση ενός προβλήματος.

Αναδρομικός αλγόριθμος. Ο αλγόριθμος που καλεί άμεσα ή έμμεσα τον εαυτό του μία ή περισσότερες φορές, επιλύοντας κάθε φορά ένα πρόβλημα της ίδιας φύσης με το αρχικό, αλλά μικρότερου μεγέθους.

Αναζήτηση. Λειτουργία σε πίνακα, που συνίσταται στην εξέταση των στοιχείων του για την εύρεση ενός ζητούμενου.

Αναλυτική μεθοδολογία. Μέθοδος σχεδιασμού με διαδοχικά βήματα.

Αντικειμενοστραφής προγραμματισμός. Μεθοδολογία ανάπτυξης προγραμμάτων προστακτικού τύπου με κεντρική ιδέα την κλάση αντικειμένων.

Βάση Δεδομένων. Μια οργανωμένη συλλογή από δεδομένα αποθηκευμένα στον Η/Υ που σχετίζονται μεταξύ τους.

Βρόχος. Μια επαναληπτική διαδικασία.

Γλώσσες προγραμματισμού. Αναπαράσταση αλγορίθμου με κείμενο σε γλώσσα περιορισμένου συνόλου κωδικοποιημένων εντολών, που είναι κατανοητή από τον υπολογιστή.

Δεδομένα (data). Τα στοιχεία που αναπαρίστανται σε αλγορίθμους και προγράμματα και υφίστανται επεξεργασία.

Δεδομένο προβλήματος .Ένα γνωστό ή αποδεκτό στοιχείο το οποίο χρησιμοποιείται ως βάση ή προϋπόθεση στην επίλυση προβλημάτων.

Διάγραμμα ροής. Αναπαράσταση αλγορίθμου με χρήση γεωμετρικών σχημάτων.

Διαδίκτυο. Δίκτυο που σχηματίζεται από την ενοποίηση όλων των μικρών και μεγάλων δικτύων υπολογιστών.

Διερμηνευτής. Μεταφραστικό πρόγραμμα που αναλύει, μεταφράζει και εκτελεί το πηγαίο πρόγραμμα εντολή προς εντολή.

Δίκτυα ευρείας περιοχής. Δίκτυο που οι κόμβοι του βρίσκονται σε ακτίνα εκατοντάδων χιλιάδων μέτρων.

Δίκτυο υπολογιστών. Σύνδεση δύο ή περισσότερων Η/Υ με σκοπό την ανταλλαγή δεδομένων και τη διαμοίραση πόρων.

Δομή δεδομένων. Τρόπος οργάνωσης, συσχέτισης και αποθήκευσης των δεδομένων.

Δομημένος προγραμματισμός. Στυλ προγραμματισμού το οποίο βασίζεται στην έννοια της κλήσης διαδικασιών.

Δρομολογητής. Συσκευή δικτύου που χωρίζει τα δεδομένα σε πακέτα και τα προωθεί προς τον παραλήπτη.

Δυναμικές δομές δεδομένων. Δομές δεδομένων που δεν έχουν σταθερό μέγεθος και αυξομειώνονται με τη μέθοδο της δυναμικής παραχώρησης μνήμης.

Εκσφαλματωτής. Ειδικό πρόγραμμα που επιτρέπει την άμεση εκτέλεση και παρακολούθηση του πηγαίου προγράμματος με σκοπό τον εντοπισμό των λογικών λαθών.

Επιστήμη Υπολογιστών. Η επιστήμη που ασχολείται με τους υπολογιστές και τους υπολογισμούς.

Κατανεμημένο υπολογιστικό σύστημα. Υπολογιστικό σύστημα που αποτελείται από επί μέρους τμήματα, τα οποία συνδέονται μέσω δικτύου υπολογιστών και συνεργάζονται μεταξύ τους για να επιλύσουν συγκεκριμένα προβλήματα.

Κεντρική Μονάδα Επεξεργασίας. Κρίσιμος πόρος του υπολογιστή που εκτελεί τις βασικές πράξεις.

Κύρια Μνήμη. Βασικό μέσο προσωρινής αποθήκευσης στον Η/Υ που αποτελεί ξεχωριστό άρθρωμα της μητρικής πλακέτας.

Λειτουργικό Σύστημα. Ομάδα προγραμμάτων που λειτουργεί ως σύνδεσμος ανάμεσα στο υλικό και το λογισμικό.

Λογικά λάθη. Λάθη στη λογική του προγράμματος που οδηγούν σε αδυναμία εκτέλεσης ή σε λανθασμένα αποτελέσματα. Συμβαίνουν και εντοπίζονται κατά την εκτέλεση του προγράμματος.

Λογισμικό. Τα προγράμματα που περιέχουν τις εντολές προς τις συσκευές του υπολογιστή.

Μεταβλητή. Μια ποσότητα που αναπαριστά ένα στοιχείο που έχει νόημα στον πραγματικό κόσμο και η τιμή της

μπορεί να αλλάζει κατά τη διάρκεια εκτέλεσης του αλγορίθμου.

Μεταγλωττιστής .Πρόγραμμα ποιυ διαβάζει και ελέγχει όλο το πηγαίο πρόγραμμα, και παράγει ένα ισοδύναμο πρόγραμμα σε γλώσσα μηχανής..

Μεταφραστής. Πρόγραμμα που αναλαμβάνει να μετατρέψει το πηγαίο πρόγραμμα σε γλώσσα μηχανής.

Νεφοπληροφορική. Τεχνολογία που προσφέρει στο χρήστη υπολογιστικούς πόρους (υλικό, λογισμικό, δεδομένα) ως υπηρεσίες του Διαδικτύου.

Παράμετρος. Μεταβλητή που επιτρέπει το πέρασμα τιμών από ένα τμήμα προγράμματος σε ένα άλλο.

Πίνακας. Διάταξη στοιχείων του ίδιου τύπου με συγκεκριμένο μέγεθος, στην οποία όλα τα στοιχεία αναφέρονται με το ίδιο όνομα και έναν ή περισσότερους δείκτες.

Πληροφοριακό Σύστημα. Ένα σύνολο από ανθρώπους, διαδικασίες, δεδομένα, λογισμικό και υλικό υπολογιστών, που αλληλεπιδρούν μεταξύ τους με οργανωμένο τρόπο για να υποστηρίξουν την ανθρώπινη δραστηριότητα.

Πληροφορική. Η επιστήμη που ασχολείται με την αναπαράσταση, αποθήκευση και επεξεργασία της πληροφορίας.

Πολυπλοκότητα αλγορίθμων. Κλάδος της Θεωρίας Υπολογισμού που μελετά την απόδοση των αλγορίθμων.

Πρόβλημα. Μια κατάσταση που απαιτεί λύση αλλά η λύση της δεν είναι γνωστή ούτε προφανής

Στατικές δομές δεδομένων. Δομές δεδομένων που έχουν σταθερό μέγεθος και καθορίζεται κατά την δημιουργία (συγγραφή) του προγράμματος.

Συνθετική μεθοδολογία. Μέθοδος σχεδίασης βασισμένη στη «σταδιακή εξέλιξη». Η γενική της αρχή είναι ότι η λύση ενός προβλήματος επιτυγχάνεται από τη σύνθεση των επί μέρους στοιχείων και δεδομένων.

Συντάκτης. Ειδικό πρόγραμμα που βοηθάει στη συγγραφή ενός προγράμματος.

Συντακτικά λάθη. Λάθη που παραβιάζουν τους κανόνες ορθογραφίας ή σύνταξης της γλώσσας με την οποία εκφράζεται ένας αλγόριθμος ή ένα πρόγραμμα. Συμβαίνουν κατά τη σύνταξη του προγράμματος και εντοπίζονται κατά τη μετάφραση του.

Σύστημα αρχείων. Ο τρόπος με τον οποίο διαχειρίζεται το Λ.Σ. τα δεδομένα της δευτερεύουσας μνήμης του Η/Υ.

Σύστημα Διαχείρισης Βάσεων Δεδομένων. Σύνολο προγραμμάτων που διευκολύνουν τη δημιουργία και τη λειτουργία μιας βάσης δεδομένων

Ταξινόμηση. Λειτουργία σε πίνακα, που συνίσταται στην αλλαγή της θέσης των στοιχείων του ώστε να ακολουθούν μια συγκεκριμένη διάταξη, π.χ. αύξουσα ή φθίνουσα.

Τελεστής. Το σύμβολο που χρησιμοποιείται για την εκτέλεση μιας πράξης.

Τεχνητή Νοημοσύνη. Η επιστήμη που επιχειρεί να δημιουργήσει οντότητες με ευφυΐα, συστηματοποιώντας και αυτοματοποιώντας τις νοητικές λειτουργίες.

Τοπικό δίκτυο. Δίκτυο που οι κόμβοι του βρίσκονται σε ακτίνα μερικών δεκάδων ή εκατοντάδων μέτρων.

Τοπολογία δικτύου. Ο τρόπος με τον οποίο είναι συνδεδεμένοι οι κόμβοι ενός δικτύου.

Τύπος δεδομένων. Το είδος δεδομένων που μπορεί να παραστήσει μια γλώσσα προγραμματισμού.

Υλικό. Τα στοιχεία ενός υπολογιστή που έχουν υλική υπόσταση.

Υπολογισιμότητα. Κλάδος της επιστήμης των υπολογιστών που αναζητά τι μπορεί να υπολογισθεί και τι όχι

Υπολογιστικά προβλήματα. Προβλήματα που ζητάμε να βρούμε την απάντηση που ικανοποιεί τα δεδομένα του.

Υποπρόγραμμα. Τμήμα πηγαίου κώδικα που εκτελεί συγκεκριμένη λειτουργία, μπορεί να διαχωριστεί από το υπόλοιπο πρόγραμμα και να χρησιμοποιηθεί πολλές φορές σε διαφορετικά σημεία ενός ή περισσότερων προγραμμάτων.

Ευρετήριο εικόνων

Ευρετήριο αλγορίθμων

Ευρετήριο προγραμμάτων

Βιογραφικά Συγγραφικής Ομάδας

Γιώργος Γώγουλος
Γεννήθηκε στην Τρίκαλα και σπούδασε στο Μαθηματικό τμήμα του Αριστοτέλειου Πανεπιστήμιου Θεσσαλονίκης. Κάτοχος Μεταπτυχιακού Διπλώματος του Τμήματος Επιστημών της Προσχολικής Αγωγής και του Εκπαιδευτικού Σχεδιασμού της σχολής Ανθρωπιστικών Επιστημών του Πανεπιστημίου Αιγαίου. Διορισμένος στη δευτεροβάθμια εκπαίδευση και από το 1997 Σχολικός Σύμβουλος Πληροφορικής στη Δυτική Κρήτη. Εκπαιδευτής ενηλίκων και συγγραφέας.
Συγγραφική δράση:
1. «Λειτουργικά Συστήματα». ΟΕΔΒ, ISBN: 460-06-1433-4
2. «Το Σχολικό Εργαστήριο Πληροφορικής». ISBN: 960-7251-05-9

Γιώργος Κοτσιφάκης
Γεννημένος το 1970 στα Χανιά. Πτυχιούχος από το Τμήμα Επιστήμης Υπολογιστών του Πανεπιστημίου Κρήτης το 1992. Μεταπτυχιακές σπουδές (MSc by research) στην Επιστήμη Υπολογιστών στο Πανεπιστήμιο UMIST στο Manchester. Από το 2001 διορισμένος στη Δευτεροβάθμια Εκπαίδευση Χανίων ως Καθηγητής Πληροφορικής.
Έχει εργαστεί ως Αναλυτής/ Προγραμματιστής στη διεύθυνση Μηχανογράφησης της Τράπεζας Εργασίας Α.Ε. στην Αθήνα, την περίοδο 1998-2000, και στην εταιρία Διάλογος Α.Ε. – Συστήματα Επικοινωνίας Φωνής στα Χανιά την περίοδο 2000-2001.

Γεωργία Κυριακάκη
Απόφοιτος του τμήματος Ηλεκτρονικών και Μηχανικών Υπολογιστών του Πολυτεχνείου Κρήτης με Μεταπτυχιακό Δίπλωμα Ειδίκευσης από τον Τομέα Πληροφορικής του ίδιου τμήματος. Έχει εργαστεί σε ερευνητικά προγράμματα του Πολυτεχνείου Κρήτης για το Εργαστήριο Διανεμημένων Πληροφοριακών Συστημάτων - TUC/MUSIC του τμήματος Ηλεκτρονικών Μηχανικών & Μηχανικών Υπολογιστών και για το Εργαστήριο Συστημάτων Υποστήριξης Αποφάσεων - ΕΡΓΑΣΥΑ της Σχολής Μηχανικών Παραγωγής και Διοίκησης. Έχει 12 δημοσιεύσεις σε διεθνή περιοδικά και συνέδρια. Είναι μόνιμος εκπαιδευτικός ΠΕ19 από το 2000.

Απόστολος Παπαγιάννης
Γεννήθηκε στην Αθήνα και σπούδασε στο τμήμα Μηχανικών Η/Υ και Πληροφορικής του Πανεπιστημίου Πατρών απ' όπου έλαβε και το μεταπτυχιακό του δίπλωμα. Εργάστηκε σε ερευνητικά προγράμματα του Ινστιτούτου Τεχνολογίας Υπολογιστών, αλλά και ως μηχανικός λογισμικού σε εταιρείες πληροφορικής. Από το 2003 εργάζεται ως καθηγητής πληροφορικής στη δευτεροβάθμια εκπαίδευση.

Μανόλης Φραγκονικολάκης
Πτυχιούχος του Τμήματος Επιστημόνων Ηλεκτρονικών Υπολογιστών του Πανεπιστημίου Κρήτης και κάτοχος Μεταπτυχιακού Διπλώματος του Τμήματος Μηχανικών Ηλεκτρονικής και Υπολογιστών του Πολυτεχνείου Κρήτης. Διορισμένος στη δευτεροβάθμια εκπαίδευση και από το 1997 Υπεύθυνος του ΚΕντρου ΠΛΗροφορικής και ΝΕων Τεχνολογιών στα Χανιά.
Έχει εργαστεί σαν επιστημονικός συνεργάτης στο Πολυτεχνείο Κρήτης με συμμετοχή σε Ευρωπαϊκά προγραμμάτων έρευνας και ανάπτυξης, όπως και σαν Τεχνικός Διευθυντής στην εταιρεία πολυμέσων και υπηρεσιών διαδικτύου Multimedia Systems Center A.E.

Παναγιώτα Χίνου
Πτυχιούχος του Τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ (1992) και κάτοχος Μεταπτυχιακού Διπλώματος του Τμήματος Μηχανικών Παραγωγής και Διοίκησης (1996).
Το διάστημα από το 1994 έως το 1997 Βοηθός στο εργαστήριο Οργάνωσης & Διοίκησης του Πολυτεχνείου Κρήτης. Διορισμένη από το 1999 στη Β/θμια εκπαίδευση.

ISBN: 978-618-81242-4-0

ΕΚΔΟΣΕΙΣ
ΠΥΞΙΔΑ
www.pyxida.gr